ECE 6254
Statistical Signal Processing
Spring 2014

Mark A. Davenport

Georgia Institute of Technology
School of Electrical and Computer Engineering

Statistical Signal Processing

- Modern “data-driven” approach
- How can we
  - learn effective models from data?
  - apply these models to practical signal processing problems?
- Practical algorithms and theoretical analysis of many popular statistical learning techniques
- Applications including classification, prediction, regression, clustering, modeling, and data exploration/visualization

What is learning?

learn: gain or acquire knowledge of or skill in (something) by study, experience, or being taught

How do we learn that this is a tree?

Definition?

(A perennial plant with an elongated stem, or trunk, supporting leaves or branches.)

EXAMPLES!

A good definition of learning for this course:

"using a set of examples to infer something about an underlying process"

Why learn from data?

Traditional signal processing is “top down”

Given a model for our data, derive the optimal algorithm

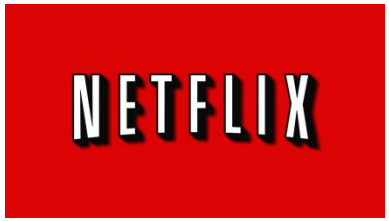
A learning approach is more “bottom up”

Given some examples, derive a good algorithm

Sometimes a good model is ***really*** hard to derive
from first principles

Examples of learning

The Netflix prize: Predict how a user will rate a movie



10% improvement = $1 million prize

- Some pattern exists
  - users do not assign ratings completely at random – if you like Godfather I, you'll probably like Godfather II
- It is hard to pin down the pattern mathematically
- We have lots and lots of data
  - we know how a user has rated other movies, and we know how other users have rated this (and other) movies

Examples of learning

Handwritten digit recognition

Types of learning

Our definition leaves considerable room for particular variations of “learning”

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

We are given input data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

Each $\mathbf{x}_i$ represents a measurement or observation of some natural or man-made phenomenon

- may be called input, pattern, signal, feature vector, instance, or independent variable
- the coordinates may be called features, attributes, predictors, or covariates

In the supervised case, we are also given output data $y_1, \ldots, y_n$

- may be called output, label, response, or dependent variable

The data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ are called the ***training data***

Supervised learning

We can think of a pair $(\mathbf{x}_i, y_i)$ as obeying a (possibly noisy) input-output relationship

The goal of supervised learning is usually to ***generalize*** the input-output relationship so that we can predict the output y associated with a previously unseen input $\mathbf{x}$

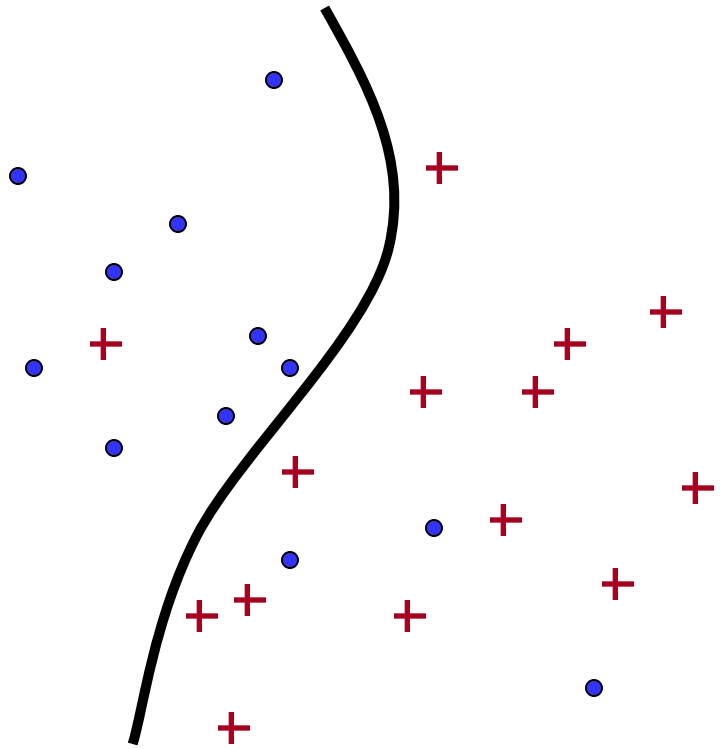
The primary supervised learning problems are

- classification: $y \in \{1, \ldots, m\}$
- regression: $y \in \mathbb{R}$

Examples of supervised learning

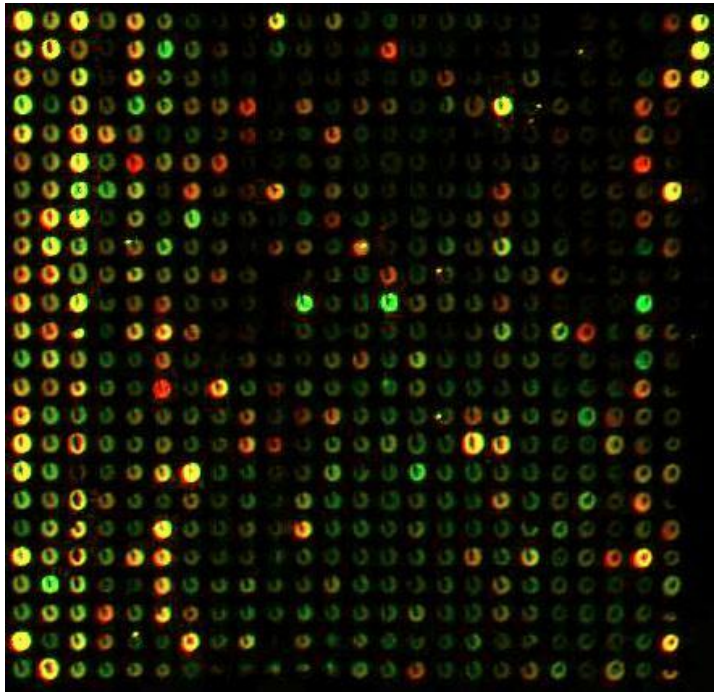
Labeled data

Examples of supervised learning



Examples of supervised learning

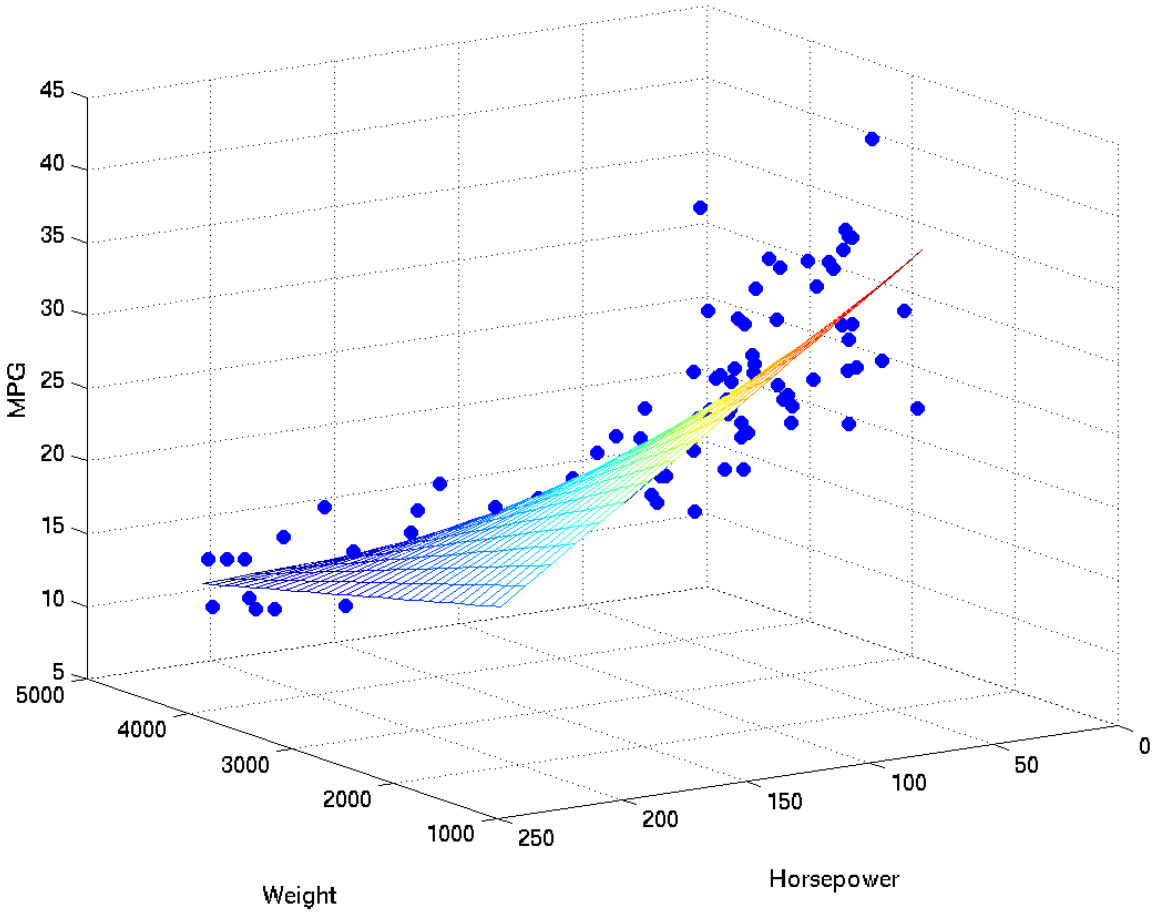
DNA microarrays



$\mathbf{x}_1, \ldots, \mathbf{x}_n$ are DNA expression levels for n patients

$y_1, \ldots, y_n$ is some quantity we would like to predict (discrete or continuous)

Examples of supervised learning



Unsupervised learning

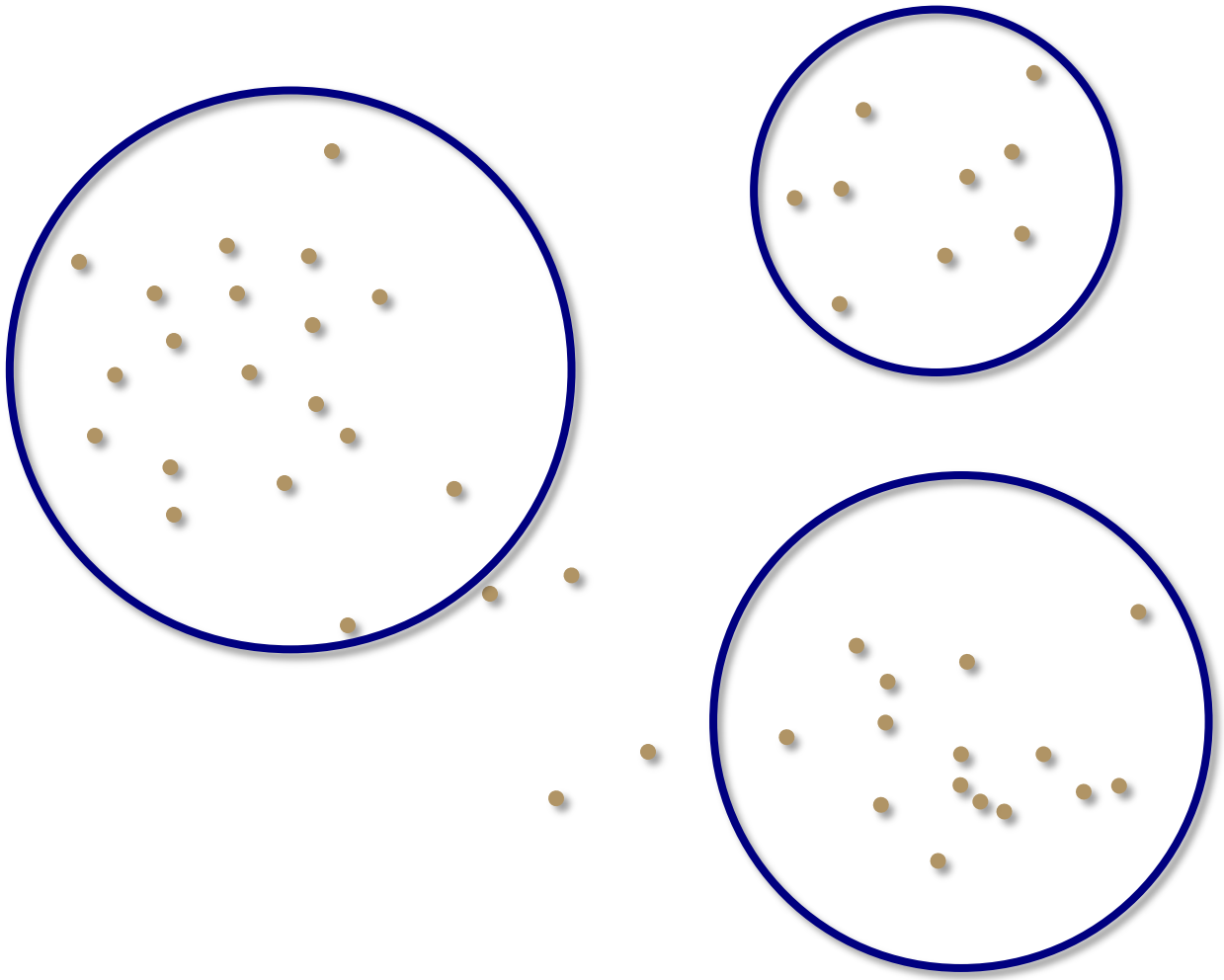
The inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are not accompanied by labels

The goal of unsupervised learning is typically not related to future observations. Instead we just want to understand that structure in the data sample itself, or to infer some characteristic of the underlying probability distribution.

The primary unsupervised learning problems are

- clustering
- desnsity estimation
- dimensionality reduction
- visualization

Unsupervised learning examples



Unsupervised learning examples

Reinforcement learning

- The inputs are observed sequentially
- After each $\mathbf{x}_i$ is observed the learner must take an action
- After each action, the learner receives a reward
- The goal is to determine a policy for selecting actions to maximize long-term reward

- Instead of (input, ***correct output***)
  we get (input, ***some output***, ***reward for this output***)

- Important in robotics (navigation/path planning) economics, and other areas

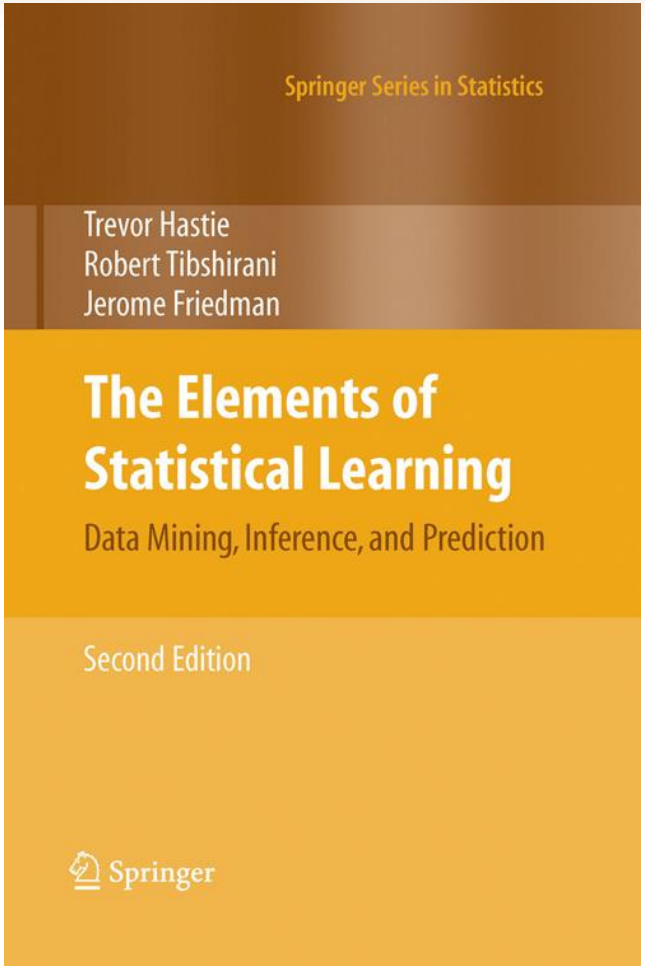
Other variants of learning

- semi-supervised learning
- active learning
- online learning
- anomaly detection
- ranking
- transfer learning
- multi-task learning
- …

Prerequisites

- Probability
  - random variables, expectation, joint distributions, independence, conditional distributions, Bayes rule, …
- Linear algebra
  - norms, inner products, orthogonality, linear independence, eigenvalues/vectors, eigenvalue decompositions, …
- MATLAB

Text



Available online

(Link on course webpage)

A list of other useful books and links to relevant papers will be posted on the course webpage.

Lecture notes will also be posted on the course webpage.

Grading

- Homework (15%)
- Midterm exam (25%)
- Mini-project (25%)
- Final project (30%)
- Participation (5%)

Outline

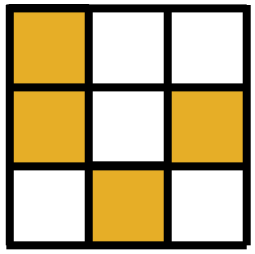
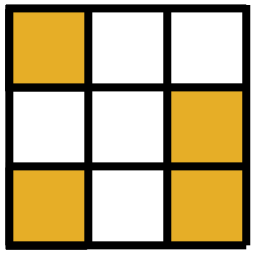
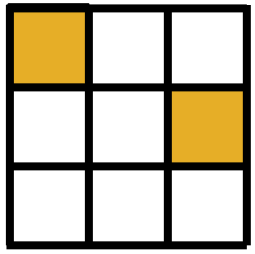
- Supervised learning
  - The Bayes classifier
  - Linear classifiers
    - Plugin classifiers (Linear discriminant analysis, Logistic regression, Naïve Bayes)
    - The perceptron algorithm and single-layer neural networks
    - The maximum margin principle and separating hyperplanes
  - Linear regression
    - Least-squares linear regression
  - Theory of generalization
    - VC dimension
    - Overfitting and the bias-variance tradeoff
    - Regularization
  - Nonlinear methods
    - Nearest neighbor classification
    - Multi-layer neural networks
    - Locally linear regression
    - Nonlinear feature maps
    - The "kernel trick"
    - Kernel ridge regression
    - Support vector machines
  - Error estimation and validation
  - Subset selection methods

- Unsupervised learning
  - Principle component analysis
  - Multidimensional scaling
  - K-means clustering
  - EM algorithm for Gaussian mixture models
  - Kernel density estimation
  - Nonlinear dimensionality reduction
- Other topics (as time and interest permits)
  - Matrix factorizations
  - Graphical models
  - Decision trees and CART
  - Ensemble methods and boosting
  - Spectral clustering

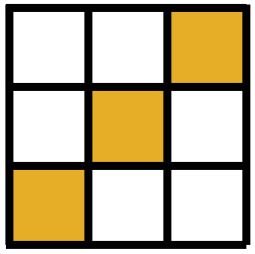
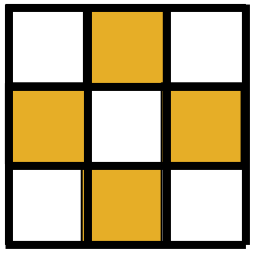
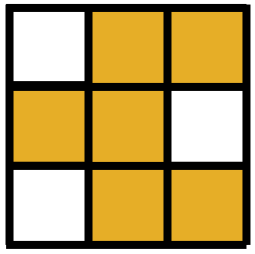
What is not covered?

