10-716: Advanced Machine Learning: Theory and Methods

Spring 2023 • Prof. Pradeep Ravikumar

Notes by Fan Pu Zeng

Contents

Lecture 1 — January 17, 2023 **5**
1 Statistical Decision Theory 5

Lecture 2 — January 19, 2023 **9**
1 Decomposable Losses 9
2 Classification 10
3 A Criticism of the Likelihood Principle 11
4 Priors over All Possible Distributions 11
5 The Empirical CDF 12

Lecture 3 — January 24, 2023 **13**
1 Non-parametric Bayes 13
2 Gaussian Processes 17

Lecture 4 — January 26, 2023 **18**
1 Point Masses 18
2 Gaussian Processes 18
3 Consistency of Non-Parametric Bayes 20
4 Frequentist Non-Parametric Methods 20

Lecture 5 — January 31, 2023 **22**
1 Non-Parametric Density Estimation 22
2 Histograms 25
3 Curse of Dimensionality 26
4 Orthogonal Basis Expansions 26

Lecture 6 — February 2, 2023 **28**
1 Non-Parametric Density Estimation 28
2 Choosing the Bandwidth 30

Lecture 7 — February 7, 2023 **33**
1 Non-Parametric versus Parametric Rates . . . 33
2 Regression . . . 34

Lecture 8 — February 9, 2023 **38**
1 Non-Parametric Regression . . . 38
2 Hilbert Spaces . . . 40

Lecture 9 — February 14, 2023 **42**
1 Reproducing Kernel Hilbert Spaces . . . 42
2 Linear Smoothers . . . 44
3 Wavelets . . . 45
4 Classification . . . 45

Lecture 10 — February 16, 2023 **47**
1 Non-Parametric Classification . . . 47
2 Three Key Ideas in Non-Parametric Estimation . . . 49
3 Greedy Algorithms . . . 49

Lecture 11 — February 21, 2023 **52**
1 Greedy Algorithms and Boosting . . . 52
2 Optimal Transport . . . 54

Lecture 12 — February 23, 2023 **57**
1 Optimal Transport . . . 57
2 Wasserstein Distance . . . 58
3 Integral Probability Metrics . . . 59

Lecture 13 — February 28, 2023 **61**
1 Midterm Review . . . 61
2 Optimal Transport . . . 64

Lecture 14 — March 14, 2023 **65**
1 Deep Density Estimation . . . 65

Lecture 15 — March 16, 2023 **68**
1 Generating Data from Noise . . . 68
2 Normalizing Flows . . . 69
3 Non-Invertible Generators . . . 70
4 Gaussianization . . . 71

Lecture 16 — March 21, 2023 **72**
1 Kernels and Eigenvalue Decay . . . 72
2 Choosing a Kernel . . . 73
3 Representation Learning . . . 74
4 Random Features . . . 75

Lecture 17 — March 23, 2023 **77**
1 Modern Paradigm Shift . . . 77
2 Infinitely Wide Networks as Gaussian Processes . . . 77
3 The Neural Tangent Kernel . . . 80

Lecture 18 — March 28, 2023 **82**
1 Dimension Reduction (Guest Lecture) . . . 82
2 Principal Component Analysis (PCA) . . . 82
3 Multi-Dimensional Scaling (MDS) . . . 84
4 Kernel PCA . . . 84

Lecture 19 — March 30, 2023 **87**
1 Dimension Reduction . . . 87
2 Local Linear Embeddings (LLE) . . . 87
3 Isomap . . . 88
4 Laplacian Eigenmaps (Belkin–Niyogi 2003) . . . 88
5 Diffusion Maps (Coifman–Lafon 2006) . . . 89
6 Random Projections . . . 89

Lecture 20 — April 4, 2023 **90**
1 Clustering . . . 90
2 Mixture Models . . . 93
3 Score Matching . . . 94

Lecture 21 — April 6, 2023 **96**
1 Mixture Models . . . 96
2 Level Set Clustering . . . 97
3 Mode-Based Clustering . . . 99
4 Hierarchical Clustering . . . 100

Lecture 22 — April 11, 2023 **101**
1 Online Learning . . . 101
2 Restricting the Adversary: Strongly Convex Losses . . . 102
3 Randomized Play . . . 104
4 Best Response . . . 105

Lecture 23 — April 18, 2023 **106**
1 Online Learning with Mixed Strategies . . . 106
2 Games . . . 107
3 Online Classification . . . 109

Lecture 24 — April 20, 2023 **110**
1 Causal Effects . . . 111
2 Causal Graphs . . . 112

Lecture 25 — April 25, 2023 **115**
1 Aggregation . . . 115
2 Random Forests . . . 117

Lecture 26 — April 27, 2023 **119**
1 Mercer Kernels . . . 119
2 Clustering . . . 121
3 Online Learning . . . 123

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 1 — January 17, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Statistical Decision Theory

1.1 State of nature

There is an unknown *state of nature* θ. We do not see θ directly. Instead, an observation process produces an observation X.

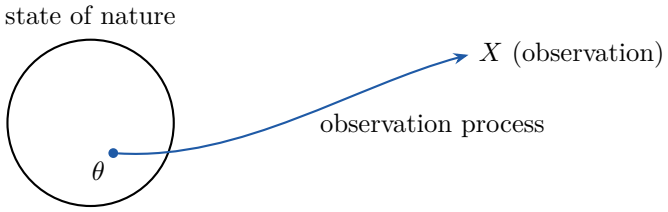

Figure 1: The state of nature θ generates the observation X through an observation process.

The setup is:

- the observation is drawn as $X \sim P(\cdot\,; \theta)$;
- an *action set* $\mathcal{A}$;
- a *loss function* $L(\theta, a)$, with $L : \Theta \times \mathcal{A} \to \mathbb{R}$.

Definition (Decision rule)**.** A *decision rule* is a map $\delta : \mathcal{X} \to \mathcal{A}$ from observations to actions.

The loss $L(\theta, \delta(X))$ of a decision rule is random, and we don't want a random objective. Instead we use the risk.

Definition (Risk)**.** The *risk* of a decision rule δ at state of nature θ is

$$R(\theta, \delta) = \mathop{\mathbb{E}}_{X \sim P(\cdot;\theta)} L(\theta, \delta(X)).$$

1.2 Minimax risk

Given two decision rules δ_1, δ_2, it could be the case that

$$R(\theta_1, \delta_1) < R(\theta_1, \delta_2) \qquad \text{and} \qquad R(\theta_2, \delta_1) > R(\theta_2, \delta_2).$$

So in state θ_1 we prefer δ_1, and in state θ_2 we prefer δ_2. But we don't know the state of nature!

Definition (Minimax risk)**.** The *minimax risk* is

$$\inf_{\delta} \underbrace{\max_{\theta \in \Theta} R(\theta, \delta)}_{\text{worst-case risk}},$$

where the infimum is over all possible functions from observations to actions.

This set of functions is too large! The goal is to come up with an estimator whose worst-case risk is close to the minimax risk.

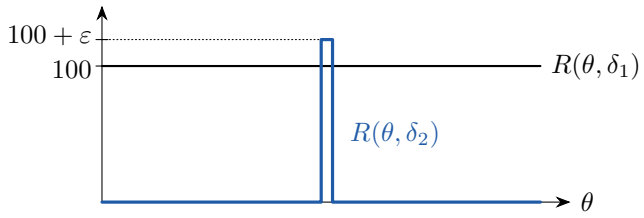

Figure 2: Minimax risk prefers δ_1 (worst case 100 versus $100 + \varepsilon$), but δ_2 looks better almost everywhere.

1.3 Bayes risk

Given a *prior* π over states of nature, the *Bayes risk* is

$$r(\pi, \delta) = \int R(\theta, \delta)\, \pi(\theta)\, d\theta.$$

Having the prior is contentious: how do we know what distribution the states of nature follow? Often one uses the uniform distribution if there is no idea. Bayesians argue that using $\int R(\theta, \delta)\pi(\theta)\, d\theta$ is rational, since it is how humans work.

Writing $p(x \mid \theta)\pi(\theta) = \pi(\theta \mid x)\, p(x)$,

$$\begin{aligned} \int R(\theta, \delta)\, \pi(\theta)\, d\theta &= \int_\theta \int_x L(\theta, \delta(x))\, p(x \mid \theta)\, \pi(\theta)\, dx\, d\theta \\ &= \int_x \left[\int_\theta L(\theta, \delta(x))\, \pi(\theta \mid x)\, d\theta \right] p(x)\, dx. \end{aligned}$$

So the Bayes risk is minimized by minimizing the inner integral separately for each x:

$$\delta_\pi(x) \in \arg\min_{a \in \mathcal{A}} \int_\theta L(\theta, a)\, \pi(\theta \mid x)\, d\theta,$$

where the subscript π refers to the prior. Then

$$\delta_\pi \in \arg\min_{\delta} r(\pi, \delta),$$

that is, δ_π minimizes the Bayes risk.

1.4 Problems with the Bayesian formulation

- Computing the integral over the posterior is intractable.
- The prior is unknown.

But even worse, the formulation itself is flawed. Recall

$$R(\theta, \delta) = \mathop{\mathbb{E}}_{X \sim P(\cdot;\theta)} L(\theta, \delta(X)).$$

Example 1. Let the true state be θ^*, and let each observation be

$$X = \begin{cases} \theta^* - 1 & \text{with probability } 1/2, \\ \theta^* + 1 & \text{with probability } 1/2. \end{cases}$$



Figure 3: Each observation lands at $\theta^* - 1$ or $\theta^* + 1$ with equal probability.

We observe $\bar{X} := (X_1, X_2)$ and use the estimator

$$\delta(\bar{X}) = \frac{X_1 + X_2}{2} \mathbb{I}[X_1 \neq X_2] + (X_1 + 1)\, \mathbb{I}[X_1 = X_2].$$

The action space is $\mathcal{A} = \Theta = \mathbb{R}$, and the loss is $L(\theta, a) = \mathbb{I}[\theta \neq a]$. Then

$$R(\theta^*, \delta) = \underbrace{\tfrac{1}{2} \times 0}_{\text{if } X_1 \neq X_2} + \tfrac{1}{2} \times \tfrac{1}{2} = \tfrac{1}{4}.$$

But if you see $X_1 \neq X_2$, you know exactly what θ^* is. If you see $X_1 = X_2$, you are just randomly guessing either $X_1 + 1$ or $X_1 - 1$, which is correct with probability 50%.

1.5 The conditionality principle

The *conditionality principle* says that we should condition on the information that we have. The risk above clearly fails that.

Definition (Weak conditionality principle)**.** Suppose there are two experiments E_1, E_2, and the experiment E_J that is performed is chosen by a random index J with $\mathbb{P}(J = 1) = \mathbb{P}(J = 2) = 1/2$. Say we observe $j \in \{1, 2\}$ and the outcome of E_j. Then any inference about the state of nature θ should only depend on E_j.

So the Bayesian approach fails this, since it considers observations that we never observed.

Example 2 (Electrician with a voltmeter)**.** An electrician takes 100 observations of 75–90V, so $X_1, \ldots, X_{100} \in [75, 90]$. Model them as $X \sim N(\mu, \sigma^2)$, with $\theta = (\mu, \sigma^2)$.

The electrician then says: "My voltmeter is clamped at 100V! But I have a backup one that goes above 100V. . . "

So if we take an expectation, we consider experiments that we didn't actually perform. Recall

$$r(\pi, \delta) = \mathop{\mathbb{E}}_{\theta \sim \pi} R(\theta, \delta),$$
$$\delta_\pi(x) \in \arg\inf_{a \in \mathcal{A}} \int_\Theta L(\theta, a)\, \pi(\theta \mid x)\, d\theta.$$

The Bayes optimal estimator satisfies the conditionality principle: the minimizer $\delta_\pi(x)$ does not worry about averaging over x.

1.6 The likelihood principle

The *likelihood principle* says that inference about the state of nature θ should only be based on the likelihood

$$\ell(\theta) = P(x; \theta).$$

The maximum likelihood estimator is

$$\hat{\theta}_{\mathrm{MLE}} \in \arg\max_{\theta \in \Theta} P(x; \theta).$$

The *sufficiency principle* says to use only sufficient statistics.

The MLE respects the conditionality principle, the likelihood principle and the sufficiency principle.

The problem with the MLE is that you try to recover the entire state of nature, but this is more than you need.

1.7 PAC: probably approximately correct

We would like, with probability at least $1 - \delta$,

$$L(\theta, \delta(X)) \le \inf_{a \in \mathcal{A}} L(\theta, a) + \varepsilon.$$

Is this possible? If we knew θ, we could take

$$a_\theta \in \arg\min_{a \in \mathcal{A}} L(\theta, a).$$

In general this cannot be optimized: we cannot optimize over θ if we don't even see it.

But for a specific class of losses that we care about, this works:

$$L(\theta, a) = \mathop{\mathbb{E}}_{X \sim P(\cdot; \theta)} \ell(X, a).$$

Given samples $X_1, \dots, X_n \sim P(\cdot\,; \theta)$, we can minimize the *empirical loss*

$$L_n(\theta, a) := \frac{1}{n} \sum_{i=1}^n \ell(X_i, a),$$

since $L(\theta, a) \approx L_n(\theta, a)$. This gives the *empirical risk minimizer*

$$\hat{\delta}_{\mathrm{ERM}}(X) \in \arg\inf_{a \in \mathcal{A}} L_n(\theta, a).$$

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 2 — January 19, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Decomposable Losses

Recall the PAC goal: for all $\theta \in \Theta$, with probability at least $1 - \delta$,

$$L(\theta, \delta(X)) \leq \inf_a L(\theta, a) + \varepsilon.$$

A loss is *decomposable* if we are able to write it as

$$L(\theta, a) = \mathop{\mathbb{E}}_{X \sim P(\cdot;\theta)} \ell(X, a).$$

Given samples $\{X_i\}_{i=1}^n \sim P(\cdot\,;\theta)$, the empirical loss is

$$L_n(\theta, a) = \frac{1}{n} \sum_{i=1}^n \ell(X_i, a).$$

For a specific a, $L_n(\theta, a)$ is close to $L(\theta, a)$ by the law of large numbers. Uniformly over θ and a, they are close by *uniform deviation bounds*:

$$\sup_{\theta,a} \big| L_n(\theta, a) - L(\theta, a) \big| \asymp \frac{\mathrm{comp}(\Theta, \mathcal{A})}{n^\alpha},$$

where $\mathrm{comp}(\Theta, \mathcal{A})$ is the complexity of the class (for example, a rate of $1/\sqrt{n}$). The ERM action is

$$\hat{a}_{\mathrm{ERM}} \in \arg\min_{a \in \mathcal{A}} L_n(\theta, a).$$

1.1 Which losses are decomposable?

For a fixed action a, write

$$L_a(\theta) = \mathop{\mathbb{E}}_{X \sim P(\cdot;\theta)} \ell_a(X).$$

Dropping the subscript, the question is whether we can write

$$L(\theta) = \mathop{\mathbb{E}}_{X \sim P_\theta} \big[\ell(X)\big].$$

It is not always possible to express a loss this way, for example $L(\theta) = \|\theta\|_p$.

Axiom 1. If $L(\theta_1) < L(\theta_2) < L(\theta_3)$, then there exist $\alpha, \beta \in (0, 1)$ such that

$$L(\alpha\theta_1 + (1-\alpha)\theta_3) < L(\theta_2) < L(\beta\theta_1 + (1-\beta)\theta_3).$$

Axiom 2. If $L(P_1) < L(P_2)$, then for all $\alpha \in (0, 1)$ and all P_3,

$$L(\alpha P_1 + (1-\alpha)P_3) < L(\alpha P_2 + (1-\alpha)P_3).$$

Theorem 1. *If these two axioms are satisfied, then there exists ℓ such that*

$$L(P) = \mathop{\mathbb{E}}_{X\sim P}[\ell(X)].$$

Example 3 (KL loss)**.** Let $\theta \in \Theta$, $\mathcal{A} = \Theta$ and $X \sim P_\theta$. Let θ^* be the true state of nature, and take the loss

$$\begin{aligned} L(\theta^*, \theta) = \mathrm{KL}(P_{\theta^*}, P_\theta) &= \mathop{\mathbb{E}}_{X\sim P_{\theta^*}} \log \frac{P_{\theta^*}(X)}{P_\theta(X)} \\ &= \underbrace{\mathop{\mathbb{E}}_{X\sim P_{\theta^*}} \log P_{\theta^*}(X)}_{\text{constant}} - \mathop{\mathbb{E}}_{X\sim P_{\theta^*}} \log P_\theta(X) \\ &= C + \mathop{\mathbb{E}}_{X\sim P_{\theta^*}} [-\log P_\theta(X)]. \end{aligned}$$

So the KL loss is decomposable, with

$$\ell(\theta, x) = -\log P_\theta(x).$$

Minimizing the empirical version of this loss is maximum likelihood. So the MLE is pointwise close to the optimal parameter, which is a reason why it is so influential.

2 Classification

Let $X \in \mathcal{X}$ and $Y \in \{-1, +1\}$, with $(X, Y) \sim P \in \mathcal{P}$. A *classifier* is a map $f : \mathcal{X} \to \{-1, +1\}$. The *0/1 loss* is

$$L(P, f) = P\big[f(X) \neq Y\big] = \mathop{\mathbb{E}}_{(X,Y)\sim P} \ell((X, Y), f), \qquad \text{where } \ell((X, Y), f) = \mathbb{I}\big[f(X) \neq Y\big].$$

The empirical loss is

$$L_n(P, f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\big[f(X_i) \neq Y_i\big],$$

and the ERM classifier is

$$\hat{f}_{\mathrm{ERM}} \in \arg\min_{f\in\mathcal{F}} L_n(P, f).$$

Here the distribution P is the state of nature, and the actions are classifiers.

2.1 The Bayesian approach

Put a prior π over $\mathcal{P}$, the set of all possible data distributions. Given data $D := \{(X_i, Y_i)\}_{i=1}^n$, we get a posterior $\pi(P \mid D)$. The Bayesian estimator is

$$\hat{f}_\pi = \arg\inf_{f\in\mathcal{F}} \int_{P\in\mathcal{P}} L(P, f)\, d\pi(P \mid D).$$

3 A Criticism of the Likelihood Principle

Example 4. Consider words over the letters $\{a, a^{-1}, b, b^{-1}\}$, where adjacent opposite letters annihilate. For example,

$$a\,b\,a^{-1}a\,b\,b^{-1}b = abb, \qquad a^{-1}a^{-1}aa \to \emptyset.$$

Suppose the unknown parameter is the word $\theta = abb$. We draw another letter, here b^{-1}, append it, and observe the reduced word $x = ab$.

What could be the θ that gave rise to $x = ab$? It must be one of

$$\theta \in \{a,\ aba,\ aba^{-1},\ abb\},$$

and for each of these $P(x \mid \theta) = 1/4$. With the uniform prior $\pi(\theta_i) = 1/4$, the posterior is also uniform over these four values of θ (the Bayesian approach). The MLE $\max_\theta P(x;\theta)$ is also indifferent between them. So the MLE and the Bayesian approach would both agree.

But now fix a parameter, say $\theta_2 = aba$, and look at the observations it can produce.

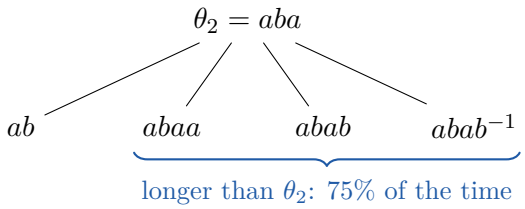

Figure 4: The four equally likely observations from $\theta_2 = aba$; in three of them, x is longer than θ.

So 75% of the time, x is longer than θ. Then we should say that 75% of the time, the choice that θ is shorter than x would have been right.

This is a criticism of the likelihood principle itself, especially so when there is a lot of structure in the problem in addition to the likelihood. Frequentist thinking asks: what is the optimal action if I need to perform this many times?

4 Priors over All Possible Distributions

Let $\mathcal{P}$ be the set of all possible distributions (for example, over images and labels). We want

- a prior over $\mathcal{P}$, and
- a posterior over $\mathcal{P}$.

4.1 Parametric case

Let the parameters be drawn from the prior, $\theta \sim \pi$, and $X_i \sim P_\theta$ with density f_θ. The posterior is

$$\pi(\theta \mid \mathbf{X}) = \frac{\prod_{i=1}^n f_\theta(X_i)\,\pi(\theta)}{\int_\Theta \prod_{i=1}^n f_\theta(X_i)\,\pi(\theta)\,d\theta}, \qquad \mathbf{X} = (X_1, \ldots, X_n),$$

where $\prod_{i=1}^{n} f_\theta(X_i)$ is the likelihood of the data. The class of densities is $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$.

4.2 All possible densities

Now let $\mathcal{F}$ be the set of all possible densities. Why? We do not really believe that our data is Gaussian, but it will be in this set for sure. For example,

$$\mathcal{F} = \left\{ f \;\middle|\; \int (f''(x))^2 dx < \infty \right\}.$$

Put a prior π over $\mathcal{F}$, a distribution over densities. The posterior is

$$\pi(f \mid \mathbf{X}) = \frac{\prod_{i=1}^{n} f(X_i)\, \pi(f)}{\int \prod_{i=1}^{n} f(X_i)\, d\pi(f)}.$$

The denominator needs us to normalize over an infinite-dimensional set. But we don't need a closed form for the normalization: we could just as well have algorithms to sample from π.

5 The Empirical CDF

Consider the set of all possible distributions, and draw $X_1, \ldots, X_n \sim F$ from the distribution defined by the CDF F,

$$F(t) = \mathbb{P}[X \leq t],$$

which is always well-defined. The *empirical CDF* is

$$F_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[X_i \leq t].$$

One of the earliest uniform deviation bounds says that

$$\sup_{t \in (-\infty, \infty)} |F_n(t) - F(t)| \asymp \frac{1}{\sqrt{n}}.$$

Precisely,

$$\mathbb{P}\Big[\sup_t |F_n(t) - F(t)| > \varepsilon\Big] \leq 2\exp(-2n\varepsilon^2),$$

so the deviation scales as $1/\sqrt{n}$.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 3 — January 24, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Non-parametric Bayes

1.1 Dirichlet process

Let $\mathcal{P}$ be the set of all distributions, with a prior π over $\mathcal{P}$.

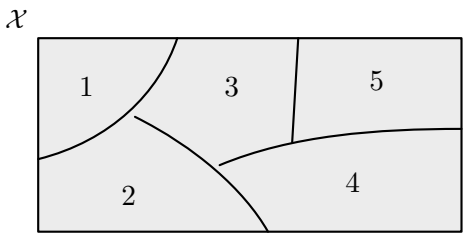

Figure 5: Discretizing the space $\mathcal{X}$ into a finite number of regions (here 5).

If you give me a categorical distribution, then a multinomial distribution is as general as it can be. So we discretize a continuous distribution by a categorical distribution:

$$X \in \{1, \ldots, k\}, \qquad P = (P_1, \ldots, P_k),$$

where P lies in the k-dimensional simplex:

$$\sum_{j=1}^{k} P_j = 1, \qquad P_j \geq 0.$$

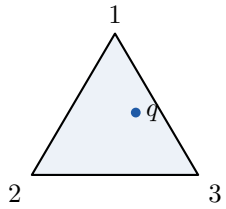

Figure 6: The simplex for $k = 3$, with a point q in it.

The *Dirichlet distribution* is a distribution over the simplex. We write

$$P \sim \mathrm{Dir}(\alpha, q), \qquad q \in \Delta_k, \quad \alpha \in \mathbb{R}^+,$$

where Δ_k is the k-dimensional simplex. Its density is

$$f(p) = \frac{\Gamma(\alpha)}{\prod_{j=1}^{k} \Gamma(\alpha q_j)} \prod_{j=1}^{k} p_j^{\alpha q_j - 1}.$$

Problems: how do we discretize the space in higher dimensions? How do we compute the normalizing constant?

Discretize $\mathcal{X}$ to k values, using a partition $A_1, \ldots, A_k$.

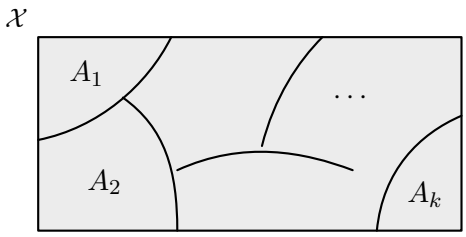

Figure 7: A partition $A_1, \ldots, A_k$ of $\mathcal{X}$.

Given a *base distribution* F_0, set

$$q = (F_0(A_1), \ldots, F_0(A_k)),$$

and draw

$$p \sim \mathrm{Dir}(\alpha, q), \qquad (p_1, \ldots, p_k) \in \Delta_k.$$

Definition (Dirichlet process)**.** A *Dirichlet process* $F \sim \mathrm{DP}(\alpha, F_0)$ is a random distribution F such that for every partition $(A_1, \ldots, A_k)$ of $\mathcal{X}$,

$$(F(A_1), \ldots, F(A_k)) \sim \mathrm{Dir}\Big(\alpha, (F_0(A_1), \ldots, F_0(A_k))\Big).$$

1.2 Stick-breaking construction

A random draw $F \sim \mathrm{DP}(\alpha, F_0)$ is made as follows.

- Draw $S_1, S_2, \cdots \sim F_0$.
- Draw $V_1, V_2, \cdots \sim \mathrm{Beta}(1, \alpha)$, so $V_i \in [0, 1]$.
- Set

  $$w_1 = V_1, \qquad w_2 = (1 - V_1)V_2, \qquad w_j = \prod_{i=1}^{j-1} (1 - V_i)\, V_j.$$

  This is the *stick-breaking process* on the V_i.
- Set

  $$F = \sum_{j=1}^{\infty} w_j\, \delta_{S_j},$$

  where δ_{S_j} is the Dirac delta at S_j. This F satisfies the Dirichlet property.

We break the stick $[0, 1]$ at V_1, then break the remaining piece $(V_1, 1)$ at a fraction V_2 along it, and so on.

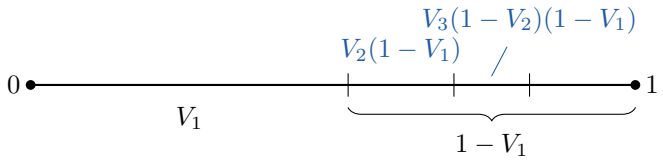

Figure 8: Stick breaking: the j-th piece of $[0, 1]$ has length w_j.

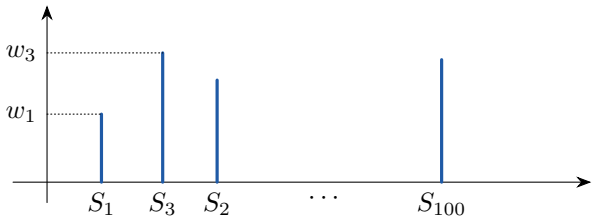

Figure 9: A draw $F = \sum_j w_j \delta_{S_j}$: a point mass of weight w_j at each S_j.

In practice, we truncate at N atoms:

- $S_1, \ldots, S_N \sim F_0$;
- $V_1, \ldots, V_N \sim \text{Beta}(1, \alpha)$;
- $w_N = 1 - \sum_{j=1}^{N-1} w_j$;
- $F = \sum_{j=1}^{N} w_j \, \delta_{S_j}$.

1.3 Posterior

With prior $F \sim \text{DP}(\alpha, F_0)$ and observations $X_1, \ldots, X_n \sim F$, the posterior is

$$\text{DP}(\alpha, \bar{F}_n), \qquad \bar{F}_n = \frac{n}{n+\alpha} F_n + \frac{\alpha}{n+\alpha} F_0,$$

where F_n is the empirical distribution and F_0 is the base distribution. Here $\bar{F}_n$ is a *mixture*: a convex combination of two distributions. As $n \to \infty$, the posterior goes back to becoming a noisy empirical CDF.

1.4 Chinese restaurant process

Problem: if n is small, drawing $F \sim \mathrm{DP}(\alpha, F_0)$ and then $X_1, \ldots, X_n \sim F$ is expensive. Can we just sample X_1 directly? Yes, with the *Chinese restaurant process*:

$$X_1 \sim F_0 \qquad \text{(like drawing from the base distribution)},$$
$$X_2 \mid X_1 \sim \begin{cases} F_0 & \text{with probability } \frac{\alpha}{\alpha+1}, \\ X_1 & \text{with probability } \frac{1}{\alpha+1}, \end{cases}$$
$$X_3 \mid X_1, X_2 \sim \begin{cases} F_0 & \text{with probability } \frac{\alpha}{\alpha+2}, \\ F_2 & \text{with probability } \frac{2}{\alpha+2}, \end{cases}$$

where F_2 is the empirical distribution of the first 2 draws.

