36-708: Statistical Methods in Machine Learning

Spring 2023 • Prof. Aaditya Ramdas

Notes by Fan Pu Zeng

Contents

Lecture 2 — January 19, 2023 **4**
1 Setup and Notation 4
2 Least Squares and Ridge Regression 5
3 Generalization, Memorization and ERM 6

Lecture 3 — January 24, 2023 **8**
1 Recap 8
2 The Regression Function Minimizes the MSE 8
3 k-Nearest Neighbors 8
4 Test Error 11
5 Consistency 11

Lecture 4 — January 26, 2023 **12**
1 Oracle Regressors and Classifiers 12
2 k-NN as Two Approximations 13
3 Drawbacks of k-NN 14
4 High Dimensions 15

Lecture 5 — January 31, 2023 **17**
1 Recap 17
2 Predictive Uncertainty Quantification 17
3 Probabilistic Calibration 17
4 What Makes a Good Classifier? 18

Lecture 6 — February 2, 2023 **21**
1 Histogram Binning 21
2 Split Conformal Prediction (Regression) 22

Lecture 7 — February 7, 2023 **24**
1 Leo Breiman Week 24
2 Bagging (Bootstrap Aggregation) 24

3 Algorithmic Stability . . . 26
4 Decision Stumps, Lists, Trees and Forests . . . 26

Lecture 8 — February 9, 2023 **28**
1 Outline . . . 28
2 Decision Trees . . . 28
3 Decision Trees Are Very Variable . . . 30
4 Random Forests . . . 30

Lecture 9 — February 14, 2023 **31**
1 Stacking . . . 31

Lecture 10 — February 16, 2023 **32**
1 Model Selection . . . 32
2 Boosting . . . 33

Lecture 11 — February 21, 2023 **35**
1 Edge of a Classifier . . . 35
2 Weak Learning Hypothesis . . . 35
3 AdaBoost . . . 37

Lecture 13 — February 28, 2023 **39**
1 Boosting: VC Theory Is Unsatisfactory . . . 39
2 Convex Surrogate Loss Minimization by Coordinate Descent . . . 40
3 Margin . . . 41
4 VC Theory . . . 41
5 Empirical Rademacher Complexity . . . 42

Lecture 14 — March 2, 2023 **43**
1 Margin and Boosting . . . 43
2 Recap of the First Half of the Course . . . 44

Lecture 15 — March 14, 2023 **47**
1 RKHS . . . 47
2 Banach Spaces . . . 49
3 Hilbert Spaces . . . 50

Lecture 16 — March 16, 2023 **51**
1 RKHS . . . 51
2 Kernels . . . 53

Lecture 17 — March 21, 2023 **55**
1 The Moore–Aronszajn Theorem . . . 55
2 "Closure" of Kernels . . . 56
3 Kernel Ridge Regression . . . 57
4 Representer Theorem . . . 57

Lecture 18 — March 23, 2023 **59**
1 Kernel Ridge Regression . . . 59
2 The "Kernel Trick" . . . 61

Lecture 20 — March 30, 2023 **62**
1 Cooperative Game Theory: Shapley Values . . . 62
2 Shapley Values in Machine Learning . . . 64

Lecture 21 — April 4, 2023 **65**
1 Unsupervised Learning . . . 65
2 SVD and PCA . . . 65
3 PCA . . . 66

Lecture 22 — April 6, 2023 **70**
1 Modal Clustering . . . 70
2 Hierarchical Clustering . . . 70

Lecture 23 — April 11, 2023 **71**
1 Graph Laplacians . . . 71
2 Spectral Clustering . . . 72

Lecture 27 — April 27, 2023 **74**
1 Deep Learning . . . 74
2 End-of-Semester Recap . . . 76
3 Paradigms . . . 76

Lecture 2 — January 19, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Setup and Notation

We observe pairs

$$(X_i, Y_i) \overset{\text{iid}}{\sim} P_{XY}.$$

The X_i are called *features*, *covariates*, *predictors* or *inputs*, and usually live in $\mathbb{R}^d$ (some authors write $\mathbb{R}^p$). The Y_i are called the *response*, the *dependent variable* or the *output*. We want to go from X to Y.



Figure 1: We want to predict the response Y from the features X.

The response can take several forms:

- binary, $Y \in \{0, 1\}$ or $Y \in \{-1, 1\}$;
- one of k classes, $Y \in \{1, \ldots, k\}$;
- a value in an interval, such as $Y \in [0, 1]$.

Moving between the two binary conventions is easy. If $y_i \in \{0, 1\}$, then $2y_i - 1 \in \{-1, 1\}$, so an expression such as $y_i f(x_i)$ for labels in $\{-1, 1\}$ becomes $(2y_i - 1)f(x_i)$ for labels in $\{0, 1\}$.

1.1 Indexing

- Points are indexed by $i \in \{1, \ldots, n\}$, also written $[n]$ (or $[N]$).
- X_{ij} is the j-th covariate of the i-th point.
- X_j is the j-th covariate of a generic point; it is a scalar.
- $X_i \in \mathbb{R}^d$ is the i-th point.
- X, X' denote two generic points, with $(X, Y), (X', Y') \sim P_{XY}$.
- $X_i, X_{i'}$ denote two different training points, and similarly for $f(X_i), f(X_j)$.

Vectors are column vectors:

$$X_i = \begin{pmatrix} X_{i1} \\ \vdots \\ X_{id} \end{pmatrix}, \qquad \text{so } X_i^T X_j \text{ is an inner product.}$$

We write $v \in S^{d-1}$ for a unit vector, $\|v\|_2 = 1$.

The *data matrix* is the $n \times d$ matrix

$$\mathbb{X} = \begin{pmatrix} X_{11} & \cdots & X_{1d} \\ \vdots & & \vdots \\ X_{n1} & \cdots & X_{nd} \end{pmatrix} = \begin{pmatrix} X_1^T \\ \vdots \\ X_n^T \end{pmatrix}.$$

Below we write X for the data matrix when this causes no confusion.

2 Least Squares and Ridge Regression

Ordinary least squares solves

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (Y_i - X_i^T \beta)^2 = \min_{\beta \in \mathbb{R}^d} \|Y - X\beta\|_2^2,$$

where Y is $n \times 1$, X is $n \times d$ and β is $d \times 1$ (note $X_i^T \beta = \beta^T X_i$). Its solution is

$$\hat{\beta}_{\text{OLS}} = (X^T X)^{-1} X^T Y,$$

where $X^T X$ is $d \times d$ (it is the covariance, up to scaling) and X^T is $d \times n$.

Adding a penalty gives *ridge regression*:

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (Y_i - X_i^T \beta)^2 + \lambda \|\beta\|_2^2,$$

with solution

$$\hat{\beta}_{\text{ridge}} = (X^T X + \lambda I_d)^{-1} X^T Y.$$

2.1 Kernel ridge regression formulation

An equivalent formulation is

$$\hat{\beta}_{\text{ridge}} = \underbrace{X^T}_{d \times n} (\underbrace{X X^T}_{n \times n} + \lambda I_n)^{-1} \underbrace{Y}_{n \times 1},$$

where XX^T is the *kernel Gram matrix*.

This is computationally nice: we can choose whether to invert a $d \times d$ matrix or an $n \times n$ matrix.

3 Generalization, Memorization and ERM

Let $(X, Y) \sim P_{XY}$, and write

$$Y = f(X) + \varepsilon, \qquad \text{where } \mathop{\mathbb{E}}_{P_{XY}}[\varepsilon] = 0.$$

Is this an assumption? **No!** We can always write

$$Y = \underbrace{Y - \mathbb{E}[Y \mid X]}_{\text{mean 0, ``}\varepsilon\text{''}} + \underbrace{\mathbb{E}[Y \mid X]}_{f^*(X)},$$

where $f^*(X) = \mathbb{E}[Y \mid X]$ is called the *regression function*.

If you see the noise assumed to be *homoskedastic*, $\varepsilon \perp\!\!\!\perp X$, then that is an actual assumption. So are statements such as: ε is light-tailed, $\mathbb{E}[\varepsilon^2] < \infty$, or the ε are i.i.d.

To learn $f^*(x)$, however, we will need some assumptions on P_{XY}. Hopefully f^* is smooth or continuous in some sense; it can be step-wise discontinuous in some places.

3.1 Algorithms

An *algorithm* is a map

$$\mathcal{A} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \times U \longrightarrow (\mathcal{X} \to \mathcal{Y}),$$

where, for example, $\mathcal{X} = \mathbb{R}^d$.

- The input is any number n of labelled data points.
- U is a single uniform random variable. It can hold infinite randomness (for instance, through its binary expansion).
- The output is a function: on a new x, how would you predict y?

Note that $\mathcal{A}$ has the freedom to do different things for different n; for instance, it could depend on whether $n < d$, $n = d$ or $n > d$.

Generalization means doing well on something out of sample.

3.2 Empirical risk minimization

In *empirical risk minimization* (ERM), "empirical" means based on the data, and the "risk" is the expected loss (for example, the expectation of the squared loss). For linear regression, ERM is

$$\operatorname*{argmin}_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} (Y_i - X_i^T \beta)^2.$$

More generally, given a class $\mathcal{F}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$,

$$\hat{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(f(X_i), Y_i) + \lambda R(f),$$

where $R(f)$ penalizes the complexity of f. That is, we pick the function that minimizes the (penalized) empirical risk. For example, for a neural network we optimize over the parameterized functions.

We then look at

$$\mathbb{E}\ \ell(\hat{f}_n(X), Y).$$

If $\hat{f}_n$ does well out of sample, it has "generalized".

ERM is one specific strategy to come up with an estimate of the regression function. It may or may not generalize.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 3 — January 24, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Recap

Last lecture, we discussed ridge regression:

$$\hat{\beta}_{\text{ridge}} = (X^T X + \lambda I_d)^{-1} X^T Y = X^T (X X^T + \lambda I_n)^{-1} Y.$$

Writing $Y = f(X) + \varepsilon$ is not an assumption. We can use

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

the *regression function*. It is an "oracle", because we don't know P.

2 The Regression Function Minimizes the MSE

Why is f^* the best choice? Because it minimizes the mean squared error. Define

$$\text{MSE}(f) = \mathop{\mathbb{E}}_{(X,Y)\sim P}\big[(Y - f(X))^2\big], \qquad f^{\text{MSE}} := \mathop{\text{argmin}}_{f \text{ measurable}} \text{MSE}(f).$$

Indeed, $f^{\text{MSE}} = f^*$! To see this, write

$$\text{MSE}(f) = \mathop{\mathbb{E}}_{X}\Big[\underbrace{\mathop{\mathbb{E}}_{Y|X}\big[(Y - f(X))^2\big]}_{(\star)}\Big].$$

It suffices to minimize $(\star)$ separately for each value of X. For intuition, take numbers $a_1, \ldots, a_k \in \mathbb{R}$. Then

$$\mathop{\text{argmin}}_{a^*} \sum_{i=1}^{k} (a_i - a^*)^2 = \frac{1}{k} \sum_{i=1}^{k} a_i,$$

while, for k odd,

$$\mathop{\text{argmin}}_{a^*} \sum_{i=1}^{k} |a_i - a^*| = \text{median}(\{a_i\}) = a_{(\lceil k/2 \rceil)},$$

the $\lceil k/2 \rceil$-th largest point. In the same way, the squared loss in $(\star)$ is minimized by the mean of Y given X, that is, by $f(X) = \mathbb{E}[Y \mid X] = f^*(X)$.

3 k-Nearest Neighbors

We want

$$f^*(x_{\text{new}}) = \mathbb{E}[Y \mid X = x_{\text{new}}].$$

But what if we have never seen x_{new} before?

3.1 k-NN regression

Define

$$\hat{f}(x_{\text{new}}) := \mathbb{E}[Y \mid X \approx x_{\text{new}}] \approx \frac{1}{k} \sum_{i\,:\,X_i \text{ is a } k\text{-NN of } x_{\text{new}}} Y_i,$$

where $X \approx x_{\text{new}}$ means that X is one of the k nearest neighbors of x_{new} according to the metric d. There are two steps of approximation here. (Here Y is real-valued, as in regression, so averaging makes sense since $\mathbb{R}$ is convex.)

We need $(\mathcal{X}, d)$ to be a metric space, since we need to define what "nearest" means.

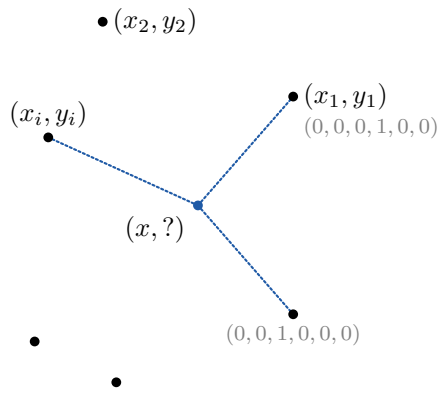


Figure 2: k-NN with $k = 3$: the label of the query $(x, ?)$ is predicted from its three nearest training points. For classification, the labels can be one-hot vectors.

3.2 Two extremes

Let's take a look at two extremes.

1-nearest neighbor $\hat{f}_1$. What is the training error? It is

$$\frac{1}{n} \sum_{i=1}^{n} \big(Y_i - \hat{f}_1(X_i)\big)^2 = 0,$$

assuming that the X_i are not repeated, since $\hat{f}_1$ will just find the item in the data set.

In general, k-NN(x) finds the k nearest neighbors of x in $\{x_i\}_{i=1}^n$ and returns the average of their y_i.

n-nearest neighbor $\hat{f}_n$. If $k = n$, then

$$\frac{1}{n} \sum_{i=1}^{n} \big(Y_i - \hat{f}_n(X_i)\big)^2 = \text{empirical variance of the } Y_i.$$

3.3 Voronoi tessellation

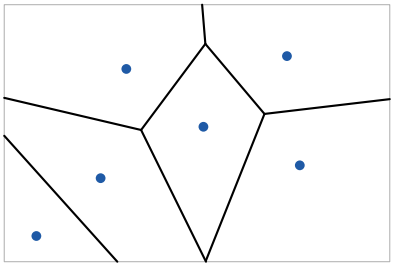


Figure 3: The Voronoi tessellation of the training points: 1-NN predicts a constant value on each cell.

1-NN is piecewise constant (constant on each Voronoi cell), while n-NN is constant.

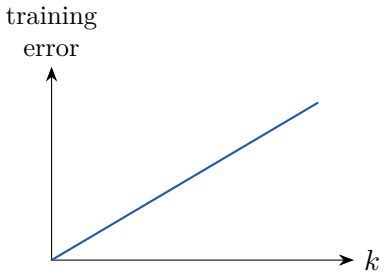


Figure 4: The training error of k-NN grows with k.

3.4 The bias–variance tradeoff

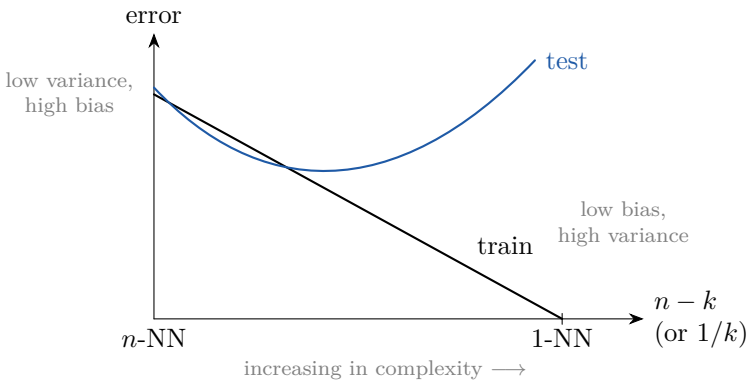


Figure 5: The bias–variance tradeoff for k-NN: training error falls to 0 as k decreases to 1, while test error is U-shaped.

The training error curve might not be exactly linear. Moving from n-NN (low variance, high bias) to 1-NN (low bias, high variance) increases the complexity of the model. This is the "bias–variance" tradeoff.