- Adaptive filters
- Spectral estimation
- A huge number of other topics

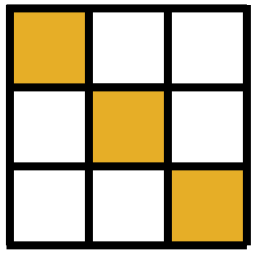
A learning puzzle?

$y = +1$

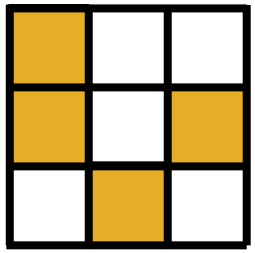
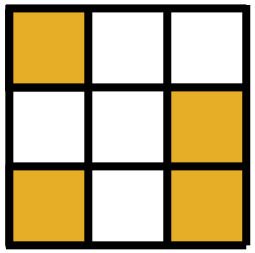
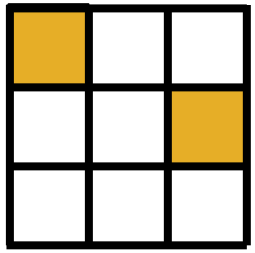
$y = -1$



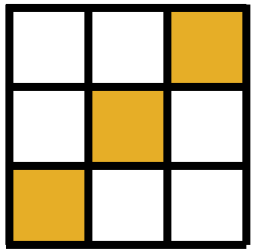
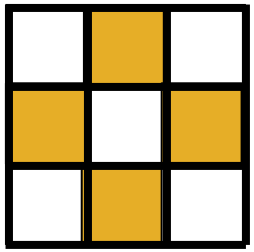
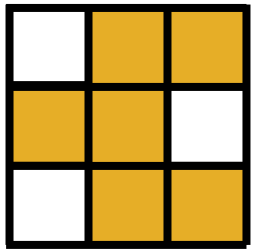
$y = ?$

Lecture 2: Is Learning Possible?

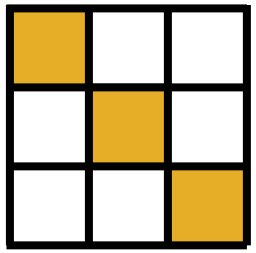
A learning puzzle?

$y = +1$

$y = -1$



$y = ?$

Is learning even possible?

or: How I learned to stop worrying and love statistics

Supervised learning

Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, we would like to learn a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) = y$ for $\mathbf{x}$ other than $\mathbf{x}_1, \ldots, \mathbf{x}_n$

but…

as we have just seen, this is impossible. Without any additional assumptions, we conclude ***nothing*** about f except (maybe) for its value on $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Probability to the rescue!

Any f agreeing with the training data may be ***possible***

but that does not mean that any f is equally ***probable***

Simple example

- Suppose that I have a biased coin, which lands on heads with some unknown probability p
  - $\mathbb{P}[\text{heads}] = p$
  - $\mathbb{P}[\text{tails}] = 1 - p$
- I toss the coin n times (independently)
  - $\widehat{p} = \frac{\#\text{ of heads}}{n}$

Does $\widehat{p}$ tell us anything about p?

What can we learn from $\widehat{p}$?

Given enough tosses (large n), $\widehat{p}$ is probably close to p

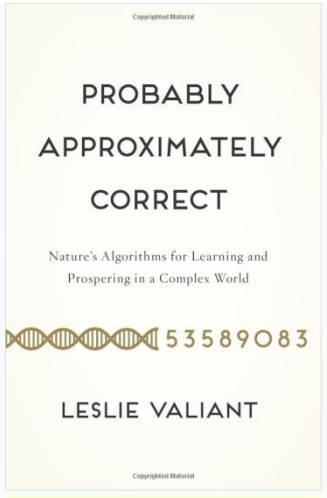
Hoeffding's Inequality

$$\mathbb{P}\left[|\widehat{p} - p| > \epsilon\right] \leq 2e^{-2\epsilon^2 n}$$

In other words, the statement that

$$\widehat{p} = p$$

is ***probably, approximately correct*** **(PAC).**



Connection to learning

Coin tosses: We want to estimate p

Learning: We want to estimate a function $f : \mathcal{X} \to \mathcal{Y}$

Suppose we have a ***hypothesis*** h

Think of the $(\mathbf{x}_i, y_i)$ as a series of independent coin tosses, where the $(\mathbf{x}_i, y_i)$ are drawn from a probability distribution

- heads: our hypothesis is correct, i.e., $h(\mathbf{x}_i) = y_i$
- tails: our hypothesis is wrong, i.e., $h(\mathbf{x}_i) \neq y_i$

Define

Risk: $R(h) := \mathbb{P}[h(X) \neq Y]$

Empirical risk: $\widehat{R}_n(h) := \dfrac{1}{n} \sum_{i=1}^{n} 1_{\{h(\mathbf{x}_i) \neq y_i\}}$

Trust, but verify

Applying Hoeffding's inequality we have

$$\mathbb{P}\left[\left|\widehat{R}_n(h) - R(h)\right| > \epsilon\right] \leq 2e^{-2\epsilon^2 n}$$

This means that we can use $\widehat{R}_n(h)$ to verify whether h was a good hypothesis

- Where did h come from?
- What if $R(h)$ is large?

Verification is not learning

From many hypotheses, we'd like to pick a good one

E pluribus unum

Consider an ensemble of many hypotheses

$$\mathcal{H} = \{h_1, \ldots, h_m\}$$

If we fix a hypotheses h_j before drawing our data, then

$$\mathbb{P}\left[\left|\widehat{R}_n(h_j) - R(h_j)\right| > \epsilon\right] \leq 2e^{-2\epsilon^2 n}$$

However, the assumption that the hypotheses is fixed ***before*** drawing our data is critical.

If h_j is selected ***after*** looking at the data, the assumptions underlying Hoeffding (independence) no longer hold.

Back to the coin analogy

Question 1: If I toss a fair coin 10 times, what is the probability that I get 10 heads?

Answer: ≈ 0.001

Question 2: If I toss 1000 fair coins 10 times each, what is the probability that ***some*** coin will get 10 heads?

Answer: ≈ 0.63

…and back to learning

If we have many hypotheses (large m), then

even though for any fixed hypothesis h_j it is likely that

$$\widehat{R}_n(h_j) \approx R(h_j)$$

it is also likely that there will be at least ***one*** hypothesis h_k where $\widehat{R}_n(h_k)$ is very different from $R(h_k)$

How to adapt our approach to handle many hypotheses?

A crude approach

Suppose that h^* represents the hypothesis that we ultimately pick from $\mathcal{H}$ (perhaps by taking the h_j with smallest $\widehat{R}_n(h_j)$)

We are interested in

$$\mathbb{P}\left[\left|\widehat{R}_n(h^*) - R(h^*)\right| > \epsilon\right] \leq \mathbb{P}\Big[\left|\widehat{R}_n(h_1) - R(h_1)\right| > \epsilon$$
$$\text{or } \left|\widehat{R}_n(h_2) - R(h_2)\right| > \epsilon$$
$$\vdots$$
$$\text{or } \left|\widehat{R}_n(h_m) - R(h_m)\right| > \epsilon\Big]$$

Union bound

Recall that for any sequence of events $\mathcal{E}_1, \ldots, \mathcal{E}_m$,

$$\mathbb{P}\left[\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_m\right] \leq \mathbb{P}\left[\mathcal{E}_1\right] + \cdots + \mathbb{P}\left[\mathcal{E}_m\right]$$

Thus,

$$\begin{aligned}
\mathbb{P}\left[\left|\widehat{R}_n(h^*) - R(h^*)\right| > \epsilon\right] \leq \mathbb{P}\Big[& \left|\widehat{R}_n(h_1) - R(h_1)\right| > \epsilon \\
& \textbf{or } \left|\widehat{R}_n(h_2) - R(h_2)\right| > \epsilon \\
& \vdots \\
& \textbf{or } \left|\widehat{R}_n(h_m) - R(h_m)\right| > \epsilon\Big] \\
\leq & \sum_{j=1}^{m} \mathbb{P}\left[\left|\widehat{R}_n(h_j) - R(h_j)\right| > \epsilon\right]
\end{aligned}$$

Putting it all together

$$\mathbb{P}\left[\left|\widehat{R}_n(h^*) - R(h^*)\right| > \epsilon\right] \leq \sum_{j=1}^{m} \mathbb{P}\left[\left|\widehat{R}_n(h_j) - R(h_j)\right| > \epsilon\right]$$

$$\leq \sum_{j=1}^{m} 2e^{-2\epsilon^2 n}$$

$$= 2me^{-2\epsilon^2 n}$$

Bottom line: As long as m isn't too big ($m \lesssim e^n$) then we can be reasonably confident that

$$\widehat{R}_n(h^*) \approx R(h^*)$$

Is that learning?

It's not quite enough to have $\widehat{R}_n(h^*) \approx R(h^*)$

We ultimately want $h^* \approx f$, which means that $R(h^*) \approx 0$

Note that if $\widehat{R}_n(h^*) \approx R(h^*)$, then $\widehat{R}_n(h^*) \approx 0$ implies that $R(h^*) \approx 0$

The learning problem

1. Can we ensure that $R(h^*)$ is close to $\widehat{R}_n(h^*)$? ✓
2. Can we make $\widehat{R}_n(h^*)$ small enough?

Empirical risk minimization

We want to make $\widehat{R}_n(h^*)$ as small as possible

Pick $h^* = \arg\min_{h_j \in \mathcal{H}} \widehat{R}_n(h_j)$

Is this a good idea?

It will make $\widehat{R}_n(h^*)$ small only if there is some $h_j \in \mathcal{H}$ such that $h_j \approx f$

We need a rich set of possible hypotheses…

but…

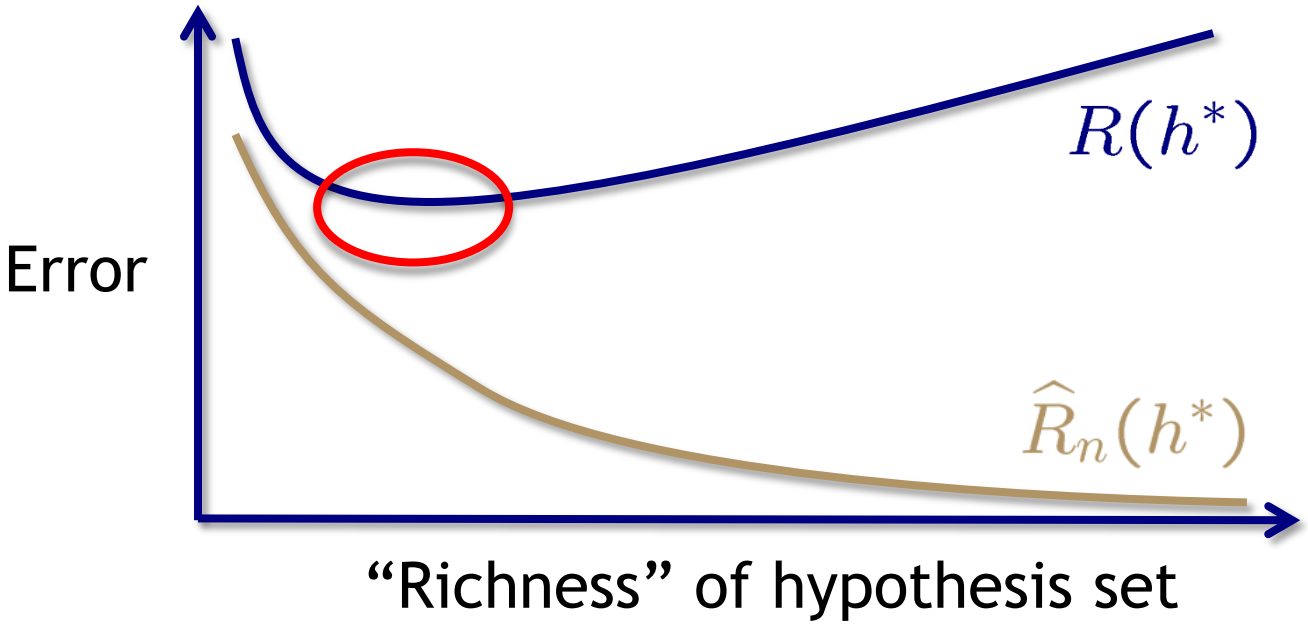
Fundamental tradeoff

More hypotheses ultimately sacrifices our guarantee that $\widehat{R}_n(h^*) \approx R(h^*)$

Richer set of hypotheses ➡

$$\widehat{R}_n(h^*) \downarrow$$

$$\widehat{R}_n(h^*) - R(h^*) \uparrow$$



Lecture 3: Bayes' Plugins

The learning problem

Training data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$

We want to estimate a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(\mathbf{x}) = y$ for new (previously unobserved) $\mathbf{x}$

From a list of hypotheses $\mathcal{H} = \{h_1, \ldots, h_m\}$, select one (denoted h^*) by taking the h_j that minimizes

$$\widehat{R}_n(h_j) := \frac{1}{n} \sum_{i=1}^{n} 1_{\{h_j(\mathbf{x}_i) \neq y_i\}}$$

We want to argue that $R(h^*) := \mathbb{P}[h^*(X) \neq Y]$ is small

1. Ensure that $R(h^*)$ is close to $\widehat{R}_n(h^*)$
2. Make $\widehat{R}_n(h^*)$ as small as possible

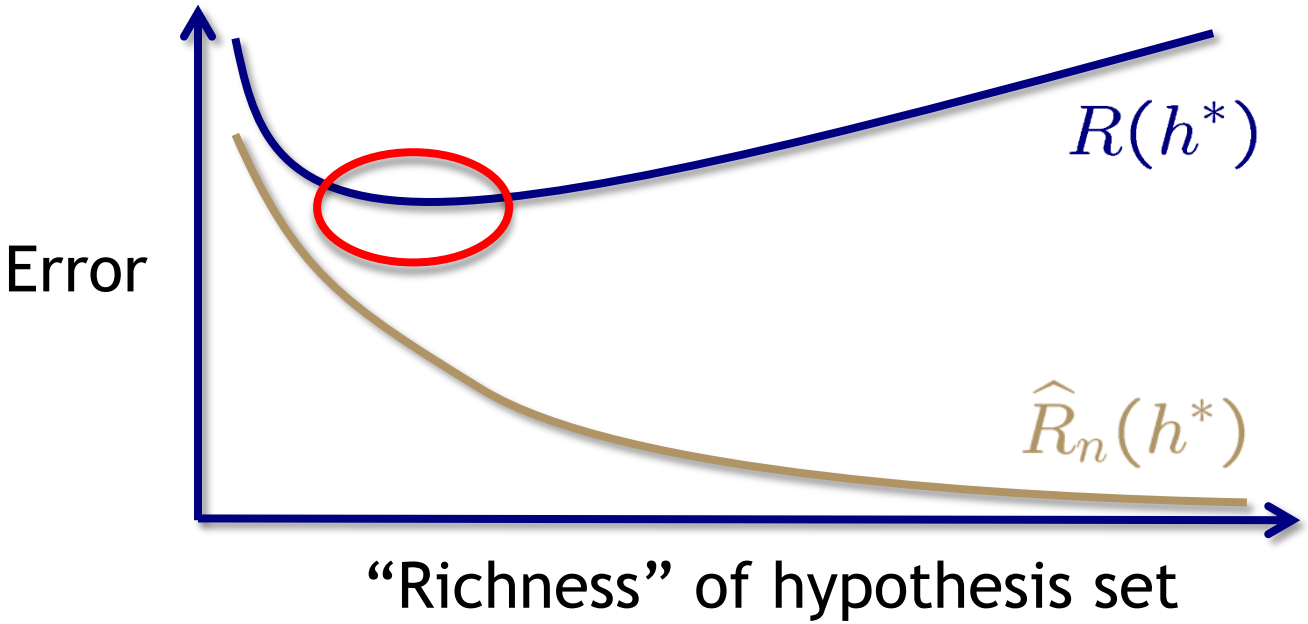
Fundamental tradeoff

More hypotheses ultimately sacrifices our guarantee that $\widehat{R}_n(h^*) \approx R(h^*)$

Richer set of hypotheses ➡

$$\widehat{R}_n(h^*) \downarrow$$

$$\widehat{R}_n(h^*) - R(h^*) \uparrow$$



Limits of performance

Ideally, we would like to have a small number of hypotheses, so that $\widehat{R}_n(h^*) \approx R(h^*)$, while also being lucky (or smart) enough to have $\widehat{R}_n(h^*) \approx 0$

Together these imply that $R(h^*) \approx 0$

Unfortunately, in general this will not be possible, since there may not be ***any*** function f with $R(f) \approx 0$

Why not?

Noise: $Y = f(X) + N$

What are the fundamental limits on how small $R(f)$ can be?

The Bayes classifier

Consider (X, Y) where

- X is a random vector in $\mathbb{R}^d$
- $Y \in \{0, \ldots, K-1\}$ is a random variable (depending on X)

Let $f : \mathbb{R}^d \to \{0, \ldots, K-1\}$ be a ***classifier***
with ***probability of error/risk*** given by

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

Denote the ***a posteriori class probabilities*** by

$$\eta_k(\mathbf{x}) := \mathbb{P}[Y = k | X = \mathbf{x}]$$

for $k = 0, \ldots, K-1$

The Bayes classifier

Theorem

The classifier $f^*(\mathbf{x}) := \arg\max_k \eta_k(\mathbf{x})$ satisfies

$$R(f^*) = \min R(f)$$

where the min is over all possible classifiers.