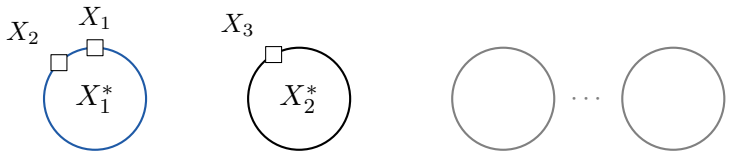

Figure 10: The restaurant after three draws: X_1 and X_2 share the first table (value X_1^*), X_3 sits at a second table (value X_2^*), and further tables are empty.

1.5 Dirichlet process mixture models

A draw from a Dirichlet process is a collection of spikes.

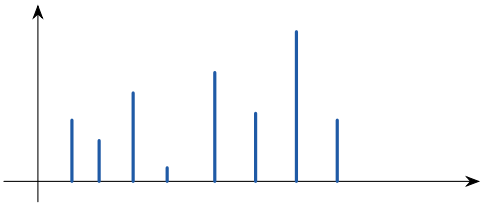

Figure 11: A draw from a Dirichlet process: a discrete distribution made of point masses.

A *mixture model* has density

$$f(x) = \sum_{j=1}^{k} w_j \, \phi(x; \theta_j),$$

a mixture of densities. In the Dirichlet process,

$$f = \sum_{j=1}^{\infty} w_j \, \delta_{S_j}$$

is an infinite mixture model of delta distributions. The delta distributions are the problem. Instead, take

$$f(x) = \sum_{j=1}^{\infty} w_j \, g(x; S_j),$$

where

- $S_1, S_2, \cdots \sim F_0$;
- $w_1, w_2, \cdots \sim \text{SBP}(\alpha)$, the stick-breaking process;
- $f(x) = \sum_{j=1}^{\infty} w_j \, g(x; S_j)$.

Instead of treating $S_j \sim F_0$ as values, we treat them as parameters. This is the *Dirichlet process mixture model*:

$$f \sim \text{DPM}(\alpha, F_0, g), \qquad X_1, \ldots, X_n \sim f.$$

But sadly, the posterior of a DPM is not a DPM.

Why care about densities? Distributions with densities are better behaved. For example, in the discrete case, labels could change if one pixel changes.

2 Gaussian Processes

Consider regression, with

$$\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}.$$

We want a distribution over such functions. For a random $f : \mathcal{X} \to \mathbb{R}$ and a fixed input x, the value $f(x)$ is random, since f is random. The collection of random variables $\{f(x)\}_{x \in \mathcal{X}}$ is a *stochastic process*. One of the simplest such processes is the *Gaussian process*.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 4 — January 26, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Point Masses

Consider a random distribution of the form

$$F = \sum_{j=1}^{\infty} w_j \, \delta_{s_j},$$

a weighted sum of point masses δ_{s_j}.

The point mass δ_0 is the limit of increasingly narrow uniform densities. Start from the uniform density on $[-1, 1]$, and shrink the interval while keeping the total mass equal to 1. At step n the density has height 2^n on an interval of width 2^{-n}. Taking $n \to \infty$ gives the delta distribution δ_0.

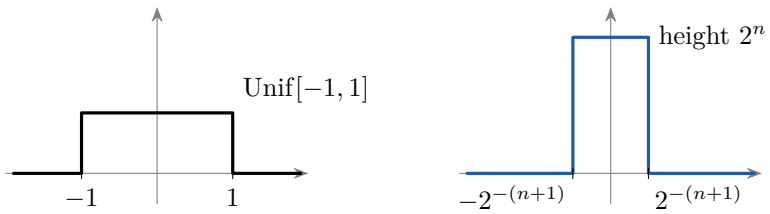

Figure 12: Uniform densities of shrinking width and growing height, each with total mass 1; in the limit they give δ_0.

2 Gaussian Processes

Consider a class of functions $\{f : \mathcal{X} \to \mathbb{R}\}$, and the collection of values $\{f(x)\}_{x \in \mathcal{X}}$. Here $f(x)$ is random because f comes from a distribution over functions. The input x is not random.

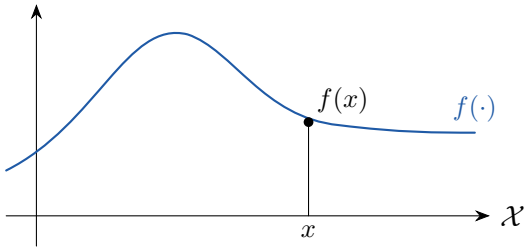

Figure 13: A random function f drawn from a distribution over functions, and its value $f(x)$ at a fixed input x.

Definition (Gaussian process)**.** We write $f \sim \mathrm{GP}(\mu(\cdot), k)$, read "$f$ is a *Gaussian process* with mean function μ and kernel k", where $\mu : \mathcal{X} \to \mathbb{R}$ and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, if for all n and all $(x_1, \ldots, x_n) \in \mathcal{X}^n$,

$$\big(f(x_1), f(x_2), \ldots, f(x_n)\big)^T \sim N\left(\begin{pmatrix}\mu(x_1)\\ \vdots \\ \mu(x_n)\end{pmatrix}, \Sigma\right), \qquad \Sigma_{ij} = k(x_i, x_j).$$

The function k is the *kernel function.*

2.1 Gaussian process regression

We are given data $\{(x_i, y_i)\}_{i=1}^n$. Suppose our model is

$$y_i = f(x_i) + \varepsilon_i, \qquad \varepsilon_i \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2),$$

with the prior $f \sim \mathrm{GP}(0, k)$. That is, f is a random function specified by the GP. Then

$$(y_1, \ldots, y_n) \sim N(0, K_n + \sigma^2 I),$$

where $K_n \in \mathbb{R}^{n \times n}$ has entries $(K_n)_{ij} = k(x_i, x_j)$.

For a new point,

$$y_{n+1} = f(x_{n+1}) + \varepsilon_{n+1}, \qquad y_{n+1} \sim N\big(0, k(x_{n+1}, x_{n+1}) + \sigma^2\big).$$

The posterior distribution of y_{n+1} given $D = \{(x_i, y_i)\}_{i=1}^n$ is also Gaussian; the posterior over f is again a GP. To compute it, note that

$$(y_1, \ldots, y_{n+1}) \sim N\big(0, K_{n+1} + \sigma^2 I\big),$$

where K_{n+1} is the $(n+1) \times (n+1)$ kernel matrix. Let

$$\mathbf{k} = \big(k(x_{n+1}, x_1), \ldots, k(x_{n+1}, x_n)\big) \in \mathbb{R}^n, \qquad y = (y_1, \ldots, y_n) \in \mathbb{R}^n.$$

Then

$$y_{n+1} \mid D \sim N\Big(\underbrace{\mathbf{k}^T (K_n + \sigma^2 I)^{-1}}_{\text{linear combination}} y,\ k(x_{n+1}, x_{n+1}) + \sigma^2 - \mathbf{k}^T (K_n + \sigma^2 I)^{-1} \mathbf{k}\Big).$$

In particular,

$$\mathbb{E}[y_{n+1} \mid D] = \sum_{i,i'=1}^n k(x_{n+1}, x_i)\ \underbrace{(K_n + \sigma^2 I)^{-1}_{ii'}}_{\text{for normalization}}\ y_{i'}.$$

Think of it as a weighted combination of the observed $y_{i'}$.

People are wary of non-parametric Bayes because it is not very scalable, but a lot of people are working on scaling these methods up.

3 Consistency of Non-Parametric Bayes

Let $f \sim \Pi$ be drawn from the prior, and $X_1, \ldots, X_n \sim f$, so $(X_1, \ldots, X_n, f)$ has a joint density. Let $A \subseteq \mathcal{F}$, where $\mathcal{F}$ is the set of all densities, and let Π_n be the posterior after n samples. We would like

$$\lim_{n\to\infty} \Pi_n(A) = \mathbb{I}[f^* \in A],$$

where $\mathbb{I}$ is the indicator function and f^* is the true density. Consider the pair (X^∞, f) of infinitely many samples together with f, and let μ be its joint distribution.

Theorem 2 (Doob)**.**

$$\mu\Big[\lim_{n\to\infty} \Pi_n(A) = \mathbb{I}[f \in A]\Big] = 1.$$

This means that non-parametric Bayes can be used. The problem is that the result holds with respect to its own prior.

Example 5. Let $X_1, \ldots, X_n \sim N(\theta, 1)$, with the prior $\theta \sim \delta_0 \equiv \Pi$. Then the posterior is $\Pi_n = \delta_0$, whatever the data.

Doob's theorem is only about the f that can be drawn from the prior with non-zero probability mass. In essence: if you have a bad prior, you have a bad posterior, and Bayesian inference will not help you. This is the "looking for your keys under the lamppost" analogy.

4 Frequentist Non-Parametric Methods

A *parametric model* has a finite-dimensional class of models. A *non-parametric* model allows the model class to be as large as it needs to be. (Similarly, in non-parametric Bayes our prior was over all possible distributions.)

4.1 Density estimation

The simplest approach is rote learning, that is, memorization. Given $X_1, \ldots, X_n \in \mathcal{X}$, take the empirical distribution

$$\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i},$$

which makes zero assumptions. But this does not give a density, since each point gets mass $\frac{1}{n}$.

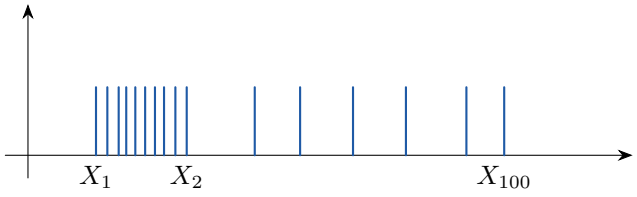

Figure 14: The empirical distribution puts a spike of mass $\frac{1}{n}$ at each sample, so it has no density.

Idea: can we use some kernel to smooth it out, to estimate the density? (Kernel methods.)

Smoothing is the same thing as non-parametric density estimation. It is difficult in high dimensions: what does "nearby" mean, for example for images?

Lecture 5 — January 31, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Non-Parametric Density Estimation

1.1 Recap

We have n points $X_i \overset{\text{i.i.d.}}{\sim} P$, $i \in [n]$, and the empirical distribution

$$\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i}.$$

A parametric approach assumes $p \in \mathcal{P} = \{p_\theta\}_{\theta\in\Theta}$ (for example, all Gaussians), and projects $\mathbb{P}_n$ onto this class.

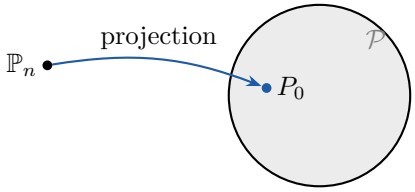

Figure 15: A parametric approach projects the empirical distribution $\mathbb{P}_n$ onto a point P_0 of the model class $\mathcal{P}$.

Goal: do some sort of smoothing, to distribute out the $\frac{1}{n}$'s. (This is what a density is!)

1.2 Smoothness and Hölder spaces

If x is close to x', we want the density to be smooth: the density at x and at x' should be close. That is, we want functions with bounded gradient.

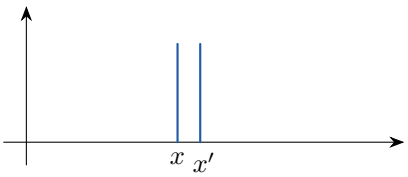

Figure 16: Nearby points x and x' should get similar density.

Definition (Hölder space)**.** *Hölder spaces* are spaces of smooth functions:

$$H(1, L) = \{p : |p(x) - p(x')| \leq L \|x - x'\|\},$$

and more generally

$$H(\beta, L) = \{p : |p^{(j)}(x) - p^{(j)}(x')| \leq L|x - x'|,\ j \in \{1, \ldots, \beta - 1\}\}.$$

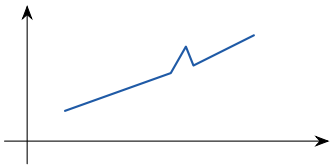

Figure 17: A function with bounded variation whose gradient changes a lot.

Hölder spaces are "classical" function spaces.

1.3 Measuring the quality of an estimate

The true density is $p \in \mathcal{P}$, and the estimate is $\hat{p} \in \mathcal{P}$. The estimate depends on the training data, so it is random. How do we determine how good our estimate $\hat{p}$ is? Classically, we use the squared loss

$$\ell(p, \hat{p}) = \int \big(p(x) - \hat{p}(x)\big)^2 \, dx,$$

with risk

$$\begin{aligned}
R(p, \hat{p}) = \mathbb{E}\ \ell(p, \hat{p}) &= \mathbb{E} \int_{\mathcal{X}} \big(p(x) - \hat{p}(x)\big)^2 \, dx \\
&= \mathbb{E} \int \Big(\big(\hat{p}(x) - \mathbb{E}\,\hat{p}(x)\big) + \big(\mathbb{E}\,\hat{p}(x) - p(x)\big) \Big)^2 \, dx \\
&= \mathbb{E} \int \big(\hat{p}(x) - \mathbb{E}\,\hat{p}(x)\big)^2 \, dx + \mathbb{E} \int \big(\mathbb{E}\,\hat{p}(x) - p(x)\big)^2 \, dx \\
&\quad + 2\,\mathbb{E} \int \big(\hat{p}(x) - \mathbb{E}\,\hat{p}(x)\big) \underbrace{\big(\mathbb{E}\,\hat{p}(x) - p(x)\big)}_{\text{constant (not random)}} \, dx \\
&= \int \sigma_n^2(x) \, dx + \int b_n^2(x) \, dx.
\end{aligned}$$

The cross term is 0: exchanging $\mathbb{E}$ and the integral, the factor $\mathbb{E}\,\hat{p}(x) - p(x)$ is a constant, and $\mathbb{E}\big[\hat{p}(x) - \mathbb{E}\,\hat{p}(x)\big] = 0$.

The first term is the *variance term*:

$$\mathbb{E} \int \big(\hat{p}(x) - \mathbb{E}\,\hat{p}(x)\big)^2 \, dx = \int_{\mathcal{X}} \mathbb{E}\big(\hat{p}(x) - \mathbb{E}\,\hat{p}(x)\big)^2 \, dx = \int_{\mathcal{X}} \underbrace{\mathrm{Var}\big(\hat{p}(x)\big)}_{\sigma_n^2(x)} \, dx.$$

The second term is the *bias term*:

$$\int \underbrace{\big(\mathbb{E}\,\hat{p}(x) - p(x)\big)^2}_{\mathrm{bias}^2(\hat{p}(x))} \, dx = \int b_n^2(x) \, dx.$$

If your function space is very big, your bias goes down, but the variance grows larger because you have more parameters.

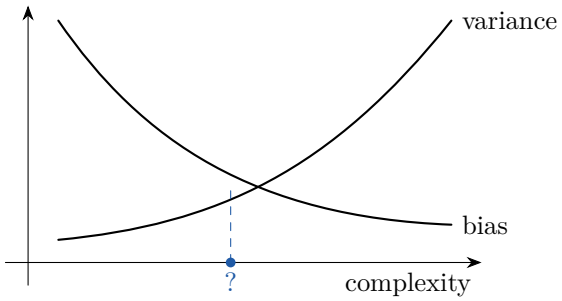

Figure 18: The bias–variance tradeoff: bias falls and variance grows with model complexity, and we want to minimize their sum.

We want to minimize the sum of the two. This is the *bias–variance tradeoff*!

But we study the mean squared error and its bias–variance tradeoff not because we think it is better, but because it is easier to study.

1.4 The L_1 distance

A better loss for densities is the L_1 distance,

$$\|p - \hat{p}\|_1 = \int |p(x) - \hat{p}(x)|\, dx.$$

We would like a distance d with

$$d(P_X, P_Z) \stackrel{?}{=} d(P_{f(X)}, P_{f(Z)})$$

for any invertible f. The L_1 distance satisfies this property.

The *total variation distance* is

$$d_{TV}(P, Q) = \sup_A |P(A) - Q(A)|.$$

Let $A = \{x : p(x) > q(x)\}$ and $B = \{x : q(x) > p(x)\}$. Then

$$\begin{aligned}\int |p(x) - q(x)|\, dx &= \int_A \big(p(x) - q(x)\big)\, dx + \int_B \big(q(x) - p(x)\big)\, dx \\ &= \big(P(A) - Q(A)\big) + \big(Q(B) - P(B)\big) \\ &= 2\big(P(A) - Q(A)\big).\end{aligned}$$

The last step holds because p and q both integrate to 1 and agree outside $A \cup B$, so $Q(B) - P(B) = P(A) - Q(A)$. The set A attains the supremum in d_{TV}, so $\|p - q\|_1 = 2\, d_{TV}(P, Q)$.

2 Histograms

General function spaces are hard to understand and too big. We can instead discretize the domain: $\mathcal{X} = \{1, \dots, k\}$. The set of densities is then a k-dimensional simplex.

Partition the space into regions $R_1, R_2, \dots, R_m$.

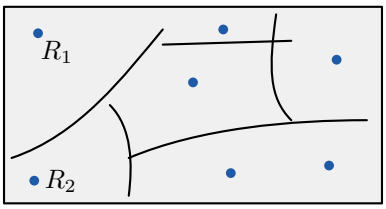

Figure 19: Partitioning the space into regions $R_1, R_2, \dots$; the dots are data points.

Then

$$p(x) \approx \sum_{j=1}^{m} p(x \mid X \in R_j)\, P(X \in R_j).$$

We estimate each factor:

$$\hat{p}(X \in R_j) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[X_i \in R_j],$$

$$\hat{p}(x \mid X \in R_j) \sim \text{Unif}(R_j) = \frac{1}{h^d} \mathbb{I}[x \in R_j],$$

where each R_j is a cube of side length h in $\mathbb{R}^d$, as in a histogram.

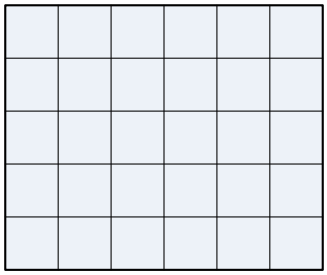

Figure 20: A histogram in $\mathbb{R}^2$ partitions the domain into a regular grid of cells.

Definition (Histogram estimator)**.** The *histogram estimator* is

$$\hat{p}(x) = \sum_{j=1}^{m} \frac{\hat{\theta}_j}{h^d} \mathbb{I}[x \in R_j],$$

where $\hat{\theta}_j = \hat{p}(X \in R_j)$ is the fraction of points in R_j.

It turns out that this is minimax-optimal:

$$\sup_{p \in H(1,L)} \underbrace{R_2(p, \hat{p})}_{\text{risk}} \lesssim n^{-\frac{2}{d+2}}, \qquad \inf_{\hat{p}} \sup_{p \in H(1,L)} R_2(p, \hat{p}) \gtrsim n^{-\frac{2}{d+2}}.$$

Therefore the histogram estimator is minimax-optimal. But this rate decays very slowly. For low risk we need

$$n^{-\frac{2}{d+2}} \le \varepsilon \iff n^{\frac{2}{d+2}} \ge \frac{1}{\varepsilon} \iff n \ge \left(\frac{1}{\varepsilon}\right)^{\frac{d+2}{2}},$$

so we need exponentially many samples (in d).

2.1 Adaptive partitioning

Neural networks partition the input space with lines; this is *adaptive partitioning*. On the other hand, the histogram is non-adaptive: it requires an exponential number of cells, possibly most of them empty.

Decision trees split the input space by axis-parallel lines.

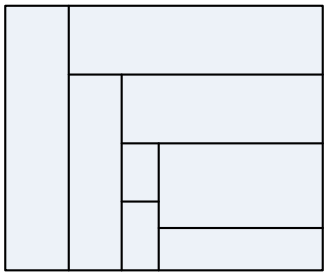

Figure 21: A decision tree recursively splits the input space with axis-parallel lines.

But adaptive partitioning is not well understood.

3 Curse of Dimensionality

In 1D, things are okay. But in higher dimensions, suppose

$$p(x_1, \dots, x_d) = \prod_{j=1}^{d} p_j(x_j)$$

(that is, assume the coordinates are independent). Then we can just estimate each p_j separately. For example, for pixels: take a 100×100 image; maybe we could estimate the density of each pixel separately. The problem is that this does not capture the dependencies. To capture them, we can use graphical models.

4 Orthogonal Basis Expansions

Another approach is brute-force calculation. Take

$$\mathcal{P} = \left\{p : [0,1] \to \mathbb{R},\ \int p^2(x)\,dx < \infty\right\} = L_2[0,1],$$

or more generally

$$L_2(\mathcal{X}) = \Big\{p : \int_{\mathcal{X}} p^2(x)\,dx < \infty\Big\}.$$

This space is infinite dimensional. We can express its elements in terms of a basis $\{\phi_j(x)\}_{j\in\mathbb{N}}$ (for example the Fourier basis, the cosine basis, ...):

$$p(x) = \sum_{j=1}^{\infty} \beta_j \phi_j(x).$$

The coefficients are

$$\beta_j = \int p(x)\phi_j(x)\,dx = \mathop{\mathbb{E}}_{x\sim p}[\phi_j(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \phi_j(X_i) \equiv \hat{\beta}_j.$$

How good the approximation is can be controlled by uniform deviation bounds. The formula for β_j holds because the basis is orthonormal:

$$\int \sum_{j'} \beta_{j'}\phi_{j'}(x)\phi_j(x)\,dx = \sum_{j'} \mathbb{I}[j' = j]\,\beta_{j'} = \beta_j.$$

This gives the estimators

$$\hat{p}(x) = \sum_{j=1}^{\infty} \hat{\beta}_j \phi_j(x), \qquad \hat{p}_k(x) = \sum_{j=1}^{k} \hat{\beta}_j \phi_j(x).$$

4.1 Sobolev spaces

Consider the class

$$\Big\{p : \sum_{j=1}^{\infty} \beta_j^2 a_j < \infty\Big\}, \qquad a_j \to \infty,$$

with a_j growing like a power of j. This is a *Sobolev* space, which is another way of stating smoothness.

Lecture 6 — February 2, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Non-Parametric Density Estimation

Idea: take an infinite-dimensional class, and find a density in this class.

1.1 Kernel density estimation

The kernel method aims to create a density by smoothing out the points.

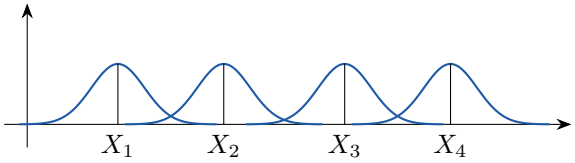

Figure 22: The kernel method places a smooth bump at each data point X_i.

Instead of the empirical distribution

$$\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i},$$

we use the *kernel density estimator*

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\frac{1}{h^d}}_{\text{scaling}} K\Big(\frac{x - X_i}{h}\Big),$$

where d is the dimension and h is a *scaling parameter*. The kernel $K : \mathbb{R} \to \mathbb{R}$ satisfies

$$\begin{aligned} K(t) &\geq 0, \\ \int t\, K(t)\, dt &= 0 \qquad \text{(mean zero)}, \\ \int t^2 K(t)\, dt &< \infty, \\ \int K(t)\, dt &= 1. \end{aligned}$$

Example 6 (Gaussian kernel)**.**

$$K\Big(\frac{t}{h}\Big) = \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{t^2}{2h^2}\Big).$$

1.2 Rates over Hölder classes

Recall the Hölder class

$$H(\beta, L) = \{p : |p^{(j)}(x) - p^{(j)}(x')| \leq L|x - x'|,\ j \in \{0, \ldots, \beta - 1\}\}.$$

These are functions that are smooth and whose derivatives do not change too fast. (If we do not have this assumption, we cannot generalize.)

With $h = n^{-\frac{1}{4+d}}$, the kernel density estimator achieves

$$\sup_{p \in H(2,L)} R(\hat{p}_n, p) \lesssim n^{-\frac{4}{4+d}}.$$

We want h to decay to 0 as we get more samples. This is minimax-optimal:

$$\inf_{\hat{p}} \sup_{p \in H(2,L)} R(\hat{p}_n, p) \gtrsim n^{-\frac{4}{4+d}}.$$

Recall that

$$R(\hat{p}, p) = \mathbb{E} \int (\hat{p}(x) - p(x))^2 \, dx.$$

If we know our distribution is very smooth and has smooth β derivatives, then

$$\inf_{\hat{p}} \sup_{p \in H(\beta,L)} R(\hat{p}, p) \asymp n^{-\frac{2\beta}{2\beta+d}}.$$

1.3 Higher-order kernels

The rate above is achieved by a higher-order kernel.

Definition (β-order kernel)**.** A *β-order kernel* K satisfies

$$\begin{aligned} \int K(t)\, dt &= 1, \\ \int t^j K(t)\, dt &= 0, \qquad j \in \{1, \ldots, \beta - 1\}, \\ \int |t|^\beta K(t)\, dt &< \infty. \end{aligned}$$

That is, the moments up to the $(\beta - 1)$-th are 0.

Example 7 (4th-order kernel)**.**

$$K(t) = \frac{3}{8}(3 - 5t^2)\, \mathbb{I}(|t| \leq 1).$$

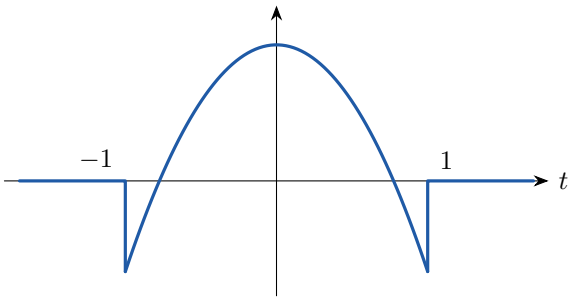

Figure 23: The 4th-order kernel $K(t) = \frac{3}{8}(3 - 5t^2)\,\mathbb{I}(|t| \leq 1)$ takes negative values near ± 1.

But we do not expect most functions to be very smooth. If β is too large, then we restrict our class of functions a lot.

2 Choosing the Bandwidth

The question is how to pick h. This choice is usually more important than the kernel that you choose.

2.1 Histograms and the partition estimator

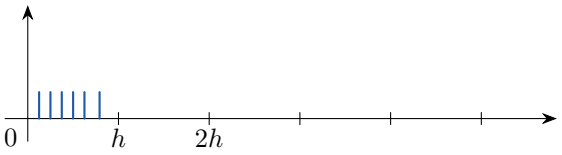

Figure 24: A one-dimensional histogram with bins of width h starting at 0; the data points shown fall in the first bin.

In one dimension, the histogram estimator is

$$\hat{p}(x) = \sum_{j=1}^{N} \hat{p}_j \, \frac{1}{h} \, \mathbb{I}(x \in R_j).$$

The *partition estimator* is

$$\hat{p}_h(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbb{I}[x \in [X_i - h, X_i + h]]}{2h}.$$

2.2 Data splitting

Problem: we have a class of estimators $\hat{p}_h$, and want to pick the best h. Split the $2n$ data points into a training half and a validation half.

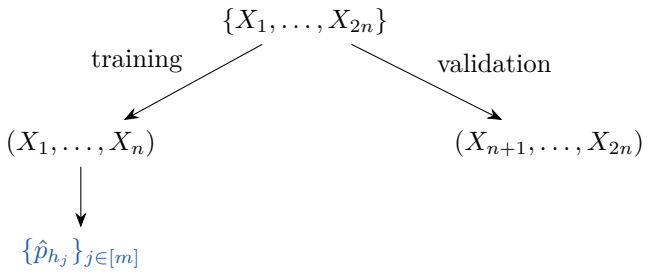

Figure 25: Data splitting: the training half produces the candidate estimators $\hat{p}_{h_j}$, and the validation half is used to choose among them.

One criterion is the validation log-likelihood,

$$\frac{1}{n} \sum_{i=n+1}^{2n} \log \hat{p}_h(X_i).$$

2.3 L_2 loss

Consider the loss

$$L(\hat{p}_h, p) = \int \left(\hat{p}_h(x) - p(x)\right)^2 dx.$$

We choose the L_2 distance because it is easy to compute minimax rates with it.