4 Test Error

But we are really interested in the test error. Given the training data D_n,

$$\text{test error} = \mathop{\mathbb{E}}_{(X,Y)\sim P}\Big[\big(Y - \hat{f}_k(X)\big)^2 \;\Big|\; D_n\Big].$$

Averaging over the training data as well,

$$\text{test error} = \mathop{\mathbb{E}}_{D_n}\mathop{\mathbb{E}}_{(X,Y)}\Big[\big(Y - \hat{f}_{k,n}(X)\big)^2\Big].$$

5 Consistency

Definition. A sequence of regressors $\{g_n\}_{n\geq 1}$ is L_1-*consistent* if

$$\mathop{\mathbb{E}}_{X}|f^*(X) - g_n(X)| \xrightarrow{n\to\infty} 0$$

whenever $\mathbb{E}|Y| < \infty$, where f^* is the oracle regression function. Here n indexes the number of data points.

Remark 1. k-NN with any fixed k is not consistent.

Theorem 1 (Stone's theorem)**.** *If $k_n \to \infty$ and $k_n/n \to 0$, then k_n-NN is L_1-consistent.*

So k_n must increase as a function of n, but it cannot be any constant fraction of n.

We could also look at L_2-*consistency*:

$$\mathop{\mathbb{E}}_{P}|Y - g_n(X)|^2 \xrightarrow{n\to\infty} \mathop{\mathbb{E}}_{P}|Y - f^*(X)|^2.$$

So this is the first non-parametric model, with weak assumptions, that is universally consistent! This is a very powerful result.

5.1 Non-parametric models

A model is *non-parametric* ("not a fixed, finite number of parameters") if:

1. the class of functions grows with the sample size; or
2. the class of functions cannot be represented or parameterized by a fixed, finite number of parameters.

5.2 Classification

With L classes, the guess is a point in the probability simplex over the L classes. If forced to give a single label, take the index of the largest entry. But taking the argmax loses information about confidence.

1 Oracle Regressors and Classifiers

The "best" or oracle regressor for the MSE is

$$f^*(x) := \mathbb{E}[Y \mid X = x].$$

The "best" or oracle classifier (binary) for the 0/1 loss is

$$\mathbf{1}\Big(\mathbb{P}(Y=1 \mid X) > \tfrac{1}{2}\Big) \equiv \underbrace{\begin{cases} 1 & \text{if } \mathbb{P}(Y=1 \mid X) > \mathbb{P}(Y=0 \mid X), \\ 0 & \text{if } \mathbb{P}(Y=0 \mid X) > \mathbb{P}(Y=1 \mid X) \end{cases}}_{\text{argmax}_{k\in\{0,1\}}\, \mathbb{P}(Y=k|X)}.$$

Note that

$$\mathbb{E}[Y \mid X] = 1 \cdot \mathbb{P}(Y=1 \mid X) + 0 \cdot \mathbb{P}(Y=0 \mid X) = \mathbb{P}(Y=1 \mid X).$$

Remark 2. The "best" always depends on the loss function. If it is not mentioned, the implicit choices are:

- regression: the MSE;
- binary classification: the 0/1 loss $\ell(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y}, \\ 0 & \text{if } y = \hat{y}. \end{cases}$

1.1 Other loss functions

What if we use some other loss function that is not the 0/1 loss? For example, take the following loss.

y \ $\hat{y}$	0	1
0	0	50
1	0.1	0

Figure 6: An asymmetric loss $\ell(y, \hat{y})$: predicting $\hat{y} = 1$ when $y = 0$ costs 50, while the opposite mistake costs only 0.1.

In this case, we want to bias towards predicting $\hat{y} = 0$ more often.

In general, with L classes we have a *loss matrix* ℓ, where

$$\ell(i, j) = \text{the loss if the truth is } i \text{ but you say } j.$$

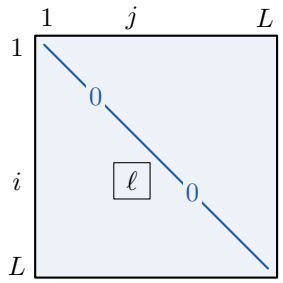


Figure 7: The $L \times L$ loss matrix: the diagonal (correct predictions) is zero, and entry (i, j) is the loss for saying j when the truth is i.

1.2 The Bayes classifier

If this is the 0/1 loss (1 on all off-diagonal entries), then the optimal classifier in this multi-class setting is

$$f^*(x) := \operatorname*{argmax}_{k \in \{1,\dots,L\}} \mathbb{P}(Y = k \mid X = x),$$

the *Bayes classifier*.

Proof. We have

$$\min_{f:\mathcal{X}\to\mathcal{Y}} \mathop{\mathbb{E}}_{(X,Y)\sim P_{XY}} \big[\ell(Y, f(X))\big] = \min_{f:\mathcal{X}\to\mathcal{Y}} \mathop{\mathbb{E}}_{X} \Big[\sum_{k=1}^{L} \underbrace{\ell(k, f(X))}_{0/1 \text{ loss}} \mathbb{P}(Y = k \mid X)\Big].$$

The inner sum can be minimized for each x separately, and it is minimized by $f^*(x)$. This works for the 0/1 loss since choosing $f(x) = k$ zeros out the k-th term, so the loss becomes $\mathbb{P}(Y \neq k \mid X)$:

$$\sum_{k'=1}^{L} \mathbf{1}(k' \neq k)\ \mathbb{P}(Y = k' \mid X) = 1 - \mathbb{P}(Y = k \mid X),$$

which is smallest when we zero out the largest term, that is, when $k = \operatorname{argmax}_k \mathbb{P}(Y = k \mid X) = f^*(X)$. ∎

2 k-NN as Two Approximations

k-NN makes two types of approximations. Writing m for the number of neighbors (since k now indexes the classes), we want

$$f^*(x) = \operatorname*{argmax}_{k \in \{1,\dots,L\}} \mathbb{P}(Y = k \mid X = x).$$

But we don't have this x when we are out of sample, so we approximate

$$\begin{aligned}\mathbb{P}(Y = k \mid X = x) &\approx \mathbb{P}(Y = k \mid X \approx x) && \text{(first approximation)} \\ &\approx \frac{1}{m} \sum_{i \in N_m(x)} \mathbf{1}(Y_i = k) && \text{(second approximation: empirical average),}\end{aligned}$$

where $N_m(x)$ is the set of indices i such that X_i is one of the m nearest neighbors of x.

2.1 Kernel smoothing

But this cuts off pretty sharply. Instead, we could use an exponentially weighted kernel, and take the weighted average

$$\frac{\sum_{i=1}^n h(X_i, x)\, Y_i}{\sum_{i=1}^n h(X_i, x)}, \qquad \text{for example } h(x, x') = e^{-d^2(x,x')},$$

where h is the *kernel*. Stone's theorem applies here too. A reasonable choice of kernel gives a softening of k-NN.

3 Drawbacks of k-NN

Stone's theorem is very general, with no distributional assumptions other than the existence of the first and second moments. But because it assumes so little, it has limitations. What are the drawbacks?

1. **Finite-sample guarantees?** Without assumptions, no. With assumptions, yes.

   The *inductive bias* of a method is its bias towards the kind of functions that are learned. For example, linear regression has a bias for learning linear functions. This is different from *statistical bias*,

   $$\mathop{\mathbb{E}}_{X}[\hat{f}_k(X) - \mathbb{E}[Y \mid X]], \qquad \text{where } \mathbb{E}[Y \mid X] = f^*(X)$$

   (or the same quantity at a point x).

   For k-NN: for small k, it learns a piecewise constant function. For $k = n$, it learns a constant function. But if you think the data is supported on a low-dimensional manifold, this may not be appropriate. Note that the assumptions are implicit in the method.

2. **Memory and computation.**

3. **How to pick k?** By cross-validation.

4. **How to pick the distance d?** For instance, what is the right distance metric for images?

5. **High dimensions.** k-NN does badly here.

4 High Dimensions

Take data uniform on the cube, $U[-1,1]^d$, and $k = n/10$. How large does a sphere around the center have to be to capture k of the points?

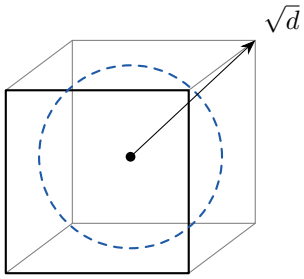


Figure 8: The cube $[-1,1]^d$ with the unit sphere inside it. The farthest point of the cube, a corner, is at distance $\sqrt{d}$ from the center.

The furthest distance from the center to something in a corner is $\sqrt{d}$. When the sphere radius is 1, it already goes past each of the coordinates.

Question: How much volume does the unit sphere have within the cube $[-1,1]^d$?

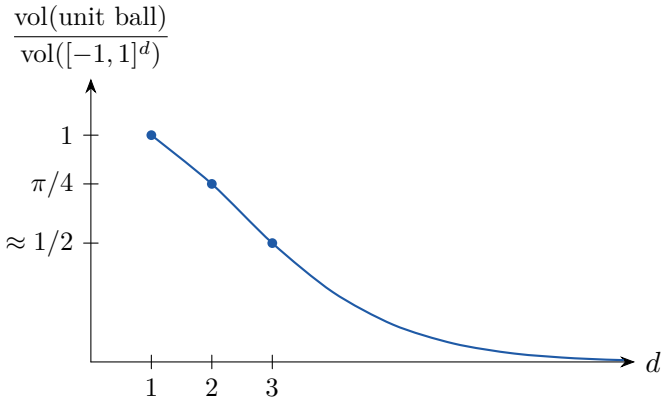


Figure 9: The fraction of the cube $[-1,1]^d$ taken up by the unit ball goes down as d grows.

When d increases, it goes down. For $d = 2$ it is $\pi/4$, and for $d = 3$ it is

$$\frac{\frac{4}{3}\pi}{8} = \frac{\pi}{6} \approx \frac{1}{2}.$$

As you go to higher dimensions, the sphere occupies almost nothing in the cube. In high dimensions, uniform points are mostly not in the sphere, but in the corners!

A high-dimensional cube can be visualized like this:

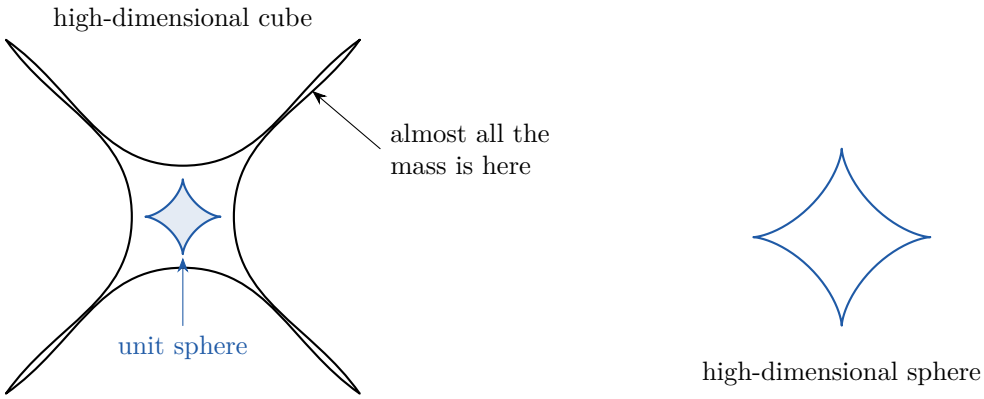


Figure 10: A picture of high dimensions: almost all of the cube's volume is in its corners (the spikes), and the unit sphere takes up a tiny part of it.

Fact 1. The uniform distribution on the unit sphere puts all its mass on the boundary as $d \to \infty$.

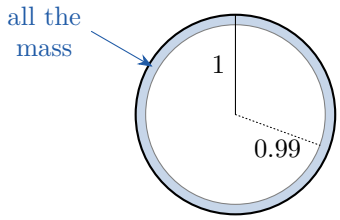


Figure 11: In high dimensions, the thin shell between radius 0.99 and 1 holds almost all of the mass of the unit ball.

In high dimensions, if you shave off the boundary, you lose all the mass. The volume of the ball of radius 0.99 is $c \cdot (0.99)^d$, and the volume of the ball of radius 1 is $c \cdot 1^d$. The ratio goes to 0 as $d \to \infty$.

Fact 2. Almost all the mass is on the equator.

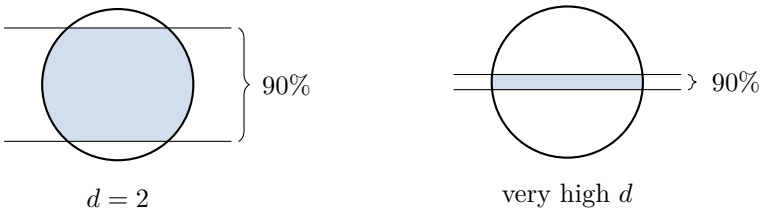


Figure 12: The band around the equator that holds 90% of the mass: wide in two dimensions, very thin in high dimensions.

Next: the volume of the disc at the equator.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 5 — January 31, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Recap

We recapped the questions about high dimensions from last lecture.

Fact 3 (Thomas Cover)**.** The asymptotic risk of the 1-NN classifier is at most 2 times the *Bayes risk*, the optimal (oracle) risk, attained using $\mathbb{P}(Y \mid X)$.

2 Predictive Uncertainty Quantification

Predictive uncertainty quantification tells you how sure you are about a prediction. We will look at two "distribution-free" approaches.

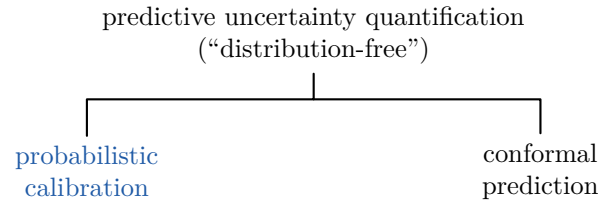


Figure 13: Two distribution-free approaches to predictive uncertainty quantification. This lecture covers probabilistic calibration.

3 Probabilistic Calibration

The original application was weather forecasting.

3.1 Post hoc (look-back) assessment

How do you determine whether a prediction was good, given that it was a probabilistic statement? For a weather forecast, look at all the days, and compare the predictions with what actually happened. That is, on the days that you predicted a 10% chance of rain, did it really rain on only 10% of those days?

For all the days on which the forecaster F predicted "p", did it rain on p% of those days? If yes, then F is *calibrated*.

3.2 Calibration and sharpness

Sharpness ("variation") measures how much the prediction varies. For example, suppose it rains 50% of the time in Pittsburgh, and you just say "50%" all the time without considering anything else. Then your forecast is not sharp, but it is still calibrated. There is a tradeoff between calibration and sharpness.

- **Calibration** can be corrected on the fly when you realize your bias. It doesn't need any assumptions on the distribution.
- **Sharpness** depends on $P_{Y|X} \neq P_Y$, that is, on how informative X is.

An application of calibration: suppose you have a deep neural network (which is not explainable) and it makes a prediction. You can calibrate it, and then interpret its output as a probabilistic statement.

3.3 Post-hoc calibration

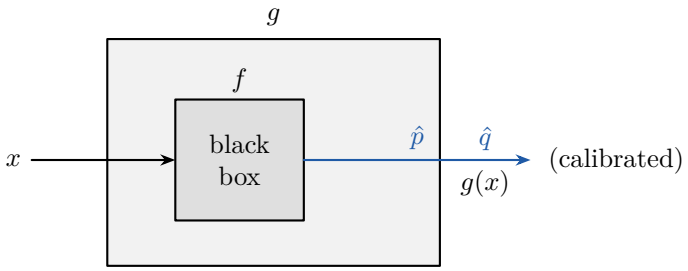


Figure 14: Post-hoc calibration: a wrapper g turns the output $\hat{p}$ of a black-box predictor f into a calibrated prediction $\hat{q} = g(x)$.

Even though k-NN is explainable, it does not have to be calibrated.

4 What Makes a Good Classifier?

We need a loss for probabilistic predictions $p \in [0, 1]$ of a label $y \in \{0, 1\}$.

Brier score.

$$\ell(y, p) = (y - p)^2.$$

It decomposes into a calibration term plus a sharpness term.