Terminology:

- f^* is called a ***Bayes classifier***
- $R(f^*)$ is called the ***Bayes risk***

Proof

For convenience, assume $X|Y = k$ is a continuous random variable with density $g_k(\mathbf{x})$

Let $\pi_k = \mathbb{P}[Y = k]$ denote the ***a priori class probabilities***

Consider an arbitrary classifier f. Denote the decision regions

$$\Gamma_k(f) := \{\mathbf{x} : f(\mathbf{x}) = k\}$$

Then $1 - R(f) = \mathbb{P}[f(X) = Y]$

$$= \sum_{k=0}^{K-1} \pi_k \cdot \mathbb{P}[f(X) = k | Y = k]$$

$$= \sum_{k=0}^{K-1} \pi_k \cdot \int_{\Gamma_k(f)} g_k(\mathbf{x})\, d\mathbf{x}$$

Proof (… continued)

$$1 - R(f) = \sum_{k=0}^{K-1} \pi_k \cdot \int_{\Gamma_k(f)} g_k(\mathbf{x})\, d\mathbf{x}$$

To maximize this expression, we should select f such that

$$\mathbf{x} \in \Gamma_k(f) \quad \Longleftrightarrow \quad \pi_k g_k(\mathbf{x}) \text{ is maximal}$$

Therefore, the optimal f has

$$\begin{aligned} f^*(\mathbf{x}) &= \arg\max_k \pi_k g_k(\mathbf{x}) \\ &= \arg\max_k \frac{\pi_k g_k(\mathbf{x})}{\sum_{\ell=0}^{K-1} \pi_\ell g_\ell(\mathbf{x})} \\ &= \arg\max_k \mathbb{P}[Y = k | X = \mathbf{x}] \end{aligned}$$

Bayes rule!

Variations

Different ways of expressing the Bayes classifier

- $f^*(\mathbf{x}) = \arg\max_k \eta_k(\mathbf{x})$
- $f^*(\mathbf{x}) = \arg\max_k \pi_k g_k(\mathbf{x})$
- When $K = 2$

$$\frac{g_1(\mathbf{x})}{g_0(\mathbf{x})} \underset{1}{\overset{0}{\lessgtr}} \frac{\pi_0}{\pi_1}$$

likelihood ratio test

- When $\pi_0 = \pi_1 = \cdots = \pi_{K-1}$

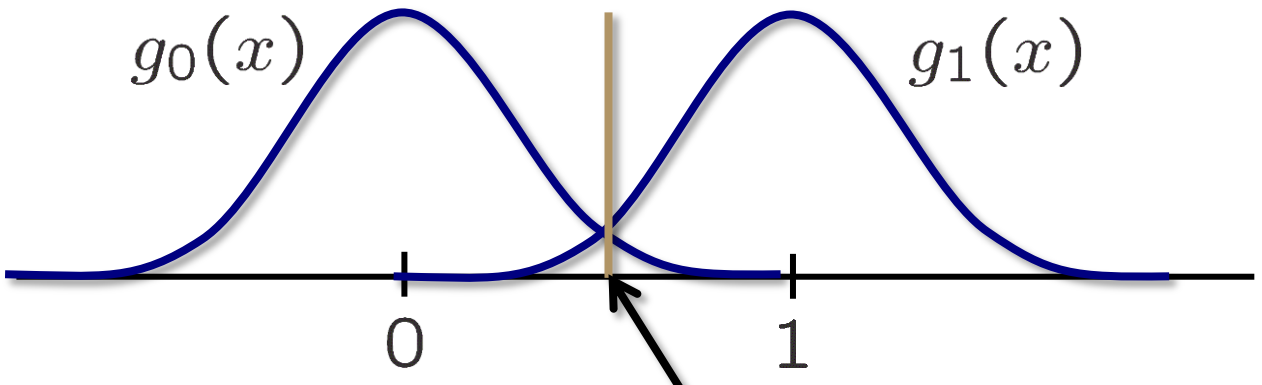
$$f^*(\mathbf{x}) = \arg\max_k g_k(\mathbf{x})$$

maximum likelihood classifier/detector

Example

Suppose that $K = 2$ and that

$$X|Y = 0 \sim \mathcal{N}(0, 1) \qquad X|Y = 1 \sim \mathcal{N}(1, 1)$$



$$\frac{g_1(\mathbf{x})}{g_0(\mathbf{x})} \underset{1}{\overset{0}{\lessgtr}} \frac{\pi_0}{\pi_1}$$

If $\pi_0 = \pi_1$

Generative models and plug-in methods

The Bayes classifier requires knowledge of the joint distribution of (X, Y)

In learning, all we have is the training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$

A generative model is an assumption about the unknown distribution

- typically ***parametric***
- build classifier by estimating the parameters via training data
- plug the result into formula for Bayes classifier
- often called “plug-in” methods

Linear discriminant analysis (LDA)

In linear discriminant analysis, we assume

$$X|Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$$

for $k = 0, \ldots, K-1$

Here $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is the multivariate Gaussian/normal distribution with parameters $\boldsymbol{\mu}$ and Σ and pdf

$$\phi(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Note: Each class has the same covariance matrix Σ

Parameter estimation

In LDA, we assume that the prior probabilities π_k, the mean vectors $\boldsymbol{\mu}_k$, and the covariance matrix Σ are all unknown

To estimate these from the data, we use

$$\widehat{\pi}_k = \frac{|\{i : y_i = k\}|}{n}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{|\{i : y_i = k\}|} \sum_{i:y_i=k} x_i$$

$$\widehat{\Sigma} = \frac{1}{n} \sum_{k=0}^{K-1} \sum_{i:y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^T$$

"pooled covariance estimate"

Resulting classifier

The LDA classifier is then

$$
\begin{aligned}
f(\mathbf{x}) &= \arg\max_k \hat{\pi}_k \cdot \phi(\mathbf{x}; \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}) \\
&= \arg\max_k \log \hat{\pi}_k + \log \phi(\mathbf{x}; \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}) \\
&= \arg\max_k \log \hat{\pi}_k - \frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)^T \hat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \hat{\boldsymbol{\mu}}_k) \\
&= \arg\min_k \underbrace{(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)^T \hat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \hat{\boldsymbol{\mu}}_k)}_{\text{squared } \textit{Mahalanobis distance}} - 2 \log \hat{\pi}_k
\end{aligned}
$$

between $\mathbf{x}$ and $\boldsymbol{\mu}$

$$
d_M(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}
$$

Example

Suppose that $K = 2$

$$d_M(\mathbf{x}; \widehat{\boldsymbol{\mu}}_0, \widehat{\boldsymbol{\Sigma}}) - 2 \log \widehat{\pi}_0 \underset{1}{\overset{0}{\lessgtr}} d_M(\mathbf{x}; \widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\Sigma}}) - 2 \log \widehat{\pi}_1$$

It turns out that by setting

$$\mathbf{a} = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_0 - \widehat{\boldsymbol{\mu}}_1)$$

$$b = -\tfrac{1}{2}\widehat{\boldsymbol{\mu}}_0^T \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_0 + \tfrac{1}{2}\widehat{\boldsymbol{\mu}}_1^T \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_1 + \log \frac{\widehat{\pi}_0}{\widehat{\pi}_1}$$

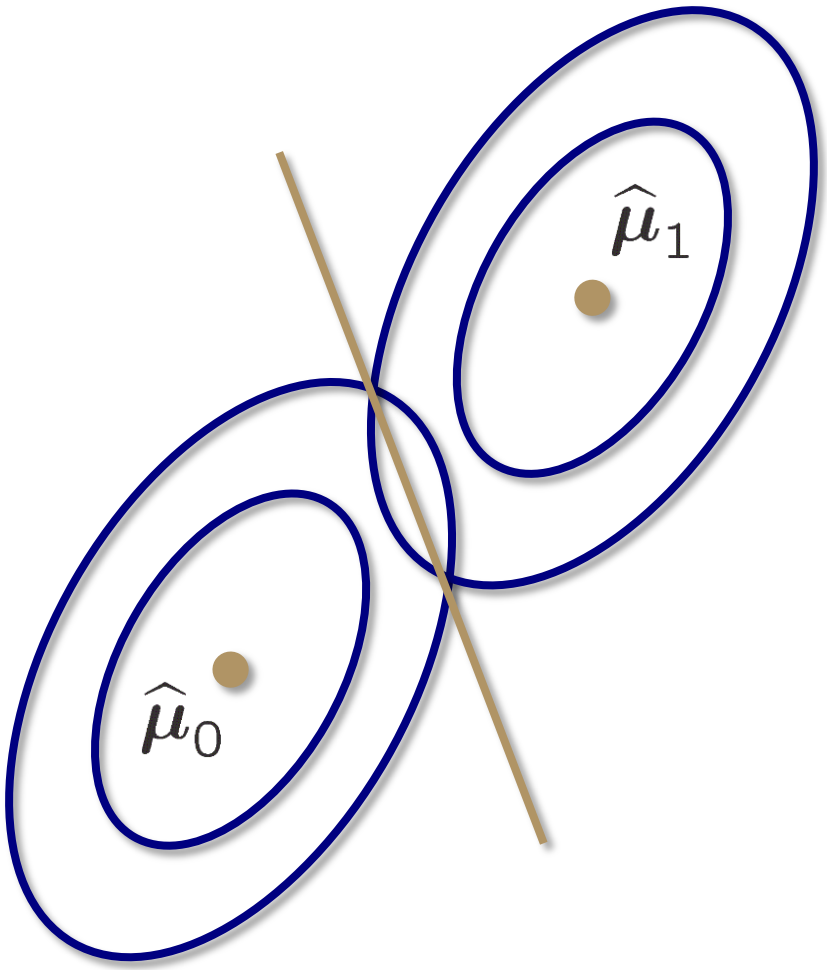
we can re-write this as

$$\mathbf{a}^T \mathbf{x} + b \underset{0}{\overset{1}{\lessgtr}} 0$$

linear classifier

Example

Recall that the contour $\{\mathbf{x} : d_M(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = c\}$ is an ***ellipse***

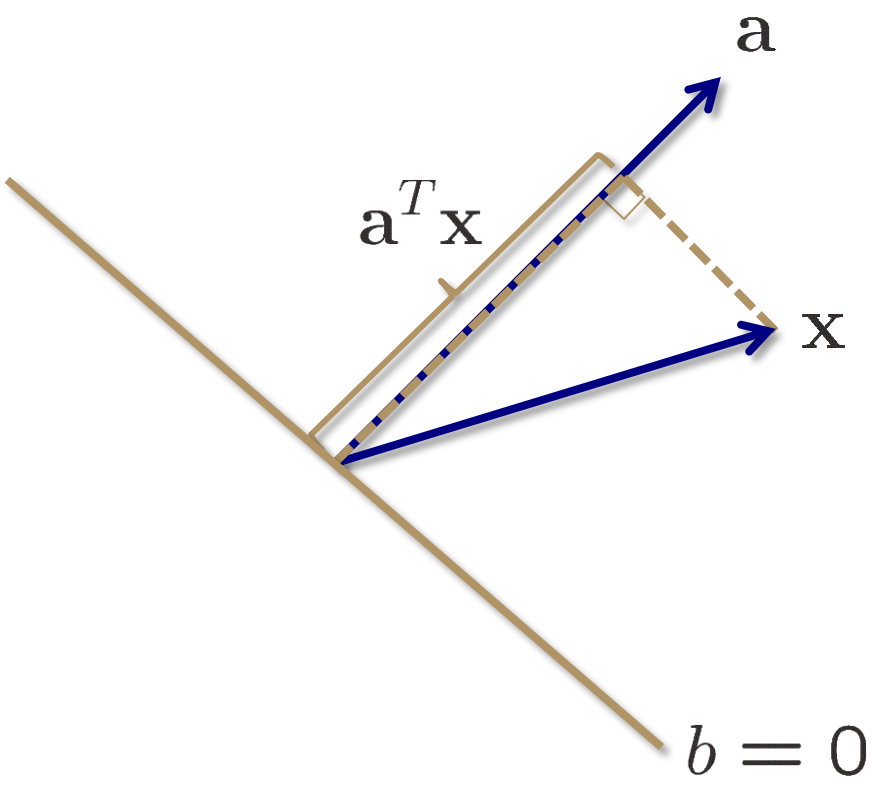


picture assumes

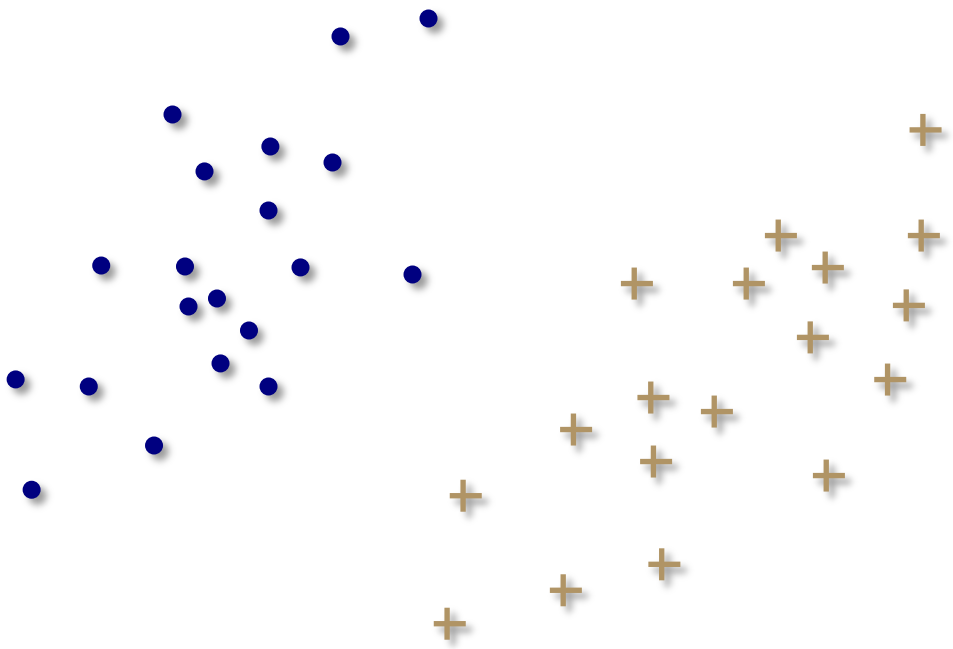
$$\widehat{\boldsymbol{\mu}}_0 = \widehat{\boldsymbol{\mu}}_1$$

Linear classifiers

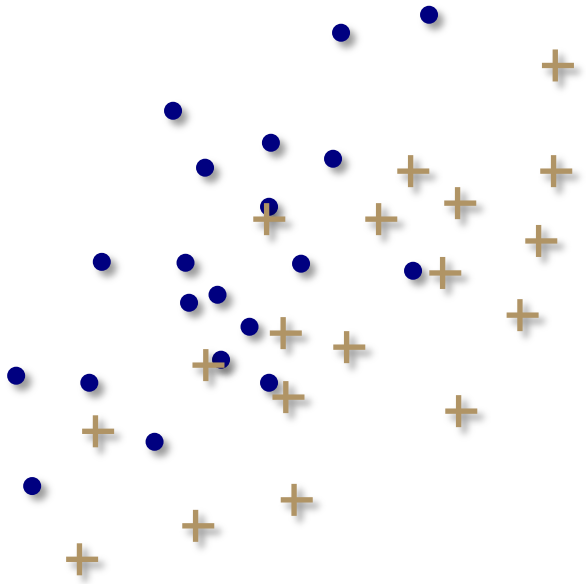
In general, why does $\mathbf{a}^T\mathbf{x} + b \underset{0}{\overset{1}{\lessgtr}} 0$ describe a linear classifier?



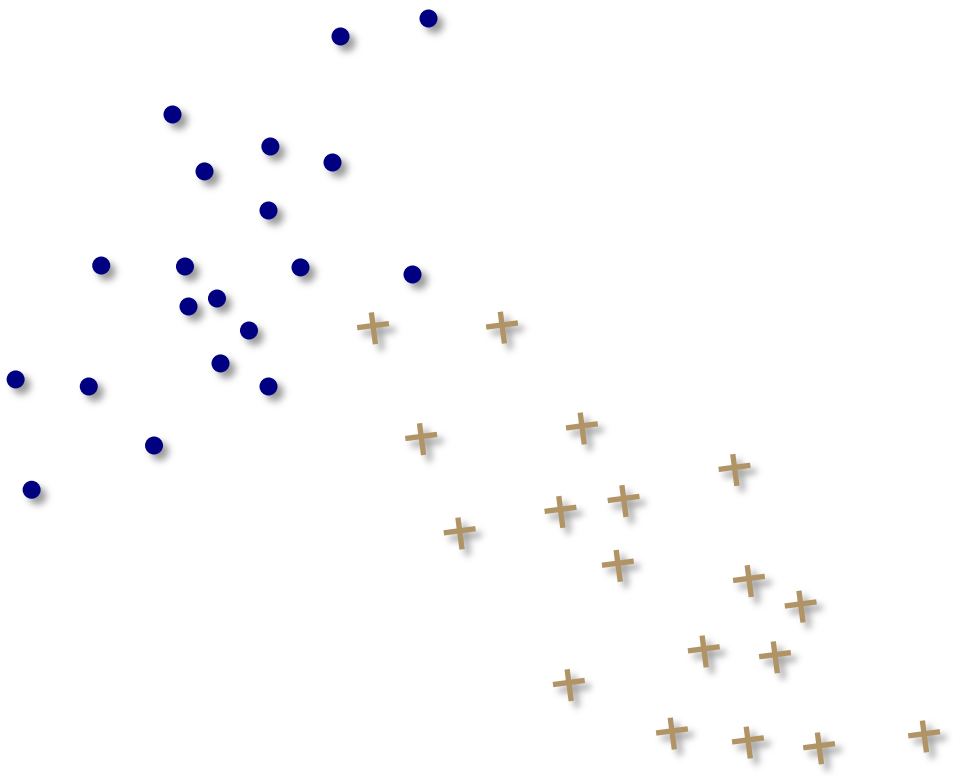
When is LDA appropriate?