The KL divergence is not good, as it is bad at dealing with tails that do not match (the KL divergence will blow up):

$$\int p(x) \log \underbrace{\frac{p(x)}{q(x)}}_{\text{kaboom}} dx.$$

Expanding the L_2 loss,

$$L(\hat{p}_h, p) = \int \hat{p}_h(x)^2 \, dx - 2 \underbrace{\int \hat{p}_h(x)\, p(x)\, dx}_{\mathbb{E}_{x \sim p}[\hat{p}_h(x)]} + \underbrace{\int p^2(x)\, dx}_{\text{does not depend on the estimator}}.$$

The middle term can be approximated with Monte Carlo sampling. We are minimizing over h, so the last term can be dropped. We like expectations because we can optimize them using Monte Carlo samples.

2.4 L_1 loss

The error of h under the L_1 loss is

$$\int |\hat{p}_h(x) - p(x)| \, dx,$$

which is twice

$$\Delta(h) = \sup_A |\hat{P}_h(A) - P(A)|.$$

Replacing P with the validation points gives

$$\Delta_n(h) = \sup_A \Big| \hat{P}_h(A) - \frac{1}{n} \sum_{i=n+1}^{2n} \mathbb{I}[X_i \in A] \Big|,$$

but the supremum over all sets A is hard to optimize over.

So suppose we restrict to a class $\mathcal{A}$ with VC dimension r. Then

$$\Delta_n(h) = \sup_{A \in \mathcal{A}} |\hat{P}_h(A) - P_n(A)|,$$

but $\mathcal{A}$ may still be very large. We have

$$\Delta_n(h) \leq \Delta(h) + \sqrt{\frac{r}{n}}.$$

Definition (Yatracos sets)**.** Given candidate bandwidths $\{h_1, \ldots, h_m\}$, let

$$A(i,j) = \big\{x : \hat{p}_{h_i}(x) > \hat{p}_{h_j}(x)\big\},$$

and take the class of *Yatracos sets*

$$\mathcal{A} = \{A(i,j) : i, j \in \{1, \ldots, m\},\ i \neq j\}.$$

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 7 — February 7, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Non-Parametric versus Parametric Rates

Compare the risk of non-parametric and parametric estimators as a function of the number of samples n.

- **Non-parametric rate.** The (minimax) risk scales as

$$n^{-\frac{c_1}{c_2+d}}.$$

- **Parametric rate.** The risk scales as d/n^a. To get risk at most ε we need

$$\frac{d}{n^a} \leq \varepsilon \quad \iff \quad n \geq \left(\frac{d}{\varepsilon}\right)^{1/a},$$

  which is polynomial in d and $1/\varepsilon$.

For the non-parametric rate, take for example a risk of $n^{-1/d}$. Then

$$n^{-1/d} \leq \varepsilon \quad \iff \quad n^{1/d} \geq \frac{1}{\varepsilon} \quad \iff \quad n \geq \left(\frac{1}{\varepsilon}\right)^d,$$

so we need exponentially much data in d!

What this says is that the space of functions is too large for non-parametric models. The unfortunate message is that you need to make assumptions.

1.1 Recap

We have seen two methods:

1. **Partitioning-based** methods: fixed partitions, adaptive partitions, decision trees. Neural networks also fall in this family.
2. **Kernel density estimation.** This is largely based on smoothing, and is not going to be amazing unless you have a Mercer kernel.

The *inductive bias* of a method is its model assumptions.

2 Regression

We want a function

$$m : \mathcal{X} \to \mathcal{Y} = \mathbb{R},$$

where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. The pair $(X, Y) \sim P$ follows the ground-truth distribution.

Definition (Regression function)**.** The *regression function* is

$$m(x) = \mathbb{E}[Y \mid X = x],$$

the mean of $P(Y \mid X = x)$, the random response given an input. It is optimal for the squared loss.

We can write

$$Y = m(X) + \varepsilon,$$

where $\varepsilon = Y - m(X)$ is the *residual*. It has mean zero:

$$\begin{aligned}\mathbb{E}[\varepsilon] &= \mathbb{E}\big[\mathbb{E}[\varepsilon \mid X]\big] = \mathbb{E}\big[\mathbb{E}[Y - m(X) \mid X]\big] \\ &= \mathbb{E}\big[\mathbb{E}[Y \mid X] - m(X)\big] = 0.\end{aligned}$$

The loss and the risk of an estimator $\hat{m}$ are

$$\begin{aligned}L(\hat{m}, m) &= \int \big(\hat{m}(x) - m(x)\big)^2 p(x)\, dx, \\ R(\hat{m}, m) &= \mathbb{E} \int \big(\hat{m}(x) - m(x)\big)^2 p(x)\, dx.\end{aligned}$$

2.1 Partition estimator

Partition the input space into regions $R_1, \ldots, R_k$.

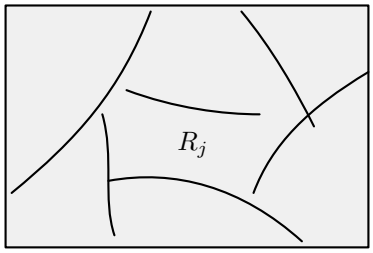

Figure 26: A partition of the input space into regions.

The *partition estimator* is

$$\hat{m}(x) = \sum_{j=1}^{k} c_j \, \mathbb{I}[x \in R_j], \qquad c_j = \frac{\sum_{i=1}^{n} \mathbb{I}[X_i \in R_j]\, Y_i}{\sum_{i=1}^{n} \mathbb{I}[X_i \in R_j]},$$

so c_j is the average response of the training points that fall in R_j.

2.2 Splines

Splines are piecewise polynomials. (The name comes from graphics.)

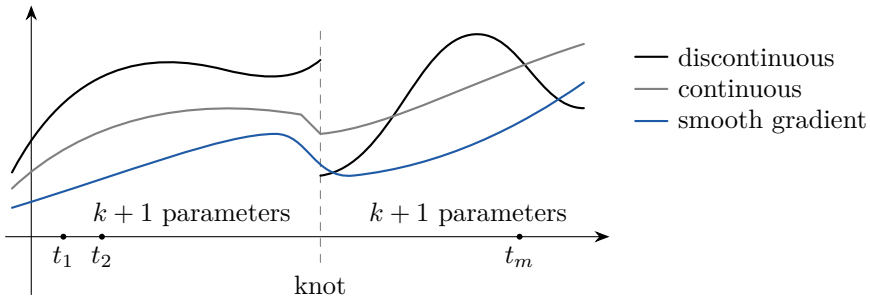

Figure 27: Piecewise polynomials meeting at a knot, with increasing smoothness at the knot.

- The black fit is discontinuous at the knot. This is undesirable; we want it to be continuous.
- The gray fit is continuous but not smooth: its gradients are not continuous.
- The blue fit has smooth gradients, but what about higher derivatives?

We want each piece to be a degree-k polynomial, with the derivatives up to order $k-1$ matching at the knots.

Definition (Spline)**.** With m knots $t_1 < \cdots < t_m$ there are $m+1$ regions

$$(-\infty, t_1],\ (t_1, t_2],\ \ldots,\ (t_m, \infty).$$

A degree-k spline is a separate degree-k polynomial in each region, such that the derivatives up to order $k-1$ are continuous at the knots.

This makes the function very smooth: it is hard to tell that it is not the same polynomial!

Number of parameters. In total there are

$$\begin{aligned} &(k+1)(m+1) && \text{(parameters)} \\ -\ &km && \text{(constraints from matching derivatives)} \\ =\ &k+m+1 = O(k+m). \end{aligned}$$

Truncated power basis

A basis for degree-k splines with knots $t_1, \ldots, t_m$ is

$$\begin{aligned} B_1(x) = 1,\ \ldots,\ B_{k+1}(x) = x^k, \\ B_{k+1+j}(x) = (x - t_j)_+^k, \qquad j = 1, \ldots, m, \end{aligned}$$

where $(z)_+ = \max(z, 0)$. This gives $k+1+m$ basis functions. Another choice of basis is the *B-spline basis*.

2.3 Smoothing splines

Given data $\{(x_i, y_i)\}_{i=1}^n$, place the knots at the data points: $t_i = x_i$ for $i = 1, \ldots, n$.

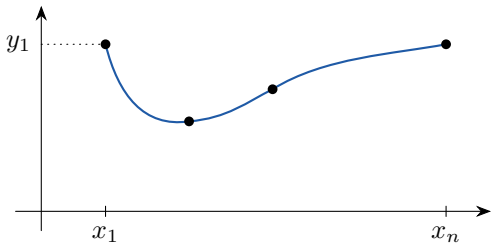

Figure 28: A function passing through every data point (x_i, y_i).

We want $\hat{m}$ with small L_2 error with respect to y:

$$\min_m \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - m(x_i))^2 \right\}.$$

But there are arbitrarily many functions that achieve this, so we need to constrain it:

$$\min_m \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - m(x_i))^2 + \lambda \int (m''(x))^2 dx \right\}.$$

The solution is a degree-3 *smoothing spline*.

2.4 Kernel regression

Take a kernel k with

$$\int k(t)\, dt = 1, \qquad \int t\, k(t)\, dt = 0, \qquad \int t^2 k(t)\, dt < \infty,$$

for example

$$k(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right).$$

The kernel regression estimator is

$$\hat{m}(x) = \frac{\sum_{i=1}^n k\left(\frac{x - x_i}{h}\right) y_i}{\sum_{i=1}^n k\left(\frac{x - x_i}{h}\right)} = \frac{\sum_{i=1}^n w_i(x)\, y_i}{\sum_{i=1}^n w_i(x)},$$

a weighted average of all the training responses. We can think of it as a "soft" nearest-neighbor method.

k-nearest neighbors

Let $N_k(x)$ be the k nearest neighbors of x. The k-*NN* estimator is

$$\hat{m}(x) = \frac{1}{k} \sum_{i=1}^n \mathbb{I}[x_i \in N_k(x)]\, y_i = \frac{\sum_{i=1}^n \mathbb{I}[x_i \in N_k(x)]\, y_i}{\sum_{i=1}^n \mathbb{I}[x_i \in N_k(x)]}.$$

So k-NN is kernel regression with a box-shaped weight, while a smooth kernel gives smoothed weights.

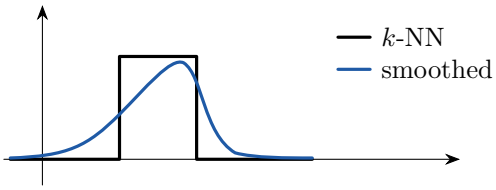

Figure 29: The box-shaped weight of k-NN and a smoothed weight.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 8 — February 9, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Non-Parametric Regression

Non-parametric regression is what is done in modern ML methods.

The kernel regression estimator is

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} k_h\left(\frac{x-x_i}{h}\right) y_i}{\sum_{i=1}^{n} k_h\left(\frac{x-x_i}{h}\right)},$$

with risk

$$R(\hat{m}, m) \asymp n^{-\frac{4}{4+d}},$$

which is minimax for smooth functions. In low dimensions this is OK; in high dimensions it is bad.

1.1 Universal consistency

Suppose m^* is the true regression function, so that

$$Y = m^*(X) + \varepsilon.$$

Definition (Universal consistency)**.** An estimator $\hat{m}$ is *universally consistent* if

$$\mathbb{E}\,\|\hat{m} - m^*\| \xrightarrow{n\to\infty} 0,$$

under the only (minimal) assumption that $\mathbb{E}[Y^2] < \infty$.

Suppose we use the k-NN kernel, with k-NN parameter k_n. If

$$k_n \xrightarrow{n\to\infty} \infty \qquad \text{and} \qquad \frac{k_n}{n} \to 0,$$

then the k-NN estimator is universally consistent.

1.2 Kernels as soft partitions

Kernel methods seem very different from partition-based methods, which take the mean of the responses in your region. But you could also motivate kernel methods from the lens of softening the notion of a partition.

A partition gives a piecewise simple model, but one that is globally non-simple.

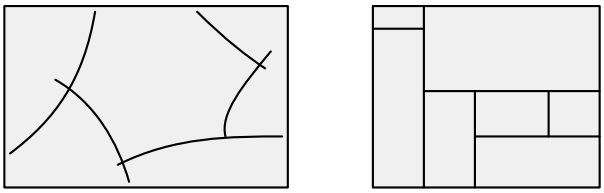

Figure 30: Two partitions of the input space: an irregular one, and an axis-aligned one.

Kernels provide a notion of locality: given any test point $x \in \mathcal{X}$, they can adaptively figure out what your locality is.

Given any test point $x \in \mathcal{X}$, consider

$$\min_c \sum_{i=1}^n w_i(x)\,(Y_i - c)^2,$$

where c is a constant given the test point. Different methods use different weights:

- partition-based: $w_i(x) = \mathbb{I}[R(x) = R(x_i)]$, where $R(x)$ is the region containing x;
- kernel: $w_i(x) = k\left(\frac{x - x_i}{h}\right)$.

Setting the derivative with respect to c to zero,

$$\sum_{i=1}^n w_i(x)\,(\hat{c} - Y_i) = 0 \quad \Longrightarrow \quad \hat{c} = \frac{\sum_{i=1}^n w_i(x)\,Y_i}{\sum_{i=1}^n w_i(x)}.$$

1.3 Local polynomial regression

We can also look at OLS from this weighted perspective, where $w_i(x)$ is the weight given by the test point x:

$$\sum_{i=1}^n w_i(x)\,\left(Y_i - \beta_0 - \beta_1^T x_i\right)^2.$$

More generally, *local polynomial regression* fits a polynomial:

$$\sum_{i=1}^n w_i(x)\Big(Y_i - \sum_{j=0}^{\gamma} \beta_j x_i^j\Big)^2,$$

where the polynomial $\sum_{j=0}^{\gamma} \beta_j x_i^j$ is what you are trying to fit.

If the true function satisfies $m \in H(\beta, L)$, then local polynomial regression of degree $\beta - 1$ (or an order-β kernel) suffices, and

$$R(\hat{m}, m) = n^{-\frac{2\beta}{2\beta + d}}.$$

The estimate is

$$\hat{m}(x) = \sum_{j=0}^{\gamma} \hat{\beta}_j(x)\,x^j,$$

where the polynomial coefficients depend on the test point. The choice of β_j falls out from $w(x)$:

$$\beta(x) = \left(X^T W(x) X\right)^{-1} \left(X^T W(x)\, y\right), \qquad \beta(x) = \{\beta_j(x)\}_j,$$

where X is the design matrix with rows $(1, x_i, \ldots, x_i^\gamma)$ and $W(x) = \text{diag}(w_1(x), \ldots, w_n(x))$.

2 Hilbert Spaces

In non-parametric regression, you want to make as few assumptions about your function class as possible. A natural choice is

$$L_2(\mathcal{X}) = \left\{ m : \mathcal{X} \to \mathbb{R} \;\middle|\; \int m^2(x)\, dx < \infty \right\},$$

with inner product

$$\langle f, g \rangle = \int f(x)\, g(x)\, dx,$$

which expresses closeness between points, and norm

$$\|f\|^2 = \langle f, f \rangle .$$

2.1 Evaluation is not continuous in L_2

Consider the non-smooth function

$$m_n(x) = \sqrt{n}\, \mathbb{I}\left[x < \frac{1}{n^2}\right].$$

Then $m_n(0) = \sqrt{n}$, but

$$\|m_n - 0\|^2 = \int m_n^2(x)\, dx = n \cdot \frac{1}{n^2} = \frac{1}{n} \xrightarrow{n\to\infty} 0.$$

So $m_n \to m_\infty = 0$, while $m_n(0) = \sqrt{n} \to \infty$. Thus

$$m_n(0) \not\to m_\infty(0).$$

Consider functionals $R : L_2(\mathcal{X}) \to \mathbb{R}$. The *evaluation functional*

$$\delta_x : L_2(\mathcal{X}) \to \mathbb{R}, \qquad \delta_x(m) = m(x),$$

evaluates a function at a point. Our data are point evaluations of the true function, $\{(X_i, \delta_{X_i}(m^*))\}$ with $\delta_{X_i}(m^*) = m^*(X_i)$. But the example above shows that δ_x is not even continuous on $L_2(\mathcal{X})$.

2.2 Reproducing kernel Hilbert spaces

Instead, work in a Hilbert space $\mathcal{H}$ where

$$\delta_x(m) = \langle k_x, m \rangle \qquad \text{for some } k_x \in \mathcal{H}.$$

Then evaluation is continuous: if $m_n \to m$, then

$$\begin{aligned} \langle k_x, m_n \rangle - \langle k_x, m \rangle &= \langle k_x, m_n - m \rangle \\ &\leq \|k_x\| \, \|m_n - m\| \to 0 \end{aligned}$$

by Cauchy–Schwarz. Moreover,

$$\langle k_x, k_y \rangle = k_y(x) = k_x(y) = k(x, y).$$

Definition (Reproducing kernel property)**.**

$$\langle k_x, m \rangle = m(x).$$

The kernel gives a representation of the function evaluation.

2.3 Mercer kernels

Definition (Mercer kernel)**.** A *Mercer kernel* is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is

- symmetric: $k(x, y) = k(y, x)$;
- positive semi-definite:
$$\int_{\mathcal{X}} \int_{\mathcal{X}} f(x)\, k(x, y)\, f(y)\, dx\, dy \geq 0.$$

If $\mathcal{X}$ were finite, then $k(x, y)$ would be a matrix K and $f(x)$, $f(y)$ vectors, and the condition would read $f^T K f \geq 0$.

Given a Mercer kernel, define

$$\mathcal{H}_0 = \Big\{ f \;\Big|\; f(x) = \sum_{j=1}^{N} \alpha_j \, k_{x_j}(x) \text{ for some } N,\ (\alpha_1, \ldots, \alpha_N),\ \{x_j\}_{j=1}^{N} \Big\}.$$

For $f = \sum_j \alpha_j k_{x_j}$ and $g = \sum_j \beta_j k_{y_j}$ in $\mathcal{H}_0$, define

$$\langle f, g \rangle = \sum_{j,j'} \alpha_j \beta_{j'} \, k(x_j, y_{j'}).$$

The inner product is always unique, even though the representation of each of f, g may not be unique.

One can show that for this inner product, the reproducing kernel property holds: for $f = \sum_j \alpha_j k_{x_j}$,

$$\langle f, k_x \rangle = \sum_j \alpha_j \, k(x_j, x) = f(x).$$

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 9 — February 14, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Reproducing Kernel Hilbert Spaces

1.1 Recap

The *evaluation functional* is

$$\delta_x : \mathcal{L}_2(\mathcal{X}) \to \mathbb{R}, \qquad \delta_x(f) = f(x).$$

On $\mathcal{L}_2(\mathcal{X})$, δ_x is *not continuous*! To make it continuous, we ask for

$$\delta_x(f) = \langle k_x, f \rangle ,$$

where k_x is the *representer* of δ_x, and

$$k(x, y) = k_x(y).$$

An RKHS is a big class of spaces where the kernels are the representers.

1.2 Representer theorem

Given data $\{(x_i, y_i)\}_{i=1}^n$, suppose we want to find

$$\min_{f \in \mathcal{H}_k} \left[\frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i) + \lambda \|f\|_k^2 \right].$$

Theorem 3 (Representer theorem)**.** *The minimizer has the form*

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \alpha_i \, k(x, x_i).$$

This is parametric, based on $(\alpha_1, \ldots, \alpha_n)$. We love RKHS because it converts a non-parametric estimation problem into a parametric one.

For

$$f(x) = \sum_{i=1}^n \alpha_i \, k(x, x_i)$$

we have

$$\|f\|_k^2 = \sum_{i,i'=1}^n \alpha_i \alpha_{i'} \, k(x_i, x_{i'}) = \alpha^T K \alpha,$$

where

$$\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}, \qquad K = \begin{bmatrix} k(x_1, x_1) & \cdots & \\ & k(x_i, x_j) & \\ & & \ddots \end{bmatrix}.$$

1.3 Comparison with kernel regression

Kernel regression gives

$$\hat{f}(x) = \sum_{i=1}^{n} w_i(x)\, y_i, \qquad w_i(x) \propto k\left(\frac{x - x_i}{h}\right),$$

a linear combination of the y_i.

For RKHS regression with squared loss, we solve

$$\min_{\alpha}\Big\{\|K\alpha - y\|^2 + \lambda\, \alpha^T K \alpha\Big\},$$

which gives

$$\alpha = (K + \lambda I)^{-1} y$$

and

$$\hat{f}(x) = k_x^T (K + \lambda I)^{-1} y = \sum_{i=1}^{n} w_i(x)\, y_i,$$

where $k_x = (k(x, x_1), \ldots, k(x, x_n))^T$. So this is again a linear combination of the y_i.

Issue: the estimator could end up fitting noise, and we get slapped by the $n^{-1/d}$ rates of non-parametric methods.

1.4 Mercer expansion and feature maps

Suppose

$$\sup_{x, x' \in \mathcal{X}} k(x, x') < \infty.$$

Then

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i\, \psi_i(x)\, \psi_i(x'),$$

where $\{\psi_i\}$ is an orthonormal basis for $\mathcal{L}_2(\mathcal{X})$. This extracts the most important basis functions.

Consider the feature map

$$x \mapsto \big\{\sqrt{\lambda_i}\, \psi_i(x)\big\}_{i \in \mathbb{N}} \in \ell_2.$$

It lies in ℓ_2 because

$$\sum_i \lambda_i\, \psi_i^2(x) = k(x, x) < \infty,$$

and

$$\Big\langle (\sqrt{\lambda_i}\, \psi_i(x)), (\sqrt{\lambda_i}\, \psi_i(x')) \Big\rangle = \sum_i \lambda_i\, \psi_i(x)\, \psi_i(x') = k(x, x').$$

This is the *kernel trick*. The appeal of RKHS is that you can actually work in these infinite-dimensional feature maps.

So we can represent infinite-dimensional maps ϕ: for $f = \sum_i \alpha_i k(\cdot, x_i)$,

$$f(x) = \sum_i \alpha_i\, k(x, x_i) = \sum_i \alpha_i\, \langle \phi(x), \phi(x_i) \rangle$$

$$= \Big\langle \sum_{i=1}^{n} \alpha_i\, \phi(x_i),\ \phi(x) \Big\rangle.$$

1.5 Which kernels give a useful RKHS?

What is a good kernel so that the RKHS is useful? With $\{\lambda_i, \psi_i\}_{i\in\mathbb{N}}$ as above, the RKHS norm can be written as

$$\|f\|_k^2 = \sum_{i=1}^{\infty} \frac{a_i^2}{\lambda_i} \qquad \text{for } f = \sum_{i=1}^{\infty} a_i\, \psi_i(x).$$

So a bounded RKHS norm describes an ellipsoid in $\mathcal{L}_2$. For the norm to be bounded, the a_i have to decay faster than the λ_i: your function's coefficients should decay fast.

In general, any $f = \sum_{i=1}^{\infty} a_i\psi_i(x) \in \mathcal{L}_2(\mathcal{X})$, and

$$\mathcal{H}_k = \Big\{ f = \sum_i a_i\psi_i \;\Big|\; \sum_i a_i^2 c_i < \infty \Big\}.$$

Example 8 (Sobolev space)**.** Take $\{\psi_i\}$ to be the Fourier basis and $c_i \asymp i^m$. Then the condition is

$$\sum_{i=1}^{\infty} a_i^2\, i^m < \infty.$$

Remark 1. RKHS regression uses a Mercer kernel (which must be PSD). Kernel regression uses a smoothing kernel.

2 Linear Smoothers

A surprising fact: all our estimators studied so far are of the form

$$\hat{f}(x) = \sum_{i=1}^{n} \ell_i(x)\, y_i, \qquad \sum_{i=1}^{n} \ell_i(x) = 1.$$

So this is a linear combination of the y_i's. In matrix form,

$$\begin{bmatrix} \hat{f}(x_1) \\ \vdots \\ \hat{f}(x_n) \end{bmatrix} = \underbrace{\begin{bmatrix} \ell_1(x_1) & \ell_2(x_1) & \cdots & \ell_n(x_1) \\ \vdots & & & \vdots \\ \ell_1(x_n) & \cdots & \cdots & \ell_n(x_n) \end{bmatrix}}_{L} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

It turns out that

$$\mathrm{tr}(L) = \sum_{i=1}^{n} L_{ii}$$

captures the complexity of the function class. It is the *effective degrees of freedom* of a non-parametric method.

2.1 Choosing the tuning parameter

The risk of the estimator with tuning parameter h is

$$R(h) = \mathbb{E}[(Y - \hat{f}_h(X))^2].$$

Leave-one-out (LOO) validation.

$$R_{\mathrm{LOO}}(h) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}_{h,-i}(x_i)\big)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - L_{ii}}\right)^2,$$

so essentially we scale each residual by $1 - L_{ii}$. Here

$$\hat{f}(x) = \sum_{i=1}^{n} \ell_i(x)\, y_i, \qquad \hat{f}_{-i}(x) = \frac{\sum_{i' \neq i} \ell_{i'}(x)\, y_{i'}}{\sum_{i' \neq i} \ell_{i'}(x)}.$$

Generalized cross-validation (GCV). Replacing each L_{ii} by the average $\frac{1}{n}\sum_{i=1}^{n} L_{ii} = \frac{1}{n}\operatorname{tr}(L)$ gives the approximation $R_{\mathrm{LOO}}(h) \approx R_{\mathrm{GCV}}(h)$, where

$$R_{\mathrm{GCV}}(h) = \frac{1}{n}\sum_{i=1}^{n}\frac{\big(y_i - \hat{f}(x_i)\big)^2}{\big(1 - \frac{1}{n}\operatorname{tr}(L)\big)^2} \approx \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}(x_i)\big)^2\left(1 + \frac{2\operatorname{tr}(L)}{n}\right),$$

using $(1-x)^{-2} \approx 1 + 2x$.

3 Wavelets

Write

$$f(x) = \sum_i a_i\, \psi_i(x).$$

We want the coefficients to be sparse: $\|a\|_1 < C$ or $\|a\|_0 < C$. This leads to

$$\min_a \left\{ \frac{1}{n}\sum_{i=1}^{n}\big(y_i - f_a(x_i)\big)^2 + \lambda \|a\|_1 \right\}.$$

Brains are not like Sobolev spaces, since those can have a lot of large activations.

4 Classification

We want $f : \mathcal{X} \to \mathcal{Y} \in \{1, \ldots, k\}$, with risk

$$R(f) = \mathbb{P}\big[f(X) \neq Y\big].$$

The minimizer

$$f^* = \arg\min_f R(f)$$

is the *Bayes classifier*:

$$f^*(x) = \arg\max_{i \in \{1,\ldots,k\}} \mathbb{P}[Y = i \mid X = x].$$

Estimating this directly is the *discriminative approach*. Compare with regression, where $R(f) = \mathbb{E}\big[(Y - f(X))^2\big]$ and $\arg\min_f R(f) = \mathbb{E}[Y \mid X]$.

By Bayes' rule,

$$f^*(x) = \arg\max_{i \in \{1,\dots,k\}} \mathbb{P}(Y = i \mid X = x) = \arg\max_{i \in \{1,\dots,k\}} \mathbb{P}(X = x \mid Y = i)\ \mathbb{P}(Y = i).$$

If you have access to non-parametric density estimation, you can estimate $\mathbb{P}(X \mid Y = i)$, and use a plug-in estimator for the categorical $\mathbb{P}(Y = i)$:

$$\hat{f}(x) = \arg\max_{i \in \{1,\dots,k\}} \hat{\mathbb{P}}(x \mid Y = i)\, \hat{\mathbb{P}}(Y = i).$$

This is the *generative approach*. It is actually easy for non-parametric estimators, and easy for anything that involves smoothness.

A lot of methods that seem to be using a discriminative approach actually use generative approaches, and leverage the geometry of the problem.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 10 — February 16, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Non-Parametric Classification

Let $Y \in \{0, 1\}$. Then

$$\mathbb{E}[Y \mid X] = \mathbb{P}[Y = 1 \mid X],$$

and the Bayes classifier is

$$f^*(x) = \mathbb{I}\Big[\underbrace{\mathbb{P}[Y = 1 \mid X = x]}_{m(x)} > \tfrac{1}{2}\Big].$$

A neural network with a fixed number of layers is parametric, but one whose number of layers is allowed to grow is not.