Log-loss (*cross-entropy loss*, closely related to the Shannon entropy).

$$\ell(y, p) = -y \log p - (1 - y) \log(1 - p).$$

If $y = p = 1$ or $y = p = 0$, the loss is 0. If $y = 0, p = 1$ or $y = 1, p = 0$, the loss is ∞. So this penalizes overconfident predictions which are wrong.

0/1 **loss.**

$$\ell(y, y') = \mathbf{1}(y \neq y').$$

A classifier can have good 0/1 loss but be very poorly calibrated.

4.1 Calibration

Definition. A (pre-trained) classifier $f : \mathcal{X} \to [0, 1]$ is *calibrated* if

$$\mathop{\mathbb{E}}_{Y|X}[Y \mid f(X)] = f(X).$$

Here $f(X)$ is the prediction, and the left-hand side is the (empirical) average of Y when I predict $f(x)$. In other words, we ask whether

$$\mathbb{E}[Y \mid f(X) = p] \stackrel{?}{=} p \qquad \text{for all } p \in [0, 1].$$

Definition. f is (ε, α)-*calibrated* if

$$\mathop{\mathbb{P}}_{X}\Big(\big|\mathop{\mathbb{E}}_{Y|X}[Y \mid f(X) = p] - p\big| > \varepsilon\Big) \leq \alpha, \qquad \text{where } p = f(X).$$

Equivalently, with probability at least $1 - \alpha$, f is ε-calibrated.

Goal: a *post-hoc calibration algorithm*

$$\mathcal{A} : (f, D_m) \longrightarrow g$$

such that g is (ε, α)-calibrated. Here D_m is independent, labelled calibration data.

Idea: use most of your data to train f, and use the remaining data for calibration.

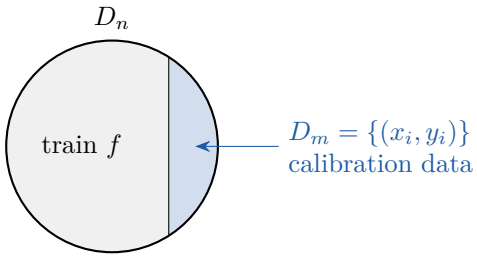


Figure 15: The data D_n is split: most of it trains f, and the rest, D_m, is held out for calibration.

4.2 Histogram binning

Create buckets for the predictions $f(X_i)$, $i \in D_m$: $[0, 0.1], [0.1, 0.2), \ldots, [0.9, 1)$.

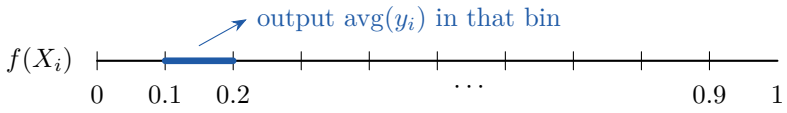


Figure 16: Histogram binning: the range of f is split into ten bins, and on each bin we output the average label of the calibration points that fall in it.

1. Use f to divide D_m into 10 bins:

$$B_j = \{i \in D_m : f(X_i) \in [l_j, u_j]\}.$$

2. On seeing a new point X, don't predict $f(X)$. Instead, predict

$$\frac{1}{|B_j|} \sum_{i \in B_j} Y_i, \qquad \text{where } f(X) \text{ falls in bin } B_j.$$

With this algorithm we can get

$$\varepsilon = \sqrt{\frac{\log(1/\alpha)}{m}},$$

where α trades off with ε.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 6 — February 2, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Histogram Binning

Recall that f is (ε, δ)-*calibrated* if for all distributions P,

$$\mathop{\mathbb{P}}_{X}\Big(\big|\mathbb{E}[Y \mid f(X)] - f(X)\big| > \varepsilon\Big) \leq \delta.$$

The histogram binning procedure is:

1. Divide the (labelled) calibration data into B bins, by dividing $[0, 1]$ based on the values $f(X_i)$.
2. On seeing a test point X_{new}, return the empirical average of the Y's in the bin that $f(X_{\text{new}})$ falls into.

There are two ways to choose the bins:

1. *Equal probability binning*: $[0, 0.1], [0.1, 0.2], \ldots$
2. *Equal mass binning*: the first bin holds the first 0.1 fraction of the values $f(X_i)$, the next bin holds the next 0.1 fraction, and so on.

Theorem 2. *This procedure is*

$$\left(\sqrt{\frac{\log(1/\delta)}{m/B}},\ \delta\right) \textit{-calibrated},$$

where m is the number of calibration points.

The more calibration data you keep aside, the better your ε. The guarantee is independent of the binning strategy, but equal mass binning does better in practice.

- Gupta, Podkopaev, Ramdas [NeurIPS'20] proves an impossibility result: if Range(f) is uncountable, then f cannot be distribution-free asymptotically calibrated.
- Gupta, Ramdas [ICML'21] proves that equal mass binning is valid. Previously it was only a heuristic.

1.1 Reliability diagrams

A *reliability diagram* plots, for each bin of predicted probabilities, the fraction of 1's among the points in that bin. If the predictor is calibrated, the bars should lie on the diagonal line.

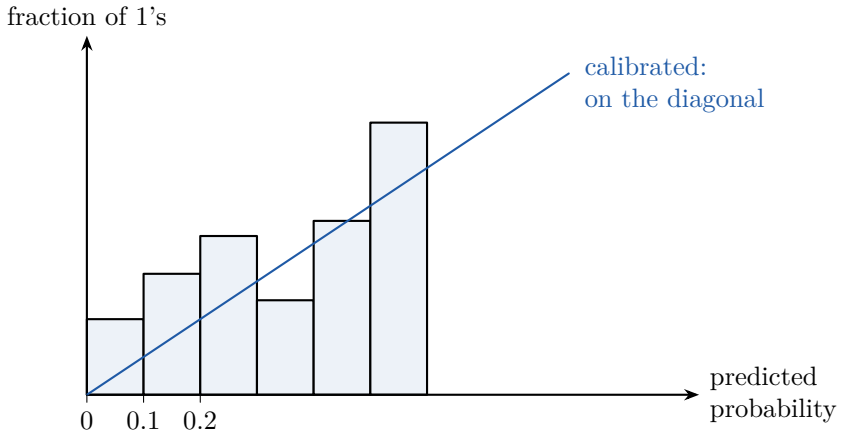


Figure 17: A reliability diagram: each bar is the fraction of 1's in one bin of predicted probabilities. A calibrated predictor has bars on the diagonal.

1.2 Other ways of calibration

1. Isotonic regression.
2. Platt scaling.

Post-hoc calibration can fix calibration, but it can't fix sharpness. If the original f was bad (for example, it always outputs the same thing), then we can't fix the sharpness problem.

2 Split Conformal Prediction (Regression)

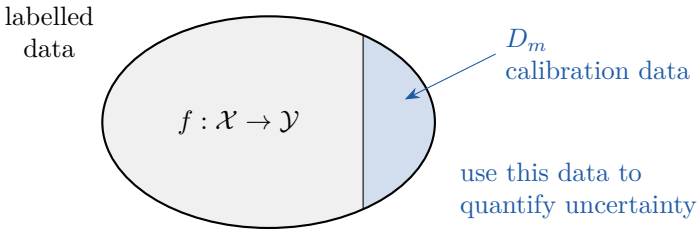


Figure 18: Split conformal prediction: f is trained on most of the labelled data, and the held-out calibration data D_m is used to quantify uncertainty.

Goal: output a set (a *prediction interval*) C_m such that, for all P (that is, distribution-free),

$$\mathbb{P}(Y_{\text{new}} \in C_m(X_{\text{new}})) \geq 1 - \alpha.$$

There are $m+1$ sources of randomness: the m data points used to train C_m, and X_{new}.

So by outputting an interval, we learn about the confidence of the prediction.

Definition (Vovk 2000)**.** Given

1. data $(X_1, Y_1), \ldots, (X_n, Y_n) \overset{\text{iid}}{\sim} P$, and a test point $(X, Y) \overset{\text{iid}}{\sim} P_{XY}$;
2. an algorithm
$$\mathcal{A} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \longmapsto (\mathcal{X} \to \mathcal{Y}),$$

produce a set $C : \mathcal{X} \to 2^{\mathcal{Y}}$ such that

$$\mathbb{P}(Y \in C(X)) \geq 1 - \alpha$$

(and this claim holds for any P and $\mathcal{A}$).

Ideal goal:

$$\mathbb{P}(Y \in C(X) \mid X = x) \geq 1 - \alpha.$$

But if this holds for almost surely all x, then $\mathcal{L}(C(x)) = \infty$ almost surely. This is too good to be true, so we settle on our definition. [On the impossibility of distribution-free conditional inference: Barber, Candès, Ramdas, Tibshirani, *Information and Inference* 2020.]

2.1 The split conformal algorithm

1. Calculate the *residuals*
$$R_i = |Y_i - f(X_i)|, \qquad i \in D_m,$$
where D_m is the calibration data and f is trained on the other $n - m$ points. If Y is not real valued, define a score $S_f(X_i, Y_i)$ instead.
2. Output
$$C(x) = \left[f(x) \pm q_{1-\alpha}\{R_i\}_{i \in D_m} \right] = \left\{ y \in \mathcal{Y} : S_f(x, y) \leq q_{1-\alpha}\{R_i\}_{i \in D_m} \right\},$$
where $q_{1-\alpha}$ is the $1 - \alpha$ quantile.

To compute the quantile, sort the residuals, then take $R_{\lceil (1-\alpha)(m+1) \rceil}$. This is the $1 - \alpha$ quantile value: $1 - \alpha$ of the time, the residual is less than this quantity.

But $|\cdot|$ is not a great score function: it assumes that the errors are homogeneous everywhere, but they are actually not in practice.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 7 — February 7, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Leo Breiman Week

This week we discuss ideas from Leo Breiman:

1. bagging;
2. random forests;
3. stacking.

2 Bagging (Bootstrap Aggregation)

Goal: reduce variance (typically not bias).

Let $Z = (X, Y)$, and let the original dataset be

$$Z^{1:N} = Z_1^N = Z^N = \{\underbrace{(X_1, Y_1)}_{Z_1}, \ldots, \underbrace{(X_N, Y_N)}_{Z_N}\},$$

where the first three expressions are different notations for the same thing.

Definition (Bootstrap sample)**.** The *bootstrap sample* Z^* is obtained by sampling with replacement N times from Z^N.

So we pretend that Z^N is the empirical distribution, and sample N times from it i.i.d. The bootstrap sample Z^* is the same size as Z^N, and is the multiset

$$Z^* = \{(X_1^*, Y_1^*), \ldots, (X_N^*, Y_N^*)\}.$$

Do this B times, each of size N, to get

$$Z^{*1}, Z^{*2}, \ldots, Z^{*B}.$$

Question: is $Z^{*1} \perp\!\!\!\perp Z^{*2}$? No: in the graphical model they share the common parent Z^N.

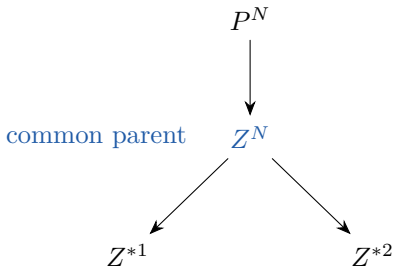


Figure 19: Graphical model of two bootstrap samples: both are drawn from the common parent Z^N, which is drawn from P^N.

But

$$Z^{*1} \perp\!\!\!\perp Z^{*2} \mid Z^N!$$

2.1 The idea of bagging

Suppose you have an algorithm $\mathcal{A}$ that produces a classifier, $\mathcal{A}(Z^N) \to f$. For $b = 1, \ldots, B$, let

$$\mathcal{A}(Z^{*b}) \to f^b,$$

and output

$$f_{\text{bag}} = \frac{\sum_{b=1}^{B} f^b}{B}.$$

This cannot change the bias, but will change the variance.

2.2 Why averaging reduces variance

Let $A_1, \ldots, A_K$ be (dependent) random variables with

$$\text{mean}(A_i) = \mu, \qquad \text{var}(A_i) = V, \qquad \text{cor}(A_i, A_j) = \rho.$$

If $\rho = 0$, then

$$\text{Var}\left(\frac{\sum_{k=1}^{K} A_k}{K}\right) = \frac{V}{K}, \qquad \text{Mean}\left(\frac{\sum_{k=1}^{K} A_k}{K}\right) = \mu \text{ still.}$$

So averaging helps with concentration. For general ρ,

$$\text{Var}\left(\frac{\sum_{k=1}^{K} A_k}{K}\right) = \rho V + \frac{(1-\rho)V}{K} \leq V.$$

For bagging, we have $Z^{*1} \overset{\text{dist}}{=} Z^{*2}$, and

$$\text{cor}(Z^{*1}, Z^{*2}) = \text{cor}(Z^{*3}, Z^{*4}),$$

so the f^b have a common variance and a common correlation ρ, and

$$\text{var}(f_{\text{bag}}) = \rho \,\text{var}(f^b) + (1-\rho)\frac{\text{var}(f^b)}{B} < \text{var}(f^b).$$

So whatever algorithm you have, you can just bag it for lower variance. The cost is computational: B can be as large as your computation can afford.

3 Algorithmic Stability

The reference is Bousquet and Elisseeff (2002). An algorithm is stable if changing a couple of points does not change the output by too much.

Definition. $\mathcal{A}$ is *stable* if $\mathcal{A}(Z^N) \approx \mathcal{A}(Z'^N)$, where Z^N and Z'^N are *neighboring datasets*, which differ in only one observation.

This notion is used widely in differential privacy.

Definition. $\mathcal{A}$ is (ε, δ)-*stable* if

$$\mathop{\mathbb{P}}_{\substack{Z,Z' \\ (X,Y)\sim P}} \left(|f(X) - f'(X)| > \varepsilon\right) \leq \delta,$$

where f is trained on Z and f' is trained on Z'.

There are many different definitions, but they share the same idea.

- k-NN is generally a stable algorithm, and its stability increases with k.
- Many natural algorithms are stable.
- Not stable: OLS (ordinary least squares) when $N \approx d$. Then $(X^TX)^{-1}X^TY$ is low rank and ill-conditioned (the condition number $\sigma_{\max}/\sigma_{\min}$ is large). OLS is provably unstable in this region.

A recent paper (Soloff, Barber, Willett) gives a distribution-free result: "bagging improves stability".

We usually bag decision stumps or trees.

4 Decision Stumps, Lists, Trees and Forests

Definition (Decision stump)**.** A *decision stump* is the classifier

$$\text{if } X_j > b \text{ then 1 else 0.}$$

For a categorical feature, it is

$$\text{if } X_j \in \{a, b\} \text{ then 1 else 0.}$$

From stumps we build decision lists, decision trees, and forests.

4.1 Decision stumps

How do we choose j and b? Choose the one with the lowest empirical risk (ERM). There are d choices of the coordinate j and $N+1$ choices of the threshold b, and the comparison can go either way ($<$ or $>$), so in total we need to try $2(N+1)d$ points.

4.2 Decision lists

For a test point X:

$$\begin{aligned}&\text{if } X_j > b \text{ then } 1\\&\text{else if } X_i > c \text{ then } 0\\&\text{else } 1.\end{aligned}$$

Each cut is a stump. A list keeps subdividing the space.

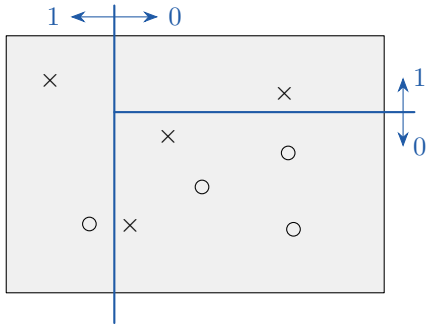


Figure 20: A decision list on two features: each cut is a stump, and the list keeps subdividing the space.

4.3 Decision trees

A *decision tree* can also add stumps in the "then" statement.