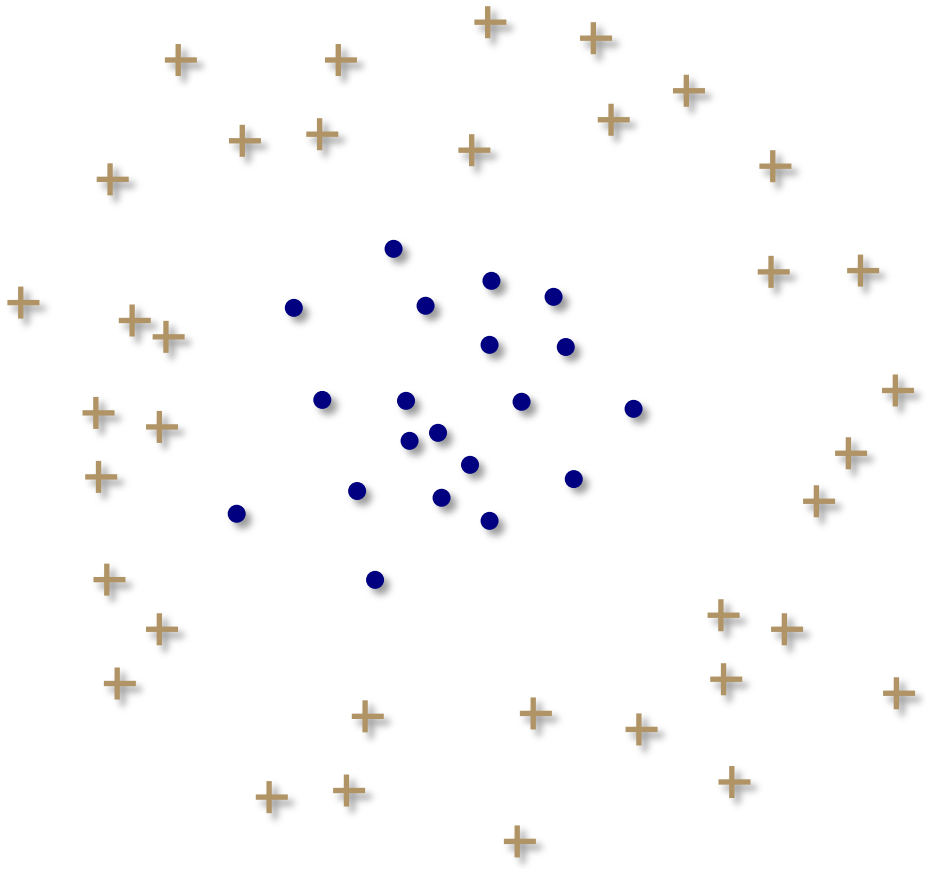
When is LDA appropriate?



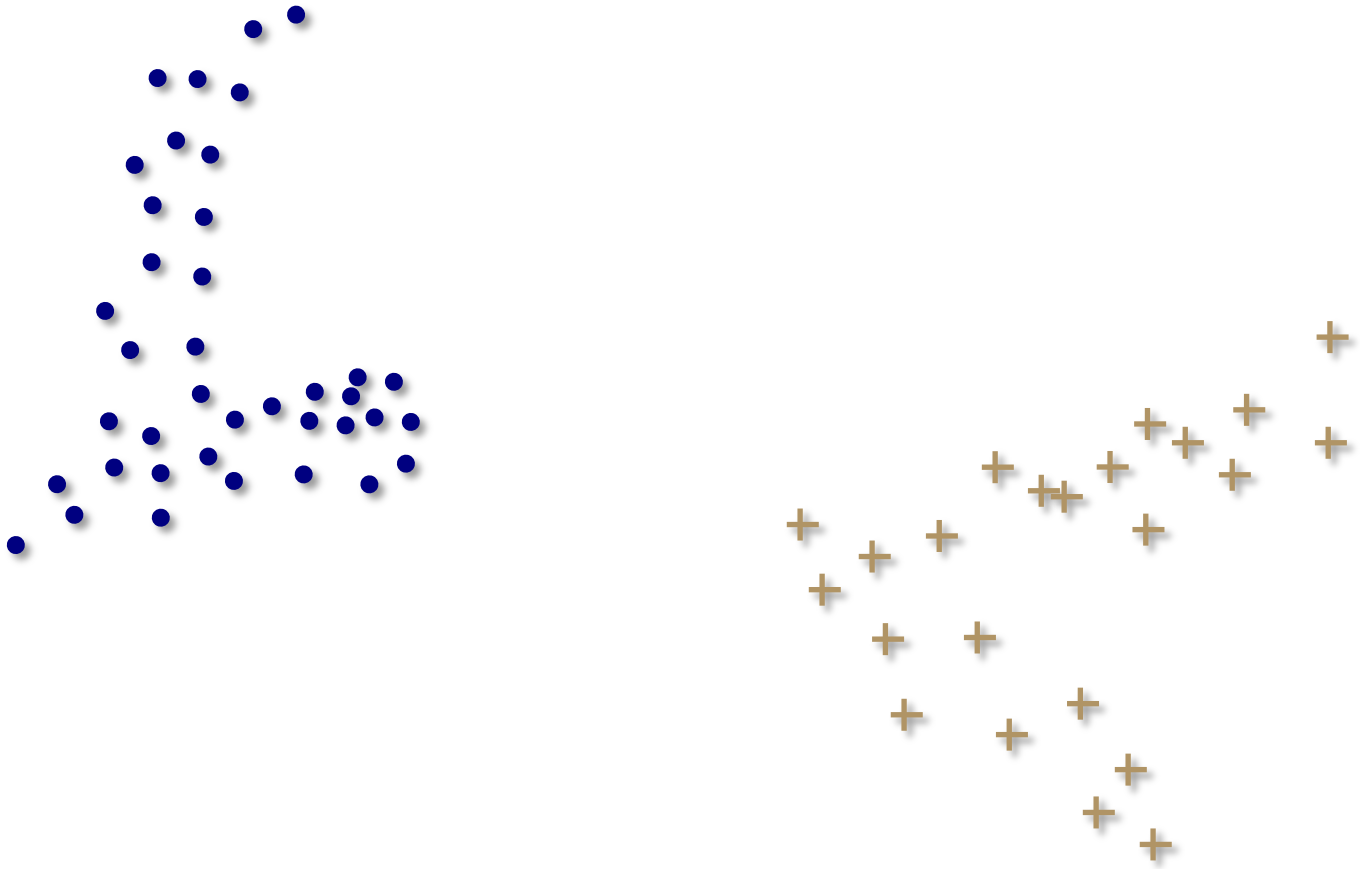
When is LDA appropriate?



When is LDA appropriate?

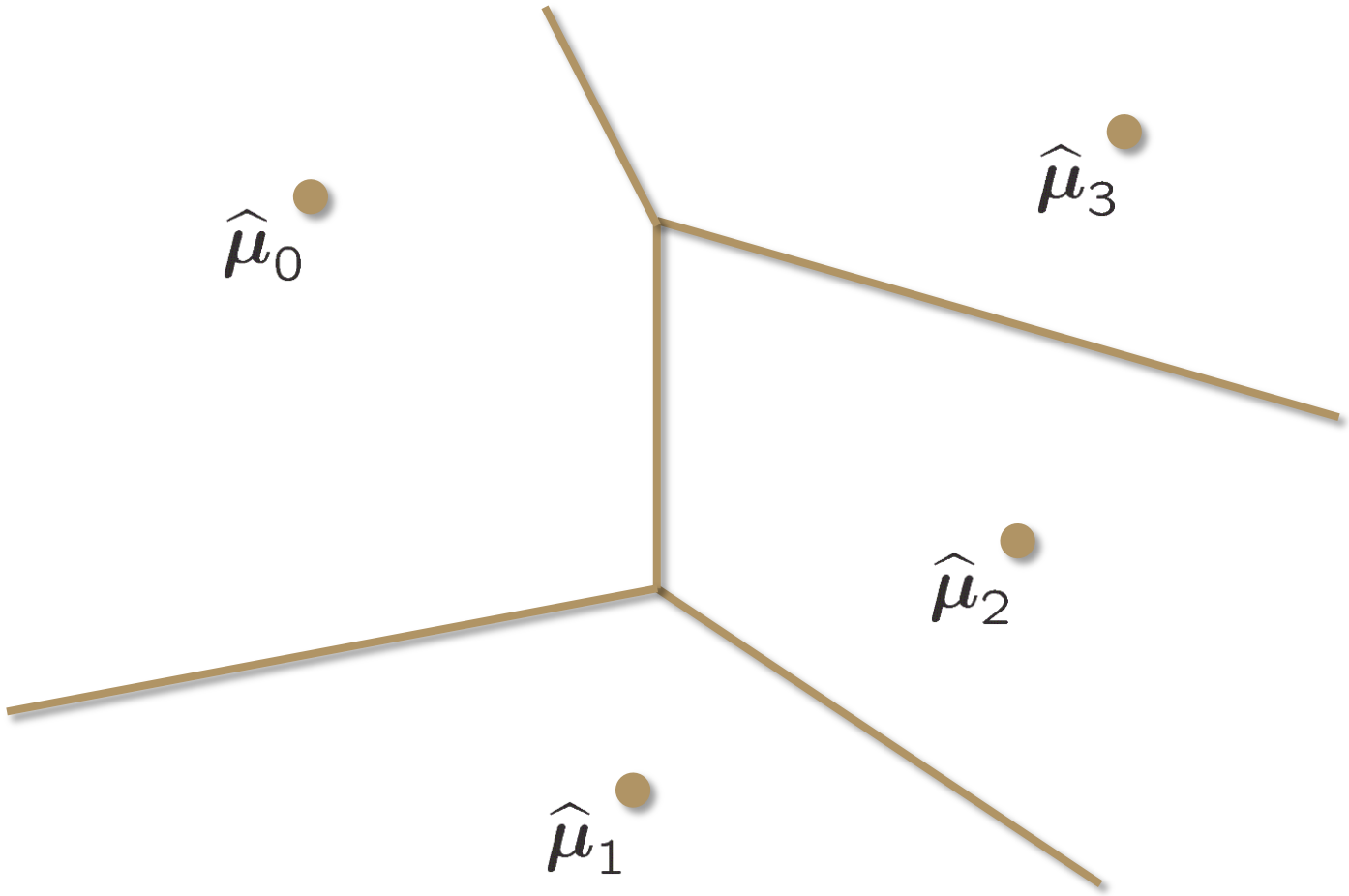


When is LDA appropriate?



More than two classes

The ***decision regions*** are convex polytopes (intersections of linear half-spaces)



Challenges for LDA

The generative model is rarely valid

Moreover, the number of parameters to be estimated is

- class prior probabilities: $K - 1$
- means: Kd
- covariance matrix: $\frac{1}{2}d(d+1)$

If n is small and d is large, then we have more parameters than observations, and will likely obtain poor estimates

- first apply a dimensionality reduction technique like PCA to reduce d
- assume a more structured covariance matrix

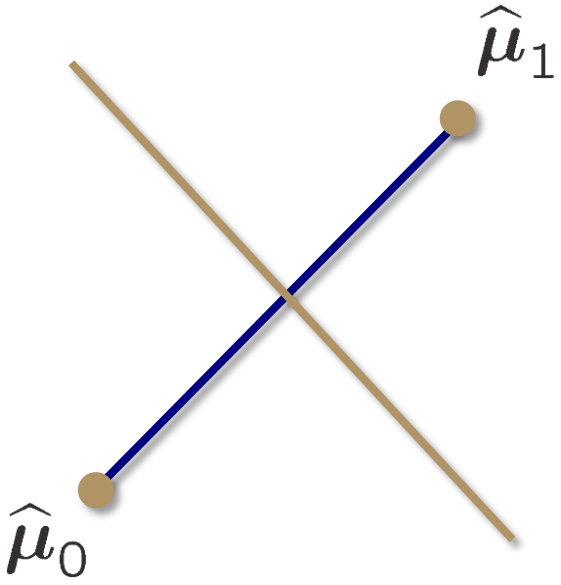
Example

Structured covariance matrix:

Assume $\Sigma = \sigma^2 \mathbf{I}$ and estimate $\widehat{\sigma}^2 = \frac{1}{d}\text{tr}(\widehat{\Sigma})$

If $K = 2$ and $\widehat{\pi}_0 = \widehat{\pi}_1$, then LDA becomes

$$\frac{1}{\sigma^2}\|\mathbf{x} - \widehat{\boldsymbol{\mu}}_0\|^2 \underset{1}{\overset{0}{\lessgtr}} \frac{1}{\sigma^2}\|\mathbf{x} - \widehat{\boldsymbol{\mu}}_1\|^2 \iff \|\mathbf{x} - \widehat{\boldsymbol{\mu}}_0\|^2 \underset{1}{\overset{0}{\lessgtr}} \|\mathbf{x} - \widehat{\boldsymbol{\mu}}_1\|^2$$



nearest centroid classifier

Quadratic discriminant analysis (QDA)

What happens if we expand the generative model to

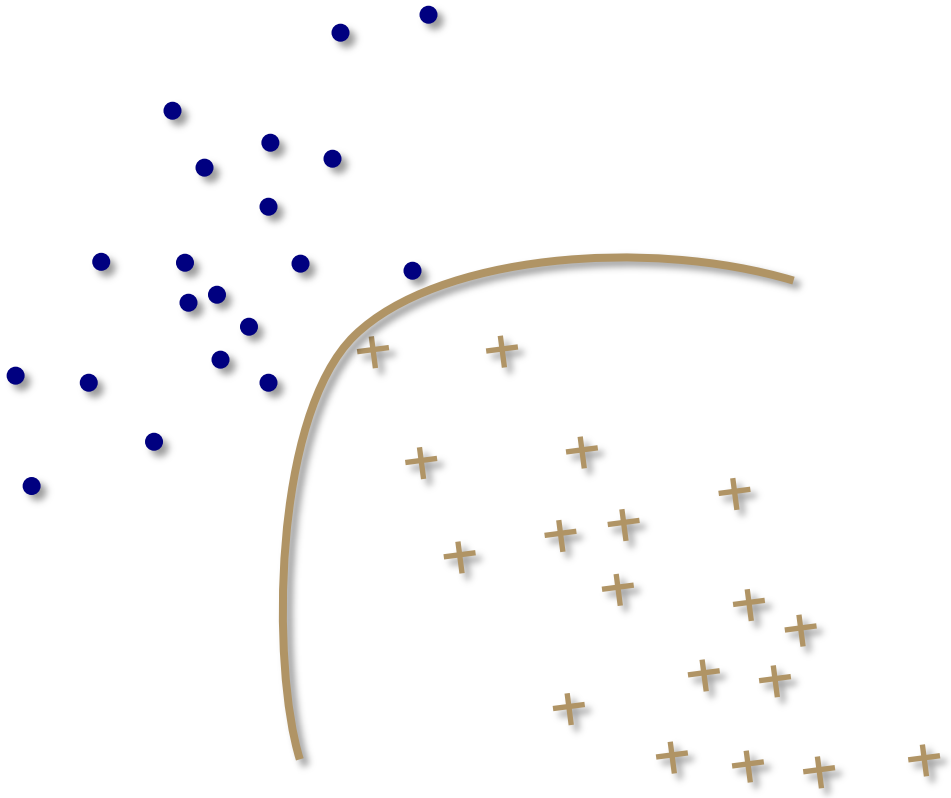
$$X|Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

for $k = 0, \ldots, K-1$?

Set $\widehat{\boldsymbol{\Sigma}}_k = \dfrac{1}{|\{i : y_i = k\}|} \displaystyle\sum_{i:y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^T$

Proceed as before, only this case the decision boundaries will be ***quadratic***

Example



Another look at plugin methods

Suppose $K = 2$

Define $\eta(\mathbf{x}) = \mathbb{P}[Y = 1 | X = \mathbf{x}]$

$$= 1 - \mathbb{P}[Y = 0 | X = \mathbf{x}]$$

In this case, another way to express the Bayes classifier is as

$$f^*(\mathbf{x}) = \begin{cases} 1 & \text{if } \eta(\mathbf{x}) \geq 1/2 \\ 0 & \text{if } \eta(\mathbf{x}) < 1/2 \end{cases}$$

Note that we do not actually need to know the full distribution of (X, Y) to express the Bayes classifier

All we really need is the distribution of $Y|X$

Logistic regression

This observation gives rise to another class of plugin methods, the most important of which is logisitic regression, which implements the following strategy

1. Assume $\eta(\mathbf{x}) = \dfrac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$ $\quad (\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R})$

2. Compute the MLE of $\theta = (\mathbf{w}, b)$

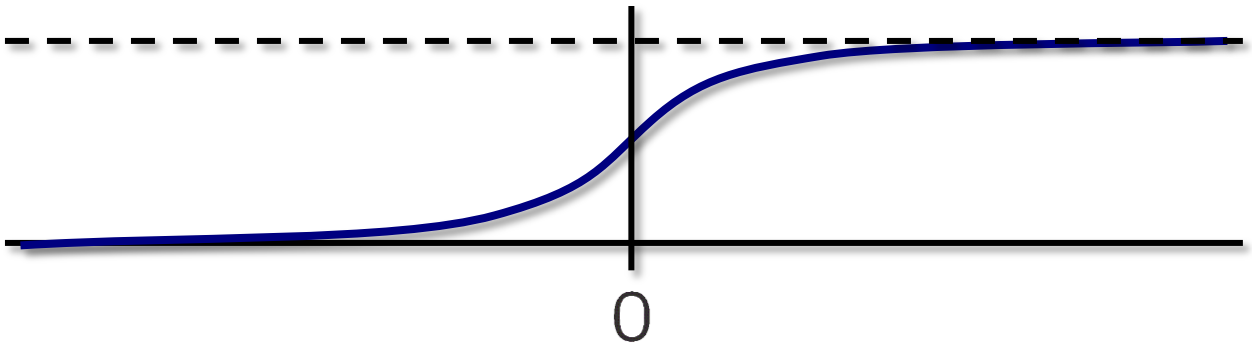
3. Plug the estimate

$$\widehat{\eta}(\mathbf{x}) = \frac{1}{1 + e^{-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})}}$$

   into the formula for the Bayes classifier

The logistic function

The function $\frac{1}{1+e^{-t}}$ is called a ***logistic*** function (or a ***sigmoid*** function in other contexts)



The logistic regression classifier

Denote the logistic regression classifier by

$$\widehat{f}(\mathbf{x}) = 1_{\left\{\widehat{\eta}(\mathbf{x}) \geq 1/2\right\}}$$

Note that $\widehat{f}(\mathbf{x}) = 1 \iff \widehat{\eta}(\mathbf{x}) \geq \frac{1}{2}$

$$\iff \frac{1}{1+\exp\left(-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})\right)} \geq \frac{1}{2}$$