Given an estimate $\hat{m}$ of the regression function, the plug-in classifier is

$$\hat{f}(x) = \mathbb{I}\big[\hat{m}(x) > \tfrac{1}{2}\big].$$

If $\hat{m}(x) = \sum_{i=1}^n \ell_i(x) Y_i$ (any linear smoother), then

$$\hat{f}(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^n \ell_i(x)\, \mathbb{I}[Y_i = 1] > \sum_{i=1}^n \ell_i(x)\, \mathbb{I}[Y_i = 0], \\ 0 & \text{otherwise.} \end{cases}$$

In k-NN, you don't have a finite set of parameters: it grows with the data.

1.1 Classification versus regression

What kind of guarantees can you get for your classifier based on the guarantees for your regression estimate?

Intuition. In classification we could tolerate more error in the regression estimate, as long as it is on the same side of $\frac{1}{2}$. Let

$$m^*(x) = \mathbb{P}[Y = 1 \mid X = x].$$

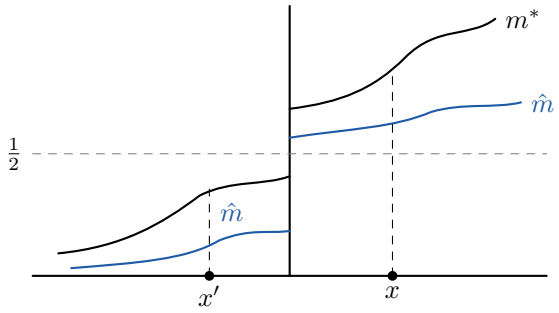

Figure 31: The estimate $\hat{m}$ is far from m^*, but at every point it lies on the same side of $\frac{1}{2}$, so the plug-in classifier is correct.

In this case, even though $\hat{m}$ and m^* are far away from each other, as long as they lie on the same side of $\frac{1}{2}$ it is fine. This gives some idea of why classification may be easier than regression.

Define the regression and classification risks

$$R_{\text{reg}}(\hat{m}, m^*) \equiv \mathbb{E}\,\|\hat{m} - m^*\|_2^2\,, \qquad R_{\text{class}}(\hat{m}, m^*) = \mathbb{E}\,\mathbb{P}\big[\hat{f}(X) \neq f^*(X)\big].$$

Then

$$R_{\text{class}}(\hat{m}, m^*) \leq 2\sqrt{R_{\text{reg}}(\hat{m}, m^*)},$$

so if you can solve regression, you can also solve classification. But R_{class} can be small even when R_{reg} is large.

For the minimax risks over a class $\mathcal{P}$, one can show

$$R^{\min}_{\text{class}}(\mathcal{P}) \leq 2\sqrt{R^{\min}_{\text{reg}}(\mathcal{P})}.$$

But is classification strictly easier?

- If $\mathcal{P}$ is simple, the inequality is strict.
- If $\mathcal{P}$ is sufficiently rich (e.g. Hölder classes), then there is equality in rate:

$$R^{\min}_{\text{class}}(\mathcal{P}) \asymp \sqrt{R^{\min}_{\text{reg}}(\mathcal{P})}.$$

This is why people are happy with the plug-in classifier $\hat{f}(x)$: we already have equality in complexity.

1.2 Low noise condition

When is classification problematic?

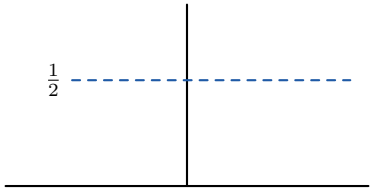

Figure 32: The threshold $\frac{1}{2}$ that the regression function is compared against.

Definition (Low noise condition)**.**

$$\mathbb{P}\Big[\big|m^*(X) - \tfrac{1}{2}\big| < t\Big] \leq t^\alpha.$$

When we don't have a lot of noise near the boundary, classification is easier.

1.3 Learning the classifier directly

Recall that in an RKHS,

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \Big\{ \frac{1}{n} \sum_{i=1}^{n} L\big(f(x_i), y_i\big) + \lambda \left\| f \right\|_K^2 \Big\}.$$

Then

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i \, k(x, x_i).$$

Idea: we can directly train $\hat{m}$ using a classification loss instead of the regression loss. Let $f : \mathcal{X} \to \mathbb{R}$ be a classifier score, and minimize for example

- the logistic loss $\hat{\mathbb{E}}_n \log\big(1 + \exp(-yf(x))\big)$;
- the exponential loss $\hat{\mathbb{E}}_n \exp\big(-yf(x)\big)$;
- the hinge loss $\hat{\mathbb{E}}_n \big[(1 - yf(x))_+\big]$.

We don't learn a regression function and then try to threshold it; we just learn the classifier directly.

When training neural network classifiers, there is an RKHS under the hood.

2 Three Key Ideas in Non-Parametric Estimation

- Smooth and hard partitions.
- Mercer kernels.
- Greedy algorithms.

3 Greedy Algorithms

Instead of an orthonormal basis for your function space, suppose you have a dictionary.

Definition (Dictionary)**.** A *dictionary* is a set of functions

$$\mathcal{H} = \{h : \mathcal{X} \to \mathbb{R}\}.$$

Its elements are called *atoms*. Let $\mathcal{F} = \overline{\operatorname{span}(\mathcal{H})}$.

It turns out you can use greedy algorithms to optimize over linear combinations of functions from this function space! For $F \in \mathcal{F}$, we want to solve

$$\min_{F \in \mathcal{F}} L(F) = \hat{\mathbb{E}}_n \, \ell\big(F, (x, y)\big).$$

3.1 The greedy algorithm

- Start with some $F_0 \in \mathcal{F}$.
- For $t = 1, 2, \ldots$:
$$\alpha_t, h_t \in \operatorname*{arg\,min}_{\alpha \in \mathbb{R},\, h \in \mathcal{H}} L(F_{t-1} + \alpha h).$$
This could be approximated with gradient descent.
- Output $F_T = \sum_{t=1}^{T} \alpha_t h_t$.

How do we pick the best h? It is good enough to pick a "reasonable" h. To first order,

$$L(F_{t-1} + \alpha h) \approx L(F_{t-1}) + \langle \nabla L(F_{t-1}), \alpha h \rangle,$$

and on the samples

$$\langle \nabla L(F_{t-1}), h \rangle_n \approx \frac{1}{n} \sum_{i=1}^{n} \frac{\partial L(F_{t-1})}{\partial F_{t-1}}(x_i, y_i)\, h(x_i).$$

Then find the h that maximizes this:

$$h_t \in \operatorname*{arg\,max}_{h \in \mathcal{H}} \big|\langle \nabla L(F_{t-1}), h \rangle\big|,$$

where L must be Fréchet-differentiable. With α_t the step size, update

$$F_t \leftarrow F_{t-1} + \alpha_t h_t.$$

3.2 Example: exponential loss with binary atoms

Let $\mathcal{H} = \{h : \mathcal{X} \to \{-1, +1\}\}$ and use the exponential loss

$$L(F) = \frac{1}{n} \sum_{i=1}^{n} \exp\big(-y_i F(x_i)\big).$$

Then

$$L(F_{t-1} + \alpha h) = \frac{1}{n} \sum_{i=1}^{n} \exp\big(-y_i (F_{t-1}(x_i) + \alpha h(x_i))\big) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\exp\big(-y_i F_{t-1}(x_i)\big)}_{w_i \text{ (scalar)}} \exp\big(-y_i \alpha h(x_i)\big).$$

Since $y_i, h(x_i) \in \{-1, +1\}$,

$$\exp\big(-\alpha y_i h(x_i)\big) = \mathbb{I}[y_i = h(x_i)]\, e^{-\alpha} + \mathbb{I}[y_i \neq h(x_i)]\, e^{\alpha},$$

so

$$L(F_{t-1} + \alpha h) = \frac{1}{n} \sum_{i=1}^{n} w_i \Big[e^{\alpha} \, \mathbb{I}[y_i \neq h(x_i)] + e^{-\alpha} \, \mathbb{I}[y_i = h(x_i)] \Big].$$

Write

$$\varepsilon_t(h) = \frac{1}{n} \sum_{i=1}^{n} w_i \, \mathbb{I}[h(x_i) \neq y_i].$$

Then, with the weights normalized so that $\frac{1}{n} \sum_i w_i = 1$,

$$L(F_{t-1} + \alpha h) = e^{\alpha} \varepsilon_t + e^{-\alpha}(1 - \varepsilon_t).$$

Taking gradients to find the best α:

$$e^{\alpha} \varepsilon_t - e^{-\alpha}(1 - \varepsilon_t) = 0 \quad \Longrightarrow \quad \alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}.$$

Substituting back,

$$L(F_{t-1} + \alpha_t h) = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}.$$

So minimizing over h is equivalent to minimizing this as a function of h:

$$\arg\min_{h \in \mathcal{H}} L(F_{t-1} + \alpha_t h) = \arg\min_{h \in \mathcal{H}} \varepsilon_t(h).$$

Lecture 11 — February 21, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Greedy Algorithms and Boosting

Let $\mathcal{H}$ be a dictionary of simple functions, and let

$$\mathcal{F} = \text{span}(\mathcal{H})$$

be the function class of linear combinations of simple functions.

1.1 Prediction

For prediction, $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$. Given data $\{(X_i, Y_i)\}_{i=1}^n$, the loss is

$$L(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), Y_i).$$

1.2 Weakly supervised learning

In weakly supervised learning we have

- labelled data $\{(X_i, Y_i)\}_{i=1}^n$;
- classifiers $\{g_j : \mathcal{X} \to \mathcal{Y}\}$ from experts. We want our predictor f to be close to these.

The loss is

$$L(f) = \lambda \sum_{i=n+1}^{m} \text{KL}\big(\underbrace{g(X_i)}_{\text{from expert}}, \underbrace{f(X_i)}_{\text{ours}} \big) + \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), Y_i),$$

where each KL term becomes, up to terms that do not depend on f,

$$-g(X_i) \log f(X_i) - \big(1 - g(X_i)\big) \log\big(1 - f(X_i)\big).$$

1.3 Semi-supervised learning

In semi-supervised learning we have

- labelled data $\{(X_i, Y_i)\}_{i=1}^n$;
- unlabelled data $\{X_i\}_{i=n+1}^m$.

The loss is

$$L(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i) + \sum_{i=n+1}^{m} \phi(|f(X_i)|), \qquad |f(X_i)| = f(X_i) \cdot \operatorname{sign}\big(f(X_i)\big).$$

On the unlabelled data the predictor should be confident, i.e. have a large classifier score.

Many loss functions are functions of the *margin*, the product of the label and the prediction:

$$\phi\big(F(X_i)\, Y_i\big).$$

So the unlabelled term treats $\operatorname{sign}(f(X_i))$ as the label.

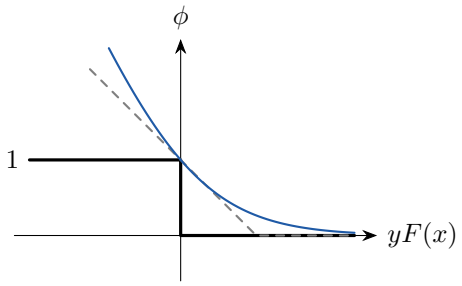

Figure 33: Margin losses as functions of the margin $yF(x)$: the 0–1 loss (black), a piecewise-linear surrogate (gray, dashed) and a smooth surrogate (blue).

Common trick: when the predictor is confident on some unlabelled data, use its prediction as the label and add the point to the training data.

1.4 The greedy algorithm

For $t = 1, \ldots, T$:

- $\alpha_t, h_t \in \underset{h \in \mathcal{H},\, \alpha \in \mathbb{R}}{\arg\min}\, L(f_{t-1} + \alpha h)$;
- $f_t \leftarrow f_{t-1} + \alpha_t h_t$.

The atom is chosen as

$$h_t \in \underset{h \in \mathcal{H}}{\arg\max} \big|\langle \nabla L(f_{t-1}), h \rangle\big|.$$

Eventually, we only want an approximate maximizer:

$$\big|\langle \nabla L(f_{t-1}), h_t \rangle\big| \geq (1 - \varepsilon) \max_{h \in \mathcal{H}} \big|\langle \nabla L(f_{t-1}), h \rangle\big|.$$

Even if you have no labels, the confidence criterion allows the method to learn a separator.

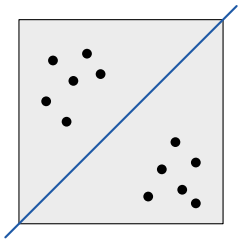

Figure 34: Unlabelled points in two clusters; asking for confident predictions places the boundary between them.

2 Optimal Transport

Consider three distributions P_1, P_2, P_3 with disjoint supports.

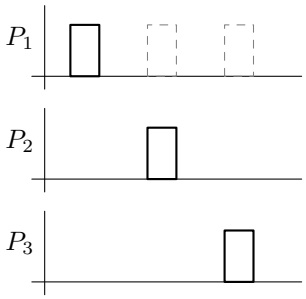

Figure 35: Three distributions with disjoint supports; the dashed boxes in the top panel mark where P_2 and P_3 sit.

If you look just at the total variation distance,

$$\|P_1 - P_2\|_{\mathrm{TV}} = \|P_1 - P_3\|_{\mathrm{TV}}\,, \qquad \text{where} \qquad \mathrm{TV}(P_1, P_2) = \sup_A |P_1(A) - P_2(A)|.$$

Idea: in the input space, your random variable can exhibit geometry (i.e. some notion of distance between the points). If I want to interpolate between P_1 and P_3, I want the mass to shift through the middle, not just move from P_1 to P_3.



Figure 36: Interpolating from P_1 to P_3: the mass should pass through the middle (blue).

A lot of distances are also not good for singular distributions (i.e. those without support everywhere). For example,

$$\mathrm{KL}(P_1, P_2) = \int_{\mathcal{X}} P_1(x) \log \frac{P_1(x)}{P_2(x)}\, dx$$

blows up if P_2 has no mass where P_1 does.

Most data we have are singular! For example, natural images are a small set in the space of all possible images.

2.1 The assignment problem

Suppose there are n warehouses $x_1, \dots, x_n$ and n factories $y_1, \dots, y_n$, and it costs $c(x_i, y_j)$ to transport from x_i to y_j.

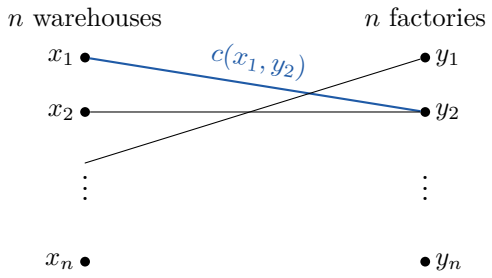

Figure 37: Transporting from n warehouses to n factories; each edge has a cost $c(x_i, y_j)$.

Given the costs, what is the optimal way of transporting? With $\sigma : \{1, \dots, n\} \to \{1, \dots, n\}$, solve

$$\min_{\sigma \in \mathrm{Perm}(n)} \sum_{i=1}^{n} c\big(x_i, y_{\sigma(i)}\big).$$

2.2 Monge maps

Let's relax some constraints. Suppose now there are more warehouses than factories, $n > m$, and warehouse x_i holds mass a_i while factory y_j needs mass b_j.

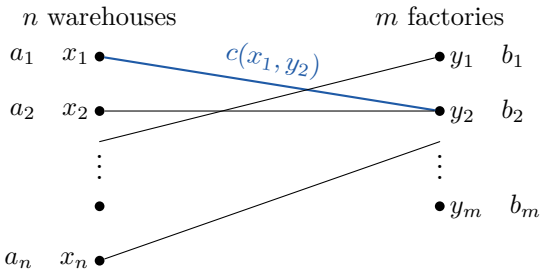

Figure 38: Warehouses with masses a_i and factories with demands b_j.

We can have a map, the *Monge map*,

$$T : \{x_1, \dots, x_n\} \to \{y_1, \dots, y_m\},$$

and solve

$$\min_T \sum_{i=1}^{n} a_i \, c(x_i, T(x_i)) \quad \text{subject to} \quad \sum_{i=1}^{n} a_i \, \mathbb{I}[T(x_i) = y_j] = b_j \text{ for all } j.$$

We don't want to transport more or less, as that leads to wastage.

Problem: suppose you have the following probability distribution, and want to move it to a different probability distribution.

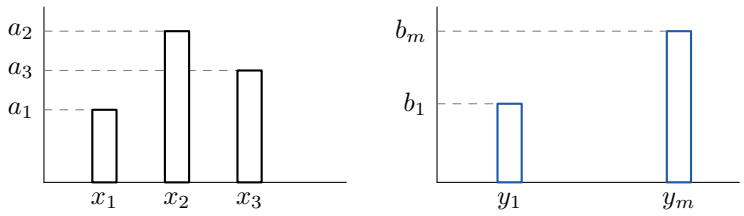

Figure 39: A discrete distribution with masses a_i at points x_i (left), to be moved to one with masses b_j at points y_j (right).

Write $\alpha = \sum_{i=1}^n a_i \delta_{x_i}$ and $\beta = \sum_{j=1}^m b_j \delta_{y_j}$. The constraint says that the *pushforward* of α with respect to T equals β:

$$T_\# \alpha = \sum_{i=1}^{n} a_i \, \delta_{T(x_i)} = \sum_{j=1}^{m} b_j \, \delta_{y_j}.$$

The Monge problem is then

$$\min_T \mathop{\mathbb{E}}_{x \sim \alpha} c(x, T(x)) \quad \text{subject to} \quad T_\# \alpha = \beta.$$

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 12 — February 23, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Optimal Transport

1.1 Recap: Monge maps

Recall that we had a distribution α over $\mathcal{X}$, a distribution β over $\mathcal{Y}$, and a cost

$$c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}.$$

We solve

$$\min_{T:\mathcal{X}\to\mathcal{Y}} \mathop{\mathbb{E}}_{x\sim\alpha} c(x, T(x)) \quad \text{subject to} \quad T_{\#}\alpha = \beta,$$

where $T_{\#}\alpha$ is the pushforward. This is a mass conservation constraint. The pushforward $T_{\#}\alpha$ is a distribution over $\mathcal{Y}$: for $B \subseteq \mathcal{Y}$,

$$T_{\#}\alpha(B) = \alpha(\{x \in \mathcal{X} \mid T(x) \in B\}) = \alpha(T^{-1}(B)).$$

1.2 Kantorovich relaxation

Let P_{ij} be the number of units sent from the i-th warehouse to the j-th factory, so $P \in \mathbb{R}^{n\times m}$. For the i-th warehouse,

$$a_i = \sum_{j=1}^{m} P_{ij},$$

and summing over warehouses,

$$\sum_{i=1}^{n} P_{ij} = b_j.$$

Definition (Kantorovich relaxation)**.**

$$\begin{aligned} \min_{P} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij}\, c(x_i, y_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} P_{ij} = b_j \quad \forall j \in [m], \\ & \sum_{j=1}^{m} P_{ij} = a_i \quad \forall i \in [n]. \end{aligned}$$

In general,

$$\begin{aligned} L_c(\alpha,\beta) = \min_{\pi:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}} \quad & \int_{\mathcal{X}}\int_{\mathcal{Y}} c(x,y)\,\pi(x,y)\,dx\,dy \\ \text{subject to} \quad & \pi(x,y) \geq 0, \\ & \iint \pi(x,y)\,dx\,dy = 1, \\ & \int \pi(x,y)\,dy = \alpha(x), \\ & \int \pi(x,y)\,dx = \beta(y). \end{aligned}$$

1.3 The dual problem

The dual Kantorovich problem (an LP) of the discrete case has as many variables as constraints:

$$L_c(\alpha,\beta) = \max_{f\in\mathbb{R}^n,\ g\in\mathbb{R}^m} \underbrace{\sum_{i=1}^{n} f_i a_i}_{\mathbb{E}_{x\sim\alpha}[f(x)]} + \underbrace{\sum_{j=1}^{m} g_j b_j}_{\mathbb{E}_{y\sim\beta}[g(y)]} \quad \text{subject to} \quad f_i + g_j \leq c(x_i, y_j).$$

We can think of $f_i + g_j$ as "vendors": if the vendor is cheaper, go with the vendor.

So we can rewrite the dual in terms of the *Kantorovich potentials*:

$$\max_{f:\mathcal{X}\to\mathbb{R},\ g:\mathcal{Y}\to\mathbb{R}} \mathop{\mathbb{E}}_{x\sim\alpha}[f(x)] + \mathop{\mathbb{E}}_{y\sim\beta}[g(y)] \quad \text{subject to} \quad f(x) + g(y) \leq c(x,y).$$

2 Wasserstein Distance

In practice, mostly $\mathcal{Y} = \mathcal{X}$, with a distance metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

Definition (Wasserstein distance)**.**

$$W_p(\alpha,\beta) = \left[L_{d^p}(\alpha,\beta)\right]^{1/p}.$$

We mostly look at W_2 (squared distance) and W_1, with d the Euclidean distance.

2.1 Empirical distributions

Suppose $\mathcal{X} = \mathbb{R}$ and

$$\alpha = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}, \qquad \beta = \frac{1}{n}\sum_{i=1}^{n} \delta_{y_i}.$$

Then, with $x_{(i)}$ and $y_{(i)}$ the sorted points,

$$W_p(\alpha,\beta)^p = \frac{1}{n}\sum_{i=1}^{n}|x_{(i)} - y_{(i)}|^p.$$

If $\mathcal{X} = \mathbb{R}^d$,

$$W_p(\alpha,\beta)^p = \min_{\sigma\in\mathrm{Perm}(n)} \frac{1}{n}\sum_{i=1}^{n}\left\|x_i - y_{\sigma(i)}\right\|_2^p.$$

For $\mathcal{X} = \mathbb{R}$, with $\alpha \sim F_\alpha$ and $\beta \sim F_\beta$ (their CDFs),

$$W_p(\alpha,\beta)^p = \int_0^1 |F_\alpha^{-1}(t) - F_\beta^{-1}(t)|^p\,dt.$$

In 2d, we can solve for the Wasserstein distance in closed form. In general, it is an LP problem.

2.2 The case $p \in (0,1]$

For W_p with $p \in (0,1]$, we can take $g = -f$ with $\mathrm{Lip}_{d^p}(f) \le 1$, where

$$\mathrm{Lip}_{d^p}(f) = \sup_{\substack{x,y\in\mathcal{X}\\ x\neq y}} \frac{|f(x) - f(y)|}{d(x,y)^p}.$$

In this case there is only one Kantorovich potential to optimize over:

$$W_p(\alpha,\beta) = \max_{f:\mathrm{Lip}_{d^p}(f)\le 1}\Big\{\mathop{\mathbb{E}}_{x\sim\alpha} f(x) - \mathop{\mathbb{E}}_{y\sim\beta} f(y)\Big\}.$$

When we have d^p with $p \in (0,1]$, the set

$$\{(f,g) \mid f(x) + g(y) \le d^p(x,y)\}$$

gives the same constraints as

$$\{(f,-f) \mid \mathrm{Lip}_{d^p}(f) \le 1\}.$$

Similarly,

$$\mathrm{TV}(\alpha,\beta) = \max_{f:\|f\|_\infty\le 1}\Big\{\mathop{\mathbb{E}}_{x\sim\alpha}[f(x)] - \mathop{\mathbb{E}}_{y\sim\beta}[f(y)]\Big\}.$$

These compare moments, in an overloaded sense: moments through a function.

3 Integral Probability Metrics

Definition (Integral probability metric (IPM))**.** For a class of functions $\mathcal{B}$,

$$D_{\mathcal{B}}(\alpha,\beta) = \max_{f\in\mathcal{B}}\Big\{\mathop{\mathbb{E}}_{x\sim\alpha}[f(x)] - \mathop{\mathbb{E}}_{y\sim\beta}[f(y)]\Big\}.$$

For example, take the unit ball of an RKHS,

$$\mathcal{B} = \{f \in \mathcal{H}_K \mid \|f\|_{\mathcal{H}_K} \le 1\},$$

where $\|\cdot\|_{\mathcal{H}_K}$ is the RKHS norm, and consider $D_{\mathcal{B}}(\alpha,\beta)$.

3.1 Statistical estimation

Let $\mathbb{P}_n$ be the empirical distribution of $\{X_i\}_{i=1}^n$, and suppose you have some parametric family $\{P_\theta\}_{\theta\in\Theta}$. Consider

$$\min_{\theta\in\Theta} \mathrm{KL}(\mathbb{P}_n, P_\theta).$$

What if, instead of KL, you use the Wasserstein distance?

$$\min_{\theta\in\Theta} W_1(\mathbb{P}_n, P_\theta) = \min_{\theta\in\Theta} \max_{f:\mathrm{Lip}(f)\le 1} \Big\{ \mathop{\mathbb{E}}_{X\sim\mathbb{P}_n}[f(X)] - \mathop{\mathbb{E}}_{X'\sim P_\theta}[f(X')] \Big\}.$$

The inner maximization is a discriminator: it tries to find an f such that the moment of the true and generated distributions is different. This is called a *Wasserstein GAN*.

But why only smooth functions? We can also use an RKHS.

3.2 Bayes risk

Let $(X, Y) \sim P$. The Bayes risk is

$$\inf_f P(f(X)\neq Y) = \frac{1}{2} - \frac{1}{2} L_{0/1}(P_{X|Y=1}, P_{X|Y=0}) = \frac{1}{2} - \frac{1}{2} \underbrace{\mathrm{TV}(P_{X|Y=1}, P_{X|Y=0})}_{\text{how separable they are}}.$$

If the TV distance is 0, we cannot separate the classes. Otherwise, if we can separate them completely, the risk becomes 0.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 13 — February 28, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Midterm Review

1.1 Non-Parametric Methods

So far we have seen non-parametric methods for

- density estimation;
- regression.

1.2 Regression

The target is $f \in \mathcal{F}$, where $\mathcal{F}$ is infinite dimensional. We explicitly characterize and use $\mathcal{F}$.

Orthogonal series

Take $\mathcal{F}$ to be the set of linear combinations of basis functions,

$$\mathcal{F} = \Big\{ f(x) \;\Big|\; f = \sum_{j}^{\infty} \theta_j \psi_j(x) \Big\},$$

where $\{\psi_j\}$ is an orthonormal basis in L^2:

$$\int \psi_j(x)\psi_{j'}(x)\, dx = \mathbb{I}[j = j'].$$

Problem: there is an infinite set of coefficients to learn,

$$f(x) = \sum_{j=1}^{\infty} \theta_j \psi_j(x) \quad \text{such that} \quad \sum_{j=1}^{\infty} \theta_j^2 < \infty$$

(f is in the L^2 space, so the coefficients are bounded).

To enforce additional regularity, we require

$$\sum_{j=1}^{\infty} \theta_j^2 a_j < \infty, \qquad \text{where } a_j \xrightarrow[j\to\infty]{} \infty, \quad \text{e.g. } a_j = j^m.$$

Then θ_j^2 must decay fast!

If the ψ_j are orthonormal, we can express θ_j in closed form:

$$\begin{aligned}
\theta_j = \langle f, \psi_j \rangle &= \int f(x)\psi_j(x)\,dx \;\approx\; \frac{1}{n}\sum_{i=1}^{n}\psi_j(X_i) \qquad \text{(Monte Carlo approximation)} \\
&= \int \sum_{j'} \theta_{j'}\psi_{j'}(x)\psi_j(x)\,dx \\
&= \sum_{j'} \theta_{j'} \underbrace{\int \psi_{j'}(x)\psi_j(x)\,dx}_{=\mathbb{I}[j=j']} \\
&= \theta_j.
\end{aligned}$$

1.3 RKHS

$$\mathcal{F} = \Big\{ f \;\Big|\; \exists\{\alpha_i\}, \{x_i\},\ f(x) = \sum_i \alpha_i K(x, x_i) \Big\}.$$

That is,

$$f(x) = \sum_i \alpha_i K_{x_i}(x), \qquad \text{where } K_{x'}(x) = K(x', x).$$

The function $K_{x'}$ evaluates f at x':

$$\langle f, K_{x'} \rangle_{\mathcal{H}_K} = f(x').$$

For $f = \sum_i \alpha_i K(x, x_i)$ and $g = \sum_j \beta_j K(x, y_j)$, the inner product is

$$\langle f, g \rangle_{\mathcal{H}_K} = \sum_{i,j} \alpha_i \beta_j K(x_i, y_j),$$

so in particular

$$\langle f, K_{x'} \rangle = \sum_i \alpha_i K(x_i, x') = f(x').$$

This is any space where function evaluation is continuous. Define the evaluation functional

$$\delta_x(f) = f(x).$$

δ_x is linear:

$$\delta_x(af + bg) = af(x) + bg(x) = a\delta_x(f) + b\delta_x(g).$$

In general, however,

$$f_n \to f \quad \not\Rightarrow \quad \delta_x(f_n) \to \delta_x(f),$$

that is, function evaluations may not converge.