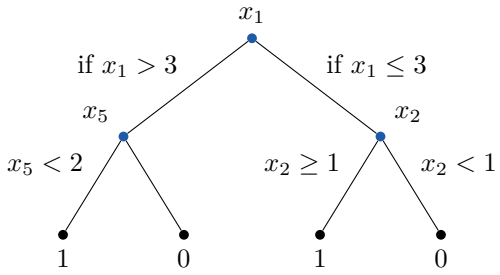


Figure 21: A decision tree: each internal node is a stump on one feature, and each leaf is a label.

Lecture 8 — February 9, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Outline

1. Greedy fitting of decision trees.
2. Inductive bias of trees.
3. Random forests (classification vs. regression).
4. OOB (out of bag) error.

2 Decision Trees

A decision tree splits the space into non-overlapping rectangles.

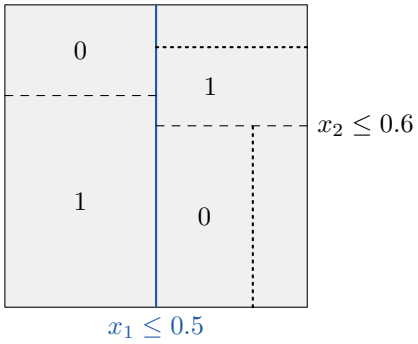


Figure 22: A decision tree on $[0,1]^2$: the first split is $x_1 \leq 0.5$, and further splits (dashed, then dotted) cut each side into smaller rectangles, each with a predicted label.

A tree of depth d computes

$$f(x) = \sum_{k=1}^{2^d} A_k \, \mathbf{1}(x \in R_k),$$

where the R_k are non-overlapping rectangles.

How do we evaluate a test point? If a test point falls into leaf 2, output the empirical average of the labelled data in leaf 2.

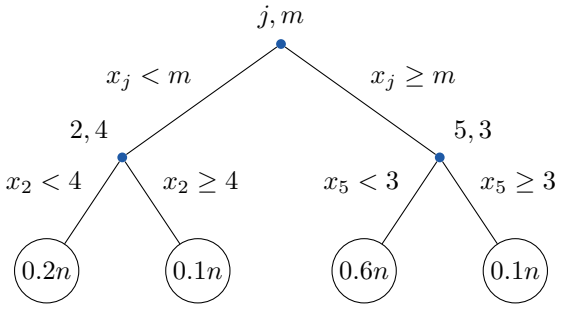


Figure 23: A depth-2 decision tree. Each internal node is labelled by the coordinate and threshold it splits on, and each leaf by the fraction of the n data points that fall into it.

There are $O(dn^3)$ possible decision trees in this case.

2.1 Inductive bias

The *inductive bias* is due to the choice of model, not the data. When trying to fit data that is not axis-aligned, we need a large tree depth.

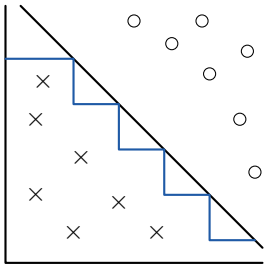


Figure 24: When the true boundary is diagonal, an axis-aligned tree can only approximate it by a staircase of many splits.

This is also why there is no one "best" ML algorithm: they all have their inductive biases. For example, the inductive bias of CNNs works well for learning functions that can classify images.

2.2 A sensible heuristic

1. Find the best root split.
2. Split D_n into left and right branches.

Recurse until the leaves are pure.

Classification metric. Use the "purity" or entropy (or the *Gini impurity*) of a discrete distribution on L labels:

$$-\sum_{l=1}^{L} p_l \log p_l.$$

Facts:

- It is maximized when $p_l = 1/L$ (uniform), where it equals $\log L$.
- It is minimized if $p_l = 1$ (point mass).

We want to find a rule that minimizes entropy.

If this is *regression* instead of *classification*, look at the MSE instead of the entropy.

3 Decision Trees Are Very Variable

If you change just one datapoint, the rules that are found can be very different.

3.1 Bagged decision trees

But recall from bagging: if the trees are correlated, the variance reduction is not as much:

$$\text{Var}(\text{avg}) = \rho V + (1-\rho)\frac{V}{B},$$

where ρ is the pairwise correlation.

4 Random Forests

A *random forest* (RF) is bagging decision trees, but first reducing the correlation between trees.

One idea: at each split, randomly pick m out of p covariates (often $m = \sqrt{p}$), and only split on those m.

- When $m = 1$: bad splits, since we are forced to use the given coordinate.
- When $m = p$: the same problem as before.

We can define the correlation between trees as how they behave on inputs:

$$\mathop{\mathbb{E}}_{x \sim P_X}\big[f_1(x) f_2(x)\big].$$

RF works well when the problem is dense, since then there are a lot of other good variables to split on. It works poorly when the problem is sparse (this affects the approximation error by a lot).

RF also loses interpretability, like neural networks, but it is more flexible in what it can take in. RF can be trained very quickly, and is quite hard to beat. You don't even need domain knowledge of the data.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 9 — February 14, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Stacking

We have the bound

$$\frac{1}{K}\sum_{v=1}^{K}\mathbb{E}\left\|\hat{\mu}(x)-\mu(x)\right\|^2 \leq (1+\delta)\ \mathbb{E}\min_{\alpha}\sum_{v}\mathbb{E}\left\|\hat{\mu}_\alpha-\mu\right\|^2 + C\,\frac{V\log n}{n}.$$

Reference: arXiv 2301.03962.

Consider an ensemble of models $q_1(x),\ldots,q_m(x)$. With leave-one-out,

$$\text{loss} = \text{average bias} + \text{average variance} - \text{diversity}.$$

For the squared loss, let

$$\bar{q}(x) = \frac{1}{m}\sum_{i=1}^{m} q_i(x).$$

Then

$$\mathbb{E}(\bar{q}-y)^2 = \frac{1}{m}\sum_i\big(\mathbb{E}[q_i]-y\big)^2 + \frac{1}{m}\sum_i\mathbb{E}\big[q_i-\mathbb{E}[q_i]\big]^2 - \mathbb{E}\left[\frac{1}{m}\sum_i(q_i-\bar{q})^2\right].$$

This makes precise the notion of diversity.

For a general loss, define

$$q^* = \arg\min_z \mathbb{E}\big[V(z,q)\big].$$

Then

$$\mathbb{E}\ L(y,\bar{q}) = \underbrace{\frac{1}{m}\sum_i L(y,q_i^*)}_{\text{average bias}} + \underbrace{\frac{1}{m}\sum_i\mathbb{E}\ V(q_i^*,q_i)}_{\text{average variance}} - \underbrace{\frac{1}{m}\sum_i V(\bar{q},q_i)}_{\text{diversity}}.$$

Lecture 10 — February 16, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Model Selection

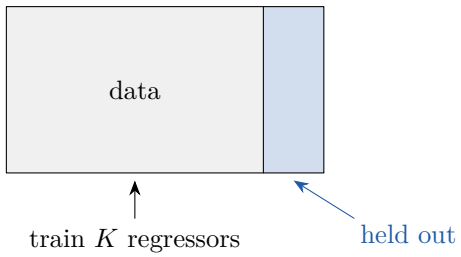


Figure 25: Model selection with a held-out set: train K regressors on most of the data, and compare them on the held-out part.

1.1 Cross-validation

10-*fold cross-validation (CV)*: split the data into 10 buckets of equal size, train on 9 of them, and evaluate on the remaining 1.

At the extreme end, this is $LOOCV$, leave-one-out cross-validation. The risk of the k-th model is

$$R^{(k)} = \sum_{i=1}^{n} (Y_i - \hat{f}_{-i}^{(k)}(X_i))^2.$$

That is, for the k-th model, leave out the i-th point, train on the rest, and evaluate on that point. Then choose

$$\hat{k} = \arg\min_{k \in [K]} R^{(k)}.$$

1.2 When is LOOCV used?

A concern is that LOOCV is computationally expensive. But it is sometimes fast to compute.

Linear smoothers. These need not be OLS: they can be a linear combination of any basis. Linear means

$$\hat{y} = HY.$$

For linear smoothers, the out-of-sample loss can be computed efficiently.

Random forests. Another class for which it is easy is RF, using the *OOB ("out of bag")* error: approximate $\hat{f}_{-i}$ by the subforest of trees that never saw (X_i, Y_i).

1.3 Risk as a function of complexity

Consider choosing among 1-NN, 5-NN, 10-NN, 50-NN, and so on.

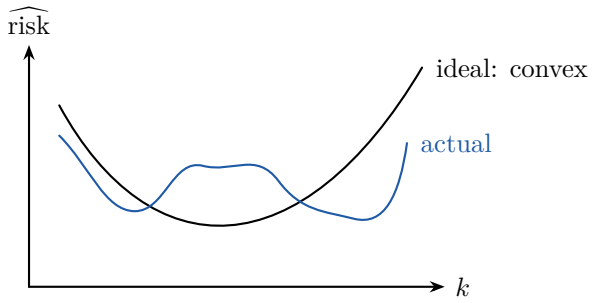


Figure 26: The estimated risk as a function of the complexity parameter k: it would be nice if it were convex, but it need not even be unimodal.

It would be nice if the risk as a function of the complexity parameter were convex. But we don't know if it is even convex or unimodal.

It is hard to have a general theory on which model to select. In general, model averaging performs better than model selection.

2 Boosting

We will look at boosting from a game-theoretic point of view.

2.1 Game theory

For the fixed game M in class,

$$\max_{r\in[R]} \min_{c\in[C]} M(r,c) = 0.$$

The play order is left to right (r moves first, then c), and the solve order is right to left.

Theorem 3. *For pure strategies,*

$$\min_{c\in[C]} \max_{r\in[R]} M(r,c) \geq \max_{r\in[R]} \min_{c\in[C]} M(r,c).$$

Proof. For all r and c,

$$\max_{r'} M(r',c) \geq M(r,c).$$

Taking the minimum over c on both sides,

$$\min_{c} \max_{r'} M(r',c) \geq \min_{c} M(r,c).$$

Since this holds for any r on the right-hand side, in particular it also holds for the maximizing r:

$$\min_{c} \max_{r} M(r,c) \geq \max_{r} \min_{c} M(r,c). \qquad \blacksquare$$

2.2 Mixed strategies

Let Δ_R and Δ_C be the sets of distributions over the actions of the two players.

Theorem 4.

$$\min_{q \in \Delta_C} \max_{p \in \Delta_R} p^T M q = \max_{p \in \Delta_R} \min_{q \in \Delta_C} p^T M q = V^*,$$

where V^* *is the* value of the game.

An equivalent formulation:

$$\begin{aligned} \exists\, p \in \Delta_R \;\; \forall\, q \in \Delta_C : \quad & p^T M q \geq V^*, \\ \exists\, q \in \Delta_C \;\; \forall\, p \in \Delta_R : \quad & p^T M q \leq V^*. \end{aligned}$$

Lecture 11 — February 21, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

Recall: the minimax theorem. **Today:** boosting.

1 Edge of a Classifier

Consider a binary classifier h that outputs -1 or 1.

Definition (Edge)**.** The *edge* of a classifier h is

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

If $h(X_i) = Y_i$ for all i, then the edge is 1. If $h(X_i)$ is independent of Y_i, then the edge is approximately 0. So the edge describes how much better than chance a classifier is. The error can be viewed as

$$\frac{1}{2}\,(1 - \text{edge}).$$

Definition (Weighted edge)**.** The *weighted edge* of h is

$$\sum_{i=1}^{n} w_i Y_i\, h(X_i), \qquad \text{where } \sum_{i=1}^{n} w_i = 1 \text{ and } w_i > 0 \text{ for all } i \in \{1, \ldots, n\}.$$

That is, w lies in the n-dimensional simplex.

2 Weak Learning Hypothesis

Let $\mathcal{H}$ be a base set of classifiers (for example, decision trees), with $|\mathcal{H}| = H$.

Definition (Weak learning hypothesis)**.** There exists $\gamma > 0$ such that for any set of weights w, there is a classifier $h \in \mathcal{H}$ with weighted edge at least γ.

Say $\gamma = 0.1$. Then for any set of weights, we can find a decision tree that is not perfect, but has weighted edge at least γ, that is, it is better than chance.

Definition (Strong learning hypothesis)**.** There exists a classifier in $\text{span}(\mathcal{H})$ with zero training error.

These are hypotheses about the data at hand.

The breakthrough: weak learning implies strong learning!

2.1 Weak learning implies strong learning

There is a large but finite set of classifiers (for example, decision trees) that we can build on the data points. Arrange them as the rows of a matrix whose columns are the data points $Z_i = (X_i, Y_i)$.

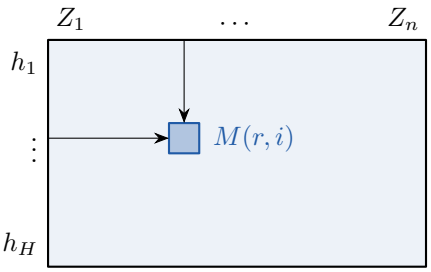


Figure 27: The game matrix: row r is a classifier h_r, column i is a data point, and entry (r, i) is $h_r(X_i)Y_i$.

Let $M(r,i) = h_r(X_i)\, Y_i$, and write Δ_n and Δ_H for the simplices over the data points and the classifiers.

Weak learning:

$$\forall\, w \in \Delta_n \;\; \exists\, h \in [H]: \quad e_h^T M w \geq \gamma, \qquad \text{where } e_h^T M w = \sum_{i=1}^{n} w_i Y_i\, h(X_i).$$

Strong learning:

$$\exists\, p \in \Delta_H \;\; \forall\, w \in \Delta_n: \quad p^T M w \geq \gamma,$$

that is,

$$\exists\, p \in \Delta_H \;\; \forall\, i \in [n]: \quad p^T M e_i \geq \gamma.$$

Then

$$f(X_i) = \operatorname{sign}\Big(\sum_{k=1}^{H} p_k h_k(X_i)\Big)$$

has zero training error, for some $p \in \Delta_H$.

So by the minimax theorem, we get the theorem! Weak learning says $\max_{p \in \Delta_H} p^T M w \geq \gamma$ for every $w \in \Delta_n$, so $\min_w \max_p p^T M w \geq \gamma$. By the minimax theorem, $\max_p \min_w p^T M w \geq \gamma$, which is strong learning.

Why does this mean that f gets everything correct? For all i,

$$p^T M e_i = \Big(\sum_{k=1}^{H} p_k h_k(X_i)\Big) Y_i \geq \gamma > 0,$$

so the vote always has the same sign as Y_i: f answers correctly on all points.

Question: how do we find this mixture $p \in \Delta_H$ of classifiers?

3 AdaBoost

Algorithm 1: AdaBoost

Initialize the weights $w_i = 1/n$ for $i = 1, \dots, n$.
for $m = 1, \dots, M$ **do**
  Compute the weighted error of each $h \in \mathcal{H}$: $\varepsilon(h) = \sum_{i=1}^{n} w_i \, \mathbb{I}\{Y_i \neq h(X_i)\}$.
  Find a classifier $h_m = \arg\min_{h \in \mathcal{H}} \varepsilon(h)$, or pick any h with nontrivial edge.
  Compute $\alpha_m = \frac{1}{2} \log\Big(\frac{1-\varepsilon_m}{\varepsilon_m}\Big)$, where $\varepsilon_m = \varepsilon(h_m)$ is the error of h_m.
  Update the weights as $w_i \leftarrow \dfrac{w_i \, e^{-\alpha_m Y_i h_m(X_i)}}{Z_m}$, where Z_m is a normalization constant.
return $f(x) = \operatorname{sign}\Big(\sum_{m=1}^{M} \alpha_m h_m(x)\Big)$.

The i-th point gets upweighted if it is classified wrongly, and downweighted otherwise. The final classifier f takes the sign of a combination of the h_m weighted by the α_m (a convex combination once the weights are normalized, which does not change the sign).