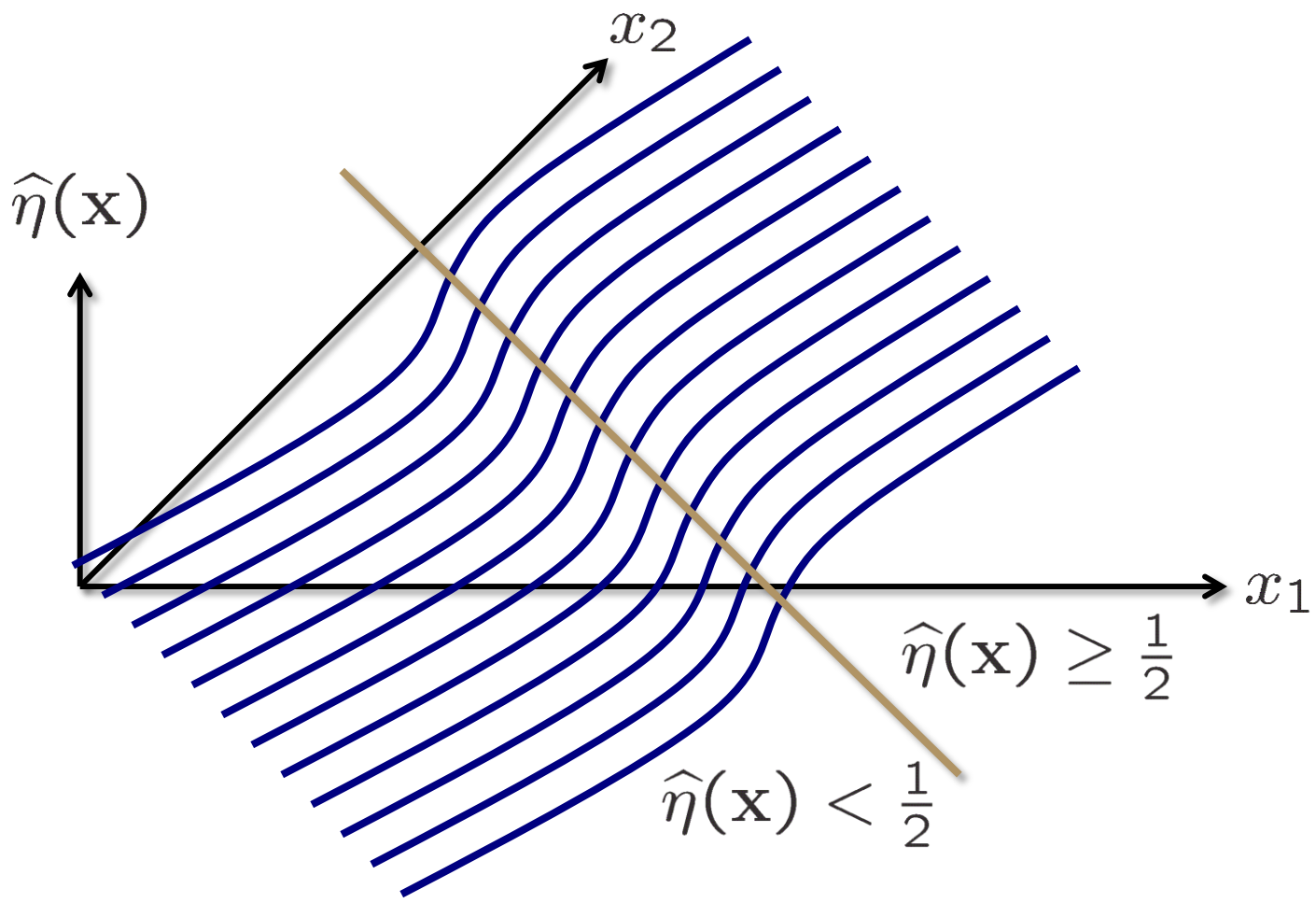
$$\iff \exp\left(-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})\right) \leq 1$$

$$\iff (\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b}) \geq 0$$

So $\widehat{f}(\mathbf{x}) = \begin{cases} 1 & \text{if } \widehat{\mathbf{w}}^T\mathbf{x}+b \geq 0 \\ 0 & \text{otherwise} \end{cases}$

linear classifier

Example


x_2
$\widehat{\eta}(\mathbf{x})$
x_1
$\widehat{\eta}(\mathbf{x}) \geq \frac{1}{2}$
$\widehat{\eta}(\mathbf{x}) < \frac{1}{2}$

Lecture 4: More Linear Classifiers

The Bayes classifier

Consider (X, Y) where

- X is a random vector in $\mathbb{R}^d$
- $Y \in \{0, \ldots, K-1\}$is a random variable (depending on X)
- Define $\eta_k(\mathbf{x}) := \mathbb{P}[Y = k | X = \mathbf{x}]$

Bayes classifier

$$f^*(\mathbf{x}) := \arg\max_k \eta_k(\mathbf{x})$$

Linear discriminant analysis (LDA)

In linear discriminant analysis, we assume

$$X|Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$$

for $k = 0, \ldots, K-1$

Here $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is the multivariate Gaussian/normal distribution with parameters $\boldsymbol{\mu}$ and Σ and pdf

$$\phi(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Note: Each class has the same covariance matrix Σ

Parameter estimation

In LDA, we assume that the prior probabilities π_k, the mean vectors $\boldsymbol{\mu}_k$, and the covariance matrix Σ are all unknown

To estimate these from the data, we use

$$\widehat{\pi}_k = \frac{|\{i : y_i = k\}|}{n}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{|\{i : y_i = k\}|} \sum_{i:y_i=k} x_i$$

$$\widehat{\Sigma} = \frac{1}{n} \sum_{k=0}^{K-1} \sum_{i:y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^T$$

"pooled covariance estimate"

Another look at plugin methods

Suppose $K = 2$

Define $\eta(\mathbf{x}) = \mathbb{P}[Y = 1 | X = \mathbf{x}]$

$$= 1 - \mathbb{P}[Y = 0 | X = \mathbf{x}]$$

In this case, another way to express the Bayes classifier is as

$$f^*(\mathbf{x}) = \begin{cases} 1 & \text{if } \eta(\mathbf{x}) \geq 1/2 \\ 0 & \text{if } \eta(\mathbf{x}) < 1/2 \end{cases}$$

Note that we do not actually need to know the full distribution of (X, Y) to express the Bayes classifier

All we really need is the distribution of $Y|X$

Logistic regression

This observation gives rise to another class of plugin methods, the most important of which is logisitic regression, which implements the following strategy

1. Assume $\eta(\mathbf{x}) = \dfrac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$ $\quad (\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R})$

2. Compute the MLE of $\theta = (\mathbf{w}, b)$

3. Plug the estimate

$$\widehat{\eta}(\mathbf{x}) = \frac{1}{1 + e^{-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})}}$$

   into the formula for the Bayes classifier

The likelihood function

Likelihood function is given by

$$\mathcal{L}(\theta; y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) = \mathbb{P}[y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n; \theta]$$

$$= \prod_{i=1}^{n} \mathbb{P}[y_i | \mathbf{x}_i; \theta]$$

conditional independence

For brevity, we'll just write $\mathcal{L}(\theta)$

Note that $Y|X$ is a Bernoulli random variable with

$$\mathbb{P}[y|\mathbf{x}; \theta] = \begin{cases} \eta(\mathbf{x}; \theta) & \text{if } y = 1 \\ 1 - \eta(\mathbf{x}; \theta) & \text{if } y = 0 \end{cases}$$

$$= \eta(\mathbf{x}; \theta)^y (1 - \eta(\mathbf{x}; \theta))^{1-y}$$

The log-likelihood

Thus

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} \eta(\mathbf{x}_i;\theta)^{y_i}(1-\eta(\mathbf{x}_i;\theta))^{1-y_i}$$

and the ***log-likelihood*** function is given by

$$\begin{aligned}\ell(\theta) &= \log \mathcal{L}(\theta) \\ &= \sum_{i=1}^{n} y_i \log \eta(\mathbf{x}_i;\theta) + (1-y_i)\log(1-\eta(\mathbf{x}_i;\theta))\end{aligned}$$

Since this is a monotonic transformation, finding maximizing $\ell(\theta)$ is equivalent to maximizing $\mathcal{L}(\theta)$

Maximum likelihood estimation

Notation

- $\widetilde{\mathbf{x}} = [1, x(1), \ldots, x(d)]^T$
- $\boldsymbol{\theta} = [b, w(1), \ldots, w(d)]^T$
- $g(t) = \frac{1}{1+e^{-t}}$

Note that $\log g(t) = -\log(1 + e^{-t})$
and $\log(1 - g(t)) = -t - \log(1 + e^{-t})$

This lets us write $\eta(\mathbf{x}) = g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}})$, and so we have

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \log g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i) + (1 - y_i) \log(1 - g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i))$$

$$= -\sum_{i=1}^{n} (1 - y_i) \boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i + \log(1 + e^{-\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})$$

Maximum likelihood estimation

To maximize the likelihood, try taking the derivatives and setting them equal to zero

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= -\sum_{i=1}^{n} \frac{\partial}{\partial \boldsymbol{\theta}} \left((1-y_i)\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i + \log(1+e^{-\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i}) \right) \\ &= -\sum_{i=1}^{n} (1-y_i)\widetilde{\mathbf{x}}_i - \widetilde{\mathbf{x}}_i (1+e^{-\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})^{-1} \\ &= \sum_{i=1}^{n} \widetilde{\mathbf{x}}_i (y_i - 1 - (1+e^{-\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})^{-1}) \;\; = 0\end{aligned}$$

This gives us $d+1$ equations, but they are ***nonlinear*** and have no closed-form solution

Newton-Raphson algorithm

The log-likelihood function is ***concave***, and therefore has a ***global maximum***

A practical way to find the maximum is using an iterative algorithm, such as the ***Newton-Raphson algorithm***

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \left(\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \right)^{-1} \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

where the derivatives are evaluated at $\boldsymbol{\theta}^{\text{old}}$

Hessian

Another plugin method: Naïve Bayes

The Naïve Bayes classifier is another approach based on estimating the distribution of the data and then plugging this into the Bayes classifier

Makes a (probably naïve) assumption:

Let $X = [X(1), \ldots, X(d)]^T \in \mathbb{R}^d$ denote the random feature vector in a classification problem and Y the corresponding label

The naïve Bayes classifier assumes that, given Y, $X(1), \ldots, X(d)$ are ***independent***

Sometimes known as “Idiot Bayes”

What does the NB assumption buy us?

The major advantage of NB is that we only need to estimate ***scalar/univariate*** densities

Let $g_k(\mathbf{x})$ be the probability law (density or mass function) of $X|Y = k, k = 1, \ldots, K$

By the NB assumption

$$g_k(\mathbf{x}) = \prod_{j=1}^{d} g_k^j(x(j))$$

where $g_k^j(x(j))$ denotes the probability law of $X(j)|Y = k$

Naïve Bayes classifier

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be training data and set

- $\widehat{\pi}_k = \dfrac{|\{i : y_i = k\}|}{n}$
- $\widehat{g}_k^j =$ any estimate of g_k^j based on $\{x_i(j) : y_i = k\}$

The NB classifier is then given by

$$\widehat{f}(\mathbf{x}) = \arg\max_k \ \widehat{\pi}_k \prod_{j=1}^{d} \widehat{g}_k^j(x(j))$$

So, how can we determine $\widehat{g}_k^j$?

Continuous features

Suppose that $X(j)|Y = k$ is a continuous random variable

Options for estimating $\widehat{g}_k^j$ include

- parametric estimate (e.g., Gaussian MLE)
- kernel density estimate
- quantize to a discrete random variable

Discrete features

Now suppose that $X(j)|Y = k$ takes only the values $z_1, \ldots, z_L$

Define $n_k = |\{i : y_i = k\}|$

$$n_{k\ell}^j = |\{i : y_i = k \text{ and } x_i(j) = z_\ell\}|$$

for $k = 0, \ldots, K-1$

Then a natural (maximum likelihood) estimate of $\mathbb{P}[X(j) = z_\ell | Y = k]$ is

$$\widehat{g}_k^j(z_\ell) = \frac{n_{k\ell}^j}{n_k}$$

Undesirable property

It is possible that after training a NB classifier, we may be given an input $\mathbf{x}$ where some feature $x(j)$ takes a value z' that was not observed in the training data for some class k

For such a class, $\widehat{g}_k^j(z') = 0$, and hence $\widehat{g}_k(\mathbf{x}) = 0$

In this case, class k will never be predicted

This is rather unfortunate…

Example: Document classification

Suppose we wish to classify documents into categories

- 0 = politics
- 1 = sports
- 2 = finance
- …

One simple (but useful) way to represent a document is as $\mathbf{x} = [x(1), \ldots, x(d)]^T$ with d representing the number of words in our "vocabulary" and

$$x(j) = \begin{cases} 1 & \text{if word } j \text{ occurs in the document} \\ 0 & \text{otherwise} \end{cases}$$

similar to "bag of words" model

Example: Document classification

It could happen that every training document in “sports” contains the word “ball”.

What happens when we get an article about hockey?

To avoid this problem, it is common to replace the maximum likelihood estimate before with

$$\widehat{g}_k^j(z_\ell) = \frac{n_{k\ell}^j + 1}{n_k + L}$$

We can think of this as the result of adding L “pseudo-samples” to our data to ensure that each feature occurs at least once

Bayesian estimate with a Dirichlet prior

(See also Laplace’s rule of succession)

Comparison of plugin methods

LDA, LR, and NB are three different ***plugin methods*** that result in ***linear*** classifiers

Linear discriminant analysis

- best if Gaussianity assumptions are valid

Logistic regression

- models only the distribution of $Y|X$, not (X, Y)
- valid for a larger class of distributions
- fewer parameters to estimate

Naïve Bayes

- scales well to high-dimensions
- naturally handles mixture of discrete and continuous features

Beyond plugin methods

Plugin methods can be useful in practice, but ultimately they are very limited

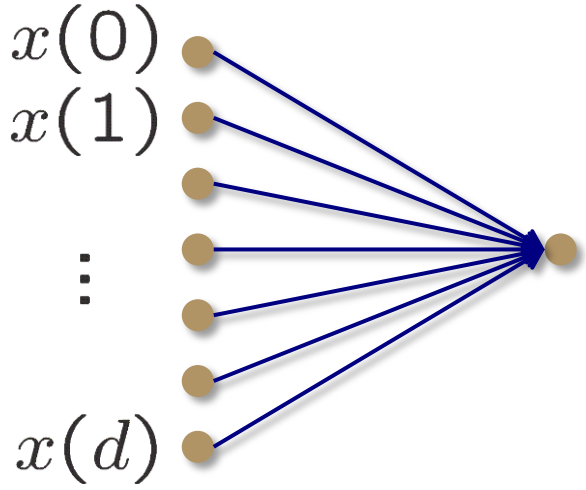
- There are always distributions where our assumptions are violated
- If our assumptions are wrong, the output is totally unpredictable
- Can be hard to verify whether our assumptions are right
- Require solving a more difficult problem as an intermediate step

Most of the remainder of this course will focus on ***nonparametric*** methods that avoid making such strong assumptions about the (unknown) process generating the data

Nonparametric linear classifiers

Suppose $K = 2$ and that $Y \in \{-1, +1\}$

By transforming $\mathbf{x}$ to $\mathbb{R}^{d+1}$ and inserting $x(0) = 1$, we can reduce any linear classifier to comparing $\mathbf{w}^T\mathbf{x}$ to a threshold for some $\mathbf{w}$



$$y = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} > \text{threshold} \\ -1 & \text{otherwise} \end{cases}$$

$$y = \text{sign}(\mathbf{w}^T\mathbf{x})$$

Single layer neural network

How to learn $\mathbf{w}$?

Perceptron Learning Algorithm (PLA)

Frank Rosenblatt (1957)

Given

- training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
- a guess for $\mathbf{w}$

Find a misclassified point, i.e., an i such that

$$y_i \neq \text{sign}(\mathbf{w}^T \mathbf{x}_i)$$

Set $\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + y_i \mathbf{x}_i$

Iterate

One iteration of the PLA: $\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + y_i \mathbf{x}_i$

At each iteration, pick another misclassified point from

$$\{i : y_i \neq \text{sign}(\mathbf{w}^T \mathbf{x})\}$$

and run another PLA iteration

That's it!

Why might this work?

$$\begin{aligned}(\mathbf{w}^{\text{new}})^T \mathbf{x}_i &= (\mathbf{w}^{\text{old}} + y_i \mathbf{x}_i)^T \mathbf{x}_i \\ &= (\mathbf{w}^{\text{old}})^T \mathbf{x}_i + y_i \mathbf{x}_i^T \mathbf{x}_i\end{aligned}$$

pushes in the right direction

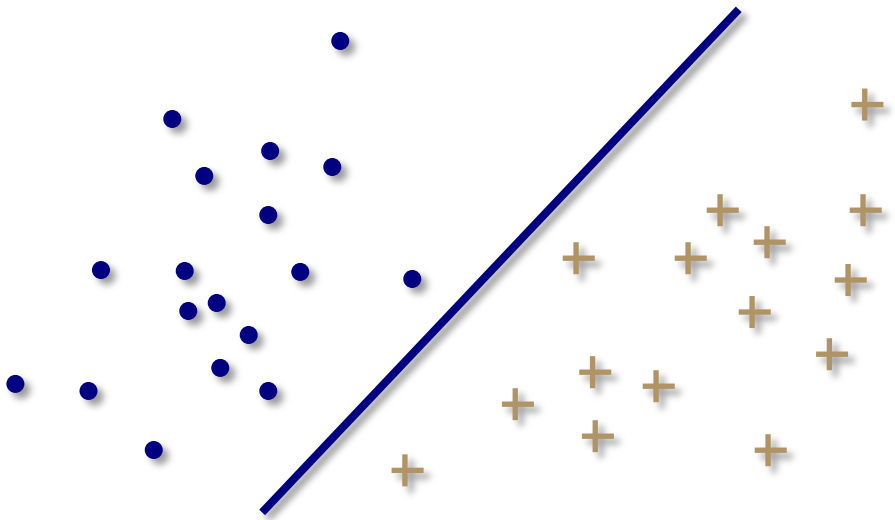
Linearly separable data sets

We say that a data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is ***linearly separable*** if there exists $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

$$y_i = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$$

for $i = 1, \ldots, n$

We refer to $\{\mathbf{x} : \mathbf{w}^T \mathbf{x} + b = 0\}$ as a ***separating hyperplane***



How can we find a separating hyperplane?

One approach is to use the PLA (see homework)

We can also take a more geometric perspective

Let $\mathbf{w}, b$ define a hyperplane, and pick two points $\mathbf{x}, \mathbf{x}'$ on the hyperplane, then

$$\begin{aligned} 0 &= (\mathbf{w}^T\mathbf{x} + b) - (\mathbf{w}^T\mathbf{x}' + b) \\ &= \mathbf{w}^T(\mathbf{x} - \mathbf{x}') \end{aligned}$$

Hence, $\mathbf{w}$ is ***orthogonal*** to all vectors that are ***parallel*** to the hyperplane

Geometry of separating hyperplanes

We call $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ the ***normal vector*** to the hyperplane

It is unique up to its sign

Question

Let $\mathbf{z} \in \mathbb{R}^d$. How far is $\mathbf{z}$ from $\{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^T\mathbf{x} + b = 0\}$?