Say δ_x is continuous. Then

$$\delta_x(f) = \langle K_x, f \rangle .$$

This is linear: recall that in 1-D Euclidean space it would be like $y = ax$. In finite dimensions this is clearly the case, since a linear functional has the form

$$h_\theta(v) = \langle \theta, v \rangle .$$

In infinite dimensions we need some condition on continuity. Here K_x is just a function:

$$K(x', x) = K_x(x').$$

We only see

$$\delta_{x_i}(f) = f(x_i),$$

but we want to infer $\delta_x(f)$ at a test point x.

Big question: choosing K. Can we make use of some domain knowledge? Using generic things like the RBF kernel is not as good.

The kernel we are concerned with here is the Mercer kernel. A smoothing kernel is different.

Mercer kernels

A Mercer kernel has a decomposition

$$K(x, x') = \sum_{j=1}^{\infty} \lambda_j \psi_j(x) \psi_j(x'), \qquad \int \psi_j(x) \psi_{j'}(x)\, dx = \mathbb{I}[j = j'].$$

Then f lies in a Hilbert space that is a subset of L^2. For

$$f = \sum_j a_j \psi_j(x), \qquad g = \sum_j b_j \psi_j(x),$$

we have

$$\langle f, g \rangle_{\mathcal{H}_K} = \sum_j \frac{a_j b_j}{\lambda_j}, \qquad \langle f, f \rangle_{\mathcal{H}_K} = \sum_j \frac{1}{\lambda_j} a_j^2,$$

whereas in L^2, $\|f\|_{L^2}^2 = \sum_j a_j^2$. Finiteness of $\langle f, f \rangle_{\mathcal{H}_K}$ requires f to be even smoother, since the coefficients a_j^2 must become smaller.

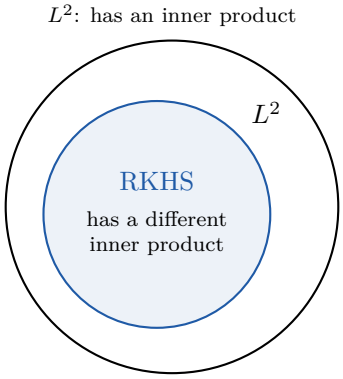

Figure 40: The RKHS is a subset of L^2 equipped with a different inner product.

For a Mercer kernel, we need K to be symmetric and PSD.

Kernels transform an orthonormal basis into an ordered basis, in order of importance. With the feature map

$$\Phi(x) := (\sqrt{\lambda_1}\,\psi_1(x), \sqrt{\lambda_2}\,\psi_2(x), \dots),$$

we have

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle = \sum_j \Phi_j(x)\Phi_j(x').$$

2 Optimal Transport

Let $X \sim \alpha$ and $Y \sim \beta$. Optimal transport looks for the joint distribution π solving

$$\min_\pi \int c(x, y)\pi(x, y)\, dx\, dy$$

subject to the marginal constraints

$$\int \pi(x, y)\, dy = \alpha(x), \qquad \int \pi(x, y)\, dx = \beta(y).$$

This can be restated as

$$L_c(\alpha, \beta) = \min_\pi \mathop{\mathbb{E}}_{(X,Y)\sim\pi} c(X, Y) \quad \text{s.t.} \quad X \sim \alpha,\ Y \sim \beta$$

(Wasserstein distance).

This is a relaxation of the *Monge transport problem*:

$$\min_T \mathop{\mathbb{E}}_{X\sim\alpha} c(X, T(X)) \quad \text{s.t.} \quad Y = T(X).$$

For example, in images, Y could be an image and X a sketch.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 14 — March 14, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Deep Density Estimation

Neural nets are poorly understood: theory exists mostly only for 1- or 2-layer neural nets. We don't have a result that says that a neural net approximates a given function.

Why do neural nets work for non-convex problems? They are heavily overparameterized function classes.

Given data $\{X_i\}_{i=1}^n \sim P$ and a model $\{P_\theta\}_{\theta\in\Theta}$, how do you fit the model to the data?

1.1 Maximum likelihood and energy-based models

The classical approach is the MLE:

$$\max_\theta \frac{1}{n}\sum_{i=1}^n \log P_\theta(X_i).$$

We need

$$\int P_\theta(x)\,dx = 1, \qquad P_\theta(x) \geq 0,$$

but how do you ensure this? This is the *normalization problem.*

The first thing people tried was the logistic transform,

$$P_\theta(x) = \frac{\exp(\eta_\theta(x))}{\int \exp(\eta_\theta(x))\,dx},$$

which gives an *energy-based model.* So now we just need to learn $\eta_\theta : \mathcal{X} \to \mathbb{R}$.

The MLE is then

$$\max_\theta \ \frac{1}{n}\sum_{i=1}^n \eta_\theta(X_i) - \underbrace{\log \int_{\mathcal{X}} \exp(\eta_\theta(x))\,dx}_{\text{log partition function}}.$$

The log partition function is intractable. People use sampling and approximations (e.g. contrastive divergence).

1.2 Avoiding the log partition function

The MLE does

$$\min_\theta \mathrm{KL}(\hat{P}, P_\theta).$$

But instead of the KL, could we do something that avoids computing the log partition function?

Consider

$$\min_\theta \mathop{\mathbb{E}}_{x\sim\hat{p}} \exp(-\eta_\theta(x)) + \mathop{\mathbb{E}}_{x\sim q}\big[\eta_\theta(x)\big],$$

where q is a noise distribution. (This is numerically a challenge: the variances are very different.)

Suppose η_θ is very flexible and can approximate all functions. Then this is

$$\min_f \Big\{ \int p(x)\exp(-f(x))\,dx + \int q(x)f(x)\,dx \Big\}.$$

Setting the derivative to 0:

$$p(x)\exp(-f(x)) = q(x) \quad \Longrightarrow \quad \exp(f(x)) = \frac{p(x)}{q(x)} \quad \Longrightarrow \quad f(x) = \log\frac{p(x)}{q(x)}.$$

So we can recover the density as

$$\hat{p}(x) = q(x)\exp(\hat{f}(x)).$$

1.3 Noise Contrastive Estimation (NCE)

$$\min_\theta \mathop{\mathbb{E}}_{x\sim\hat{p}} \log\sigma(\eta_\theta(x)) + \mathop{\mathbb{E}}_{x\sim q} \log\sigma(-\eta_\theta(x)), \qquad \sigma(z) = \frac{\exp(z)}{1+\exp(z)}.$$

Over all functions f:

$$\begin{aligned}
&\min_f \mathop{\mathbb{E}}_{x\sim\hat{p}} \log\sigma(f(x)) + \mathop{\mathbb{E}}_{x\sim q} \log\sigma(-f(x)) \\
&= \min_f \int p(x)\big[f(x) - \log(1+\exp(f(x)))\big]\,dx + \int q(x)\big[-f(x) - \log(1+\exp(-f(x)))\big]\,dx.
\end{aligned}$$

Take the derivative with respect to $f(x)$ and set it to 0:

$$p(x)(1-\sigma(f(x))) - q(x)(1-\sigma(-f(x))) = \frac{p(x)}{1+\exp(f(x))} - \frac{q(x)\exp(f(x))}{1+\exp(f(x))} = 0,$$

which gives

$$\exp(f(x)) = \frac{p(x)}{q(x)}.$$

Unless q is very close to p, the optimization landscape is very flat.

Guideline for choosing q: we don't want it to be too large, or it is hard to train. But statistically we want q to be close to p.

NCE works when the data distribution is not too complicated (so no images). But it has an appealing property: the normalization constant can be computed in finite time. NCE is used for discrete distributions.

1.4 Latent variable models

Let

$$Z \sim N(0, I), \qquad X \mid Z = z \sim N(\mu(z), \sigma^2(z) I).$$

We can view this as

$$X = \mu(Z) + \sigma(Z) w, \qquad w \sim N(0, I).$$

Then

$$\begin{aligned} p(x) &= \int p(x \mid Z = z)\, p_Z(z)\, dz \\ &= \int \phi(x; \mu(z), \sigma^2(z) I)\, \phi(z; 0, I)\, dz. \end{aligned}$$

This could be complicated, since μ and σ can be arbitrary. But the integral is still hard to evaluate in

$$\max_{\mu,\sigma} \sum_{i=1}^{n} \log p(X_i; \mu, \sigma).$$

The evidence lower bound

For any distribution $q_\phi(z \mid x)$,

$$\begin{aligned} \log p_\theta(x) &= \log \int_z \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] q_\phi(z \mid x)\, dz \\ &\geq \int_z q_\phi(z \mid x) \log \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] dz \qquad \text{(Jensen's inequality)} \\ &= \text{ELBO} = \mathcal{L}(\phi, \theta). \end{aligned}$$

The ELBO is easier to deal with: there is no integral inside the log. So

$$\max_\theta \hat{\mathbb{E}} \log p_\theta(x) \;\geq\; \max_\theta \hat{\mathbb{E}} \max_\phi \mathcal{L}(\theta, \phi).$$

1.5 VAE (variational autoencoder)

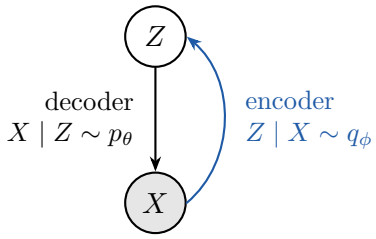

Figure 41: A VAE: the decoder p_θ generates X from the latent Z, and the encoder q_ϕ maps the observed X back to Z.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 15 — March 16, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Generating Data from Noise

Let $Z \sim N(0, I)$. Then any distribution can be modeled as applying some function f to a Gaussian:

$$X = f_\theta(Z),$$

where f_θ is deterministic.

How do we go from any distribution to noise? For $X \in \mathbb{R}$:

1. $U = F_X(X)$ (we get a uniform from the CDF);
2. $Z = \Phi^{-1}(U)$.

Hence

$$Z = \Phi^{-1} \circ F_X(X) \quad \Longrightarrow \quad X = F_X^{-1} \circ \Phi(Z).$$

In d dimensions, let $X = (X_1, X_2, \ldots, X_d)$ and set

$$\begin{aligned} U_1 &= F_{X_1}(X_1), \\ U_2 \mid U_1 &= F_{X_2}(X_2 \mid U_1), \\ &\vdots \\ U_d \mid U_1, \ldots, U_{d-1} &= \cdots \end{aligned}$$

Then $(U_1, \ldots, U_d) \sim \text{Unif}(0,1)^d$. (We need to condition so that they are jointly uniform, not just marginally uniform.)

So given any X, we can write it as $X = f(Z)$. The problem is whether f_θ is invertible.

If f_θ is invertible, then $Z = f_\theta^{-1}(X)$ and

$$p_X(x) = p_Z(f_\theta^{-1}(x)) \left|\det J f_\theta^{-1}(x)\right|,$$

where J denotes the Jacobian.

Definition (Jacobian)**.** Let $h : \mathbb{R}^d \to \mathbb{R}^d$, $h = (h_1, \ldots, h_d)$ (like having d functions). The *Jacobian* of h is

$$[Jh(x)]_{ij} = \frac{\partial h_i(x)}{\partial x_j}.$$

Compare: if $h : \mathbb{R}^d \to \mathbb{R}$, then $[\nabla h(x)]_j = \dfrac{\partial h(x)}{\partial x_j}$.

Constraining the function to be invertible is very strong: for instance, neural nets are not invertible (many inputs could give the same output).

If f_θ is invertible, we can fit it by maximum likelihood:

$$\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_{X;\theta}(x_i).$$

What are the architectures that are invertible?

2 Normalizing Flows

Example 9. Let $Z = (Z_1, Z_2)$ and

$$\begin{aligned} X_1 &= Z_1, \\ X_2 &= m_\theta(Z_1) + Z_2, \end{aligned}$$

where m_θ is any function (it does not have to be invertible). Then given (X_1, X_2), we can recover Z_1, Z_2:

$$Z_1 = X_1, \qquad Z_2 = X_2 - m_\theta(Z_1) = X_2 - m_\theta(X_1).$$

Another nice property: the Jacobian is triangular. Then $\det J$ is just the product of the diagonal entries.

The problem here: we distributed the variables into chunks and tried to model an entire chunk at once. What if we model them one by one instead?

2.1 Autoregressive Flows

For $i = 1, \ldots, d$, let

$$X_i = \mu_i(X_{<i}) + Z_i \exp\big(\alpha_i(X_{<i})\big),$$

where the μ_i are neural nets and the α_i are some functions (e.g. neural nets). Then

$$Z_i = \big(X_i - \mu_i(X_{<i})\big) \exp\big(-\alpha_i(X_{<i})\big),$$

so the map is invertible even when μ_i and α_i are very complicated.

Downside: we need to sample the X_i sequentially, since X_i depends on the previous values $X_{<i}$. But we can compute the Z_i in parallel (given the X_i's).

2.2 Masked Autoregressive Flow (MAF)

$$\begin{aligned} X_i &= \mu_i(Z_{<i}) + Z_i \exp\big(\alpha_i(Z_{<i})\big), \\ Z_i &= \big(X_i - \mu_i(Z_{<i})\big) \exp\big(-\alpha_i(Z_{<i})\big). \end{aligned}$$

When using images, we also need some notion of ordering, and then it is hard to determine what ordering is good (it could make the problem easier or harder).

In your construction, you want to bake in invertibility and place no restriction on the functions.

2.3 RNN

$$Z_i = h_\theta(Z_{i-1}), \qquad X_i = f_\theta(X_{i-1}, Z_i).$$

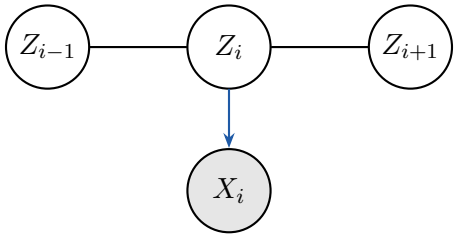

Figure 42: The RNN model: a chain of latent states Z_i, each emitting an observation X_i.

3 Non-Invertible Generators

With a general generator $Z \to X$, $X = f_\theta(Z)$, that is not invertible, how did people try to fix it?

One option is

$$\min_\theta \mathrm{KL}(P, P_\theta) = \int p(x) \log \frac{p(x)}{p_\theta(x)}\, dx.$$

A downside of the KL: we could be generating images that are unrealistic, and this is not penalized by the KL.

So use the inverse KL:

$$\int p_\theta(x) \log \frac{p_\theta(x)}{p(x)}\, dx.$$

Now if you generate unrealistic images, the $\log(p_\theta(x)/p(x))$ term blows up. But the problem is that we cannot minimize it, since we have no access to $p(x)$ (it is not decomposable).

Definition (Jensen–Shannon divergence)**.**

$$\mathrm{JSD}(P, P_\theta) = \frac{1}{2}\mathrm{KL}\Big(P, \frac{P + P_\theta}{2}\Big) + \frac{1}{2}\mathrm{KL}\Big(P_\theta, \frac{P + P_\theta}{2}\Big).$$

This is symmetric.

Let $D : \mathcal{X} \to [0, 1]$ and

$$V(P, P_\theta, D) = \mathop{\mathbb{E}}_{x \sim P} \log D(x) + \mathop{\mathbb{E}}_{x \sim P_\theta} \log(1 - D(x))$$

(the standard cross-entropy loss). D tries to discriminate between P and P_θ. If you maximize over D:

$$\max_D V(P, P_\theta, D) = 2\,\mathrm{JSD}(P, P_\theta) - \log 4.$$

So we solve

$$\min_{\theta} \max_{D} V(P, P_\theta, D).$$

Decomposable means the loss is an expectation, which we can estimate with a sample average:

$$L(P) = \mathop{\mathbb{E}}_{x \sim P}[\ell(x)] \approx \frac{1}{n} \sum_{i=1}^{n} \ell(x_i).$$

Why is minimizing over θ easy? With $X = h_\theta(Z)$,

$$\mathop{\mathbb{E}}_{X \sim P_\theta} g(X) = \mathop{\mathbb{E}}_{Z} g(h_\theta(Z))$$

(the *reparameterization trick*). So we can find

$$\min_{\theta} \mathop{\mathbb{E}}_{Z \sim N(0,I)} \log(1 - D(h_\theta(Z)))$$

using SGD, etc.

4 Gaussianization

Another approach: destroy your signal in an invertible manner until it becomes noise. This is called *Gaussianization*.

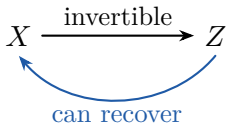

Figure 43: Gaussianization: an invertible map takes the data X to noise Z, so X can be recovered from Z.

Lecture 16 — March 21, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Kernels and Eigenvalue Decay

Let $\mathcal{X} = \mathbb{R}^d$ and consider functions $f : \mathcal{X} \to \mathcal{Y}$. A standard choice of kernel is the *RBF kernel*

$$K(x, x') = \exp\Big(-\frac{1}{2\sigma^2}\,\|x - x'\|^2\Big).$$

The RBF kernel requires an exponential number of samples.

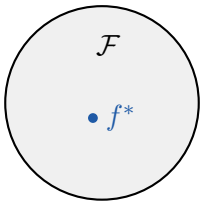

Figure 44: The function class $\mathcal{F}$, which contains the function f^*.

1.1 Spectral decomposition

The kernel has a spectral decomposition

$$K(x, x') = \sum_{i=1}^{\infty} \lambda_i\, \phi_i(x)\, \phi_i(x'), \qquad \lambda_1 \geq \lambda_2 \geq \cdots .$$

Define the feature map

$$\psi(x) = \Big(\sqrt{\lambda_1}\,\phi_1(x),\ \sqrt{\lambda_2}\,\phi_2(x),\ \ldots\Big).$$

Then

$$\langle \psi(x), \psi(y) \rangle = K(x, y),$$

and functions in the RKHS can be written as

$$f(x) = \beta^T \psi(x) = \sum_{i=1}^{\infty} \beta_i\, \psi_i(x).$$

Equivalently, in terms of the eigenfunctions,

$$f(x) = \sum_{i=1}^{\infty} \alpha_i\, \phi_i(x), \qquad \|f\|^2_{\mathcal{H}_K} = \sum_{i=1}^{\infty} \frac{\alpha_i^2}{\lambda_i} < \infty.$$

1.2 Effective dimensionality

Definition (Effective dimensionality)**.** The *effective dimensionality* of a kernel K is

$$\text{effdim}(K) = \text{tr}(K) = \sum_{i=1}^{\infty} \lambda_i.$$

If the eigenvalues of the kernel decay quickly, we effectively have a finite-dimensional kernel.

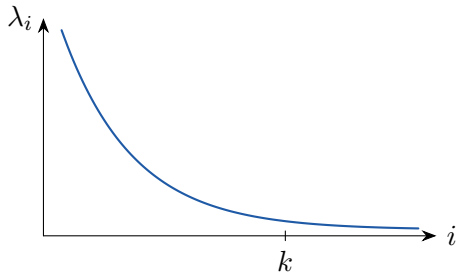

Figure 45: Fast eigenvalue decay: beyond some index k, the eigenvalues λ_i are close to zero.

2 Choosing a Kernel

So why don't we just take any kernel whose eigenvalues decay fast? Because it may not be able to characterize our function well: it could be that our function requires more than the top k eigenfunctions to be represented well.

2.1 First desideratum: sharp eigenvalue decay

Sharp eigenvalue decay helps with the estimation error. We want

$$\sum_{i=r+1}^{\infty} \lambda_i \quad \text{small.}$$

With $K(x, x') = \sum_{i=1}^{\infty} \lambda_i \, \phi_i(x) \, \phi_i(x')$, we want

$$L(\hat{f}_n) \approx \inf_{f \in \mathcal{H}_K} L(f),$$

that is, we want

$$\sup_{f \in \mathcal{H}_K} \left| L_n(f) - L(f) \right| \quad \text{small.}$$

2.2 Second desideratum: small RKHS norm of f^*

We want $\|f^*\|_{\mathcal{H}_K}$ to be small. Let $\mathcal{H}_K$ be the RKHS, and let

$$\mathcal{H}_{K,r} = \text{span}\{\text{first } r \text{ eigenfunctions}\}.$$

We want $f^* \in \mathcal{H}_{K,r}$ for small r.

The RKHS should be such that the RKHS norm of f^* is small. Writing $f^* = \sum_i \alpha_i^* \phi_i$, with eigenvalues $\lambda_i = e^{-i}$, we get

$$\sum_{i=1}^{\infty} \frac{(\alpha_i^*)^2}{\lambda_i} = \sum_{i=1}^{\infty} (\alpha_i^*)^2 e^i.$$

So α_i^* must decay more sharply than λ_i. So choose your kernel such that the RKHS norm is not too large.

2.3 Off-the-shelf kernels

Off-the-shelf kernels include cosine and Fourier bases. But these are so simple that obviously the first few basis functions cannot represent complicated things like tables.

In a lot of modelling, people implicitly specify kernels that are different from the classical kernels.

3 Representation Learning

Modern self-supervised learning pipelines do *representation learning*: they learn features by solving auxiliary tasks. Given data $\{X_i\}_{i=1}^n$:

1. Learn $\phi : \mathcal{X} \to \mathbb{R}^\gamma$. This is the unsupervised stage.
2. Fix the functions $f(x) = \beta^T \phi(x)$. This is the supervised stage.

3.1 How to learn ϕ

Suppose we have an image x. We can apply transformations to it: rotate, blur, re-color, crop, translate, etc.

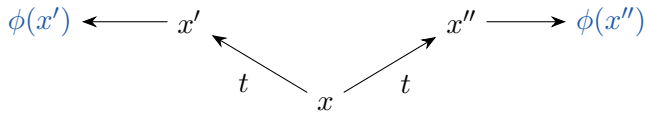

Figure 46: Two random transformations t of the same image x give x' and x'', which are then mapped to features.

Then $\phi(x')$ and $\phi(x'')$ should be close, but for some other image, its features should be far from them.

We parameterize ϕ using a neural network architecture, and maximize

$$\mathop{\mathbb{E}}_{(x',x'')\sim P_+} \log \sigma\big(\underbrace{\phi(x') \cdot \phi(x'')}_{g(x',x'')}\big) + \mathop{\mathbb{E}}_{x',x''\sim P_d} \log \sigma\big(-\phi(x') \cdot \phi(x'')\big).$$

The optimum is

$$\boxed{\phi(x') \cdot \phi(x'') = \log \frac{P_+(x', x'')}{P_d(x')\, P_d(x'')},}$$

which is a kernel $K(x', x'')$: the *PMI kernel* (pointwise mutual information). We can also use energy-based models:

$$\exp(\phi(x') \cdot \phi(x'')) = \frac{P_+(x', x'')}{P_d(x')\, P_d(x'')}.$$

Recall $K(x, y) = \langle \psi(x), \psi(y) \rangle$.

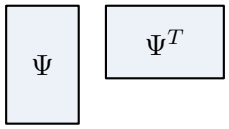

Figure 47: The feature matrix Ψ and its transpose Ψ^T.

$$\Psi B B^T \Psi^T = \Psi \Psi^T.$$

Once you learn ϕ, after that you just need to learn linear functions (which is exactly a function in the RKHS).

So here the kernel is defined in terms of the data distribution:

$$P_+(x', x'') = \int P_d(x)\, P_T(x' \mid x)\, P_T(x'' \mid x)\, dx,$$

where P_T is the transformation distribution: we have some transformation distribution over T. The factor $P_d(x)$ is why the kernel depends on the training data. So the better (larger) your training data, the better the estimation of your kernel.

4 Random Features

We had

$$K(x, y) = \sum_{i=1}^{\infty} \lambda_i\, \phi_i(x)\, \phi_i(y).$$

In the general case,

$$K(x, y) = \int \lambda(w)\, \phi_w(x)\, \phi_w(y)\, dw.$$

Theorem 4 (Bochner's theorem)**.** *If $K(x, y) = k(x - y)$ (for example, the RBF kernel), then it can be represented as*

$$K(x, y) = \int p(w)\, \phi_w(x)\, \phi_w(y)\, dw.$$

How do we get the eigenvalues from this? We can just use a Monte Carlo approximation. Draw $w_j \sim p(w)$ for $j = 1, \ldots, \gamma$. Then

$$K(x, y) \approx \frac{1}{\gamma} \sum_{j=1}^{\gamma} \phi_{w_j}(x)\, \phi_{w_j}(y).$$

Define

$$\Phi_\gamma(x) = \Big(\tfrac{1}{\sqrt{\gamma}}\, \phi_{w_1}(x),\ \tfrac{1}{\sqrt{\gamma}}\, \phi_{w_2}(x),\ \ldots,\ \tfrac{1}{\sqrt{\gamma}}\, \phi_{w_\gamma}(x) \Big).$$

Then

$$\Phi_\gamma(x) \cdot \Phi_\gamma(y) \approx K(x, y).$$

This is one of the earliest feature learning approaches.

This does not depend on the data! We design the kernel, which results in $p(w)$, and then draw samples from there.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 17 — March 23, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Modern Paradigm Shift

The modern approach has two steps:

1. learn a feature map;
2. learn a classifier based on the feature map.

In the past, we would just learn the classifier directly, assuming the features are good.

> (i) **Representation learning:** $\Phi : \mathcal{X} \to \mathbb{R}^\gamma$.
>
> (ii) **Linear probe:** $f(x) = w^T \Phi(x)$.

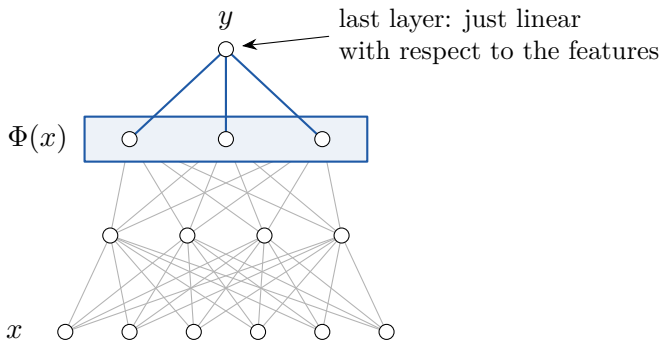

Figure 48: A deep network as a feature map $\Phi(x)$ followed by a linear last layer.

2 Infinitely Wide Networks as Gaussian Processes

Recall that an RKHS is "good" for f^* when $\|f^*\|_{\mathcal{H}_K}$ is small.

Consider a DNN with L layers. Let $X^{[0]} = x$, and let $z^{[l]} \in \mathbb{R}^{n_l}$, where n_l is the number of units in the l-th layer. Then

$$z_j^{[l]} = b_j^{[l]} + \sum_{i=1}^{n_{l-1}} w_{ji}^{[l]} X_i^{[l-1]},$$
$$X_j^{[l]} = \phi(z_j^{[l]}).$$

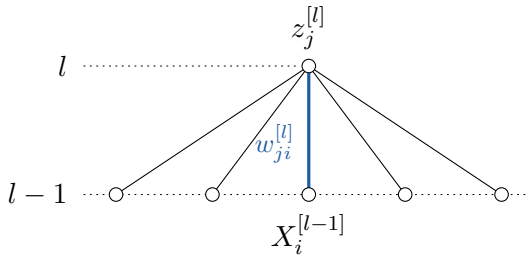

Figure 49: Unit j of layer l receives input $X_i^{[l-1]}$ from each unit i of layer $l-1$, with weight $w_{ji}^{[l]}$.

2.1 Random initialization

Initially, all weights are Gaussian:

$$b_j^{[l]} \sim N(0, \sigma_b^2), \qquad w_{ji}^{[l]} \sim N(0, \sigma_w^2).$$

How does this DNN behave when initialized like this? Let's look at the first layer:

$$z_j^{[1]} = \underbrace{b_j^{[1]} + \sum_{i=1}^{d} w_{ji}^{[1]} X_i}_{\text{linear combination of Gaussians}}$$

We have $\mathbb{E}[z_j^{[1]}] = 0$, since all components have mean 0, and

$$\mathrm{Var}[z_j^{[1]}] = \sigma_b^2 + \sigma_w^2 \, \|x\|_2^2$$

(a sum of variances).