It turns out that

$$\begin{aligned} Z_m &= \sum_{i=1}^{n} w_i \, e^{-\alpha_m Y_i h_m(X_i)} = (1-\varepsilon_m)\, e^{-\alpha_m} + \varepsilon_m \, e^{\alpha_m} \\ &= (1-\varepsilon_m)\sqrt{\frac{\varepsilon_m}{1-\varepsilon_m}} + \varepsilon_m \sqrt{\frac{1-\varepsilon_m}{\varepsilon_m}} \\ &= 2\sqrt{\varepsilon_m(1-\varepsilon_m)}. \end{aligned}$$

Theorem 5. *The empirical risk $\widehat{R}(f)$ of the AdaBoost algorithm satisfies*

$$\begin{aligned} \widehat{R}(f) &\leq \exp\Big(-2 \sum_{m=1}^{M} \Big(\frac{1}{2} - \varepsilon_m\Big)^2\Big) \\ &\leq \exp(-2M\gamma^2) \quad \textit{if the weak learning hypothesis is true.} \end{aligned}$$

Since $\widehat{R}(f)$ cannot be strictly between 0 and $1/n$, this means that if $M > \frac{\log n}{2\gamma^2}$, then $\widehat{R}(f) < \frac{1}{n}$, and hence $\widehat{R}(f) = 0$.

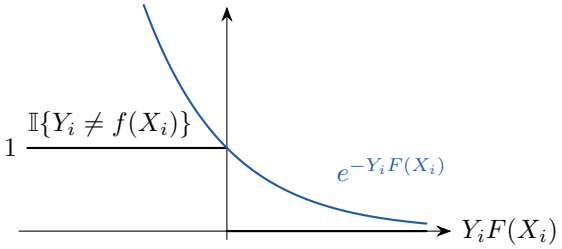


Figure 28: The exponential surrogate loss lies above the 0–1 loss.

Proof. Write $F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$, so that $f = \operatorname{sign}(F)$. Bounding the 0–1 loss by the exponential surrogate loss,

$$\begin{aligned}
\widehat{R}(f) &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\big(Y_i \neq f(X_i)\big) \\
&\leq \frac{1}{n}\sum_{i=1}^{n} \overbrace{\exp\big(-Y_i F(X_i)\big)}^{\text{surrogate loss}} \\
&= \frac{1}{n}\sum_{i=1}^{n} \Big[n \prod_{m=1}^{M} Z_m\Big] w_{M+1}(i) \\
&= \prod_{m=1}^{M} Z_m,
\end{aligned}$$

where the last step uses $\sum_i w_{M+1}(i) = 1$, and the step before it uses

$$w_{M+1}(i) = \frac{w_M(i)\, e^{-\alpha_M Y_i h_M(X_i)}}{Z_M} = \frac{e^{-Y_i \sum_m \alpha_m h_m(X_i)}}{n \prod_{m=1}^{M} Z_m}$$

by recursive unrolling (with $w_1(i) = 1/n$).

But then again, we had

$$Z_m = 2\sqrt{\varepsilon_m(1-\varepsilon_m)} = \sqrt{1 - 4\Big(\frac{1}{2} - \varepsilon_m\Big)^2} \leq \exp\Big(-2\Big(\frac{1}{2} - \varepsilon_m\Big)^2\Big),$$

using $\sqrt{1-x} \leq e^{-x/2}$. Multiplying over m gives the theorem. ∎

AdaBoost is implicitly minimizing the surrogate loss, so it does not actually matter whether the weak learner hypothesis is true or not.

Lecture 13 — February 28, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Boosting: VC Theory Is Unsatisfactory

In boosting, mistakes made by earlier classifiers are the focus of later classifiers.

1.1 An interesting phenomenon

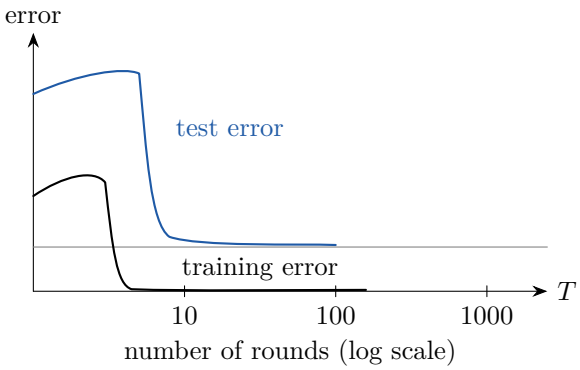


Figure 29: Training and test error of boosting against the number of rounds T, on a log scale. The training error quickly hits 0; the test error drops later and plateaus above 0.

1. The error does not decrease monotonically!
2. This is a typical graph: the training error hits 0 very quickly.
3. The test error will plateau (above 0) as well (but could also go up).

Why is my classifier still improving out of sample, even though it is no longer improving in sample?

Why does the error go up initially? Because we don't have any theorem saying it should go down.

1.2 The surrogate loss

Recall the surrogate loss in the proof from last time:

$$\begin{aligned}\widehat{R}(f) &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(Y_i \neq f(X_i)) \\ &\leq \frac{1}{n}\sum_{i=1}^{n}\underbrace{\exp(-Y_i f(X_i))}_{\text{surrogate loss}}.\end{aligned}$$

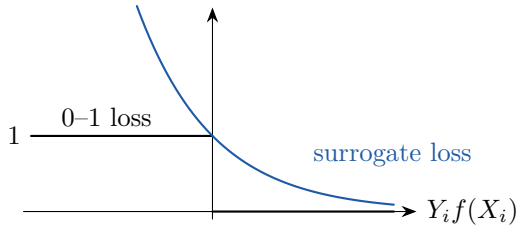


Figure 30: The exponential surrogate loss $\exp(-Y_i f(X_i))$ lies above the 0–1 loss.

The surrogate loss heavily penalizes mistakes from overconfident predictions, and still penalizes a correct prediction that is not confident.

2 Convex Surrogate Loss Minimization by Coordinate Descent

Let

$$\widehat{R}(\beta) = \frac{1}{n}\sum_{i=1}^{n} \exp\Big(-Y_i \sum_{j=1}^{H} \beta_j h_j(X_i)\Big).$$

AdaBoost solves

$$\min_{\beta \in \mathbb{R}_+^H} \widehat{R}(\beta) = \min_{f \in \mathrm{span}(\mathcal{H})} \widehat{R}(f)$$

by coordinate descent:

1. Begin at $\beta^{(0)} = [0, 0, \ldots, 0]$.
2. At step t, pick a direction $e_t \in \{e_j\}_{j \in J}$ (where $J = [H]$ indexes the base classifiers) and a step size $\alpha_t \geq 0$ to minimize
$$\widehat{R}(\beta^{(t-1)} + \alpha_t e_t).$$
3. The gradient with respect to coordinate j is
$$\widehat{R}'(\beta^{(t-1)})_j \propto (2\varepsilon_{t,j} - 1) \prod_{s=1}^{t-1} Z_s,$$
where $\varepsilon_{t,j}$ is the weighted error of h_j.
4. h_t is chosen to minimize the weighted error, and the optimal step size happens to equal $\log(\frac{1-\varepsilon_t}{\varepsilon_t})$.

In practice, you want to find the best coordinate without having to evaluate all of them.

3 Margin

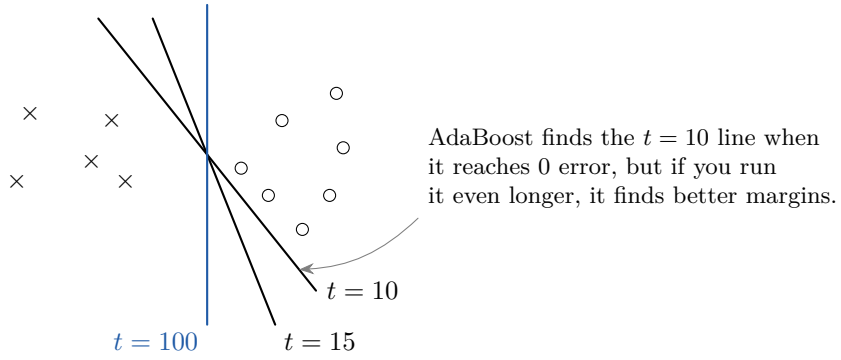


Figure 31: Separators found by AdaBoost after $t = 10$, 15 and 100 rounds. All have zero empirical risk; later ones separate the classes with a larger margin.

Consider two classifiers that both have 0 empirical risk. Which is better?

In the surrogate loss $\exp(-Y_i f(X_i))$, the quantity $Y_i f(X_i)$ is the *margin*. So not only do you want the sign of $f(X_i)$ to agree with Y_i, you also want $f(X_i)$ to be large.

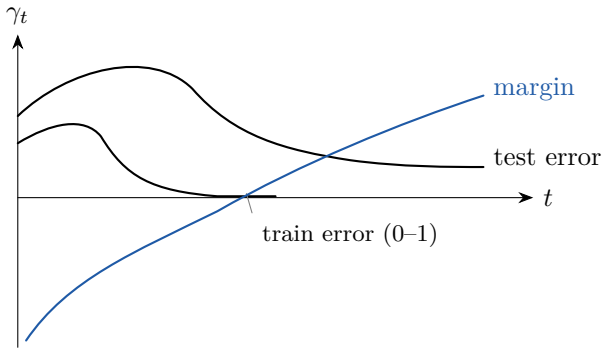


Figure 32: As boosting runs, the training error reaches 0 while the margin keeps growing, and the test error keeps falling before it plateaus.

4 VC Theory

For all $f \in \mathcal{F}$, with probability at least $1 - \delta$,

$$\text{true risk}(f) \leq \text{empirical risk}(f) + \sqrt{\frac{d}{n}\log\left(\frac{n}{d}\right) + \log\left(\frac{1}{\delta}\right)},$$

where $d = \text{VC}(\mathcal{F})$ is the VC dimension of $\mathcal{F}$ and n is the number of examples.

For boosting,

$$\text{VC(boosted class)} \leq O(dT),$$

where $d = \text{VC(base class)}$ and T is the number of iterations.

5 Empirical Rademacher Complexity

Definition (Empirical Rademacher complexity)**.** The *empirical Rademacher complexity* of $\mathcal{F}$ is

$$\widehat{\mathcal{R}}_S(\mathcal{F}) = \mathop{\mathbb{E}}_{\sigma}\Big[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(X_i)\Big].$$

We can think of σ as data labels. This measures the complexity of a function class: its ability to fit random noise.

If you get a classifier that gets 0 training error, how surprised should you be? Well, if your function class can fit any noise, then this is not surprising, and you may not generalize.

Useful fact:

$$\widehat{\mathcal{R}}_S\big(\text{conv}(\mathcal{F})\big) = \widehat{\mathcal{R}}_S(\mathcal{F}).$$

This is good, because using convex combinations of $\mathcal{F}$ doesn't hurt boosting, whereas with the VC dimension it does.

Lecture 14 — March 2, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Margin and Boosting

Theorem 6 (Margin-based theorem)**.** *For all $\rho > 0$ (the "margin"),*

$$\mathbb{P}\Big[\forall\, h \in \text{conv}(\mathcal{H}) :\ R(h) \le \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\,\widehat{\mathcal{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{n}}\Big] \ge 1-\delta,$$

where $R(h)$ is the true risk, $\widehat{R}_{S,\rho}(h)$ is the empirical margin risk, and $\widehat{\mathcal{R}}_S(\mathcal{H})$ is the empirical Rademacher complexity of $\mathcal{H}$,

$$\widehat{\mathcal{R}}_S(\mathcal{H}) = \mathop{\mathbb{E}}_{\sigma}\Big[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h(X_i)\Big].$$

Think of ρ as the margin, n as the number of data points, and δ as the failure probability.

The *empirical margin risk* is

$$\widehat{R}_{S,\rho}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell_\rho(Y_i h(X_i)),$$

where ℓ_ρ is the *margin loss*:

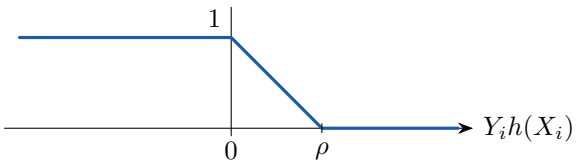


Figure 33: The margin loss ℓ_ρ: it is 1 for wrong predictions, and decreases linearly to 0 as the margin goes from 0 to ρ.

We want confidence in our answers when they are correct; otherwise we are still penalized.

Idea: we need $Y_i h(X_i) \ge \rho$ to pay no price for correct predictions.

Idea: the empirical margin risk can still be large if $\widehat{\mathcal{R}}_S(\mathcal{H})$ is too small and $\mathcal{H}$ is still simple. If $\mathcal{H}$ is too large, $\widehat{\mathcal{R}}_S(\mathcal{H})$ is large.

For the proof, see Mohri et al.

2 Recap of the First Half of the Course

	kNN	DS	DL	DT	RF	Bagging	Boosting	Stacking
A (algorithm)	✓	✓	✓	✓	✓	✓	✓	✓
B (bias–variance tradeoff)								
C (computation, calibration, conformal) D (data)								
E (explainability)	✓	✓	✓	✓	×	×	×	×

Table 1: The methods of the first half against the ABCDE themes. DS, DL and DT are decision stumps, decision lists and decision trees.

2.1 A: Algorithms

Decision trees

How do we split? Algorithm: choose a split that minimizes entropy.

Choosing splits: it is hard to get a perfect split, so we use entropy to decide what a good split is.

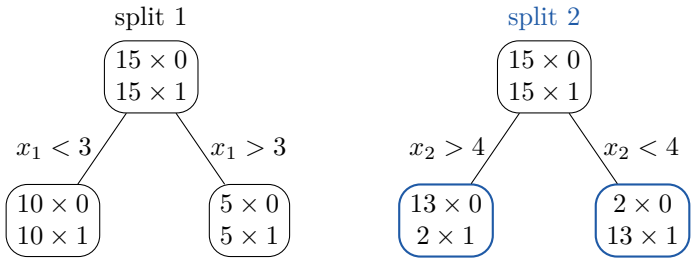


Figure 34: Two candidate splits of 15 points of class 0 and 15 of class 1. Split 1 leaves both children evenly mixed; split 2 gives nearly pure children, so it has lower entropy.

Stacking

Algorithm: take a convex combination of predictors. Solve

$$\min_{\substack{w\in\mathbb{R}_+^k \\ w_j\geq 0,\ \sum_{j=1}^k w_j=1}} \frac{1}{n}\sum_{i=1}^{n}\Big(Y_i - \sum_{j=1}^{k} w_j f_j^{(-i)}(X_i)\Big)^2,$$

where j indexes the models and i indexes the points. The sum over j is a weighted average, and $f_j^{(-i)}$ is the leave-one-out predictor. Then

$$\text{Final}(x) := \sum_{j=1}^{k} \widehat{w}_j f_j(x).$$

The weights $\widehat{w}$ optimize for the leave-one-out error. If one model dominates the others, stacking will learn to put all the weight on it.

Boosting

Coordinate descent.

Bagging

Bootstrap aggregation.

2.2 B: Bias–variance tradeoff

k-NN.

- Large k: high bias, low variance.
- Small k: low bias, high variance.

Decision trees (in terms of depth).

- Higher depth: low bias, high variance.
- Lower depth: high bias, low variance.

Splitting on variables (in random forests).

- Too few variables: high bias, because we split on irrelevant variables and don't get to split on the good ones.
- More variables: hurts the variance reduction between the trees.

Boosting (number of iterations). In practice, how many iterations should we run? See when the risk plateaus on the test set (we are not sure about the theory). It is hard to think of boosting in terms of the bias–variance tradeoff, and easier in terms of the number of iterations.

2.3 C: Computation, calibration and conformal prediction

- Conformal prediction: gives a confidence band.
- Calibration: gives predictions an interpretation as probabilities.

Both are uncertainty quantification, which is different from model selection.

Random forests vs. boosting. Random forests are embarrassingly parallel: the trees don't depend on each other. Boosting must be trained sequentially. Both have to solve for M decision trees.

Bagging is also parallelizable.

Other concerns: memory.

- Decision trees: low memory, since we can throw away the data after training.
- k-NN: needs to remember the training data, but has no training time.

This is a memory vs. computation tradeoff.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 15 — March 14, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 RKHS

We build up to reproducing kernel Hilbert spaces (RKHS) slowly.