Write $\mathbf{z} = \mathbf{z}_0 + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$ where $\mathbf{w}^T\mathbf{z}_0 + b = 0$ and $r \in \mathbb{R}$

Then

$$\begin{aligned}\mathbf{w}^T\mathbf{z} + b &= \mathbf{w}^T\left(\mathbf{z}_0 + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + b \\ &= \mathbf{w}^T\mathbf{z}_0 + b + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} \\ &= r\|\mathbf{w}\|\end{aligned}$$

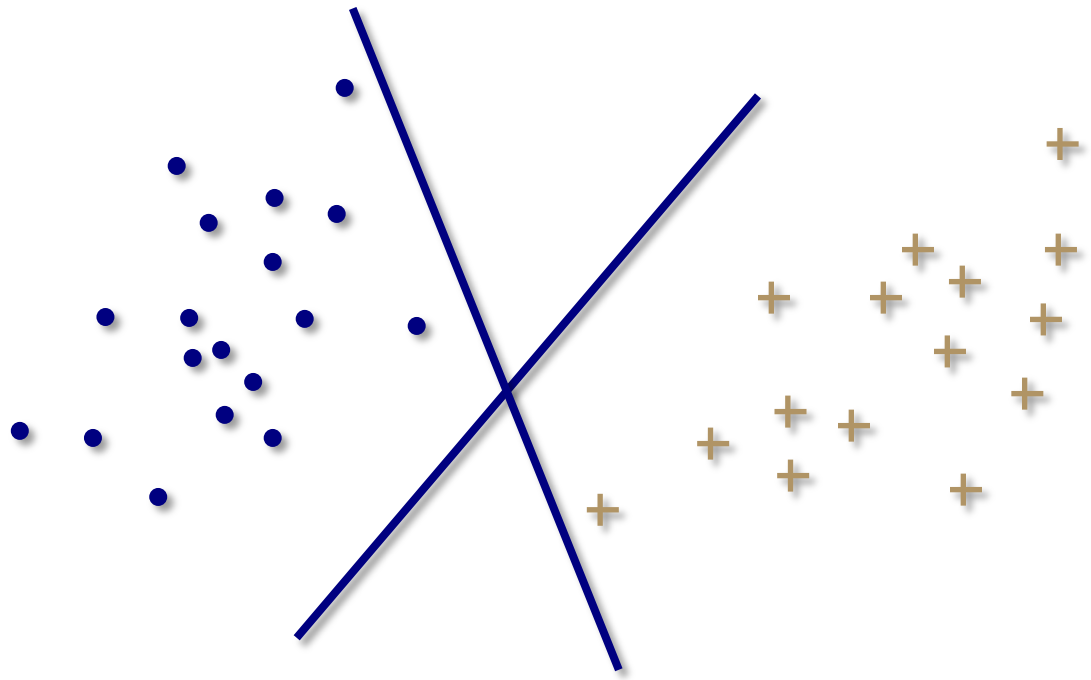
Geometry of separating hyperplanes

$$\mathbf{w}^T\mathbf{z} + b = r\|\mathbf{w}\|$$

$$|r| = \frac{|\mathbf{w}^T\mathbf{z} + b|}{\|\mathbf{w}\|}$$

distance to the hyperplane

Are all separating hyperplanes equal?



The maximum margin hyperplane

The ***margin*** ρ of a separating hyperplane is the distance from the hyperplane to the closest $\mathbf{x}_i$

$$\rho(\mathbf{w}, b) = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

The ***maximum margin*** or ***optimal*** separating hyperplane is the solution of

$$(\mathbf{w}^*, b^*) = \arg\max_{\mathbf{w},b} \rho(\mathbf{w}, b)$$

Larger margin ➡ better generalization to new data

Lecture 5: Optimal Hyperplanes and Linear Regression

The log-likelihood

Thus

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} \eta(\mathbf{x}_i;\theta)^{y_i}(1-\eta(\mathbf{x}_i;\theta))^{1-y_i}$$

and the ***log-likelihood*** function is given by

$$\begin{aligned}\ell(\theta) &= \log \mathcal{L}(\theta) \\ &= \sum_{i=1}^{n} y_i \log \eta(\mathbf{x}_i;\theta) + (1-y_i)\log(1-\eta(\mathbf{x}_i;\theta))\end{aligned}$$

Since this is a monotonic transformation, finding maximizing $\ell(\theta)$ is equivalent to maximizing $\mathcal{L}(\theta)$

Maximum likelihood estimation

Notation

- $\widetilde{\mathbf{x}} = [1, x(1), \ldots, x(d)]^T$
- $\boldsymbol{\theta} = [b, w(1), \ldots, w(d)]^T$
- $g(t) = \frac{1}{1+e^{-t}}$, which lets us write $\eta(\mathbf{x}) = g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}})$

Note that $\log g(t) = -\log(1 + e^{-t})$ and

$$\log(1 - g(t)) = -t - \log(1 + e^{-t}) = -\log(1 + e^{t})$$

Thus,

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \log g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i) + (1 - y_i) \log(1 - g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i))$$

$$= \sum_{i=1}^{n} y_i \boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i - \log(1 + e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})$$

Maximum likelihood estimation

To maximize the likelihood, try taking the derivatives and setting them equal to zero

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \sum_{i=1}^{n} \frac{\partial}{\partial \boldsymbol{\theta}} \left(y_i \boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i - \log(1 + e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i}) \right) \\ &= \sum_{i=1}^{n} y_i \widetilde{\mathbf{x}}_i - \widetilde{\mathbf{x}}_i e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i} (1 + e^{\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i})^{-1} \\ &= \sum_{i=1}^{n} \widetilde{\mathbf{x}}_i (y_i - g(\boldsymbol{\theta}^T \widetilde{\mathbf{x}}_i)) = 0\end{aligned}$$

This gives us $d + 1$ equations, but they are ***nonlinear*** and have no closed-form solution

Newton-Raphson algorithm

The log-likelihood function is ***concave***, and therefore has a ***global maximum***

A practical way to find the maximum is using an iterative algorithm, such as the ***Newton-Raphson algorithm***

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \left(\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \right)^{-1} \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

where the derivatives are evaluated at $\boldsymbol{\theta}^{\text{old}}$

Hessian

Beyond plugin methods

Plugin methods can be useful in practice, but ultimately they are very limited

- There are always distributions where our assumptions are violated
- If our assumptions are wrong, the output is totally unpredictable
- Can be hard to verify whether our assumptions are right
- Require solving a more difficult problem as an intermediate step

Most of the remainder of this course will focus on ***nonparametric*** methods that avoid making such strong assumptions about the (unknown) process generating the data

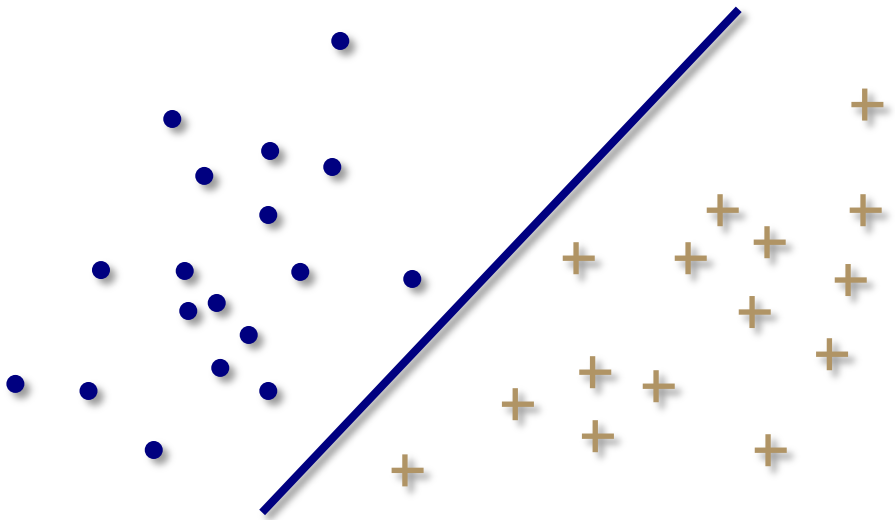
Linearly separable data sets

We say that a data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is ***linearly separable*** if there exists $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

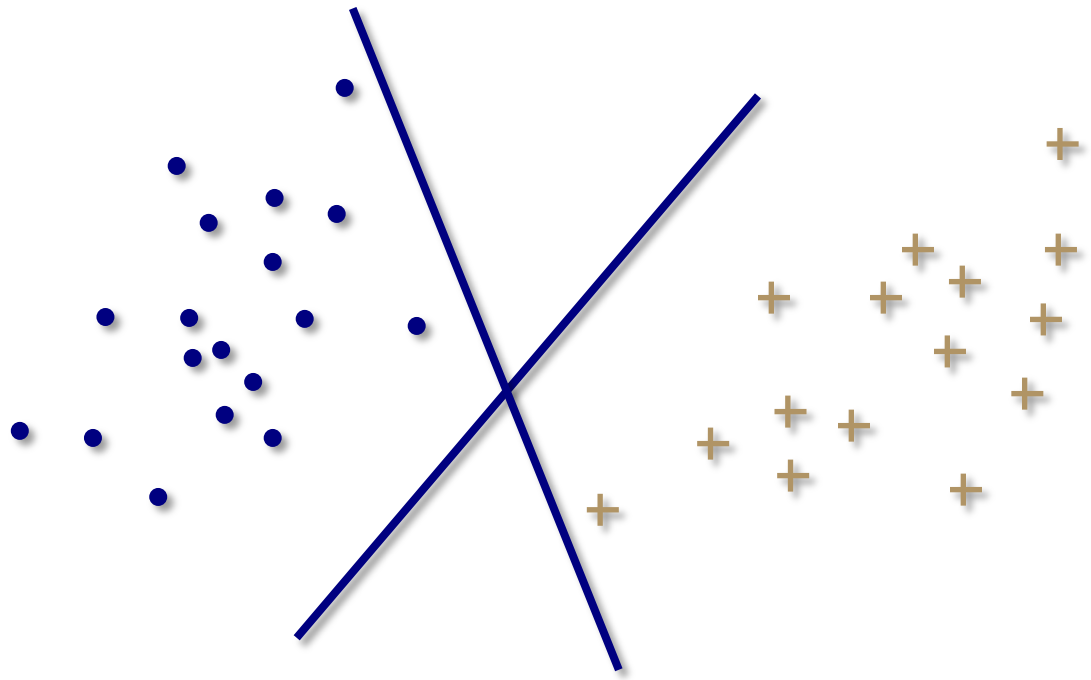
$$y_i = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)$$

for $i = 1, \ldots, n$

We refer to $\{\mathbf{x} : \mathbf{w}^T \mathbf{x} + b = 0\}$ as a ***separating hyperplane***



Are all separating hyperplanes equal?



Geometry of separating hyperplanes

We call $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ the ***normal vector*** to the hyperplane

It is unique up to its sign

Question

Let $\mathbf{z} \in \mathbb{R}^d$. How far is $\mathbf{z}$ from $\{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^T\mathbf{x} + b = 0\}$?

Answer

$$\frac{|\mathbf{w}^T\mathbf{z} + b|}{\|\mathbf{w}\|}$$

The maximum margin hyperplane

The ***margin*** ρ of a separating hyperplane is the distance from the hyperplane to the closest $\mathbf{x}_i$

$$\rho(\mathbf{w}, b) = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

The ***maximum margin*** or ***optimal*** separating hyperplane is the solution of

$$(\mathbf{w}^*, b^*) = \arg\max_{\mathbf{w},b} \rho(\mathbf{w}, b)$$

Larger margin ➡ better generalization to new data

Canonical Form

Our parameterization of a hyperplane as a normal vector $\mathbf{w}$ and an offset b is overdetermined

- e.g., in $\mathbb{R}^2$ we are using three parameters to describe a line, which really only needs two

Another way of seeing this is to realize that

$$\{\mathbf{x} : (\alpha\mathbf{w})^T\mathbf{x} + \alpha b = 0\}$$

describes the same hyperplane for all $\alpha \in \mathbb{R}$

It is often useful to remove this ambiguity by choosing a particular scaling. In the case of a separating hyperplane, we will say $(\mathbf{w}, b)$ are in ***canonical form*** if

- $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ for all i
- $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$ for some i

Maximum margin revisited

If we restrict ourselves to hyperplanes in canonical form, then

$$\rho(\mathbf{w}, b) = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

Thus we can express $(\mathbf{w}^*, b^*) = \arg\max_{\mathbf{w},b} \rho(\mathbf{w}, b)$ as

$$(\mathbf{w}^*, b^*) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \text{ for } i = 1, \ldots, n$$

Optimal separating hyperplanes

$$(\mathbf{w}^*, b^*) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \text{ for } i = 1, \ldots, n$$

- This is an example of a ***quadratic program***
- At the solution, there will always be at least some $\mathbf{x}_i$ such that $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$.

  These are called ***support vectors***
- This optimization problem forms the core idea behind ***support vector machines***

What if our data is not linearly separable?

All of the plugin methods we described can naturally accommodate that isn't perfectly linearly separable

How can we extend the notion of an optimal separating hyperplane to the case of data that is not separable?

The constraint that $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ for every training sample can only be satisfied for separable data

Idea: Introduce slack variables $\xi_1, \ldots, \xi_n \geq 0$ that allow us to violate some of the constraints by changing them to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

How should we set $\xi_1, \ldots, \xi_n$?

Optimal soft-margin hyperplane

We would ideally like for most of the ξ_i to be zero

Note that if $\mathbf{x}_i$ is misclassified, then $\xi_i > 1$

Hence, our training error $\leq \frac{1}{n}\sum_{i=1}^{n}\xi_i$

The optimal ***soft-margin*** hyperplane is given by

$$(\mathbf{w}^*, b^*) = \arg\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, n$$

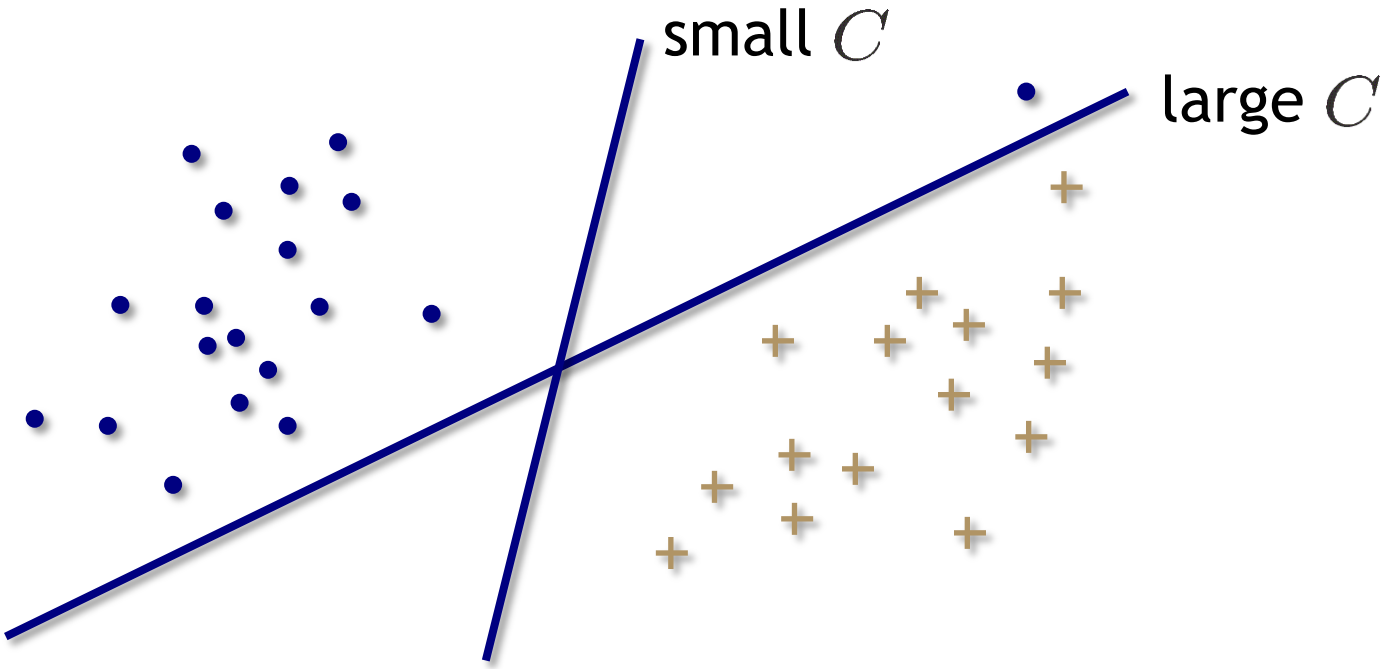
$$\xi_i \geq 0 \quad i = 1, \ldots, n$$

C is a "cost" parameter set by the user

Setting the cost parameter

C allows us to trade off between fitting the data and having a large "margin"

It also controls the influence of outliers



We will discuss strategies for setting C in more detail later on

Other linear classifiers

- Variants of the perceptron/single-layer neural nets
- Fisher's linear discriminant
- least squares approaches
- linear programming approaches
- Bayesian approaches ("relevance vector machines")
- …

Regression

In regression problems we are given training data

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$$

where as before $\mathbf{x}_i \in \mathbb{R}^d$, but now $y_i \in \mathbb{R}$

You can think of regression as being an extension of classification as the number of classes grows to ∞

A ***regression model*** typically posits that our training data are realizations of a random pair (X, Y) where

$$Y = f(X) + E$$

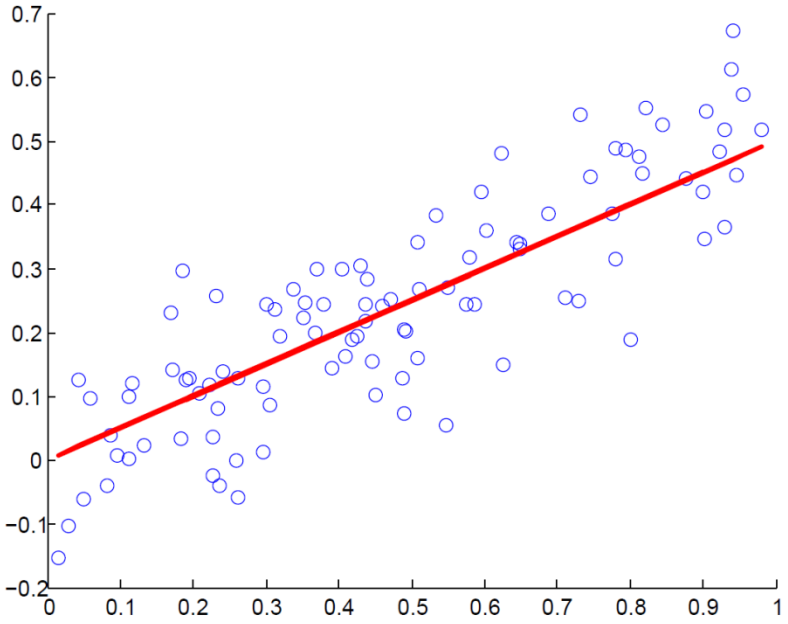
with E representing noise and f belonging to some class of functions

Linear regression

In ***linear regression***, we assume that f is an ***affine*** function, i.e.,

$$f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x} + \beta_0$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$, $\beta \in \mathbb{R}$



How can we estimate $\boldsymbol{\beta}, \beta_0$ from the training data?