What is the covariance $\mathrm{Cov}(z_j^{[1]}(x), z_j^{[1]}(x'))$? It is

$$\mathrm{Cov}\Big(b_j^{[1]} + \sum_i w_{ji}^{[1]} X_i,\ b_j^{[1]} + \sum_i w_{ji}^{[1]} X_i'\Big) = \sigma_b^2 + \sigma_w^2 \frac{x^T x'}{d}.$$

We can think of this as a linear kernel $K^{[1]}(x, x')$. So

$$(z_j^{[1]}(x_1), z_j^{[1]}(x_2), \ldots, z_j^{[1]}(x_n)) \sim N\big(0, (K^{[1]}(x_i, x_k))_{i,k}\big),$$

that is,

$$z_j^{[1]} \sim \mathrm{GP}\big(0, K^{[1]}\big).$$

2.2 Deeper layers

Now $X_j^{[1]} = \phi(z_j^{[1]})$. We cannot say that $z_j^{[2]}(x)$ is Gaussian:

$$z_j^{[2]}(x) = b_j^{[2]} + \sum_{i=1}^{n_1} w_{ji}^{[2]} X_i^{[1]}(x).$$

But suppose $n_1 \to \infty$. More generally, with

$$z_j^{[l]}(x) = b_j^{[l]} + \sum_{i=1}^{n_{l-1}} w_{ji}^{[l]} X_i^{[l-1]}(x),$$

let $n_{l-1} \to \infty$. Then

$$\begin{aligned}\mathrm{Cov}(z_j^{[l]}(x), z_j^{[l]}(x')) &= \mathbb{E}\Big[\big(b_j^{[l]}\big)^2\Big] + \sum_{i=1}^{n_{l-1}} \mathbb{E}\Big[\big(w_{ji}^{[l]}\big)^2\Big]\, \mathbb{E}\Big[X_i^{[l-1]}(x)\, X_i^{[l-1]}(x')\Big] \\ &= \sigma_b^2 + \sigma_w^2\, \mathbb{E}\Big[X_i^{[l-1]}(x)\, X_i^{[l-1]}(x')\Big].\end{aligned}$$

Here

$$\mathbb{E}\Big[X_i^{[l-1]}(x)\, X_i^{[l-1]}(x')\Big] = \mathbb{E}\Big[\phi\big(z_i^{[l-1]}(x)\big)\, \phi\big(z_i^{[l-1]}(x')\big)\Big], \qquad z_i^{[l-1]} \sim \mathrm{GP}\big(0, K^{[l-1]}\big),$$

and we can use the mean μ and covariance σ of this GP to represent any function of it. So

$$\mathbb{E}\Big[X_i^{[l-1]}(x)\, X_i^{[l-1]}(x')\Big] = F\big(K^{[l-1]}(x,x),\ K^{[l-1]}(x',x'),\ K^{[l-1]}(x,x')\big).$$

Therefore

$$K^{[l]}(x,x') = \mathrm{Cov}(z_j^{[l]}(x), z_j^{[l]}(x')) = \sigma_b^2 + \sigma_w^2\, F\big(K^{[l-1]}(x,x),\ K^{[l-1]}(x',x'),\ K^{[l-1]}(x,x')\big).$$

In the limit of infinite width (number of neurons), the kernel can be written as above, and

$$z_j^{[l]}(x) \sim \mathrm{GP}\big(0, K^{[l]}\big).$$

So all units are given by a Gaussian process.

To see why the expectation is just a function F of the covariances: if

$$(z_1, z_2) \sim N\left(0, \begin{pmatrix} K_{11} & K_{12} \\ K_{12} & K_{22} \end{pmatrix}\right),$$

then

$$\mathop{\mathbb{E}}_{z_1,z_2}\big[\phi(z_1)\, \phi(z_2)\big] = F(K_{11}, K_{22}, K_{12}),$$

which is just a function of the covariance!

Remark 2. An infinitely wide DNN is just a Gaussian process.

You could take

$$\Phi(x) = z^{[L]}(x),$$

that is, freeze all but the last layer and use that as the features. But this is not going to perform well. This theory explains that a network with randomly initialized weights is just a GP.

In practice, the CLT kicks in quite early, so we don't actually need infinite width.

The main problem is that this doesn't capture what we do in practice: we train our weights, we don't use random weights.

3 The Neural Tangent Kernel

Suppose we have a large (in θ) neural network $f_\theta(x)$, whose output is the overall prediction of the DNN. Suppose θ and θ_0 are not very far apart in terms of individual coordinates, although their distance can be large. Then

$$f_\theta(x) \approx f_{\theta_0}(x) + \nabla_\theta f_\theta(x)\big|_{\theta=\theta_0} \cdot (\theta - \theta_0),$$

with feature map

$$\phi(x) = \nabla_\theta f_\theta(x)\big|_{\theta=\theta_0}.$$

The idea is that we can linearize along each individual coordinate. This gives the kernel

$$K(x, x') = \langle \nabla f_{\theta_0}(x), \nabla f_{\theta_0}(x') \rangle,$$

called the *neural tangent kernel* (NTK). For parameters $\hat{\theta}$,

$$K_{\mathrm{NTK},\hat{\theta}}(x, x') = \langle \nabla f_{\hat{\theta}}(x), \nabla f_{\hat{\theta}}(x') \rangle.$$

For the DNN, suppose

$$\min_{l \in [L]} n_l \gtrsim \log(1/\delta).$$

Suppose that after T steps of training, for x, x', you have

$$\left| K_{\mathrm{NTK},\hat{\theta}_T}(x, x') - K_{\mathrm{NTK},\theta_0}(x, x') \right| \le \varepsilon \quad \text{with probability} \ge 1 - \delta.$$

This is saying that the DNN is roughly linear, using some feature map, and the feature map is $\nabla_\theta f_\theta(x)$. But it results in things that are all close to each other.

3.1 Lazy training

The actual issue is that the learning rate is too small. Let

$$R(f_\theta) = \mathbb{E}[\mathrm{loss}(f_\theta(x), y)], \qquad R_\alpha(\theta) = \frac{1}{\alpha^2} R(\alpha f_\theta),$$

the risk of the DNN scaled by α. Similarly, for the linearized model,

$$R_\alpha^{\mathrm{lin}}(\theta) = \frac{1}{\alpha^2} R(\alpha f_\theta^{\mathrm{lin}}), \qquad f_\theta^{\mathrm{lin}}(x) = f_{\theta_0}(x) + \nabla_\theta f_{\theta_0}(x) \cdot (\theta - \theta_0).$$

The gradient flows are

$$\frac{d\theta_\alpha}{dt} = -\nabla R_\alpha(\theta), \qquad \text{and, linearized,} \qquad \frac{d\theta_\alpha^{\mathrm{lin}}}{dt} = -\nabla R_\alpha^{\mathrm{lin}}(\theta).$$

They showed that, in continuous time t,

$$\sup_{t \in [0,T]} \|\theta_\alpha(t) - \theta_0\| = O(1/\alpha),$$

where θ_0 is the initial parameter. (When α is large, it is as if the step size is small.) And

$$\sup_{t \in [0,T]} \left\| \theta_\alpha^{\mathrm{lin}}(t) - \theta_0 \right\| = O(1/\alpha^2).$$

Why doesn't the NTK work? Because once your step size is large enough, you leave the region where the model is linear, so it does poorly. So early on in the process it behaves like a linear model, but after that it does not. This was a nail in the coffin for it.

3.2 Approximation power of the NTK

If you use an NTK kernel whose dimensionality is $N \asymp d^l$, and the data lies on the scaled unit sphere

$$\mathcal{X} = S^{d-1}(\sqrt{d}),$$

then the approximation error is $\asymp$ that of degree-l polynomial regression.

So if you choose d^l random basis functions, you do no better than degree-l polynomial regression, with the number of samples $n \asymp d^l$.

Lecture 18 — March 28, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Dimension Reduction (Guest Lecture)

We want to map data points to a lower-dimensional space:

$$x_1, \ldots, x_n \in \mathbb{R}^d \xrightarrow{T} y_1, \ldots, y_n \in \mathbb{R}^m.$$

2 Principal Component Analysis (PCA)

PCA gives a linear T (this is both an assumption and a limitation).

Say $X \sim D$ on $\mathbb{R}^d$. Let

$$\mathcal{L}_k = \{\text{all } k\text{-dimensional linear subspaces}\}.$$

The *k-th principal subspace* is

$$\ell_k := \arg\min_{\ell \in \mathcal{L}_k} \mathop{\mathbb{E}}_{X \sim D} \Big(\min_{y \in \ell} \left\| \tilde{X} - y \right\|^2 \Big),$$

where

$$\tilde{X} = X - \mathbb{E}[X]$$

centers X.

2.1 How to do PCA?

Let

$$\Sigma = \mathbb{E}[X - \mathbb{E}\,X]\,[X - \mathbb{E}\,X]^T,$$

with eigendecomposition

$$\Sigma = E \Lambda E^T = \sum_{j=1}^{d} \lambda_j e_j e_j^T, \qquad (\lambda_1 \geq \cdots \geq \lambda_d).$$

Theorem 5. *The k-th principal subspace is* $\operatorname{span}(\{e_1, \ldots, e_k\})$. *Moreover,*

$$T_k(X) = \mu + \sum_{j \leq k} \beta_j e_j, \qquad \beta_j = \langle X - \mu, e_j \rangle .$$

The "risk" is

$$R(k) := \mathbb{E} \left\| X - T_k(X) \right\|^2 = \sum_{j=k+1}^{d} \lambda_j.$$

If $k = d$, then we are not doing any dimension reduction.

Proof. Use Courant–Fisher. ■

Alternate interpretation. This T_k minimizes

$$\mathbb{E}\,\|X_i - \mu - AY_i\|^2$$

over $A \in \mathbb{R}^{d\times k}$ and $Y_i \in \mathbb{R}^k$. At the minimum, $\mu + AY_i$ is $T_k(X_i)$, and Y_i is the vector of coefficients $\beta_i \in \mathbb{R}^k$.

Lemma 1. *Let $Y_i = e_i^T(X - \mu)$. Then*

(i) $\mathbb{E}\,Y = 0$ *and* $\operatorname{var}(Y) = \Lambda$*;*

(ii) $X = \mu + EY$*, where E is the change-of-basis matrix;*

(iii) $\displaystyle\sum_{i=1}^{d} \operatorname{var}(Y_i) = \Sigma_{11} + \cdots + \Sigma_{dd}$.

2.2 "Duality" of the transformation

The ratio

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

is the percentage of variance explained by Y.

How to choose k: find when this ratio is $< 10\%$.

2.3 Sample setting

Use

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)(X_i - \mu)^T, \qquad \text{where } \mu = \frac{\sum_i X_i}{n}.$$

Theorem 6.

1. *With n samples, $\left\|\hat{\Sigma} - \Sigma\right\|_2 = O(\frac{1}{\sqrt{n}})$, where $\|\cdot\|_2$ is the operator norm.*
2. $\max_j |\lambda_j(\hat{\Sigma}) - \lambda_j(\Sigma)| \le \left\|\hat{\Sigma} - \Sigma\right\|_2$*, where $\lambda_j(\cdot)$ is the j-th eigenvalue.*
3. $\|\hat{e}_j - e_j\| \le 2^{3/2}\,\dfrac{\left\|\hat{\Sigma} - \Sigma\right\|_2}{\min(\cdots)}$.

2.4 The PCA algorithm

1. Center X.
2. Find $\hat{\Sigma}$.
3. Compute the λ_i and e_i.
4. Pick k.
5. Project the X_i.

PCA is essentially the SVD of the data matrix.

3 Multi-Dimensional Scaling (MDS)

Find "embeddings" $z_i = T(x_i)$, where $T : \mathbb{R}^d \to \mathbb{R}^k$, that minimize the loss

$$L = \sum_{\substack{1 \le i \le n \\ 1 \le j \le n}} \left(\|x_i - x_j\|^2 - \|z_i - z_j\|^2 \right)^2.$$

(This is almost like JL, but not exactly.)

Lemma 2. *If T is forced to be linear, this is PCA.*

4 Kernel PCA

Idea: do PCA on $\phi(x)$, not x. The "kernel" is

$$K(i,j) = \langle \phi(x_i), \phi(x_j) \rangle ,$$

so we perform PCA on $\phi(x)$.

4.1 Method

Assume $\phi(x)$ is already centered. Let

$$C_\phi := \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \phi(x_i)^T$$

(we do not actually compute this). Take an eigenvalue–eigenvector pair (λ, v) of C_ϕ:

$$C_\phi v = \lambda v.$$

Observation. v lies in the span of $\phi(x_1), \ldots, \phi(x_n)$. Why? We have

$$\lambda v = C_\phi v = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \underbrace{\phi(x_i)^T v}_{\in \mathbb{R}} = \sum_{i=1}^{n} \alpha_i \phi(x_i), \qquad \text{where } \alpha_i = \frac{1}{n} \langle \phi(x_i), v \rangle .$$

So now the goal is to recover the weights α.

4.2 How to find α?

It turns out that

$$K\vec{\alpha} = n\lambda\vec{\alpha}, \qquad \text{where } K = \begin{bmatrix} & \vdots & \\ \cdots & \langle\phi(x_i), \phi(x_j)\rangle & \cdots \\ & \vdots & \end{bmatrix}$$

is the *Gram matrix*. This says that $\vec{\alpha}$ is an eigenvector of K with eigenvalue $n\lambda$.

Proof. We have

$$\sum_{i=1}^{n} \alpha_i \phi(x_i) = C_\phi v.$$

Multiply by $\lambda\phi(x_k)$:

$$\lambda \sum_i \alpha_i \langle\phi(x_i), \phi(x_k)\rangle = \lambda \langle\phi(x_k), C_\phi v\rangle .$$

The left-hand side is $\lambda(K\alpha)_k$. For the right-hand side, use $v = \frac{1}{\lambda}\sum_i \alpha_i\phi(x_i)$:

$$\lambda \langle\phi(x_k), C_\phi v\rangle = \frac{\lambda}{n} \sum_j K_{kj} \langle\phi(x_j), v\rangle = \frac{1}{n} \sum_j K_{kj} \sum_i K_{ji}\alpha_i = \frac{1}{n}(K^2\alpha)_k.$$

This is the k-th row of

$$\lambda n K\alpha = K^2\alpha \quad \Longrightarrow \quad \lambda n\alpha = K\alpha.$$

■

Also,

$$\langle v, v\rangle = 1 \quad \Longrightarrow \quad \langle\alpha, \alpha\rangle = \frac{1}{\lambda}.$$

Now we have α and λ, but no ϕ.

4.3 Kernel PCA on a new x

We want to compute, using $v = \sum_i \alpha_i\phi(x_i)$,

$$\langle v, \phi(x)\rangle = \sum_i \alpha_i \langle\phi(x_i), \phi(x)\rangle = \sum_i \alpha_i \underbrace{k(x_i, x)}_{\text{just the kernel}} .$$

We can do it with just the kernel!

In practice, people try using Gaussian kernels; this works very well, much better than vanilla PCA.

4.4 Centering

To center K, we work with

$$\tilde{K} = K - \mathbf{1}_n K - K\mathbf{1}_n + \mathbf{1}_n K \mathbf{1}_n, \qquad \text{where } \mathbf{1}_n = \frac{1}{n}\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}.$$

4.5 The kernel PCA algorithm

1. Center K.
2. Diagonalize K.
3. Scale α.
4. Compute the "projections" of x using the kernel trick.

Remark 3. From a paper by Tengyu Ma: modern contrastive learning is essentially just kernel PCA.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 19 — March 30, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Dimension Reduction

So far:

1. Principal Component Analysis (PCA).
2. Multi-dimensional scaling (MDS), with loss

$$L = \sum_{i,j \leq n} \underbrace{\|x_i - x_j\|^2}_{d_{ij}^2} - \|y_i - y_j\|^2 .$$

Theorem 7. *Let $y = T(x)$. Minimizing L is equivalent to PCA when T is a projection.*

(So we don't have the $-\infty$ issue from previously.)

3. Kernel PCA. For isotropic kernels, kernel PCA is MDS on

$$d_{ij} = \sqrt{1 - 2k(x_i, x_j)}.$$

2 Local Linear Embeddings (LLE)

Input: $x_1, \ldots, x_n$.

Goal: dimension reduction.

Intuition: preserve the "local geometry".

Step 1. For each point x_i, find the k nearest neighbors. Time: $O(n^2 d)$.

Step 2. Find a weight matrix $W \in \mathbb{R}^{n \times n}$ which minimizes

$$\sum_i \left\| x_i - \sum_{j \leq n} W_{ij} x_j \right\|^2$$

subject to

$$\sum_j W_{ij} = 1 \quad \forall i,$$

and $W_{ij} = 0$ if j is not one of the k neighbors of i. This characterizes x_i as a function of its neighbors. Time: $O(nk^3)$.

Step 3. Find $y_1, \ldots, y_n \in \mathbb{R}^m$ that minimize

$$\psi(Y) = \sum_{i \le n} \left\| y_i - \sum_{j \le n} W_{ij} y_j \right\|^2$$

subject to

$$\sum_i y_i = 0, \qquad \frac{1}{n} \sum_i y_i y_i^T = I.$$

To do this, find the lowest $(m+1)$ eigenvectors of $(I-W)(I-W)^T$. Time: $O(n^3) \to O(mn^2)$.

Assumption: the graph embedded by W is "connected".

Then $y_1, \ldots, y_m$ are the lowest eigenvectors of $(I-W)(I-W)^T$ with nonzero eigenvalues.

3 Isomap

Step 1: neighborhood graph. Two options:

1. Nearest neighbors graph: $x_i \sim x_j$ if $j \in N_k(i)$ or $i \in N_k(j)$.
2. ε-neighborhood graph: $x_i \sim x_j$ if $\|x_i - x_j\| \le \varepsilon$.

Step 2. Define d_{ij} as the distance between i and j in the unweighted graph.

Step 3. Do MDS with these d_{ij}'s.

4 Laplacian Eigenmaps (Belkin–Niyogi 2003)

1. Find distances W: $W_{ij} = d(x_i, x_j)$, e.g. $e^{-\|x_i - x_j\|^2/2}$.
2. $L = D - W$, where $D = \mathrm{diag}(\sum_j W_{ij})$.
3. Find the lowest m eigenvectors $y_1, \ldots, y_m$ of L with non-zero eigenvalues.
4. Map $x_i \xrightarrow{T} (y_{1i}, y_{2i}, \ldots, y_{mi})$.

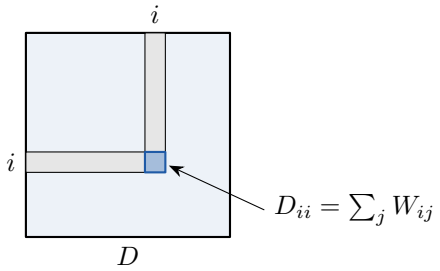

Figure 50: The degree matrix D is diagonal, with entry (i,i) equal to the sum of row i of W.

Why the low eigenvalues? We are trying to minimize $\psi(Y)$.

5 Diffusion Maps (Coifman–Lafon 2006)

Do MDS on the new distance function

$$D_t(x,y) = \sqrt{\sum_j \lambda_j^{2t}(\psi_j(x) - \psi_j(y))^2},$$

where λ_j, ψ_j are the eigenvalues and eigenvectors of L^t. Here

$$L = D^{-1}W.$$

(Intuition: the number of steps to go from x to y.)

In the continuous setting:

- L becomes the transition matrix of a diffusion process;
- D_t becomes the diffusion distance.

6 Random Projections

Let $x_1, \dots, x_n \in \mathbb{R}^d$, and

$$T(x_i) = \frac{Sx_i}{\sqrt{m}},$$

where $S \in \mathbb{R}^{m \times d}$ has $N(0,1)$ entries. Write $y_i = T(x_i)$.

Theorem 8 (Johnson–Lindenstrauss)**.** *Let $\varepsilon > 0$ and $m > \dfrac{32 \log n}{\varepsilon^2}$. Then with probability*

$$1 - e^{-m\varepsilon^2/16} \geq 1 - \frac{1}{n^2},$$

we have

$$(1-\varepsilon)\|x_i - x_j\|^2 \leq \|y_i - y_j\|^2 \leq (1+\varepsilon)\|x_i - x_j\|^2.$$

Proof. We have

$$\frac{\|y_j - y_k\|^2}{\|x_j - x_k\|^2} - 1 = \frac{1}{m}\sum_{i=1}^m Z_i^2 - 1, \qquad \text{where } Z_i = \left\langle S_i, \frac{x_j - x_k}{\|x_j - x_k\|} \right\rangle \sim N(0,1)$$

and S_i is the i-th row of S. Use the MGF to bound this:

- Bound the different parts.
- Use a union bound over all pairs x_i, x_j from the n points. ∎

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 20 — April 4, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Clustering

Given points $\{X_i\}_{i=1}^n \sim P$, we want a clustering map

$$C : \{1, \ldots, n\} \to \{1, \ldots, k\}.$$

Let

$$\mathcal{C} = \{c_1, \ldots, c_k\}$$

be a set of *codewords*. Given a sample X_i, we want to communicate the codeword closest to X_i:

$$\Pi_{\mathcal{C}}(x) = \arg\min_{c \in \mathcal{C}} \|x - c\| .$$

This is *quantization*: we want to represent x with a finite set of values.

The empirical risk is

$$\hat{R}(\mathcal{C}) = \frac{1}{n} \sum_{i=1}^{n} \|X_i - \Pi_{\mathcal{C}}(X_i)\|^2 ,$$

and we solve

$$\min_{\mathcal{C}} \hat{R}(\mathcal{C})$$

(the standard approach in signal processing). Equivalently,

$$\min_{\mathcal{C}} \min_{g:[n]\to[k]} \sum_{i=1}^{n} \left\| X_i - c_{g(i)} \right\|^2 ,$$

where $g(i)$ is the codeword we assign to X_i.

1.1 Algorithm

Alternate between optimizing $\mathcal{C}$ and g, holding the other fixed.

- With $\mathcal{C}$ fixed,
$$g(i) = \arg\min_{j \in [k]} \|X_i - c_j\| .$$
- With g fixed, the objective is
$$\sum_{i=1}^{n} \sum_{j=1}^{k} \|X_i - c_j\|^2 \, \mathbb{I}[g(i) = j],$$
and
$$\arg\min_{c_j} \sum_{i:g(i)=j} \|X_i - c_j\|^2 = \frac{1}{|\{i : g(i) = j\}|} \sum_{i:g(i)=j} X_i.$$

1.2 Population risk

$$R(\mathcal{C}) = \mathop{\mathbb{E}}_{X\sim P} \|X - \Pi_{\mathcal{C}}(X)\|^2 .$$

So this allows us to summarize the distribution using multiple vectors. We have

$$|\hat{R}(\hat{\mathcal{C}}) - R(\mathcal{C}^*)| \le \sqrt{\frac{kd\log n}{n}},$$

where

$$\hat{\mathcal{C}} \in \arg\min_{\mathcal{C}} \hat{R}(\mathcal{C}), \qquad \mathcal{C}^* = \arg\min_{\mathcal{C}} R(\mathcal{C}).$$

So this perspective is more general than clustering: it is a form of quantization.

1.3 k-means++

1. $c_1 \sim \text{Unif}[X_1, \dots, X_n]$, and $\mathcal{C} = \{c_1\}$.
2. For $j = 2, \dots, k$:
   (a) For $i \in [n]$:

$$D(X_i) = \min_{c\in\mathcal{C}} \|X_i - c\| = \|X_i - \Pi_{\mathcal{C}}(X_i)\|, \qquad p_i = \frac{D^2(X_i)}{\sum_{i'\in[n]} D^2(X_{i'})}.$$

   (b) $p = (p_1, \dots, p_n)$ and $i_j \sim \text{Mult}(p)$.
   (c) $c_j = X_{i_j}$, and add c_j to $\mathcal{C}$.

So we sample proportionally to the distance: we want the new codeword to be further away.

With $\mathcal{C}_{++}$ the codewords from k-means++,

$$\mathbb{E}[\hat{R}(\mathcal{C}_{++})] \le (\log k) \min_{\mathcal{C}} \hat{R}(\mathcal{C}).$$

So even if you just initialize and stop, the result is already pretty good.

1.4 Choosing k

Let

$$R_k = \min_{\mathcal{C}:|\mathcal{C}|=k} R(\mathcal{C}).$$

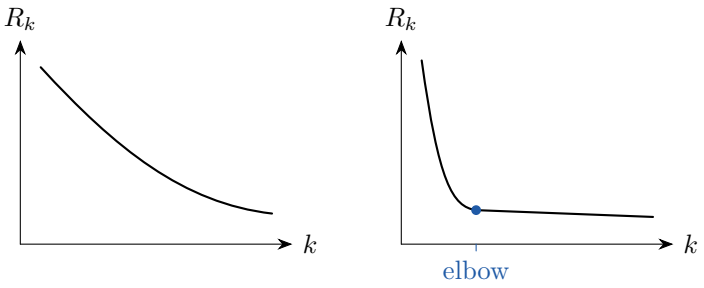

Figure 51: Left: R_k decreases as k grows. Right: a curve with a clear elbow.

How do we choose k? The typical way is to see if there is an elbow (in many real applications you won't really see this). Another choice is

$$k_\alpha = \min\Big\{k \;\Big|\; \frac{R_k - R_{k+1}}{\sigma^2} \le \alpha\Big\}, \qquad \sigma^2 = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

1.5 Stability

Suppose $k = 2$ but there are 3 actual clusters.

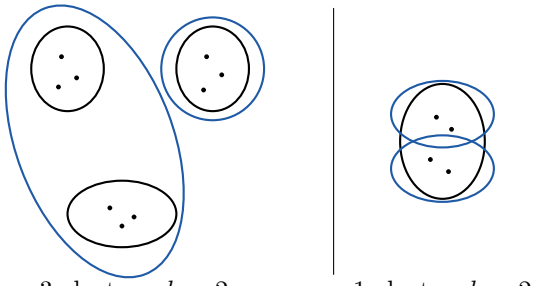

Figure 52: k-means with $k = 2$ (accent): with three true clusters it must merge two of them, and with one true cluster it must split it.

k-means can be unstable if k is too large or too small.

Suppose we have $\{X_1, \ldots, X_{2n}\}$. Split it into multiple groups

$$Y^j = \{Y_1^j, \ldots, Y_n^j\}, \qquad Z^j = \{Z_1^j, \ldots, Z_n^j\}.$$

Let

$$\hat{\Omega}(k) = \frac{1}{n}\sum_{j=1}^{n} s(\mathcal{C}_k(Y^j), \mathcal{C}_k(Z^j)),$$

where s is a similarity. If the similarity is high, the clusters are not changing.

For the similarity, let $\psi : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$ (whether two points are in the same cluster), and use

$$\sum_{\substack{z_s \neq z_t \\ z_s, z_t \in Z^j}} \|\psi_{Z^j}(z_s, z_t) - \psi_{Y^j}(y'_s, y'_t)\|, \qquad \text{where } y'_s = \arg\min_{y_s \in Y^j} \|y_s - z_s\|.$$

Goal: find the most stable clustering for $k = 1$.

But stability is not a silver bullet. Consider:

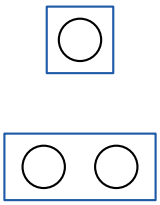

Figure 53: With $k = 2$ the clustering (accent boxes) is still stable, but there are really 3 clusters.

Here $k = 2$ is still stable, but there are really $k = 3$ clusters!

High dimensions are bad for distances. The distance becomes meaningless in high d. It's just a single value that everything was summarized into, so there is a huge loss of information! So we must do dimension reduction, or use a better notion of distance.

1.6 Silhouette score

Stability is good, but not sufficient. Define the *intra-cluster distance*

$$IC[i] = \sum_{i':C_{i'}=C_i} \|X_i - X_{i'}\|$$

(we want this small), and the *cross-cluster distance*

$$XC[i] = \min_{j \neq C[i]} \sum_{i':C[i']=j} \|X_i - X_{i'}\|$$

(we want this large).