1.1 Metric spaces

Definition (Metric space)**.** A *metric space* (M, d) consists of a set of objects M and a distance metric $d : M \times M \to \mathbb{R}$ such that for all $x, y, z \in M$:

1. $d(x, y) = d(y, x)$;
2. $d(x, x) = 0$;
3. $d(x, y) > 0$ if $x \neq y$;
4. $d(x, y) + d(y, z) \geq d(x, z)$.

1.2 Vector spaces

Definition (Vector space)**.** A *vector space* $(M, +, \times)$ is closed under addition and scalar multiplication:

$$\begin{aligned} x, y \in M &\implies x + y \in M, \\ x \in M &\implies ax \in M \quad \text{for } a \in \mathbb{R}. \end{aligned}$$

A vector space has no notion of distance between points: we can just take linear combinations. Metric spaces and vector spaces are orthogonal concepts. A vector space always contains the 0 element: $0 \in M$.

1.3 Normed vector spaces

Definition (Norm)**.** Let F be a vector space. A function $\|\cdot\| : F \to \mathbb{R}_0^+$ is called a *norm* if

1. $\|f\| = 0$ if and only if $f = \vec{0}$;
2. $\|af\| = |a| \cdot \|f\|$ for all $a \in \mathbb{R}$ and $f \in F$;
3. $\|f + g\| \leq \|f\| + \|g\|$.

Example 1. $F = \mathbb{R}^d$ with

$$\|x\|_p = \Big(\sum_i |x_i|^p\Big)^{1/p}, \qquad 1 \le p < \infty.$$

Example 2. $F = \mathbb{R}^{m\times n}$ with the operator norm

$$\|X\|_{\text{op}} = \text{maximum singular value of } X = \max_{\substack{v\in\mathbb{R}^n \\ \|v\|_2=1}} \|Xv\|_2 = \max_{\substack{v\in\mathbb{R}^n \\ v\neq 0}} \frac{\|Xv\|_2}{\|v\|_2},$$

where X is $m \times n$ and v is $n \times 1$. Another norm on this space is the *Frobenius norm*

$$\|X\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n X_{ij}^2}.$$

Example 3. $F = C[a, b]$, the continuous functions $[a, b] \to \mathbb{R}$ (this is a vector space), with

$$\|f\|_p = \Big(\int_a^b |f(x)|^p \, dx\Big)^{1/p}.$$

This is the natural definition, extending the ℓ_p norm to the continuous case.

Distance. For $x, y \in F$, let $d(x, y) = \|x - y\|$. So a norm induces a metric.

1.4 Inner product spaces

Definition (Inner product)**.** Let F be a vector space. A map $\langle\cdot,\cdot\rangle : F \times F \to \mathbb{R}$ is an *inner product* if

1. $\langle f, g\rangle = \langle g, f\rangle$ for all $f, g \in F$;
2. $\langle af_1 + bf_2, g\rangle = a\langle f_1, g\rangle + b\langle f_2, g\rangle$;
3. $\langle f, f\rangle \ge 0$ for all $f \in F$, and $\langle f, f\rangle = 0$ if and only if $f = 0$.

Intuition: the inner product tells you the angle between two things. If $\langle f, g\rangle = 0$, then $f \perp g$.

We can define a norm using the inner product:

$$\|f\| := \sqrt{\langle f, f\rangle}.$$

So an inner product induces a norm, which induces a metric.

Theorem 7 (Cauchy–Schwarz)**.**

$$|\langle f, g\rangle| \le \|f\| \cdot \|g\|.$$

So we can define the angle θ between f and g by

$$\cos\theta = \frac{\langle f, g\rangle}{\|f\| \cdot \|g\|}, \qquad \theta = \cos^{-1}(\cdots).$$

Example 4. If X, Y are matrices,

$$\langle X, Y \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij} Y_{ij}.$$

Then this induces the Frobenius norm.

Example 5. If f, g are functions,

$$\langle f, g \rangle = \int_a^b f(x) g(x)\, dx.$$

The induced norm coincides with the $p = 2$ norm above.

2 Banach Spaces

A *Banach space* is a *complete* normed vector space.

Definition (Convergent sequence)**.** A sequence $\{f_n\}_{n=1}^{\infty}$ in $(F, \|\cdot\|)$ *converges* to $f \in F$ if

$$\forall\, \varepsilon > 0\ \exists N :\ \forall\, n \geq N, \quad \|f_n - f\| \leq \varepsilon.$$

Definition (Cauchy convergent sequence)**.** $\{f_n\}$ is a *Cauchy convergent sequence (CCS)* if

$$\forall\, \varepsilon > 0\ \exists N :\ \forall\, m, n \geq N, \quad \|f_n - f_m\| \leq \varepsilon.$$

Convergence implies CCS, but CCS does not imply convergence.

Example 6. Let $F = \mathbb{Q}$. Then the following is a CCS:

$$(3,\ 3.1,\ 3.14,\ 3.141,\ 3.1415,\ 3.14159,\ \ldots).$$

But it does not converge in $\mathbb{Q}$.

Example 7. Take $C[0, 1]$ with the norm $\left(\int_0^1 f^2(x)\, dx\right)^{1/2}$, and let f_n be 0 up to $\frac{1}{2} - \frac{1}{n}$, rise linearly to 1 at $\frac{1}{2} + \frac{1}{n}$, and stay at 1 after that.

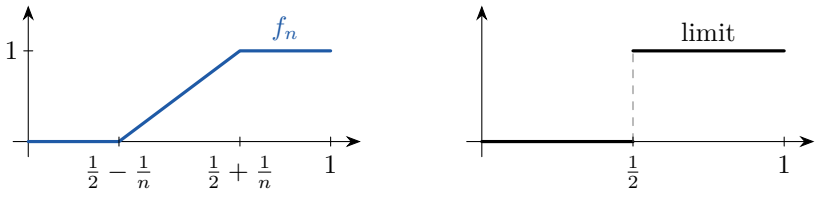


Figure 35: The continuous functions f_n (left) form a CCS, but their limit is a step function (right), which is not continuous.

The limit is a step function, which is not in $C[0, 1]$.

Definition (Complete)**.** $(F, \|\cdot\|)$ is called *complete* if every CCS is a convergent sequence, and the limit is some $f \in F$.

So $(\mathbb{Q}, |\cdot|)$ is not a Banach space.

3 Hilbert Spaces

Complete + norm = **Banach**	(can only get the size of points)
Complete + inner product = **Hilbert**	(can find the angle between points)

Example 8 (ℓ_2, the square-summable space)**.** The elements of F are infinite-dimensional vectors

$$a = (a_1, a_2, a_3, \dots) \quad \text{such that} \quad \sum_{i=1}^{\infty} a_i^2 < \infty.$$

This turns out to be a Hilbert space, with

$$\langle a, b \rangle = \sum_{i=1}^{\infty} a_i b_i, \qquad \|a\| = \sqrt{\sum_{i=1}^{\infty} a_i^2}.$$

Fact (parallelogram law): $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$.

The non-trivial part is verifying that ℓ_2 is complete: given a CCS $A_1, A_2, \dots$, we can show that its limit is also in the space.

Lecture 16 — March 16, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 RKHS

An RKHS is a set of functions satisfying certain "nice" properties. As in any Hilbert space, we can form

$$af_1 + bf_2, \qquad \langle f_1, f_2 \rangle, \qquad \|f\|, \qquad \|f_1 - f_2\|.$$

What does an RKHS have that Hilbert spaces don't?

1.1 Continuity of the evaluation functional

A *functional* is a map $\phi : F \to \mathbb{R}$. The *evaluation functional* at x is

$$\delta_x : f \mapsto f(x).$$

Example 9. Take $L_2[0,1]$ with

$$\langle f, g \rangle = \int_0^1 f(x)g(x)\,dx.$$

Consider $f_n(x) = x^n$ and $f^*(x) = 0$.

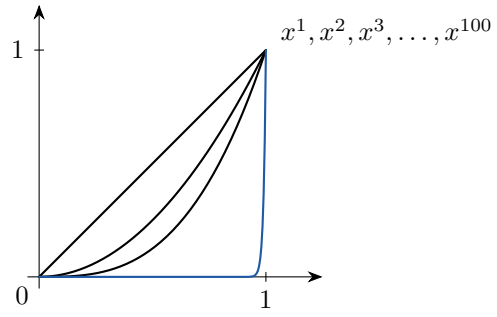


Figure 36: The functions $f_n(x) = x^n$ on $[0,1]$: as n grows they approach 0 everywhere except at $x = 1$, where they all equal 1.

So $f_n \to f^*$:

$$\lim_{n\to\infty} \|f_n - f^*\| = 0 \qquad \text{(in the } L_2 \text{ sense)}.$$

But

$$\delta_1(f_n) = 1 \quad \text{while} \quad \delta_1(f^*) = 0.$$

So convergence in L_2 does not imply pointwise convergence. Hence the evaluation functional is discontinuous (at 1).

One definition of an RKHS:

Definition (RKHS, first definition)**.** A Hilbert space is an *RKHS* if δ_x is continuous for all $x \in \mathcal{X}$.

Definition (Continuous functional)**.** A functional ϕ is *continuous* at f_0 if for all $\varepsilon > 0$ there exists $\delta > 0$ such that

$$\|f - f_0\| < \delta \implies |\phi(f) - \phi(f_0)| < \varepsilon.$$

So in an RKHS, norm convergence implies pointwise convergence:

$$\|f_n - f\| \to 0 \implies f_n(x) \to f(x) \quad \forall x \in \mathcal{X}.$$

This is why we like RKHSs!

Now we will define an RKHS in another way. (It turns out we have three definitions that coincide.)

1.2 Reproducing kernels

Definition (Reproducing kernel)**.** A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *reproducing kernel* of a Hilbert space $\mathcal{F}$ if

(a) $K(\cdot, x) \in \mathcal{F}$ for all $x \in \mathcal{X}$ (a univariate function), and K is symmetric: $K(\cdot, x) = K(x, \cdot)$;

(b) the "reproducing property" holds:

$$\langle f, K(\cdot, x)\rangle = f(x) \qquad \forall f \in \mathcal{F},\ x \in \mathcal{X}.$$

That is, $\langle \cdot\,, K(\cdot, x)\rangle$ is the evaluation functional.

Applying the reproducing property to $f = K(x, \cdot)$ gives

$$\langle K(x, \cdot), K(y, \cdot)\rangle = K(x, y).$$

Definition (RKHS, second definition)**.** A Hilbert space $\mathcal{F}$ is an *RKHS* if it has a reproducing kernel.

Theorem 8. *If a reproducing kernel exists, then it is unique.*

$\langle f_n, K(\cdot, x)\rangle$ is a linear operator, hence it maintains the continuity property.

1.3 The topological dual space

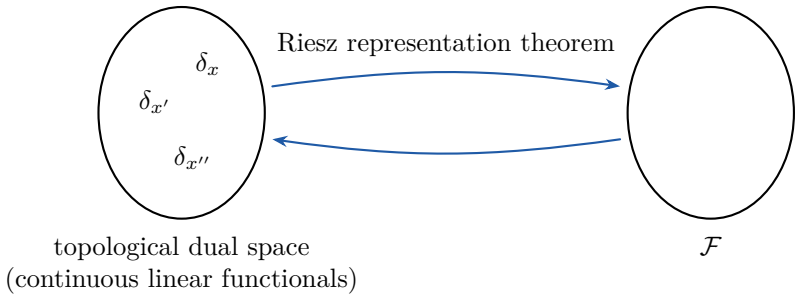


Figure 37: The Riesz representation theorem maps between the dual space of continuous linear functionals and the primal space $\mathcal{F}$.

There is a mapping between the dual space and the primal space. So functionals and functions are actually mapped to each other.

2 Kernels

Definition (Kernel)**.** A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel* for $\mathcal{F}$ if there exists a "feature map" $\phi : \mathcal{X} \to \mathcal{F}$ such that

$$K(x, y) = \langle \phi(x), \phi(y) \rangle .$$

Remark 3. Every reproducing kernel is a kernel: take $\phi(x) = K(\cdot, x)$.

Example 10 (Linear kernel)**.** Let $\mathcal{X} = \mathbb{R}^d$ and $K(x, y) = x^T y$. This is a valid kernel: use the feature map $\phi(x) = x$. This is known as the *linear kernel*.

Example 11. Let $K(x, y) = (x^T y)^2$, and consider the $\mathbb{R}^2$ case. This can be represented with either

$$\phi(x) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{bmatrix} \qquad \text{or} \qquad \phi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{bmatrix}.$$

Then $\langle \phi(x), \phi(y) \rangle = (x^T y)^2$.

So if a feature map exists, it is not unique. But the kernel is unique.

Definition (RKHS, third definition)**.** $\mathcal{F}$ is an *RKHS* if there exists a kernel for $\mathcal{F}$.

2.1 Kernels are positive definite

Important: a kernel is a *positive definite function*: for all $n \geq 1$ and all $x_1, \ldots, x_n \in \mathcal{X}$, the kernel "Gram matrix" K with $K_{ij} = K(x_i, x_j)$ satisfies $K \succeq 0$.

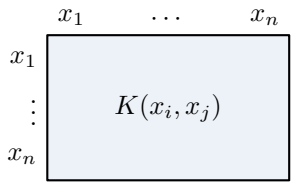


Figure 38: The kernel Gram matrix: entry (i, j) is $K(x_i, x_j)$.

That is,

$$\forall a_1, \ldots, a_n \in \mathbb{R}, \qquad a^T K a = \sum_i \sum_j a_i a_j K_{ij} \geq 0.$$

In the literature, people say either

$$\begin{array}{ll} \text{p.d.: } \geq 0, & \text{strictly p.d.: } > 0, \\ & \text{or} \\ \text{p.s.d.: } \geq 0, & \text{p.d.: } > 0. \end{array}$$

A kernel is just a way to enable us to take inner products (like a generalization of the inner product).

Kernel trick: replace inner products by kernels.

1 The Moore–Aronszajn Theorem

Theorem 9 (Moore–Aronszajn)**.** *There is a unique RKHS corresponding to each reproducing kernel K.*

So there is a one-to-one mapping between RKHSs and reproducing kernels K. The three definitions of an RKHS from last lecture coincide:

$$\begin{array}{ccccc} \text{RKHS} & \equiv & \text{reproducing kernel} & \equiv & \text{positive definite kernel} \\ \delta_x \text{ is continuous} & & \langle K(x,\cdot), f\rangle = f(x) & & K(x,y) = \langle \phi(x), \phi(y)\rangle \end{array}$$

Proof idea. Build a "pre-RKHS", then complete it (by adding the limits of Cauchy sequences).

The pre-RKHS consists of the functions

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i K(x_i, \cdot) \qquad \text{for some } x_1, \ldots, x_n \in \mathcal{X},\ \alpha_1, \ldots, \alpha_n \in \mathbb{R},\ n \geq 0.$$

Say $g(\cdot) = \sum_{j=1}^{m} \beta_j K(x'_j, \cdot)$. Then define

$$\langle f, g\rangle := \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i \beta_j K(x_i, x'_j).$$

This is only a pre-RKHS, because it may still not be complete. ■

Example 12 (Mixture of Gaussians)**.** The kernel

$$K_\gamma(x, x') = \exp\Big(-\frac{\|x - x'\|^2}{2\gamma^2}\Big)$$

is a valid kernel. A function

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

in the pre-RKHS is just a mixture of Gaussians.

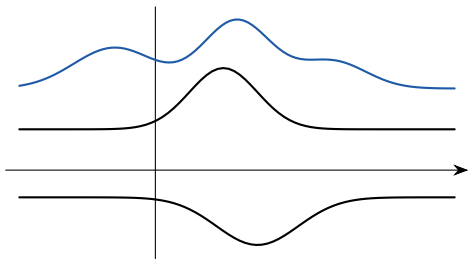


Figure 39: Functions $\sum_i \alpha_i K_\gamma(x_i, \cdot)$ are sums of Gaussian bumps, possibly with negative coefficients.