Least squares

In ***least squares*** linear regression, we select $\boldsymbol{\beta}, \beta_0$ to minimize the sum of squared errors

$$\text{SSE}(\boldsymbol{\beta}, \beta_0) := \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0\right)^2$$

Least squares is (arguably) the most fundamental tool in all of applied mathematics!

Legendre (1805)

Gauss ~~(1809)~~ (1795)

Example

Suppose $d = 1$, so that x_i, β are scalars

$$\text{SSE}(\beta, \beta_0) = \sum_{i=1}^{n} (y_i - \beta x_i - \beta_0)^2$$

How to minimize?

$$\frac{\partial \text{SSE}}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - \beta x_i - \beta_0) = 0$$

$$\frac{\partial \text{SSE}}{\partial \beta} = -2 \sum_{i=1}^{n} x_i (y_i - \beta x_i - \beta_0) = 0$$

Example

Rearranging these equations, we obtain

$$n\beta_0 + \sum_{i=1}^{n} \beta x_i = \sum_{i=1}^{n} y_i$$

$$\sum_{i=1}^{n} \beta_0 x_i + \sum_{i=1}^{n} \beta x_i^2 = \sum_{i=1}^{n} x_i y_i$$

or in matrix form

$$\begin{bmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta \end{bmatrix} = \begin{bmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{bmatrix}$$

Example

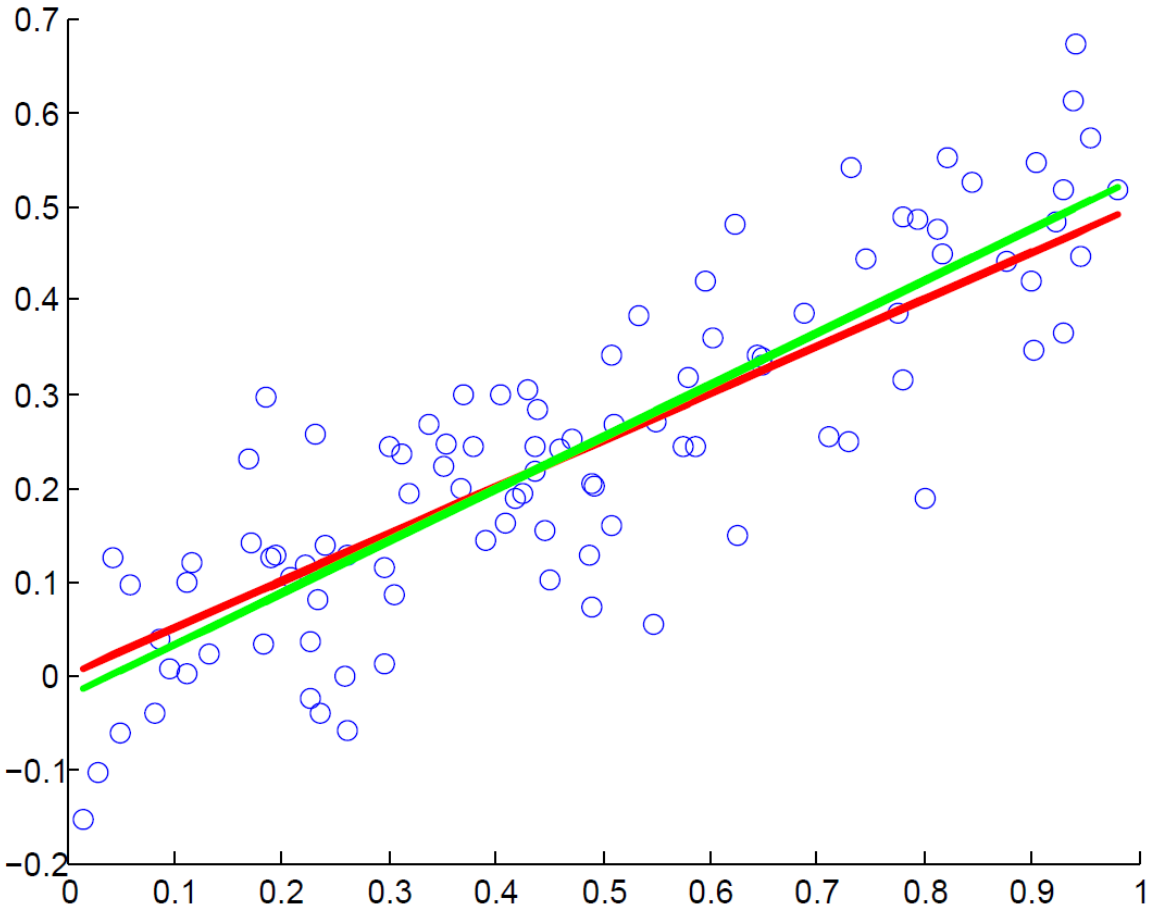
Inverting the matrix

$$\begin{bmatrix} \widehat{\beta}_0 \\ \widehat{\beta} \end{bmatrix} = \begin{bmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{bmatrix}$$

Setting $\bar{x} = \frac{1}{n}\sum_i x_i$ and $\bar{y} = \frac{1}{n}\sum_i y_i$, the solution to this system reduces to

$$\begin{bmatrix} \widehat{\beta}_0 \\ \widehat{\beta} \end{bmatrix} = \frac{1}{\sum_i x_i^2 - n\bar{x}^2} \begin{bmatrix} \bar{y}(\sum_i x_i^2) - \bar{x}\sum_i x_i y_i \\ \sum_i x_i y_i - n\bar{x}\bar{y} \end{bmatrix}$$

Example



General least squares

Suppose d is arbitrary. Set

$$\boldsymbol{\theta} = \begin{bmatrix} \beta_0 \\ \beta(1) \\ \vdots \\ \beta(d) \end{bmatrix}$$

Then $\mathrm{SSE}(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0)^2 = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\theta}\|^2$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} 1 & x_1(1) & \cdots & x_1(d) \\ 1 & x_2(1) & \cdots & x_2(d) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n(1) & \cdots & x_n(d) \end{bmatrix}$$

General least squares

The minimizer $\widehat{\theta}$ of this quadratic objective function is

$$\widehat{\theta} = \left(A^T A\right)^{-1} A^T y$$

provided that $A^T A$ is ***nonsingular***

"Proof"

$$\begin{aligned} \|y - A\theta\|^2 &= (y - A\theta)^T (y - A\theta) \\ &= y^T y - 2y^T A\theta + \theta^T A^T A\theta \end{aligned}$$

$$\frac{\partial}{\partial \theta}\|y - A\theta\|^2 = -2A^T y + 2A^T A\theta = 0$$

$$\widehat{\theta} = \left(A^T A\right)^{-1} A^T y$$

Nonlinear feature maps

Sometimes linear methods (in both regression and classification) just don't work

One way to create nonlinear estimators or classifiers is to first transform the data via a nonlinear feature map

$$\Phi : \mathbb{R}^d \to \mathbb{R}^{d'}$$

After applying Φ , we can then try applying a linear method to the transformed data $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)$

Regression

In the case of regression, our model becomes

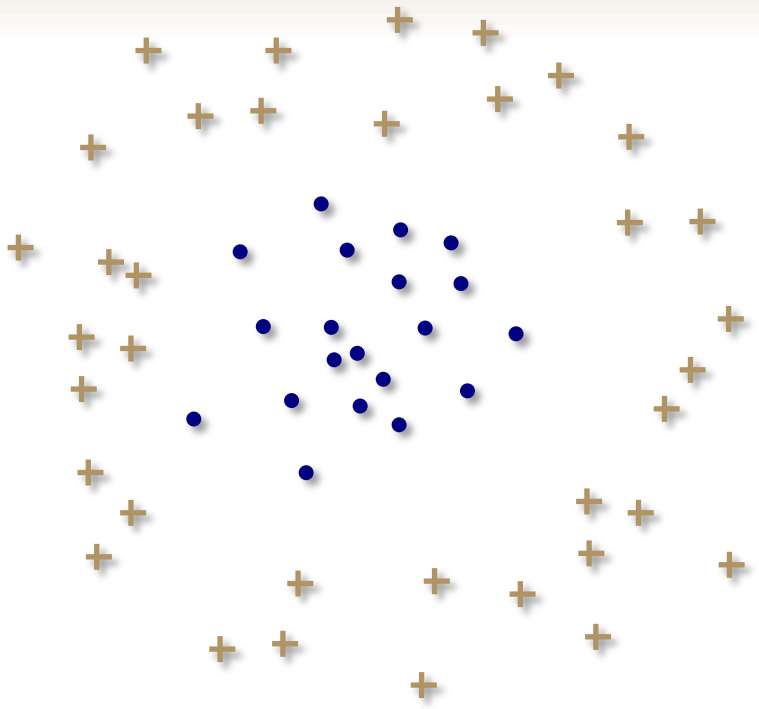
$$f(\mathbf{x}) = \boldsymbol{\beta}^T \Phi(\mathbf{x}) + \beta_0$$

where now $\boldsymbol{\beta} \in \mathbb{R}^{d'}$

Example. Suppose $d = 1$ but $f(x)$ is a cubic polynomial. How do we find a least squares estimate of f from training data?

$$\Phi_k(x) = x^k \quad \Rightarrow \quad \boldsymbol{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

Classification



This data set is not linearly separable

Consider the mapping

$$\Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x(1) \\ x(2) \\ x(1)x(2) \\ x(1)^2 \\ x(2)^2 \end{bmatrix}$$

Is the dataset linearly separable after applying this feature map?

Lecture 6: Theory of Generalization

The Bayes classifier

Consider (X, Y) where

- X is a random vector in $\mathbb{R}^d$
- $Y \in \{0, \ldots, K-1\}$is a random variable (depending on X)
- Define $\eta_k(\mathbf{x}) := \mathbb{P}[Y = k | X = \mathbf{x}]$

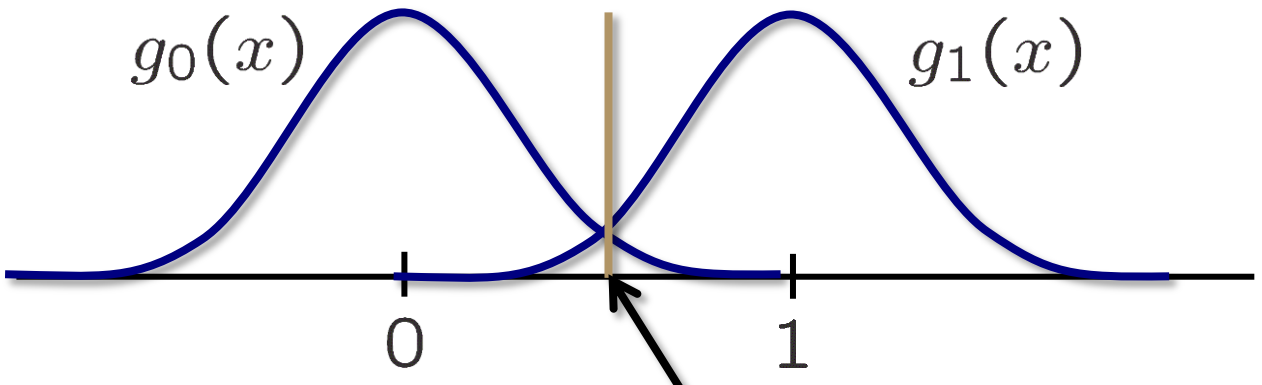
Bayes classifier

$$f^*(\mathbf{x}) := \arg\max_k \eta_k(\mathbf{x})$$

Example

Suppose that $K = 2$ and that

$$X|Y = 0 \sim \mathcal{N}(0, 1) \qquad X|Y = 1 \sim \mathcal{N}(1, 1)$$



$$\frac{g_1(x)}{g_0(x)} \underset{1}{\overset{0}{\lessgtr}} \frac{\pi_0}{\pi_1}$$

If $\pi_0 = \pi_1$

Example

How do we calculate the ***Bayes risk***?

$$\begin{aligned} R(f^*) &= \mathbb{P}\left[f^*(X) \neq Y\right] \\ &= \mathbb{P}\left[\text{declare } 0 | Y = 1\right] \cdot \pi_1 \\ &\quad + \mathbb{P}\left[\text{declare } 1 | Y = 0\right] \cdot \pi_0 \end{aligned}$$

In the case where $\pi_0 = \pi_1 = \frac{1}{2}$, our test reduced to declaring 1 iff $x \geq \frac{1}{2}$, thus

$$\begin{aligned} R(f^*) &= \tfrac{1}{2}\mathbb{P}\left[X < \tfrac{1}{2} | Y = 1\right] + \tfrac{1}{2}\mathbb{P}\left[X > \tfrac{1}{2} | Y = 0\right] \\ &= \frac{1}{2}\int_{-\infty}^{\frac{1}{2}} g_1(t)\, dt + \frac{1}{2}\int_{\frac{1}{2}}^{\infty} g_0(t)\, dt \\ &= \Phi\left(-\tfrac{1}{2}\right) \end{aligned}$$

Optimal separating hyperplanes

$$(\mathbf{w}^*, b^*) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \text{ for } i = 1, \ldots, n$$

$$(\mathbf{w}^*, b^*) = \arg\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, n$$

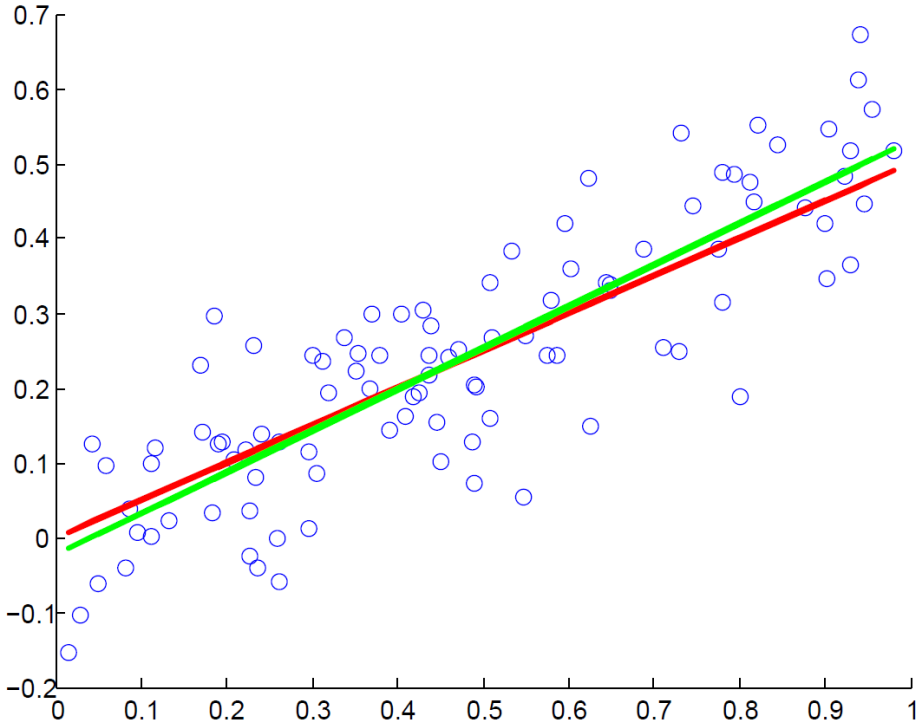
$$\xi_i \geq 0 \quad i = 1, \ldots, n$$

Linear regression

In ***linear regression***, we assume that f is an ***affine*** function, i.e.,

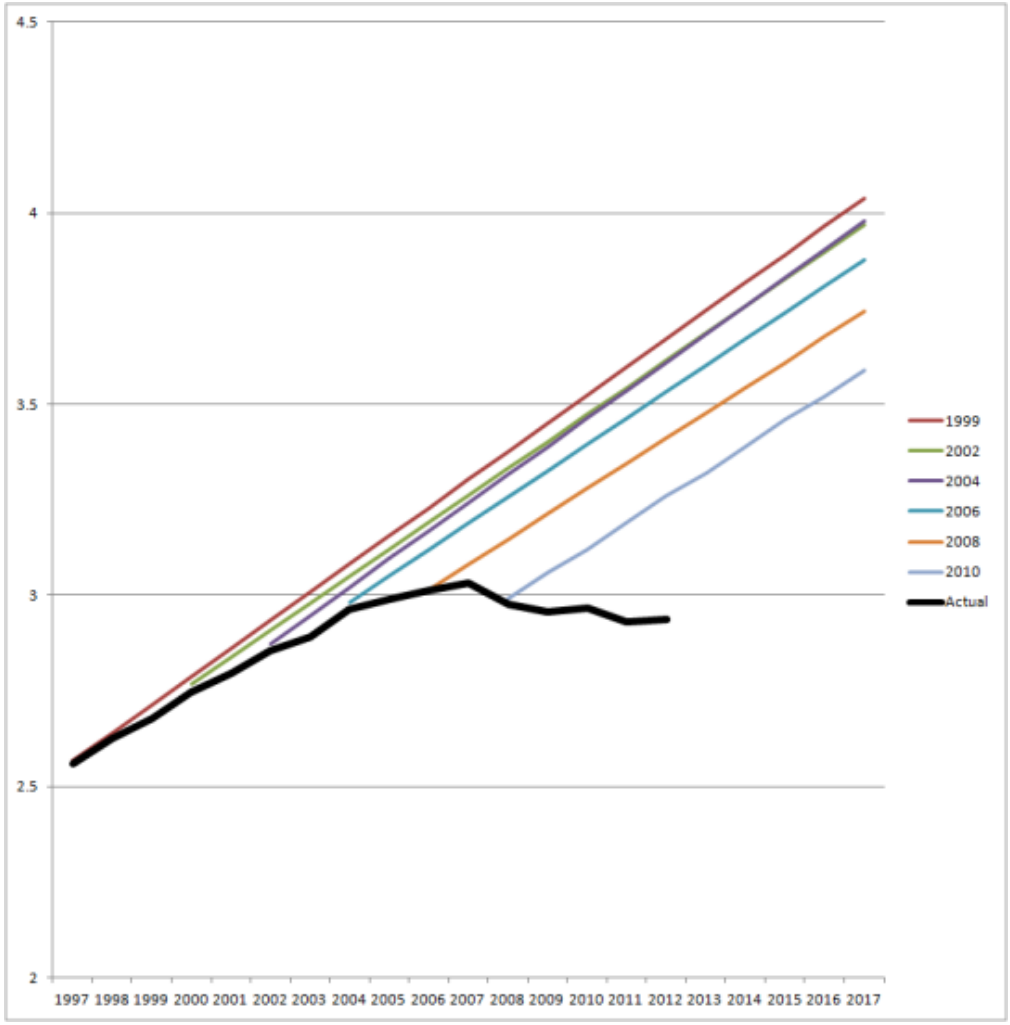
$$f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x} + \beta_0$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$, $\beta \in \mathbb{R}$



Does *linear* regression always make sense?