Definition (Silhouette score)**.**

$$\frac{1}{n} \sum_{i=1}^{n} \frac{XC[i] - IC[i]}{\max(XC[i], IC[i])}.$$

We want to maximize it.

2 Mixture Models

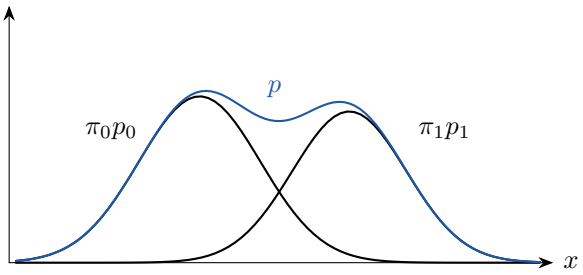

Figure 54: A mixture p (accent) is the sum of its weighted components $\pi_0 p_0$ and $\pi_1 p_1$.

$$p = \pi_0 p_0 + \pi_1 p_1.$$

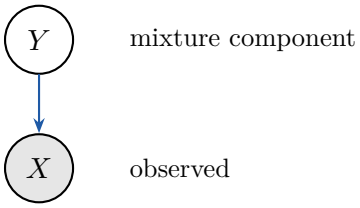

Figure 55: The mixture model as a graphical model: a latent component Y generates the observation X.

$$P(x) = \sum_{j=1}^{k} \underbrace{P(Y=j)}_{\pi_j} \underbrace{P(x \mid Y=j)}_{p_j(x)} = \sum_{j=1}^{k} \pi_j \, p_j(x).$$

Then, by Bayes' rule,

$$P(Y=j \mid x) = \frac{P(x \mid Y=j)\, P(Y=j)}{\sum_{j' \in [k]} P(x \mid Y=j')\, P(Y=j')}.$$

(Clustering is related to mixture models.)

In

$$P(x) = \sum_j P(Y=j)\, P(x \mid Y=j),$$

the component densities $P(x \mid Y=j)$ allow you to get $P(Y \mid x)$. In general, mixture models are not identifiable (identifiability would mean knowing which j a point came from). If you are given assumptions on the $P(x \mid Y=j)$, the model could be identifiable.

3 Score Matching

$$\frac{1}{2} \mathop{\mathbb{E}}_{p_{\text{data}}} \left[\| s_\theta(x) - \nabla_x \log p_{\text{data}}(x) \|_2^2 \right] = \frac{1}{2} \mathop{\mathbb{E}}_{p_{\text{data}}} \left[\| s_\theta(x) \|_2^2 \right] + \frac{1}{2} \mathop{\mathbb{E}}_{p_{\text{data}}} \left[\| \nabla_x \log p_{\text{data}}(x) \|_2^2 \right] - \mathop{\mathbb{E}}_{p_{\text{data}}} \left[\langle s_\theta(x), \nabla_x \log p_{\text{data}}(x) \rangle \right].$$

Integration by parts: with

$$v = p(x), \quad dv = \frac{\partial}{\partial x_i} p(x)\, dx, \qquad u = f_i(x), \quad du = \frac{\partial}{\partial x_i} f_i(x)\, dx,$$

we get

$$\int f_i(x) \frac{\partial}{\partial x_i} p(x)\, dx = p(x) f_i(x) \Big|_{-\infty}^{\infty} - \int p(x) \frac{\partial}{\partial x_i} f_i(x)\, dx.$$

Assumption: $p(x) f_i(x) \to 0$ as $|x| \to \infty$.

Apply this to the cross term, with $f = s_\theta$ and $p = p_{\text{data}}$:

$$\begin{aligned}
\mathop{\mathbb{E}}_{p_{\text{data}}}\Big[\langle s_\theta(x), \nabla_x \log p_{\text{data}}(x)\rangle\Big] &= \sum_i \int s_{\theta,i}(x)\frac{\partial}{\partial x_i} p_{\text{data}}(x)\,dx \\
&= -\sum_i \int p_{\text{data}}(x)\frac{\partial}{\partial x_i} s_{\theta,i}(x)\,dx \\
&= -\mathop{\mathbb{E}}_{p_{\text{data}}}\Big[\operatorname{tr}\big(\nabla_x s_\theta(x)\big)\Big].
\end{aligned}$$

The middle term does not depend on θ, so the objective needs only s_θ.

Lecture 21 — April 6, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Mixture Models

In a mixture model there is a latent class Y, and we observe only X:

$$P(X) = \sum_j \underbrace{P(X \mid Y = j)\, P(Y = j)}_{P(Y|X)}.$$

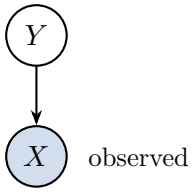

Figure 56: The mixture model as a graphical model: the latent class Y generates the observed X.

1.1 Identifiability

Consider the mixture of two Gaussians

$$p(x) = \pi\, \phi(x; \mu_1, 1) + (1 - \pi)\, \phi(x; \mu_2, 1).$$

Say $\mu_1 = \mu_2 = \mu$. Then

$$p(x) = \pi\, \phi(x; \mu, 1) + (1 - \pi)\, \phi(x; \mu, 1) = \phi(x; \mu, 1).$$

So in this case π is not identifiable.

Definition (Identifiability)**.** A distribution is *identifiable* if

$$P_{\theta_1}(x) = P_{\theta_2}(x) \implies \theta_1 = \theta_2.$$

For particular regions of the parameter space, the model becomes non-identifiable (e.g. if $\pi = 0$ or $\pi = 1$). This is called *local non-identifiability*. In this case we can have

$$\left\| \hat{\theta} - \theta^* \right\|_2 \asymp n^{-1/4}.$$

For example, take

$$p(x) = \frac{2}{3}\, \phi(x; -\theta, 1) + \frac{1}{3}\, \phi(x; 2\theta, 1).$$

The Fisher information is

$$I(\theta) = \mathop{\mathbb{E}}_{x \sim p_\theta}\left[\frac{\partial^2 p_\theta(x)}{\partial \theta^2}\right].$$

If $I(\theta^*) = 0$, then the second-order term is 0. This means the higher-order terms won't be a lot smaller, and so we need to care about higher orders. Usually in MLE we only care about the first and second orders.

1.2 Modes of a mixture

Consider

$$p(x) = \pi\, \phi(x; \mu_1, \sigma^2) + (1 - \pi)\, \phi(x; \mu_2, \sigma^2).$$

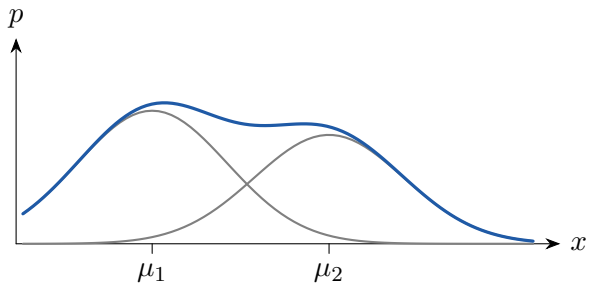

Figure 57: A mixture of two Gaussians (accent) and its two weighted components (gray).

You only get a bimodal distribution if $|\mu_1 - \mu_2| > 2\sigma$.

But in general the interaction between the number of modes and k is complex: the mixture could have more modes than k.

2 Level Set Clustering

So far we've talked about

1. vector quantization;
2. mixture modeling.

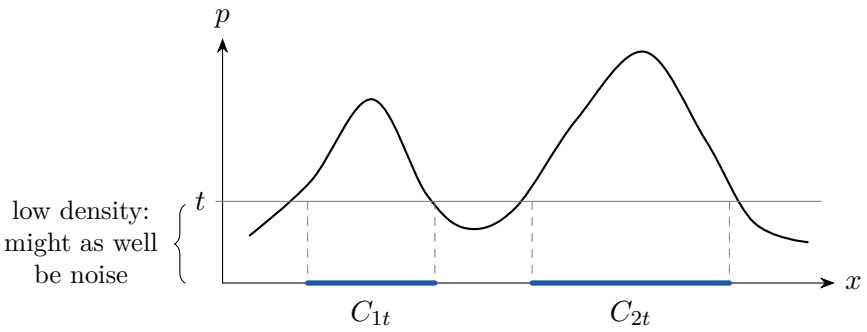

Figure 58: The level set at height t splits into the connected components C_{1t} and C_{2t}; the region of density below t is ignored.

Don't worry about the low-density region: it might as well be noise.

Definition (Level set)**.** The *level set* at level t is

$$\mathcal{L}_t = \{x \mid p(x) > t\} = \bigcup_{\gamma=1}^{R_t} C_{\gamma t},$$

a union of multiple connected components $C_{\gamma t}$. The clusters at level t are

$$\mathcal{C}_t = \{C_{\gamma t}\}_{\gamma=1}^{R_t}.$$

2.1 Estimating the level set

Given a density estimate $\hat{p}$, let

$$\hat{\mathcal{L}}_t = \{x \mid \hat{p}(x) > t\}.$$

Given samples $\{x_i\}_{i=1}^n$, keep the samples of high estimated density,

$$X_t = \{x_i \mid \hat{p}(x_i) > t\}.$$

So if $\hat{p}$ is like the number of neighbors, essentially we are saying that if a point has too few neighbors, we ignore it.

Then form a graph

$$G_t = (X_t, E_t), \qquad (x, x') \in E_t \iff \|x - x'\| \le \varepsilon.$$

That is, we connect points that are within distance ε of each other. The estimated clusters are

$$\hat{\mathcal{C}}_t = \text{the connected components of } G_t.$$

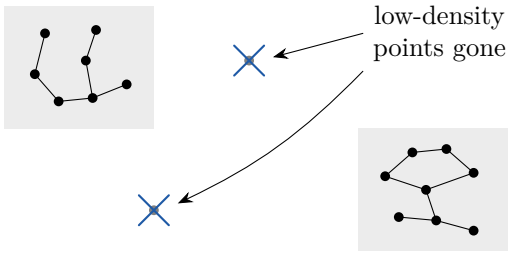

Figure 59: The graph G_t: low-density points are removed, points within distance ε are connected, and each connected component is a cluster.

2.2 Stability and Hausdorff distance

Even when the change in t is small, it could result in a big change of the level sets. We measure the distance between level sets with the Hausdorff distance.

Definition (Hausdorff distance)**.** The *Hausdorff distance* between sets U and V is

$$H(U,V) = \inf\{\varepsilon \mid U \subseteq V \oplus \varepsilon \ \text{ and } \ V \subseteq U \oplus \varepsilon\}, \qquad \text{where} \qquad U \oplus \varepsilon = \bigcup_{x \in U} B(x,\varepsilon).$$

We then have

$$H(\mathcal{L}_{t-\delta}, \mathcal{L}_{t+\delta}) \leq C\delta$$

and

$$H(\hat{\mathcal{L}}_t, \mathcal{L}_t) \asymp \sqrt{\frac{\log n}{n\varepsilon^d}}.$$

Remark 4. On smooth manifolds, the local distance is the same as the Euclidean distance.

3 Mode-Based Clustering

We go from distributions to clusters using modes.

Definition (Mode)**.** A *mode* is a point that is a local maximum of the density.

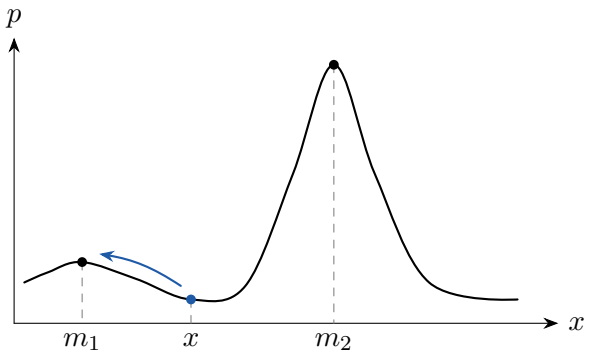

Figure 60: Following the gradient of the density from x leads uphill to the mode m_1.

How do we operationalize this? Define the curve π_x by

$$\pi_x(0) = x, \qquad \frac{d\pi_x(t)}{dt} = \nabla p(\pi_x(t)).$$

Then $\pi_x(t)$ is an *integral curve*, and

$$\lim_{t\to\infty} \pi_x(t) \equiv \text{dest}(x)$$

will be a mode of $p(x)$.

The discretized version is gradient ascent:

$$x^{(t+1)} = x^{(t)} + \eta\, \nabla p\big(x^{(t)}\big).$$

This leads you to a point where

$$\nabla p\big(\text{dest}(x)\big) = 0.$$

This could also be a saddle point instead of a maximum.

Assumption: the Hessian at stationary points is non-degenerate.

In high dimensions there can be exponentially many saddle points.

So define the modes $\{m_j\}_{j=1}^R$, and let

$$A_j = \{x \mid \text{dest}(x) = m_j\},$$

the *ascending manifold* of m_j. The collection

$$\{A_j\}_{j=1}^R \equiv \text{the } \textit{Morse complex}.$$

3.1 Sample version

Given samples $\{x_1, \ldots, x_n\}$, iterate, for each j,

$$x_j^{(t+1)} = \frac{\sum_{i=1}^n x_i\, K\big((x_j^{(t)} - x_i)/h\big)}{\sum_{i=1}^n K\big((x_j^{(t)} - x_i)/h\big)}.$$

Run this until $t \to \infty$. The estimated modes are

$$\mathcal{M} = \text{unique}\{x_i^{(\infty)}\}_{i=1}^n.$$

4 Hierarchical Clustering

Start with each point being its own cluster.

Graph-theoretic viewpoint: contract the edges of the complete graph in order of length.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 22 — April 11, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Online Learning

For rounds $t = 1, 2, \ldots, T$:

- the learner picks $\theta_t \in \Theta$;
- the environment chooses a risk $R_t : \Theta \to \mathbb{R}$.

The key idea is that the environment can give a different risk every day. The learner suffers the total risk

$$\sum_{t=1}^{T} R_t(\theta_t).$$

Let

$$\theta^* \in \arg\min_{\theta \in \Theta} R(\theta).$$

We compare against the best static action in hindsight.

Definition (Regret)**.**

$$\text{Regret} = \sum_{t=1}^{T} R_t(\theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^{T} R_t(\theta).$$

We want to get sublinear regret, $o(T)$: e.g. $\sqrt{T}$ or $\log T$.

If we make no assumptions on R_t, this is very hard to achieve.

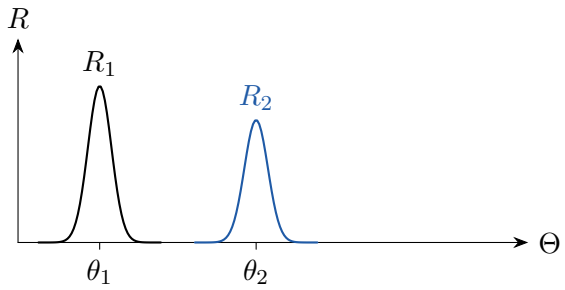

Figure 61: With no assumptions on R_t, the environment can put a sharp spike of risk exactly at the learner's choice θ_t.

2 Restricting the Adversary: Strongly Convex Losses

One way to restrict the adversary is to require strongly convex functions.

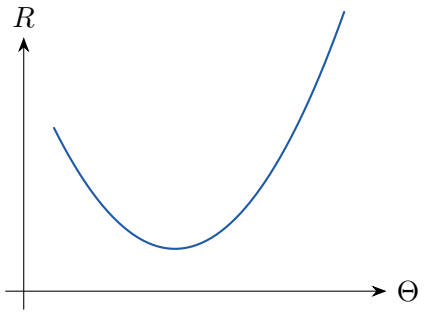

Figure 62: A strongly convex risk: it has curvature everywhere.

Definition (Strong convexity)**.** R is *strongly convex* if

$$\nabla^2 R(\theta) \succeq \gamma I.$$

It has some curvature, and can always be lower bounded by a quadratic:

$$R(\theta) \geq R(\theta_0) + \nabla R(\theta_0) \cdot (\theta - \theta_0) + \frac{\gamma}{2} \|\theta - \theta_0\|^2 .$$

If you have no curvature, you cannot lower bound the function by a quadratic.

How do we get sublinear regret if R_t is convex?

2.1 Myopic play

Myopic play optimizes for what happened recently: $\theta_{t+1} = \arg\min_{\theta \in \Theta} R_t(\theta)$.

Example 10. Suppose $\Theta = [-B, B]$, and

$$\begin{aligned} \theta_1 &= 1, & R_1(\theta) &= (\theta + 1)^2, \\ \theta_2 &= -1, & R_2(\theta) &= (\theta - 1)^2, \\ \theta_3 &= 1, & R_3(\theta) &= (\theta + 1)^2, \end{aligned}$$

and so on. Then the regret is

$$4T - 2T = 2T \neq o(T).$$

So myopic play still does not help: it results in linear regret.

2.2 Follow the Leader

Instead of being myopic, what if we consider all the previous rounds' losses?

Definition (Follow the Leader / Fictitious Play)**.** At round t, play

$$\theta_t = \arg\min_{\theta \in \Theta} \sum_{s=1}^{t-1} R_s(\theta).$$

If $\nabla^2 R_t \succeq \kappa I$, then

$$\text{Regret} = O\Big(\frac{\log T}{\kappa}\Big),$$

where the curvature κ is the strong convexity parameter of the loss.

2.3 Follow the Leader fails for linear losses

Let Θ be the simplex

$$\Theta = \{\theta \mid \theta_1 + \theta_2 = 1,\ \theta_1, \theta_2 \geq 0\}.$$

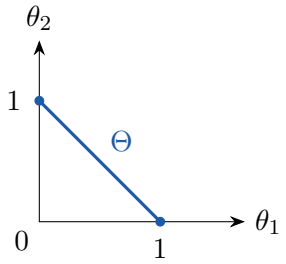

Figure 63: The parameter space Θ: the segment $\theta_1 + \theta_2 = 1$, $\theta_1, \theta_2 \geq 0$.

Take linear risks

$$R_t(\theta) = \langle a_t, \theta \rangle,$$

which are only convex, not strongly convex. Then

$$\sum_{s=1}^{t-1} R_s(\theta) = \sum_{s=1}^{t-1} \langle a_s, \theta \rangle = \langle \bar{a}_t, \theta \rangle = \underbrace{\bar{a}_{t,1}\theta_1 + \bar{a}_{t,2}\theta_2}_{\text{put weight on the smaller one}},$$

where $\bar{a}_t = \sum_{s=1}^{t-1} a_s$. So Follow the Leader wants to put all its weight on the coordinate which is smaller. The adversary responds:

$$\theta_t = (1,0) \implies a_t = (1,0), \qquad \theta_t = (0,1) \implies a_t = (0,1).$$

Then the learner suffers

$$\sum_t R_t(\theta_t) = T.$$

But for the best fixed action in hindsight,

$$\min_\theta \sum_{s=1}^{T} R_s(\theta) = \min_\theta \Big\langle \sum_{s=1}^{T} a_s, \theta \Big\rangle \leq \frac{T}{2},$$

by choosing the coordinate with the smaller total. So

$$\text{Regret} \geq \frac{T}{2}.$$

So if the loss is *only* convex, Follow the Leader does not work.

2.4 Follow the Regularized Leader

Definition (Follow the Regularized Leader (FTRL))**.** At round t, play

$$\theta_t = \arg\min_{\theta \in \Theta} \left\{ \sum_{s=1}^{t-1} R_s(\theta) + \eta\, R(\theta) \right\},$$

where the regularizer R is strongly convex. We want a healthy curvature.

Theorem 9. *Suppose the losses are L-Lipschitz,*

$$R_s(\theta) - R_s(\theta') \leq L \left\| \theta - \theta' \right\|,$$

and let

$$D = \sup_{\theta, \theta' \in \Theta} \left\| \theta - \theta' \right\|$$

be the diameter of Θ. Then FTRL has

$$\text{Regret} = O(LD\sqrt{T}).$$

3 Randomized Play

Genius idea: make θ_t a random variable. Let P be a distribution over Θ, and draw a random $\theta_t \sim P$. The expected total risk is

$$\mathbb{E}\left[\sum_{t=1}^{T} R_t(\theta_t) \right].$$

Against the best fixed distribution in hindsight,

$$\min_{P \in \Delta_\Theta} \mathop{\mathbb{E}}_{\theta \sim P} \left[\sum_{s=1}^{T} R_s(\theta) \right] = \min_{P \in \Delta_\Theta} \sum_{\theta \in \Theta} P(\theta) \sum_{s=1}^{T} R_s(\theta) \equiv \min_{\theta \in \Theta} \sum_{s=1}^{T} R_s(\theta).$$

So the static baseline is still the best: it always plays the best action.

Now the learner plays $P_t \in \Delta_\Theta$, with risk

$$R_t(P_t) = \underbrace{\sum_{\theta \in \Theta} P_t(\theta)\, R_t(\theta)}_{\text{linear, so convex}}.$$

Now your overall loss is linear, hence convex. Using FTRL, we get

$$\sum_{t=1}^{T} R_t(P_t) - \min_{\theta \in \Theta} \sum_{s=1}^{T} R_s(\theta) = o(T).$$

3.1 Multiplicative weights

Apply FTRL over distributions:

$$P_t = \operatorname*{arg\,min}_{P \in \Delta_\Theta} \Big\{ \sum_{s=1}^{t-1} R_s(P) + \eta\, R(P) \Big\}, \qquad R(P) = \sum_\theta P(\theta) \log P(\theta) \quad \text{(negative entropy)}.$$

This gives the update

$$P_{t+1}(\theta) \propto P_t(\theta) \exp\Big(-\frac{1}{\eta} R_t(\theta)\Big),$$

which is *multiplicative weights* (rediscovered every five years). Equivalently,

$$P_{t+1}(\theta) \propto \exp\left(-\frac{\sum_{s=1}^t R_s(\theta)}{\eta}\right),$$

which can be shown by induction, unrolling the update above from a uniform P_1.

4 Best Response

The best response

$$\theta_t = \operatorname*{arg\,min}_{\theta \in \Theta} R_t(\theta)$$

suffers zero regret. But in typical online learning, this is obviously not available to us.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 23 — April 18, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Online Learning with Mixed Strategies

At each round the learner takes an action $x_t \in \mathcal{X}$, and the environment picks a loss $f_t : \mathcal{X} \to \mathbb{R}$.

Regret compares against the best possible action in hindsight:

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} f_t(x).$$

A *mixed strategy* P_t is a distribution over $\mathcal{X}$. The expected loss over the mixed strategy is

$$\begin{aligned} \int P_t(x)\, f_t(x)\, dx &= \int P_t(x)(f_t(x) - f_t(x_T^*))\, dx \\ &= \int P_t(x)\, r_t(x)\, dx, \end{aligned}$$

where

$$r_t(x) := f_t(x) - f_t(x_T^*)$$

is the regret at time t if we took action x, and x_T^* is the best action in hindsight.

The regret of x_t with respect to a baseline action x_{base} is

$$\bar{r}_t(x_{\text{base}}) = f_t(x_t) - f_t(x_{\text{base}}).$$

If $x_t \sim P_t$, the regret of P_t with respect to x_{base} is

$$\bar{r}_t(x_{\text{base}}) = \int P_t(x)\, f_t(x)\, dx - f_t(x_{\text{base}}).$$

Plugging this in,

$$\begin{aligned} \int \bar{r}_t(x_B)\, P_t(x_B)\, dx_B &= \int \left[\int P_t(x)\, f_t(x)\, dx - f_t(x_B) \right] P_t(x_B)\, dx_B \\ &= 0. \end{aligned}$$

1.1 Potential functions

Let the cumulative regret be

$$R_t(x_B) = \sum_{s=1}^{t} \bar{r}_s(x_B),$$

and play

$$P_t(x) \propto \nabla\Phi(R_{t-1})(x)$$

for a potential function Φ. Then

$$\int \underbrace{\bar{r}_t(x_B)\,\nabla\Phi(R_{t-1})(x_B)}_{\text{orthogonal to the potential function}}\,dx_B = 0.$$

No matter what f_t is chosen by the environment, $\bar{r}_t$ is orthogonal to $\nabla\Phi(R_{t-1})$. So

$$\Phi(R_{t-1}+\bar{r}_t) \lesssim \Phi(R_{t-1}), \qquad \text{since} \qquad \Phi(R_{t-1}+\bar{r}_t) \approx \Phi(R_{t-1}) + \bar{r}_t \cdot \nabla\Phi(R_{t-1}) = \Phi(R_{t-1}).$$

This gives that if $\Phi(R_t) \le \delta$, then

$$R_t \le \Phi^{-1}(\delta).$$

This is useful because we can say that we are always making progress with respect to some potential function. It is fascinating because we can make progress even in a completely adversarial environment.

2 Games

- Player 1 takes actions $x \in \mathcal{X}$ and has loss $\ell(x, y)$.
- Player 2 takes actions $y \in \mathcal{Y}$ and has loss $-\ell(x, y)$.

Player 1 solves

$$\min_x \max_y \ell(x, y) = \min_x \ell(x, y_x),$$

solving the inner optimization first, where y_x is the best response to x. In general

$$\max_y \min_x \ell(x, y) \le \min_x \max_y \ell(x, y),$$

and the two are not the same.

A sufficient condition for the two to be the same:

- $\mathcal{X}, \mathcal{Y}$ are convex, bounded and closed;
- ℓ is convex in x and concave in y.

2.1 Nash equilibrium

Definition (Nash equilibrium)**.** (x^*, y^*) is a *Nash equilibrium* if

$$\max_y \ell(x^*, y) = \ell(x^*, y^*) = \min_x \ell(x, y^*).$$

If a Nash equilibrium exists, then

$$\begin{aligned}\min_x \max_y \ell(x, y) &\le \max_y \ell(x^*, y) = \ell(x^*, y^*) \\ &= \min_x \ell(x, y^*) \le \max_y \min_x \ell(x, y).\end{aligned}$$

So we have equality:

$$\max\min = \min\max.$$

2.2 Mixed strategies

"Pure" strategies are single actions $x \in \mathcal{X}$, $y \in \mathcal{Y}$. *Mixed strategies* are distributions $P_x \in \mathcal{P}_\mathcal{X}$ over $\mathcal{X}$ and $P_y \in \mathcal{P}_\mathcal{Y}$ over $\mathcal{Y}$, with

$$\ell(P_x, P_y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_x(x)\, P_y(y)\, \ell(x, y).$$

Consider

$$\min_{P_x} \max_{P_y} \ell(P_x, P_y).$$

The domain is convex, and $\ell(P_x, P_y)$ is convex-concave. It may look as if ℓ is unbounded, but it actually is bounded since $\mathcal{X}$ and $\mathcal{Y}$ are bounded. Then

$$P_x^*, P_y^* \equiv \text{"mixed" Nash equilibrium of the original game.}$$

2.3 Learning in games

To solve

$$\min_{P_x} \max_{P_y} \ell(P_x, P_y),$$

at round t the players play $P_x^{(t)}$ and $P_y^{(t)}$, and player 1 suffers

$$\sum_{t=1}^{T} \ell\big(P_x^{(t)}, P_y^{(t)}\big).$$

For player 1, $\ell\big(\cdot, P_y^{(t)}\big)$ is the loss designed by the environment. Let

$$\bar{P}_x = \frac{1}{T} \sum_{t=1}^{T} P_x^{(t)}, \qquad \bar{P}_y = \frac{1}{T} \sum_{t=1}^{T} P_y^{(t)}.$$

Then

$$\min_{P_x} \ell(P_x, \bar{P}_y) \approx \max_{P_y} \ell(\bar{P}_x, P_y),$$

so

$$(\bar{P}_x, \bar{P}_y) \equiv \text{approximate mixed Nash equilibrium.}$$

The difference from full online learning is that ℓ is fixed here. You play P_t, the adversary plays Q_t, and you suffer loss $\ell(P_t, Q_t)$. The regret is

$$\sum_{t=1}^{T} \ell(P_t, Q_t) - \min_{P} \sum_{t=1}^{T} \ell(P, Q_t).$$

3 Online Classification

The input space is $\mathcal{X}$, and the output space is $\mathcal{Y} = \{-1, +1\}$. For rounds $t = 1, 2, \ldots$:

- x_t is chosen;
- we predict $y_t \sim P_t$;
- the true label $f^*(x_t)$ is revealed.

The regret is

$$\sum_{t=1}^{T} \mathbb{P}\big[y_t \neq f^*(x_t)\big] - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \mathbb{I}\big[h(x_t) \neq f^*(x_t)\big].$$

Think of the actions not as labels, but as hypotheses. Then the update

$$P_t(h) \propto P_{t-1}(h) \exp\Big(-\eta\, \mathbb{I}\big[h(x_t) \neq f^*(x_t)\big]\Big)$$

gives sublinear regret!