Example 13. If $\mathcal{X} = \mathbb{R}^d$ and $\phi(x) = x$, then

$$K(x, x') = \langle x, x' \rangle = x^T x'.$$

Example 14. $K(x, x') = \big(\langle x, x' \rangle\big)^2$, with feature map $\phi(x) := xx^T$.

Example 15 (Gaussian kernel)**.**

$$K_\gamma(x, x') = \exp\Big(-\frac{\|x - x'\|^2}{2\gamma^2}\Big).$$

2 "Closure" of Kernels

If K_1, K_2 are kernels, then so are

- αK_1 for $\alpha > 0$;
- $K_1 + K_2$;
- polynomials in the kernels with positive coefficients, e.g. $K_1^3 + 2K_1K_2$.

Feature map of αK_1: $\sqrt{\alpha}\,\phi_1$. Indeed,

$$\langle \sqrt{\alpha}\,\phi_1(x), \sqrt{\alpha}\,\phi_1(x') \rangle = \alpha \langle \phi_1(x), \phi_1(x') \rangle = \alpha K_1(x, x').$$

Feature map for $K_1 + K_2$: we want

$$(K_1 + K_2)(x, x') = \langle \phi(x), \phi(x') \rangle .$$

What is ϕ? Take the concatenation

$$\phi(x) = \begin{pmatrix} \phi_1(x) \\ \phi_2(x) \end{pmatrix}, \qquad \phi(x') = \begin{pmatrix} \phi_1(x') \\ \phi_2(x') \end{pmatrix}.$$

Concatenation gives the sum: $\langle \phi(x), \phi(x') \rangle = \langle \phi_1(x), \phi_1(x') \rangle + \langle \phi_2(x), \phi_2(x') \rangle = K_1(x, x') + K_2(x, x')$.

For the product:

$$(K_1K_2)(x, x') = K_1(x, x')K_2(x, x'), \qquad \phi(x) := \phi_1(x) \otimes \phi_2(x).$$

To see why, let $a, b \in \mathbb{R}^d$. Then

$$a^T b = \langle a, b \rangle \in \mathbb{R}, \qquad ab^T = a \otimes b \in \mathbb{R}^{d \times d}.$$

With $A = ab^T$ and $C = cd^T$,

$$\begin{aligned} \Big\langle ab^T, cd^T \Big\rangle = \mathrm{Tr}(A^T C) &= \mathrm{Tr}(ba^T cd^T) \\ &= a^T c\, \mathrm{Tr}(bd^T) \\ &= a^T c\, \mathrm{Tr}(d^T b) \\ &= (a^T c)(b^T d). \end{aligned}$$

So $\langle \phi_1(x) \otimes \phi_2(x), \phi_1(x') \otimes \phi_2(x') \rangle = \langle \phi_1(x), \phi_1(x') \rangle \langle \phi_2(x), \phi_2(x') \rangle = K_1(x, x')K_2(x, x')$.

3 Kernel Ridge Regression

Ridge regression solves

$$\min_{\beta\in\mathbb{R}^d}\sum_{i=1}^{n}(y_i-\beta^T x_i)^2+\lambda\|\beta\|_2^2$$

on data $(x_1,y_1),\ldots,(x_n,y_n)$ with $y\in\mathbb{R}$ and $x\in\mathbb{R}^d$. For example, $x=$ songs and $y=$ year.

Kernel ridge regression solves

$$\min_{f\in\mathcal{H}_K}\sum_{i=1}^{n}\big(y_i-f(x_i)\big)^2+\underbrace{\lambda\|f\|_K^2}_{\text{regularize by norm}}.$$

$\|f\|_K$ is a measure of the complexity of f.

More generally, we can solve

$$\min_{f\in\mathcal{H}_K} L\Big(\{y_i,f(x_i)\}_{i=1}^n\Big)+\lambda J(\|f\|_K),$$

where J is superlinear and strictly increasing. The loss L could be built from $I\{y_if(x_i)<0\}$, or from $e^{-y_if(x_i)}$.

4 Representer Theorem

Claim 1. The optimum is

$$f^*(\cdot)=\sum_{i=1}^{n}\alpha_iK(x_i,\cdot),$$

where the x_i are the training data and $\alpha\in\mathbb{R}^n$.

This basically turns an ∞-dimensional optimization problem into an n-dimensional problem of finding α.

Proof. By contradiction. Assume that

$$g^*(\cdot)=f^*+f^\perp$$

is the optimum, where f^* is in the span of $\{K(x_i,\cdot)\}_{i=1}^n$ and $f^\perp$ lies in the orthogonal complement

$$\{f\in\mathcal{H}_K : f\perp\operatorname{span}\{K(x_i,\cdot)\}_{i=1}^n\}.$$

By the reproducing property,

$$f^\perp(x_i)=\Big\langle f^\perp,K(x_i,\cdot)\Big\rangle=0.$$

(1) So $g^*(x_i)=f^*(x_i)$ for every i, and the loss term is the same for g^* and f^*.

(2) By the Pythagorean theorem,

$$\|g^*\|_K^2 = \|f^*\|_K^2 + \left\|f^\perp\right\|_K^2,$$

so there is no advantage to taking anything in the orthogonal complement: if $f^\perp \neq 0$, then f^* has a strictly smaller regularization term than g^*, since J is strictly increasing.

So g^* is not the optimum, a contradiction. ■

Lecture 18 — March 23, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Kernel Ridge Regression

Kernel ridge regression (KRR) solves

$$\min_{f\in\mathcal{H}_K} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|f\|_K^2 .$$

Other function spaces could take the place of $\mathcal{H}_K$ here, such as a Besov space or $C[\mathcal{X} \to \mathbb{R}]$.

1.1 Solving KRR with the representer theorem

By the representer theorem,

$$f^* := \sum_{j=1}^{n} \alpha_j \phi(x_j) = \sum_{j=1}^{n} \alpha_j K(x_j, \cdot) \qquad \text{for some } \alpha \in \mathbb{R}^n.$$

Then

$$f^*(x_i) = \langle f^*, K(x_i, \cdot)\rangle = \sum_{j=1}^{n} \alpha_j K(x_i, x_j).$$

The $n \times n$ *kernel matrix* (or *Gram matrix*) is

$$\mathbb{K}_{ij} := K(x_i, x_j).$$

It is positive semidefinite (PSD).

So we want

$$\alpha^* := \underset{\alpha\in\mathbb{R}^n}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{n} \Big(y_i - \overbrace{\sum_{j=1}^{n} \alpha_j K(x_i, x_j)}^{(\mathbb{K}\alpha)_i}\Big)^2}_{\|y-\mathbb{K}\alpha\|_2^2} + \lambda \left\| \sum_{j=1}^{n} \alpha_j K(x_j, \cdot) \right\|_K^2 .$$

For the second term, by linearity of the inner product,

$$\begin{aligned}
\left\| \sum_{j=1}^{n} \alpha_j K(x_j, \cdot) \right\|_K^2 &= \left\langle \sum_{j=1}^{n} \alpha_j K(x_j, \cdot), \sum_{l=1}^{n} \alpha_l K(x_l, \cdot) \right\rangle \\
&= \sum_{j=1}^{n}\sum_{l=1}^{n} \alpha_j \alpha_l \underbrace{\langle K(x_j, \cdot), K(x_l, \cdot)\rangle}_{=K(x_j, x_l)} \\
&= \alpha^T \mathbb{K} \alpha,
\end{aligned}$$

where $\langle K(x_j, \cdot), K(x_l, \cdot)\rangle = K(x_j, x_l)$ by the reproducing property.

So we can rewrite the objective as

$$\|y - \mathbb{K}\alpha\|_2^2 + \lambda \alpha^T \mathbb{K}\alpha, \qquad \text{where} \quad \|y - \mathbb{K}\alpha\|_2^2 = y^T y + \underbrace{(\mathbb{K}\alpha)^T \mathbb{K}\alpha}_{\alpha^T \mathbb{K}^T \mathbb{K}\alpha} - 2y^T \mathbb{K}\alpha.$$

Take the derivative with respect to α:

$$-2\mathbb{K}^T(y - \mathbb{K}\alpha^*) + 2\lambda \mathbb{K}\alpha^* = 0 \implies \mathbb{K}y = \mathbb{K}(\mathbb{K} + \lambda I)\alpha^*.$$

$\mathbb{K}$ is PSD and λI is positive definite, so the sum is positive definite. So $\mathbb{K} + \lambda I$ is invertible, and

$$\alpha^* = (\mathbb{K} + \lambda I)^{-1} y, \qquad f^* = \sum_{j=1}^{n} \alpha_j^* K(x_j, \cdot).$$

Then $f^*(x)$ is easy to compute for all x.

1.2 Comparison with ridge regression

Recall that ridge regression solves

$$\operatorname*{argmin}_{\beta \in \mathbb{R}^d} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2,$$

with solution

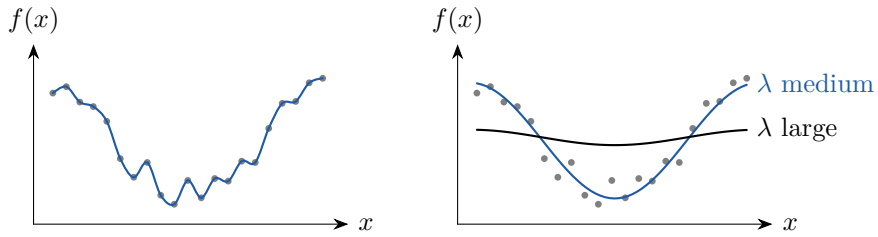
$$\beta^* = (X^T X + \lambda I_d)^{-1} X^T y = X^T \underbrace{(XX^T + \lambda I_n)^{-1} y}_{\alpha^*}.$$

This is KRR with $\phi(x) = x$ and $\mathbb{K} = XX^T$, and

$$f^*(x) = \phi(x)^T X^T \alpha^* = \phi(x)^T \beta^*.$$

In this case we could write it down because the feature space is finite dimensional.

1.3 Choosing λ



λ too small: f^* is wiggly $\qquad$ K = Gaussian kernel

Figure 40: KRR fits with a Gaussian kernel. A small λ chases the noise; a medium λ follows the trend; a large λ is nearly flat.

Use cross validation to pick a good λ. First try various λ (small, large, medium, etc.).

2 The "Kernel Trick"

The kernel trick is a way to use kernels without understanding what you're doing: whenever you see an inner product, replace it by a kernel.

So now everything that you could do linearly, you can now do non-linearly "for free" (since you don't have to solve an ∞-dimensional problem). People applied kernel techniques everywhere:

- kernel PCA;
- kernel k-means;
- kernel support vector machines (a linear classifier);
- kernel perceptron;
- kernel spectral clustering.

Difficulty: computing $(XX^T + \lambda I_n)^{-1}$ takes $O(n^3)$ time. People use

- Nyström approximations;
- random kitchen sinks.

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 20 — March 30, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Cooperative Game Theory: Shapley Values

Shapley values (1957) are a concept from cooperative game theory; Shapley received the Nobel prize in 2012.

N players together produce something of value. A *value function*

$$v : 2^{[N]} \to \mathbb{R}$$

assigns to each subset $S \subseteq [N]$ the value $v(S)$ produced by that subset. We have $v(\emptyset) = 0$.

Output: payoffs $\{\phi_i\}_{i \in [N]}$ to each player.

Bad answers:

(1) $\phi_i = 0$ for all i. (We want to preserve the value.)

(2) $\phi_1 = v([N])$ and $\phi_i = 0$ for $i > 1$.

1.1 Axioms (desiderata)

(1) **Efficiency:** $\sum_{i=1}^{N} \phi_i(v) = v([N])$ for all v.

(2) **Symmetry:** if $v(S \cup \{i\}) = v(S \cup \{j\})$ for all S, then $\phi_i(v) = \phi_j(v)$. So if i and j are interchangeable, they should get the same rewards.

(3) **Linearity:**

- $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$, i.e. a monthly or weekly payoff scheme doesn't matter;
- $\phi_i(av) = a\phi_i(v)$ for $a > 0$, i.e. the payoff is unit-free: it doesn't matter if I reward you in cents or dollars.

(4) **Freeloaders get nothing:** $v(S \cup \{i\}) = v(S)$ for all $S \implies \phi_i(v) = 0$.

If you have all four conditions, then there exists a unique ϕ.

Theorem 10 (Shapley value)**.** *The* Shapley value

$$\phi_i(v) = \frac{1}{N} \sum_{S \subseteq [N] \setminus \{i\}} \frac{v(S \cup \{i\}) - v(S)}{\binom{N-1}{|S|}}$$

is the unique ϕ satisfying axioms 1–4. Equivalently,

$$\phi_i(v) = \frac{1}{N!}\sum_R \Big[v(\pi_i^R \cup \{i\}) - v(\pi_i^R)\Big],$$

where the sum is over all rankings R of the players, and π_i^R is the set of players preceding i in the "ranking" R.

1.2 Examples

Example 16 (Business game)**.** The players are an owner o and workers $w_1, \ldots, w_N$. The value function is

$$v(S) = \begin{cases} (|S|-1)p & \text{if } o \in S, \\ 0 & \text{otherwise,} \end{cases}$$

where the value is 0 without the owner since the company wouldn't exist.

What is $\phi_o(v)$? It is $Np/2$, and $\phi_i(v) = p/2$ for each worker $i \neq o$. The total profit is Np.

Example 17 (Glove game)**.** Players 1 and 2 each have a left glove, and player 3 has a right glove. Let

$$v(S) = \begin{cases} 1 & \text{if } L, R \in S, \\ 0 & \text{if not.} \end{cases}$$

Then

$$\phi_3(v) = \frac{1}{3}\sum_{S \subseteq [3]\setminus\{3\}} \frac{v(S\cup\{3\}) - v(S)}{\binom{2}{|S|}}.$$

The possible S are:

- $S = \{1,2\}$, with weight $1/\binom{2}{2} = 1$;
- $S = \{1\}$ and $S = \{2\}$, each with weight $1/\binom{2}{1} = 1/2$;
- $S = \emptyset$, where $v(\{3\}) - v(\emptyset) = 0$.

In the first two cases, adding player 3 raises the value from 0 to 1, so

$$\phi_3(v) = \frac{1}{3}\Big(1 + \frac{1}{2} + \frac{1}{2}\Big) = \frac{2}{3}, \qquad \phi_1(v) = \phi_2(v) = \frac{1}{6}.$$

(The last values follow from symmetry and efficiency: $\phi_1 + \phi_2 = 1 - \frac{2}{3}$.)

2 Shapley Values in Machine Learning

	Regression	**Classification**
Variable importance		$v(S) :=$ train and predict only using S; held-out accuracy $-\frac{1}{2}$
Datapoint importance	$v(S) = \sum_{i \in H_0} y_i^2 - \sum_{i \in H_0} \left(y_i - f_S(x_i)\right)^2$, where H_0 is the held-out set	

For $S = \emptyset$, just output 0.

Drawbacks: the linearity assumption is not really required in the ML context (though it makes sense in economics).

Lecture 21 — April 4, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Unsupervised Learning

Unsupervised learning is huge. It includes:

- dimension reduction (which also helps to reduce noise);
- clustering;
- generative models (density estimation);
- manifold learning.

2 SVD and PCA

2.1 Singular value decomposition

An $m \times n$ matrix X of rank r can be written as $X = U\Sigma V^T$:

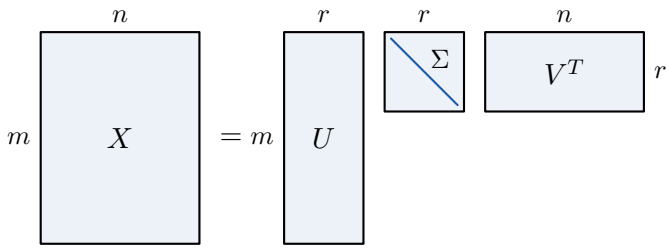


Figure 41: The SVD of an $m \times n$ matrix X of rank r.

- The columns of U are the *left singular vectors*, and $U^TU = I_r$.
- $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$, where the *singular values* satisfy $\sigma_j \geq 0$.
- The columns of V are the *right singular vectors*.