Come up with a few hypotheses, and see how people change.

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 24 — April 20, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

Let Y be a health outcome and X be whether a person takes vitamin C. There are two different questions we could ask:

- Predict Y given X.
- Predict Y when we set $X = x$.

What is interesting is if you can change the distribution and see the outcome.

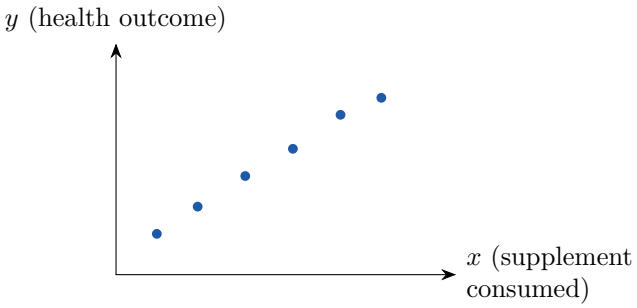

Figure 64: Observed health outcome y against the amount of supplement consumed x.

The two questions correspond to two different distributions:

$$P[Y \mid X = x] \qquad \text{versus} \qquad P[Y \mid \text{do}(X = x)].$$

Consider a treatment $X \in \{0, 1\}$ and a response $Y \in \mathbb{R}$. Define

$$Y_1 \equiv \text{the response when we set } X = 1,$$
$$Y_0 \equiv \text{the response when we set } X = 0.$$

Then

$$Y = XY_1 + (1 - X)Y_0.$$

We don't see Y_1 and Y_0 at the same time:

X	Y	Y_1	Y_0
0	1	–	1
0	1/2	–	1/2
1	3/4	3/4	–
1			
0			
1			

So you only see 50% of the response data. (You can't see what happens if you do something else.)

1 Causal Effects

The causal effect is

$$\mathbb{E}[Y_1] - \mathbb{E}[Y_0],$$

which compares the outcome if you gave treatment versus if you did not give treatment.

There is no way to compute the mean with 50% of the data missing in a very specific way. In general this is not possible, because in general

$$\mathbb{E}[Y_1] \neq \mathbb{E}[Y \mid X = 1],$$
$$\mathbb{E}[Y \mid X = 1] - \mathbb{E}[Y \mid X = 0] \neq \mathbb{E}[Y_1] - \mathbb{E}[Y_0].$$

Suppose $(X, Y, Y_0, Y_1) \sim P$, and we observe

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^n \sim P(X, Y).$$

We want to estimate

$$\theta(P) = \mathbb{E}[Y_1] - \mathbb{E}[Y_0]$$

with an estimator satisfying $\left\|\hat{\theta} - \theta\right\| \xrightarrow{P} 0$. Without assumptions, no such estimator exists.

1.1 I. Randomization

Suppose $X \perp\!\!\!\perp Y_0, Y_1$: we choose whether to give treatment or not at random. Then

$$\begin{aligned} \mathbb{E}[Y_1] &= \mathbb{E}[Y_1 \mid X = 1] && \text{(because } X \perp\!\!\!\perp Y_1\text{)} \\ &= \mathbb{E}[Y \mid X = 1] && \text{(since } Y = XY_1 + (1 - X)Y_0\text{)}. \end{aligned}$$

Similarly,

$$\mathbb{E}[Y_0] = \mathbb{E}[Y \mid X = 0].$$

This is the setting we covered previously.

1.2 II. Adjusting for confounding

Suppose instead there is a random variable Z such that

$$X \perp\!\!\!\perp (Y_0, Y_1) \mid Z.$$

Then

$$\begin{aligned} \mathbb{E}[Y_1] &= \mathbb{E}\big[\mathbb{E}[Y_1 \mid Z]\big] \\ &= \int \mathbb{E}[Y_1 \mid Z = z]\, P(Z = z)\, dz \\ &= \int \mathbb{E}[Y_1 \mid X = 1, Z = z]\, P(Z = z)\, dz && \text{(since } X \text{ is independent of } Y_1 \text{ given } Z\text{)} \\ &= \int \mathbb{E}[Y \mid X = 1, Z = z]\, P(Z = z)\, dz. \end{aligned}$$

Compare this with

$$\begin{aligned}\mathbb{E}[Y \mid X=1] &= \int \mathbb{E}[Y \mid X=1, Z=z]\, P[Z=z \mid X=1]\, dz \\ &\neq \int \mathbb{E}[Y \mid X=1, Z=z]\, P[Z=z]\, dz \\ &= \mathbb{E}[Y_1],\end{aligned}$$

since in general Z could depend on X.

To drive home the point, it could be that in reality the data look like this:

X	Y	Y_0	Y_1	Z
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Here X is clearly not independent of Y_0, Y_1, but we want it to be so. In this case, people are always healthy or always unhealthy. We adjust for confounding using Z.

The quantity

$$\int \mathbb{E}[Y \mid X=1, Z=z]\, P(Z=z)\, dz$$

is expensive to estimate, but you only care about a scalar:

$$\theta = \int f(x=1,z)\, p(z)\, dz - \int f(x=0,z)\, p(z)\, dz,$$

where $f(x,z) = \mathbb{E}[Y \mid X=x, Z=z]$. Estimating f itself only achieves the nonparametric rate $n^{-4/(4+d)}$, but it is possible to estimate θ with a parametric rate!

Y_0 and Y_1 are called *potential outcomes*: we only see one outcome (based on our X), not both. More generally,

$$Y(x) \equiv \text{the response when we set } X = x.$$

2 Causal Graphs

Let $(X_1, \ldots, X_d, Y) \sim P$. The graph is directed and acyclic.

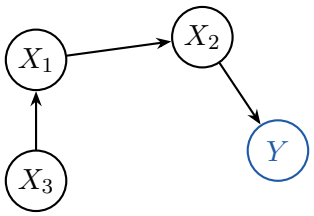

Figure 65: A causal graph over (X_1, X_2, X_3, Y): a directed acyclic graph.

Why is the DAG important? We can say that P is *faithful* to G if and only if

$$P(Z = (X, Y)) = \prod_i P(Z_i \mid Z_{\mathrm{PA}_G(i)}),$$

that is,

$$Z_i \perp\!\!\!\perp Z_{\mathrm{ND}_G(i) \setminus \mathrm{PA}_G(i)} \mid Z_{\mathrm{PA}_G(i)},$$

where $\mathrm{ND}_G(i)$ are the non-descendants of i: each variable is conditionally independent of its non-descendants given its parents.

Example 11. Consider the graph below.

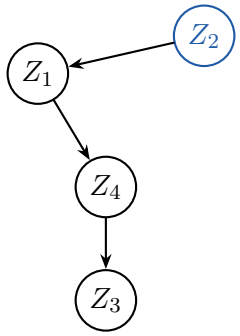

Figure 66: A DAG on $Z_1, \ldots, Z_4$; we intervene on Z_2 (highlighted).

Then

$$P(Z) = \prod_i P(Z_i \mid Z_{\mathrm{PA}_G(i)}).$$

The distribution $P(Z_1 \mid \mathrm{do}(Z_2 = 1))$ is obtained by changing the distribution, by changing the mechanism of Z_2:

$$P^{\mathrm{do}(Z_2 = z_2)}(Z) = \prod_{i \neq 2} P(Z_i \mid Z_{\mathrm{PA}_G(i)})\, \delta_{z_2}(Z_2).$$

Suppose we have variables (X, Y, Z) with some DAG structure:

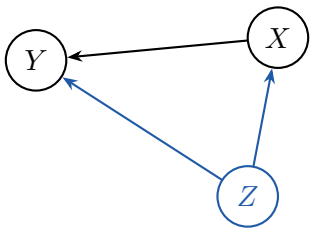

Figure 67: Z affects both the treatment X and the response Y, and X affects Y.

Then

$$P(X, Y, Z) = P(Z)\, P(X \mid Z)\, P(Y \mid X, Z).$$

How do we figure out the edge directions?

Under $\mathrm{do}(X = 1)$,

$$P^{\mathrm{do}(X=1)}(Y, X, Z) = P(Z)\, P(Y \mid X, Z)\, \delta_1(X).$$

The causal effect is

$$\mathbb{E}_{P^{\mathrm{do}(X=1)}}[Y] - \mathbb{E}_{P^{\mathrm{do}(X=0)}}[Y].$$

For the first term,

$$\begin{aligned}
\mathbb{E}_{P^{\mathrm{do}(X=1)}}[Y] &= \iiint y\, p(z)\, p(y \mid x, z)\, \delta_1(x)\, dy\, dz\, dx \\
&= \iint y\, p(z)\, p(y \mid z, x = 1)\, dy\, dz \\
&= \int \mathbb{E}[Y \mid z, x = 1]\, p(z)\, dz.
\end{aligned}$$

So the causal effect is

$$\int \mathbb{E}[Y \mid z, x = 1]\, p(z)\, dz - \int \mathbb{E}[Y \mid z, x = 0]\, p(z)\, dz.$$

This is exactly the confounding setup, with confounder Z!

That only works if you have all the confounders. With an unknown confounder U, this does not hold:

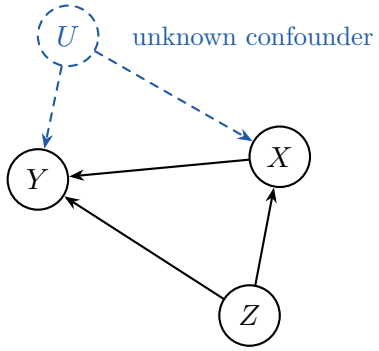

Figure 68: An unobserved confounder U (dashed) affects both X and Y, so adjusting for Z alone is not enough.

Lecture 25 — April 25, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

1 Aggregation

An *ensemble* of m models $\{f_j\}_{j=1}^m$ predicts with

$$\frac{1}{m}\sum_{j=1}^{m} f_j(x).$$

Boosting:

$$\sum_{j=1}^{m} \alpha_j h_j(x).$$

Let $D = \{(x_i, y_i)\}_{i=1}^n$, and let

$$D^{(j)} \equiv n \text{ samples drawn from } D \text{ with replacement.}$$

This gives B "datasets" $\{D^{(j)}\}_{j=1}^B$. On each $D^{(j)}$ we train a model f_j, and we predict with

$$\frac{1}{m}\sum_{j=1}^{m} f_j.$$

As an example, let $\{x_i\}_{i=1}^n \sim P$ with $\mathrm{Var}(X_i) = 1$, and let

$$\mu = \mathop{\mathbb{E}}_{x\sim P}[X], \qquad \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

Then

$$\hat{\mu}_n \to N\Big(\mu, \frac{1}{n}\Big)$$

(the distance between the two distributions goes to 0 as $n \to \infty$). So

$$\sqrt{n}\,(\hat{\mu}_n - \mu) \approx N(0,1), \qquad \hat{\mu}_n = \mu + \frac{Z}{\sqrt{n}}.$$

Consider the estimator

$$\begin{aligned}\hat{\theta} = \mathbb{I}[\hat{\mu} \le \lambda] &= \mathbb{I}\big[\sqrt{n}\,(\hat{\mu} - \mu) \le \sqrt{n}\,(\lambda - \mu)\big] \\ &\approx \mathbb{I}\big[Z \le \sqrt{n}\,(\lambda - \mu)\big].\end{aligned}$$

With

$$\lambda_n = \mu + \frac{c}{\sqrt{n}},$$

this becomes

$$\mathbb{I}[Z \le c].$$

1.1 Bagging

Let $D = \{x_i\}_{i=1}^n$, with bootstrap datasets $\{D^{(j)}\}_{j=1}^B$, where

$$D^{(j)} \equiv \{x_i^{(j)}\}_{i=1}^n, \qquad \hat{\mu}(j) = \frac{1}{n}\sum_{i=1}^n x_i^{(j)}.$$

We have $\frac{1}{n}\sum_{i=1}^n x_i \approx \mu$. For a bootstrap sample $\tilde{x}_1, \ldots, \tilde{x}_n$, let

$$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^n \tilde{x}_i.$$

Then $\tilde{\mu} \approx \hat{\mu}$, and

$$\sqrt{n}\,(\tilde{\mu} - \hat{\mu}) \to N(0,1).$$

The bootstrap version of the estimator is

$$\begin{aligned}\tilde{\theta} = \mathbb{I}[\tilde{\mu} \le \lambda] &= \mathbb{I}\big[\sqrt{n}\,(\tilde{\mu} - \hat{\mu}) \le \sqrt{n}\,(\lambda - \hat{\mu})\big] \\ &\approx \mathbb{I}\big[Z \le \sqrt{n}\,(\lambda - \hat{\mu})\big].\end{aligned}$$

Taking the expectation over bootstrap samples,

$$\mathbb{E}[\tilde{\theta}] \approx \Phi\big(\sqrt{n}\,(\lambda - \hat{\mu})\big).$$

Now take

$$\lambda = \mu + \frac{c}{\sqrt{n}}.$$

Since $\hat{\mu} \approx \mu$, we have $\lambda - \hat{\mu} \approx c/\sqrt{n}$. More precisely, $\sqrt{n}\,(\hat{\mu} - \mu) \approx N(0,1) \equiv Z$, so

$$\hat{\mu} = \mu + \frac{Z}{\sqrt{n}}, \qquad \sqrt{n}\,(\lambda - \hat{\mu}) = c - Z.$$

Therefore

$$\begin{aligned}\hat{\theta} &= \mathbb{I}[Z \le c] + o(1), \\ \mathbb{E}_B[\tilde{\theta}] &= \Phi(c - Z).\end{aligned}$$

For $c = 0$, the unbagged estimator has

$$\mathbb{E}[\hat{\theta}] = \frac{1}{2}, \qquad \mathrm{Var}(\hat{\theta}) = \frac{1}{4}.$$

Bagging smooths it out: $\Phi(Z) \sim \mathrm{Unif}(0,1)$, which has

$$\text{mean} = \frac{1}{2}, \qquad \text{variance} = \frac{1}{12}.$$

Recall that smoothing kernels are a form of averaging.

2 Random Forests

- **Bootstrap samples** (B of them): we have multiple trees, and learn each tree on a bootstrap sample.
- **Random subset of features** (a fraction p of them).

So overall, a random forest is randomized over both the bootstrap samples and the random subset of features.

There are a couple of perspectives on why random forests work so well. They all go back to smoothing.

2.1 Layered nearest neighbors (LNN)

The normal nearest neighbor is

$$\mathrm{NN}(x) = \arg\min_{x_i \in D} \|x - x_i\| .$$

A *layered nearest neighbor* (LNN) has the additional geometric requirement that the box specified by x and x_i should have no additional training points.

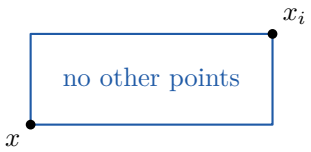

Figure 69: x_i is a layered nearest neighbor of x: the box with corners x and x_i contains no other training points.

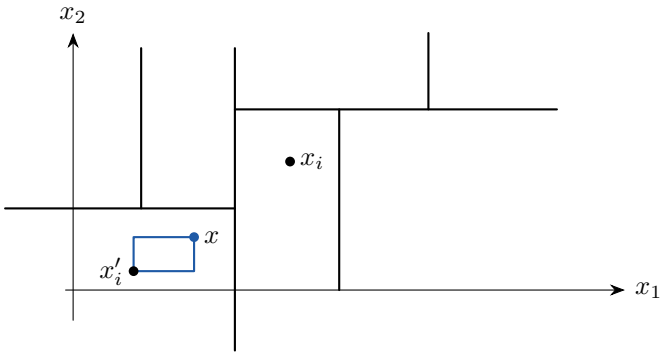

Figure 70: A tree grown until each cell holds a single training point. The box spanned by x and x_i' (highlighted) holds no other point, so x and x_i' are LNNs, and the tree's prediction at x is y_i'.

We want to stop when there is a single rectangle for each point. A tree predicts

$$\hat{f}(x) = \sum_j c_j \, \mathbb{I}[x \in R_j], \qquad c_j = \frac{\sum_{i=1}^n y_i \, \mathbb{I}[x_i \in R_j]}{\sum_{i=1}^n \mathbb{I}[x_i \in R_j]}.$$

Let

$$\mathcal{L}_n(x) = \text{the set of all LNNs of } x.$$

Then

$$\hat{f}(x) = \frac{\sum_{i=1}^n y_i \, \mathbb{I}[x_i \in \mathcal{L}_n(x)]}{\sum_{i=1}^n \mathbb{I}[x_i \in \mathcal{L}_n(x)]},$$

and

$$\mathbb{E}[(\hat{f}(x) - f(x))^2] \asymp \frac{d!}{2^d (\log n)^{d-1}}.$$

2.2 Kernel-based view of random forests

With M trees,

$$\hat{f}(x) = \frac{1}{M} \sum_{j=1}^M \frac{\sum_{i=1}^n y_i \, \mathbb{I}[x_i \in R_j(x)]}{\sum_{i=1}^n \mathbb{I}[x_i \in R_j(x)]},$$

where $R_j(x)$ is the cell of x in the j-th tree. We can think of the kernel implicitly as

$$K(x, z) = \frac{1}{M} \sum_{j=1}^M \mathbb{I}[z \in R_j(x)],$$

so that

$$\hat{f}(x) = \frac{\sum_{i=1}^n y_i \, K(x, x_i)}{\sum_{i=1}^n K(x, x_i)}.$$

In the limit of many trees,

$$K(x, z) = \mathbb{P}[z \in R(x)].$$

The region is a cell in a decision tree. This is an adaptive partitioning of your dataset, way more complicated than just a ball. We also have

$$K(x, x') = \int K(x, z) \, K(x', z) \, p(z) \, dz.$$

10-716: Advanced Machine Learning: Theory and Methods Spring 2023

Lecture 26 — April 27, 2023

Prof. Pradeep Ravikumar *Scribe: Fan Pu Zeng*

This lecture is a recap of the course.

1 Mercer Kernels

Let the input space be $\mathcal{X} \subseteq \mathbb{R}^d$, and let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A *Mercer kernel* satisfies

$$\int_{\mathcal{X}} \int_{\mathcal{X}} f(x)\, K(x, x')\, f(x')\, p(x)\, p(x')\, dx\, dx' \geq 0.$$

This is the continuous analog of

$$v^T K v \geq 0 \quad \forall v$$

for a matrix K.

1.1 Spectral decomposition of a kernel

A kernel can be decomposed as

$$K(x, x') = \sum_{j=1}^{\infty} \underbrace{\lambda_j}_{\text{eigenvalues}} \underbrace{\phi_j(x)\, \phi_j(x')}_{\text{eigenfunctions}},$$

where

$$\int_{\mathcal{X}} \phi_j(x)\, \phi_{j'}(x)\, p(x)\, dx = \mathbb{I}[j = j'],$$

so the eigenfunctions are orthogonal to each other.

With a Mercer kernel, you can define an RKHS:

$$\mathcal{H}_K = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \|f\|^2_{\mathcal{H}_K} < \infty \right\},$$

where

$$\|f\|^2_{\mathcal{H}_K} = \langle f, f \rangle_{\mathcal{H}_K}, \qquad \langle f, K(x, \cdot) \rangle_{\mathcal{H}_K} = f(x), \qquad \langle K(x', \cdot), K(x, \cdot) \rangle_{\mathcal{H}_K} = K(x, x').$$

By definition, we want the inner product to satisfy these properties. In terms of the eigenfunctions,

$$\mathcal{H}_K = \left\{ f = \sum_{j=1}^{\infty} a_j \phi_j \;\middle|\; \sum_{j=1}^{\infty} \frac{a_j^2}{\lambda_j} < \infty \right\}.$$

Compare this with

$$L_2(P) = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \int f^2(x)\, p(x)\, dx < \infty \right\},$$

for which $\{\phi_j\}$ is a basis. So one way to think about it: the RKHS contains much smoother functions than L_2.

People were looking at the RBF kernel using the standard Fourier basis as the eigenfunctions, but it required so many of the Fourier basis functions that interest died out. But recently we discovered many interesting kernels that caused results to be rediscovered, for example the following two.

1.2 Contrastive kernel

Let $P_+ : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the distribution of *positive pairs*, generated as

$$z \sim P_d, \qquad x \sim P_a(\cdot \mid z), \qquad x' \sim P_a(\cdot \mid z),$$

where P_a is the augmentation distribution. Marginalizing over z,

$$P_+(x, x') = \int P_d(z)\, P_a(x \mid z)\, P_a(x' \mid z)\, dz.$$

Contrastive kernels include

$$K_1(x, x') = \frac{P_+(x, x')}{P_+(x)\, P_+(x')}, \qquad K_2(x, x') = \frac{P_+(x, x')}{\sqrt{P_+(x)}\,\sqrt{P_+(x')}}, \qquad K_3(x, x') = \log \frac{P_+(x, x')}{P_+(x)\, P_+(x')}.$$

$K(x, x')$ has a large value if x and x' are augmentations of the same image.

1.3 Neural tangent kernel

Let $f_\theta(x)$ be a parametric model (it could be a neural network, a polynomial, etc.). The *neural tangent kernel* is

$$K_\theta(x, x') = \left\langle \frac{\partial f_\theta(x)}{\partial \theta}, \frac{\partial f_\theta(x')}{\partial \theta} \right\rangle,$$

which is huge, since it is with respect to the parameters.

Learn θ (for example by gradient descent):

$$\theta_0 \to \theta_1 \to \theta_2 \to \cdots.$$

In the overparameterized setting,

$$K_{\theta_t}(x, x') \approx K_{\theta_0}(x, x'),$$

since θ_t stays close to θ_0. Extracting features then does not rely on the data, just on $\partial f_\theta(x)/\partial\theta$.

But this only holds in the lazy training regime, where the learning rate is much smaller than what it should be. This is implicitly hidden in a lot of the analysis.

A randomly wired DNN $f_\theta(x)$ with large width starts to look like $\mathrm{GP}(K_\ell)$.

The RKHS is actually not restrictive; it is still a giant space. It is the minimal thing we require. Why do we want continuous evaluation functionals? Fix x. Then

$$f \approx f' \implies f(x) \approx f'(x).$$

2 Clustering

We observe $X_i \overset{\text{iid}}{\sim} P$ for $i \in [n]$, with $x_i \in \mathbb{R}^d$.

Let

$$\mathcal{C} = \{c_1, \ldots, c_k\} \subseteq \mathbb{R}^d$$

be a *codebook*. (This is like the input to the retina, which is very noisy: it is blocked by the nose, there is a hole in the receptors, etc.) The idea is that there exists some finite codebook $\mathcal{C}$. Then let

$$\Pi_{\mathcal{C}}(x) = \operatorname*{argmin}_{c \in \mathcal{C}} \|x - c\| .$$

Codebooks are heavily used in communication networks. Using $\Pi_{\mathcal{C}}(x)$ in this way is called *quantization* or *discretization*.

Define the risk

$$R(\mathcal{C}) = \mathop{\mathbb{E}}_{X \sim P} \|X - \Pi_{\mathcal{C}}(X)\|^2 .$$

Goal: find $\mathcal{C}^* \in \operatorname{argmin}_{\mathcal{C}} R(\mathcal{C})$. We can think of $\mathcal{C}$ as the centroids of the clusters, with k a hyperparameter. The empirical version is

$$\begin{aligned} \widehat{\mathcal{C}} \in \operatorname*{argmin}_{\mathcal{C}} \widehat{R}(\mathcal{C}) &= \operatorname*{argmin}_{\mathcal{C}} \frac{1}{n} \sum_{i=1}^n \|X_i - \Pi_{\mathcal{C}}(X_i)\|^2 \\ &= \operatorname*{argmin}_{\mathcal{C}} \frac{1}{n} \sum_{i=1}^n \min_{j \in [k]} \|X_i - c_j\|^2 , \end{aligned}$$

which is the k-means objective.

2.1 Level sets

The *level set* at level t is

$$L_t = \{x \mid p(x) > t\}.$$

We can think of t as the sea level: we want the sea level high enough that we don't have as much noise, and we only capture the islands. The clusters are

$$\mathcal{C}_t = \text{connected components of } L_t.$$

From data, let

$$\widehat{L}_t = \{x_i \mid \widehat{p}(x_i) > t\},$$

and build the graph $G_t = (V_t, E_t)$ with

$$V_t = \widehat{L}_t, \qquad E_t = \{(x, x') \in V_t \times V_t \mid \|x - x'\| \leq \varepsilon\},$$

so we only connect close points. Then

$$\widehat{\mathcal{C}}_t \equiv \text{connected components of } G_t.$$

Given points $\{x_i\}_{i=1}^n$, let

$$r_k(x_i) = \min\{r \mid |B(x_i, r)| \geq k\}, \qquad B(x_i, r) = \{x_j \mid \|x_j - x_i\| \leq r\}.$$

That is, we want a ball such that there are at least k points in it.

Fix r, and let $G_r = (V_r, E_r)$ with

$$V_r = \{x_i \mid r_k(x_i) \leq r\} \quad \text{(all the points with at least } k \text{ neighbors within radius } r\text{)},$$
$$E_r = \{(x_i, x_{i'}) \in V_r \times V_r \mid \|x_i - x_{i'}\| \leq \alpha r\}.$$

Then

$$\mathcal{C}_r \equiv \text{connected components of } G_r.$$

This still looks like level sets: V_r performs the density cutoff, and E_r defines the graph.

For $k = 2$ and $\alpha = 1$, this is single linkage (min-linkage hierarchical clustering): $r_2(x_i)$ is the distance to the nearest neighbor of x_i. So this is a generalization of single-linkage clustering, and hierarchical clustering is just capturing level sets of the distribution.

2.2 Modes

We can define a curve implicitly by

$$\frac{dx(t)}{dt} = p'(x(t)), \qquad x(0) = x.$$

Then $x(\infty)$ is a mode of p. Write $\text{dest}(x) = x(\infty)$ for the mode reached from x. Let

$$C(m) = \{x \mid \text{dest}(x) = m\} \subset \mathcal{X},$$

the set of all the points that belong to the basin of attraction of m.

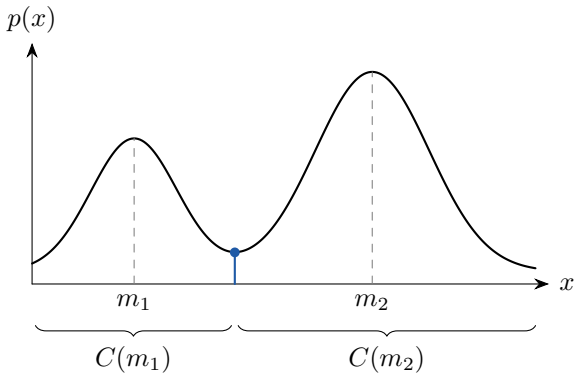

Figure 71: A density with two modes m_1 and m_2; the valley separates their basins of attraction.

This summarizes a data distribution by its basins of attraction, i.e. its modes. Other ways of summarizing a data distribution include percentiles, etc.

3 Online Learning

Online learning, and its application to games. In round $t = 1, 2, \ldots, T$:

- the learner picks $x_t \in \mathcal{X}$;
- the environment picks a loss $f_t : \mathcal{X} \to \mathbb{R}$;
- the learner suffers $f_t(x_t)$.

The total loss suffered is $\sum_{t=1}^{T} f_t(x_t)$. The average regret compares against the single best action in hindsight:

$$\frac{1}{T}\sum_{t=1}^{T} f_t(x_t) - \min_{x\in\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T} f_t(x).$$

There exist algorithms such that this is

$$o(1) \xrightarrow[T\to\infty]{} \leq 0.$$

This is impressive, since the environment can be adversarial, but we can still do as well as the single best action.

FTL (follow the leader):

$$x_t = \operatorname*{argmin}_{x\in\mathcal{X}} \sum_{s=1}^{t-1} f_s(x).$$

But this doesn't always work, as we saw in lecture. If the environment can force us to change our strategy drastically, we can get exploited.

FTRL (follow the regularized leader):

$$x_t = \operatorname*{argmin}_{x\in\mathcal{X}} \Big\{ \sum_{s=1}^{t-1} f_s(x) + \lambda R(x) \Big\}.$$

The regularizer $\lambda R(x)$ is there because we don't want to change strategy too drastically.