We can also write

$$X = \sum_{j=1}^{r} \sigma_j \underbrace{u_j v_j^T}_{m \times n}.$$

Write $L_j = u_j v_j^T$. These rank-1 matrices are orthogonal: $\langle L_i, L_j \rangle = 0$ for $i \neq j$, since

$$\left\langle u_i v_i^T, u_j v_j^T \right\rangle = \mathrm{Tr}(v_i \underbrace{u_i^T u_j}_{=0} v_j^T) = 0.$$

They also have unit norm, $\|L_i\|_F = 1$, by the cyclic property of the trace:

$$\|L_i\|_F^2 = \mathrm{Tr}(v_i u_i^T u_i v_i^T) = \mathrm{Tr}(v_i v_i^T) = \mathrm{Tr}(v_i^T v_i) = 1.$$

So the SVD is the smallest decomposition of X into rank-1 matrices, where "smallest" means using the fewest matrices.

Inner product of matrices:

$$\langle A, B \rangle = \mathrm{Tr}(A^T B) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}.$$

2.2 Key facts

Question. For some $k < r = \mathrm{rank}(X)$, what is the "best" rank-k approximation to X? "Best" can be measured in the operator norm or the Frobenius norm. Let

$$X' \in \underset{\text{rank-}k \text{ matrices}}{\mathrm{argmin}} \|X' - X\|_{\mathrm{op}},$$
$$X'' \in \underset{\text{rank-}k \text{ matrices}}{\mathrm{argmin}} \|X'' - X\|_F.$$

Then

$$X' = X'' = \text{top-}k \text{ SVD of } X.$$

An eigenvalue decomposition exists only for diagonalizable square matrices. Now let X be an $n \times d$ matrix. Then

$$\underbrace{X^T X}_{\text{covariance matrix}} = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T \qquad (d \times d),$$
$$\underbrace{X X^T}_{\text{kernel/Gram matrix}} = U \Sigma^2 U^T \qquad (n \times n),$$

and the diagonal entries of Σ^2 are the eigenvalues. So in this special case the singular values and the eigenvalues are the same. In general, singular values and eigenvalues are completely different concepts. It turns out that singular values are more useful for us.

3 PCA

There are two equivalent formulations: maximizing variance and minimizing reconstruction error.

3.1 Maximizing variance

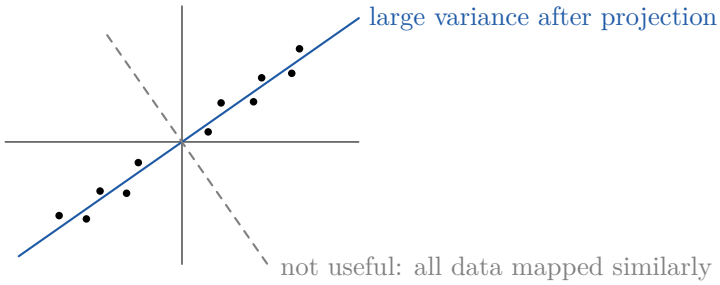


Figure 42: Projecting onto the solid direction keeps the spread of the data; projecting onto the dashed direction maps all the points to nearly the same place.

Suppose the points are re-centered so that $\frac{1}{n}\sum_i x_i = 0$. We maximize the variance of the projected data:

$$\begin{aligned}\max_{\|v\|=1} \frac{1}{n}\sum_{i=1}^{n}(x_i^T v)^2 &= \max_{\|v\|=1} v^T \underbrace{\Big(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T\Big)}_{\frac{1}{n}X^TX =: C} v \\ &= \max_{v^Tv=1} v^T C v,\end{aligned}$$

where C is the covariance matrix. We can think of this as asking which direction of C has the most energy.

Use the Lagrangian method, moving the constraint into Lagrangian form:

$$\max_v \; v^T C v - \lambda(v^T v - 1).$$

Differentiating and setting the derivative to 0 gives

$$Cv^* = \lambda v^*.$$

The optimizer v^* satisfies this, so we choose the first eigenvalue:

$$\begin{aligned}\lambda &= \text{first eigenvalue},\\ v^* &= \text{first eigenvector}.\end{aligned}$$

So the right singular vectors are the principal components, since the covariance satisfies $X^TX = V\Sigma^2V^T$.

3.2 Minimizing reconstruction error

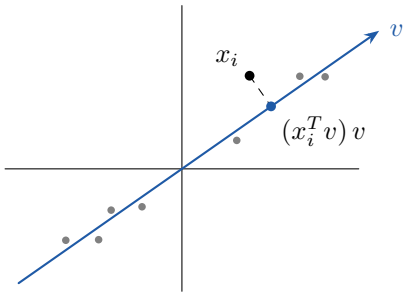


Figure 43: The reconstruction of x_i is its projection $(x_i^T v)v$ onto the direction v; the error is the dashed segment.

We minimize the reconstruction error:

$$\min_{\|v\|=1} \sum_{i=1}^{n} \left\| x_i - (x_i^T v)v \right\|_2^2 = \min_{\|v\|=1} \sum_{i=1}^{n} \Big[x_i^T x_i + (x_i^T v)^2 \underbrace{v^T v}_{=1} - 2(x_i^T v)^2 \Big]$$
$$= \min_{\|v\|=1} \sum_{i=1}^{n} \Big[x_i^T x_i - (x_i^T v)^2 \Big].$$

Since $\sum_i x_i^T x_i$ does not depend on v, flipping the sign gives

$$v^* = \operatorname*{argmax}_{\|v\|=1} \sum_{i=1}^{n} (v^T x_i)^2,$$

and this is the same as maximizing variance. Different viewpoints are more useful in different contexts.

3.3 Computation and choosing the number of components

Questions. How long does the SVD take? How many PCs should I keep?

The SVD takes time $O(\min(n^2 d, d^2 n))$. If you only want the top k components, it takes $O(nkd)$.

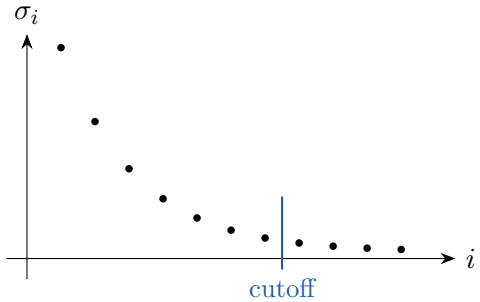


Figure 44: Singular values in decreasing order. The cutoff is the point after which we stop caring about the singular values.

The variance captured by the top k PCs is

$$\frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_d^2} \approx 0.8.$$

The denominator is fast to compute, since it is the squared Frobenius norm of X:

$$\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} X_{ij}^2} = \|X\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_d^2}.$$

So we can use $\|X\|_F$ to estimate it.

So the operator norm is like the L_∞ norm of the singular values, and the Frobenius norm is like the L_2 norm.

Next: sufficient dimension reduction (SDR).

36-708: Statistical Methods in Machine Learning Spring 2023

Lecture 22 — April 6, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Modal Clustering

This lecture covered:

- the mean shift algorithm;
- the blurred mean shift algorithm;
- k-means (Lloyd's algorithm).

2 Hierarchical Clustering

Choices for d, the distance between clusters:

- single linkage;
- average linkage;
- complete linkage.

Lecture 23 — April 11, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Graph Laplacians

Consider a weighted graph $G = (V, E)$ on n vertices.

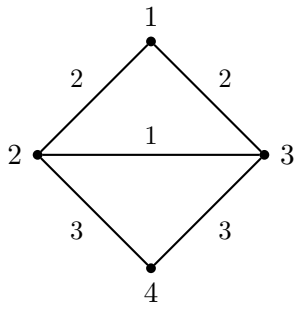


Figure 45: A weighted graph on four vertices; edge labels are the weights w_{ij}.

The *Laplacian* is the $n \times n$ matrix

$$L = D - W,$$

where W is the *adjacency matrix*,

$$W_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin E, \\ w_{ij} & \text{otherwise,} \end{cases}$$

and D is the *degree matrix*, the diagonal matrix with $D_{ii} = \sum_j W_{ij}$. For the graph above,

$$W = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 1 & 3 \\ 2 & 1 & 0 & 3 \\ 0 & 3 & 3 & 0 \end{bmatrix}, \qquad D = \begin{bmatrix} 4 & & & \\ & 6 & & \\ & & 6 & \\ & & & 6 \end{bmatrix}, \qquad L = \begin{bmatrix} 4 & -2 & -2 & 0 \\ -2 & 6 & -1 & -3 \\ -2 & -1 & 6 & -3 \\ 0 & -3 & -3 & 6 \end{bmatrix}.$$

1.1 Properties of L

- L is symmetric.
- L is PSD (all eigenvalues are ≥ 0).
- $L\mathbf{1} = 0$. So $\mathbf{1}$ is always an eigenvector ($\mathbf{1}/\sqrt{n}$ is a unit eigenvector), and 0 is always an eigenvalue.

Let $f \in \mathbb{R}^n$. Then

$$f^T L f = \sum_{(i,j)\in E} w_{ij}(f_i - f_j)^2.$$

This is used as a form of regularization for graphs: it measures graph smoothness.

Example 18 (Graph denoising)**.** Let $X_i = \mu_i + N(0,1)$ for $i \in V$, where $\mu_i \approx \mu_j$ if $(i,j) \in E$. Estimate

$$\hat{\mu} = \operatorname*{argmin}_{\beta} \sum_{i\in V} (X_i - \beta_i)^2 + \beta^T L \beta.$$

The term $\beta^T L \beta$ causes nearby nodes to have similar values.

The number of zero eigenvalues equals the number of connected components.

For repeated eigenvalues, the eigenvectors are not unique.

1.2 Three Laplacians

- $D - W$.
- $D^{-1}W$, the *random walk Laplacian*. To compute D^{-1}, just invert the diagonal entries of D. (When you are in a particular row, how do you jump to another entry?)
- $I - D^{-1/2}WD^{-1/2}$, which is "unit-free".

All three definitions are equivalent, and all are used in the field.

2 Spectral Clustering

Take the first eigenvectors $v_1, v_2, \cdots \in \mathbb{R}^n$ of the Laplacian. Their i-th entries give new coordinates for the point X_i.

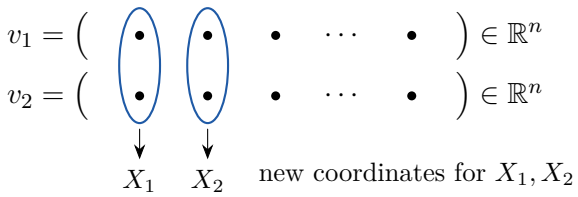


Figure 46: Each column of entries of the eigenvectors gives the new coordinates of one data point.

To construct the graph, we could add edges to the 2 nearest neighbors.

Spectral clustering is not a clustering algorithm per se. Rather, it reduces the problem to an easier one.

Spectral clustering.

Input: an $n \times n$ similarity matrix W.

1. Let D be the $n \times n$ diagonal matrix with $D_{ii} = \sum_j W_{ij}$.
2. Compute the Laplacian $\mathcal{L} = D^{-1}W$.
3. Find the first k eigenvectors $v_1, \ldots, v_k$ of $\mathcal{L}$.
4. Project each X_i onto the eigenvectors to get new points $\hat{X}_i$.
5. Cluster the points $\hat{X}_1, \ldots, \hat{X}_n$ using any standard clustering algorithm.

Lecture 27 — April 27, 2023

Prof. Aaditya Ramdas *Scribe: Fan Pu Zeng*

1 Deep Learning

Two topics:

(a) double descent;

(b) the bias–variance tradeoff (regularize).

1.1 The classical picture

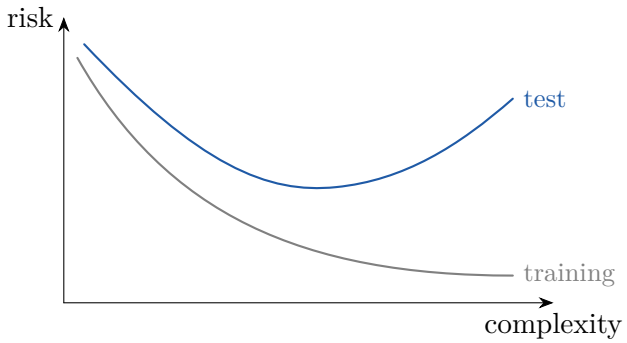


Figure 47: The "classical" picture, for fixed n: training risk falls with complexity, while test risk is U-shaped.

Think of this as a family of curves, one for each n. For smaller n, the best test error is at lower complexity.

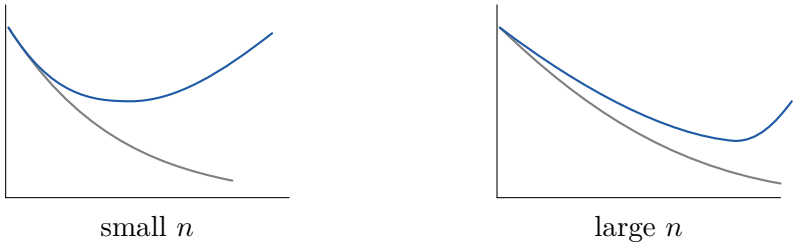


Figure 48: Test risk (accent) and training risk (gray) against complexity. With larger n, the best test risk comes at higher complexity.

1.2 Double descent

In deep learning, "double descent" is observed: the test error goes down twice.

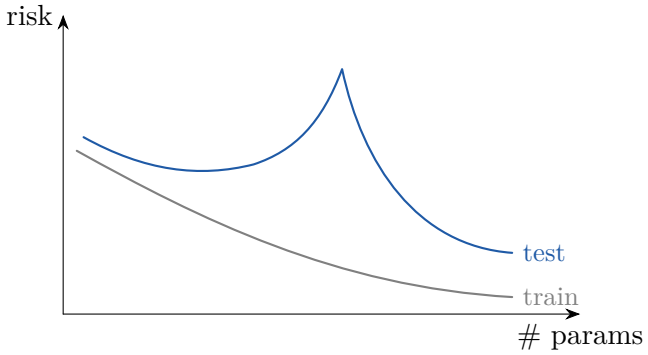


Figure 49: Double descent: test risk rises to a peak and then descends a second time as the number of parameters grows.

The number of parameters is *not* representative of the complexity of the model. Instead, we need to consider:

1. the architecture;
2. the training (SGD, Adam, SGD with momentum, dropout, etc.);
3. the data (with structured data, the optimization is easier).

For some models, the number of parameters is representative of complexity: OLS, kNN.

"Implicit regularization": the training data could have this property.

What if, instead of the number of parameters, we try the VC dimension or Rademacher complexity? We run into similar issues. There is no good notion of complexity for neural networks.

1.3 OLS

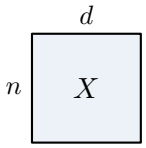


Figure 50: The $n \times d$ design matrix X.

The OLS estimator is

$$\hat{\beta} = (X^T X)^{-1} X^T y = \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|_2^2, \qquad \text{where } \|y - X\beta\|_2^2 = \sum_{i=1}^{n} (y_i - x_i^T \beta)^2.$$

We need to check whether $X^T X$ is ill-conditioned: we need to keep the ratio of its largest to its smallest eigenvalue from blowing up.

2 End-of-Semester Recap

The methods covered were k-NN, decision trees (DT), random forests (RF), bagging, boosting, stacking, RKHS methods, deep learning (DL), spectral clustering, k-means, and PCA and autoencoders. The recap lays them out against the ABCDE themes of the course (see the table in Lecture 14).

E: Explainability. LOCO, Shapley values.

Bias. The bias–variance tradeoff uses the statistical notion of bias. Inductive bias is a different notion, e.g. the axis-aligned decision boundaries in decision trees.

3 Paradigms

- Supervised learning.
- Unsupervised (self-supervised) learning.
- Reinforcement learning.
- Adversarial online learning.
- Active learning (the learner actively participates in the learning process).
- Semi-supervised learning. This is common in bioinformatics: we have millions of genomes, and a small set of genomes for people with Crohn's disease. (We could use autoencoders.)
- Transfer learning.
- Curriculum learning.
- Imitation learning.