Official US DOT forecasts of road traffic, compared to actual



Nonlinear feature maps

Sometimes linear methods (in both regression and classification) just don't work

One way to create nonlinear estimators or classifiers is to first transform the data via a nonlinear feature map

$$\Phi : \mathbb{R}^d \to \mathbb{R}^{d'}$$

After applying Φ, we can then try applying a linear method to the transformed data $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)$

If d' is big enough, linear methods can work arbitrarily well ***on the training data***

How can we choose Φ (and d') to ensure that linear methods will generalize beyond the training data?

Training versus testing

Today we will take a ***theoretical*** look at generalization

To keep life (much) simpler, we will restrict our attention to binary (two class) classification, but an analogous theory can be developed for regression

- Testing:

$$\mathbb{P}\left[\left|\widehat{R}_n(h) - R(h)\right| > \epsilon\right] \leq 2e^{-2\epsilon^2 n}$$

- Training:

$$\mathbb{P}\left[\left|\widehat{R}_n(h^*) - R(h^*)\right| > \epsilon\right] \leq 2me^{-2\epsilon^2 n}$$

$$h^* \in \mathcal{H} \qquad m = |\mathcal{H}|$$

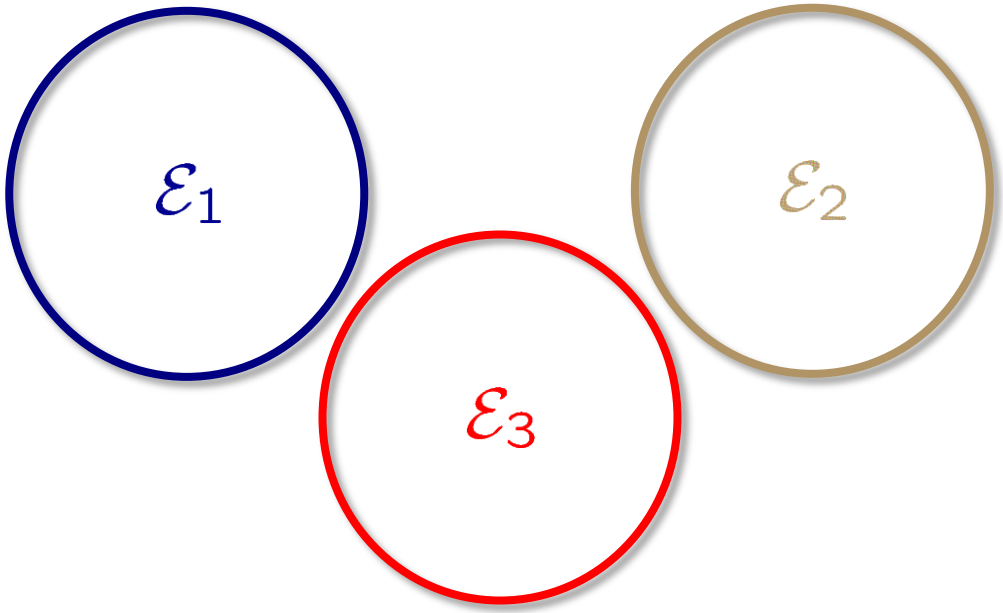
Where did m come from?

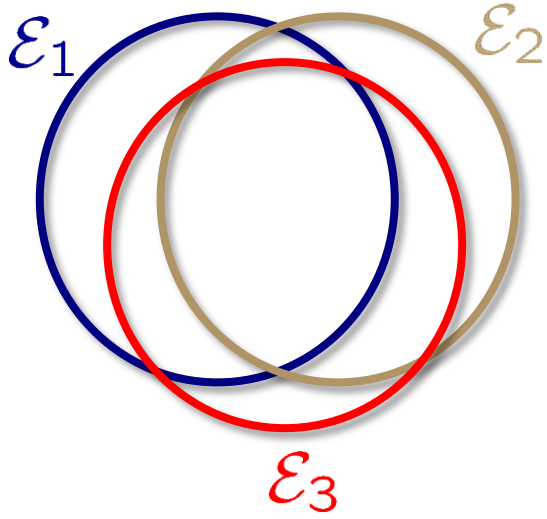
Union bound For any sequence of events $\mathcal{E}_1, \ldots, \mathcal{E}_m$

$$\mathbb{P}\left[\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_m\right] \leq \mathbb{P}\left[\mathcal{E}_1\right] + \cdots + \mathbb{P}\left[\mathcal{E}_m\right]$$

Our "bad events" were given by

$$\mathcal{E}_j = \left|\widehat{R}_n(h_j) - R(h_j)\right| > \epsilon$$



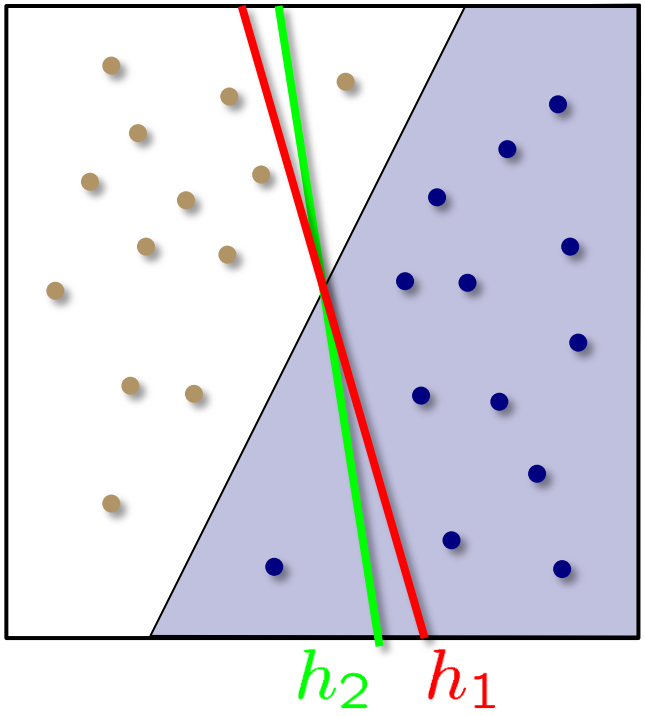


Can we improve on m?

Yes. There is tremendous overlap between our "bad events"

$$R(h_1) \approx R(h_2)$$

$$\widehat{R}_n(h_1) \approx \widehat{R}_n(h_2)$$

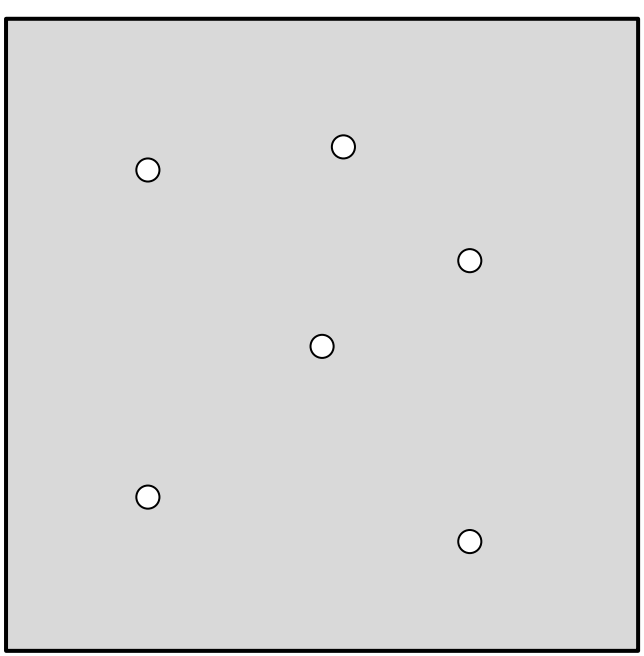


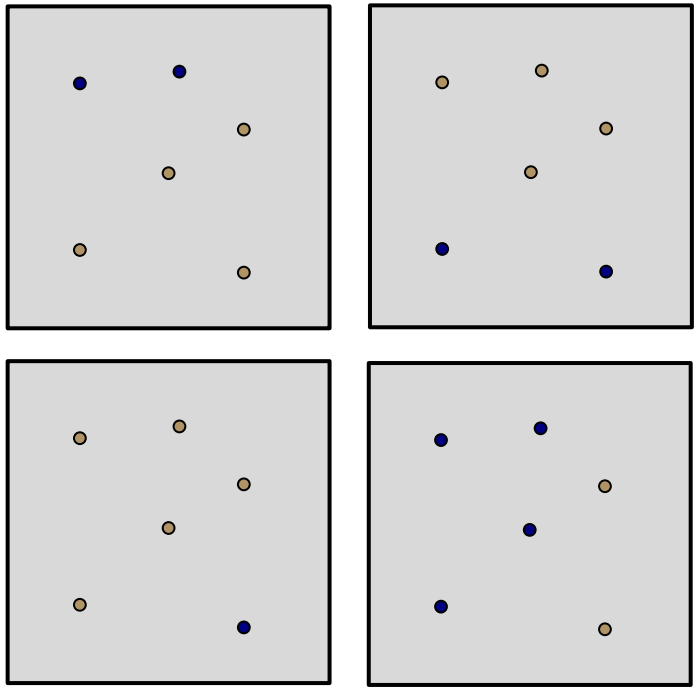
$$|\widehat{R}_n(h_1) - R(h_1)| \approx |\widehat{R}_n(h_2) - R(h_2)|$$

If not m, what?

Instead of considering all possible hypotheses in $\mathcal{H}$
we will consider a finite set of input points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
and "combine" hypotheses that result in the same labeling

We will call a particular labeling of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ a ***dichotomy***





Hypotheses vs dichotomies

Hypotheses

- $h : \mathcal{X} \to \{-1, +1\}$
- Number of hypotheses $|\mathcal{H}|$ can be infinite

$|\mathcal{H}|$ (or m) is a poor way to measure "richness" of $\mathcal{H}$

Dichotomies

- $h : \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \to \{-1, +1\}$
- Number of dichotomies $|\mathcal{H}(\mathbf{x}_1, \ldots, \mathbf{x}_n)|$ is at most 2^n

Good candidate for replacing $|\mathcal{H}|$ as a measure of "richness"

The growth function

A dichotomy is defined in terms of a particular $\mathbf{x}_1, \ldots, \mathbf{x}_n$

We would like to be able to state results that hold no matter what $\mathbf{x}_1, \ldots, \mathbf{x}_n$ turn out to be

Define the ***growth function*** of $\mathcal{H}$ as

$$m_{\mathcal{H}}(n) := \max_{\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}} |\mathcal{H}(\mathbf{x}_1, \ldots, \mathbf{x}_n)|$$

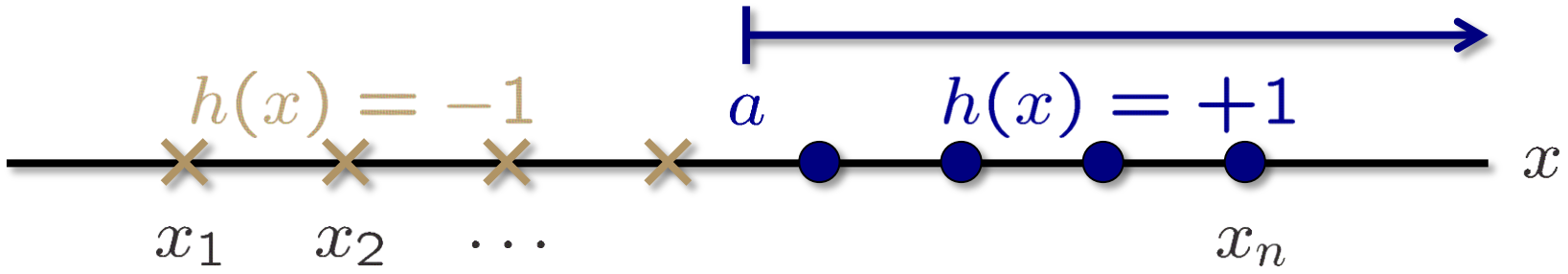
$m_{\mathcal{H}}(n)$ counts the ***most*** dichotomies that can possibly be generated on n points

It is easy to see that $m_{\mathcal{H}}(n) \leq 2^n$, but it can potentially be much smaller

Example 1: Positive rays

Candidate functions: $h : \mathbb{R} \to \{-1, +1\}$ such that

$h(x) = \text{sign}(x - a)$ for some $a \in \mathbb{R}$

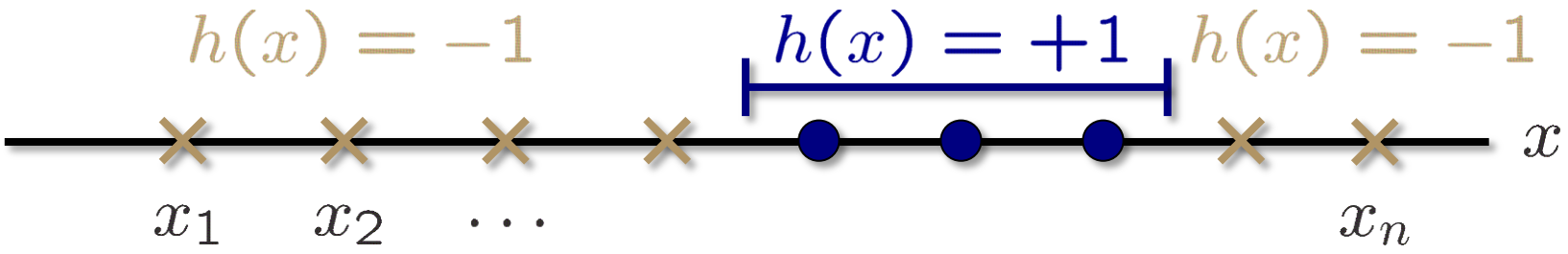


$$m_{\mathcal{H}}(n) = n + 1$$

Example 2: Positive intervals

Candidate functions: $h : \mathbb{R} \to \{-1, +1\}$ such that

$$h(x) = \begin{cases} +1 & \text{for } x \in [a, b] \\ -1 & \text{otherwise} \end{cases}$$

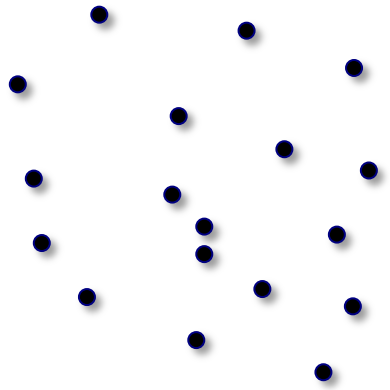


$$m_{\mathcal{H}}(n) = \binom{N+1}{2} + 1$$

$$= \tfrac{1}{2}N^2 + \tfrac{1}{2}N + 1$$

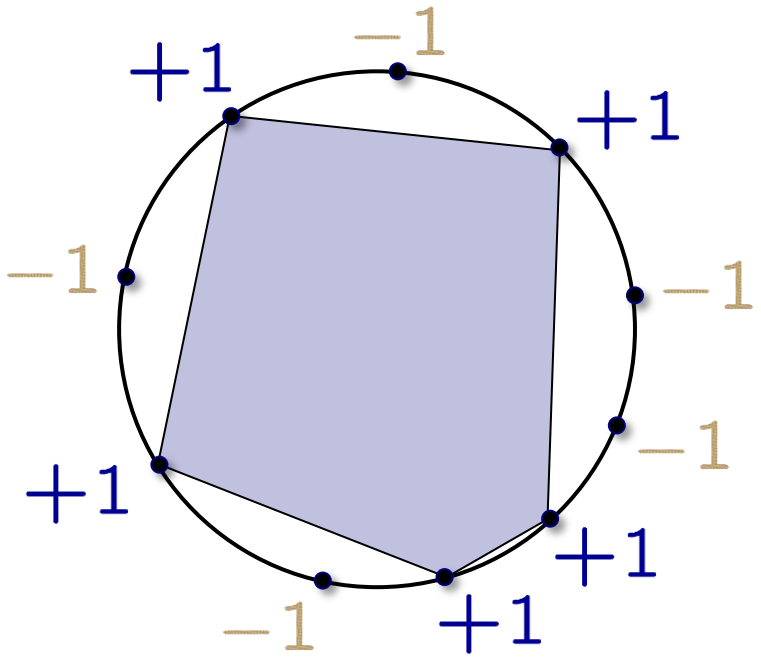
Example 3: Convex sets

Candidate functions: $h : \mathbb{R}^2 \to \{-1, +1\}$ such that $\{\mathbf{x} : h(\mathbf{x}) = +1\}$ is convex



Example 3: Convex sets

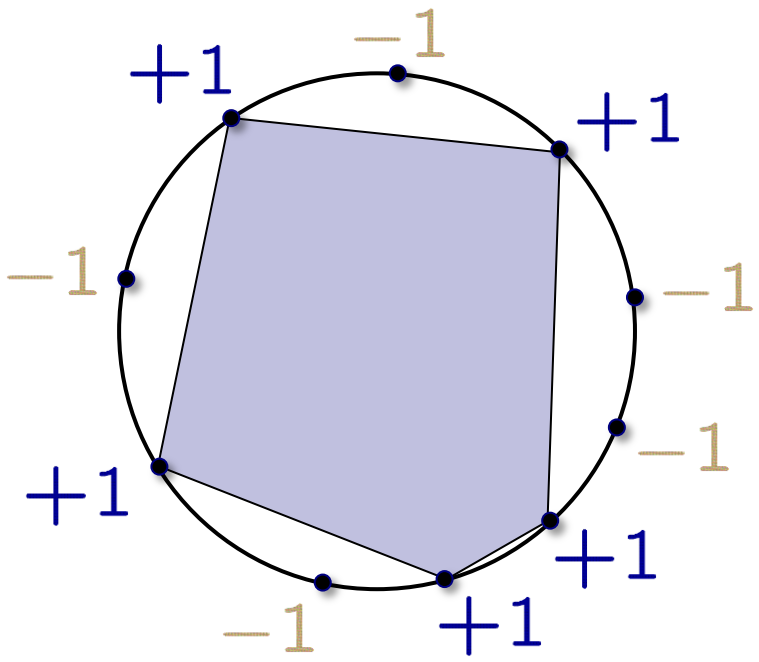
Candidate functions: $h : \mathbb{R}^2 \to \{-1, +1\}$ such that $\{\mathbf{x} : h(\mathbf{x}) = +1\}$ is convex



$$m_{\mathcal{H}}(n) = 2^n$$

Example 3: Convex sets

Candidate functions: $h : \mathbb{R}^2 \to \{-1, +1\}$ such that $\{\mathbf{x} : h(\mathbf{x}) = +1\}$ is convex

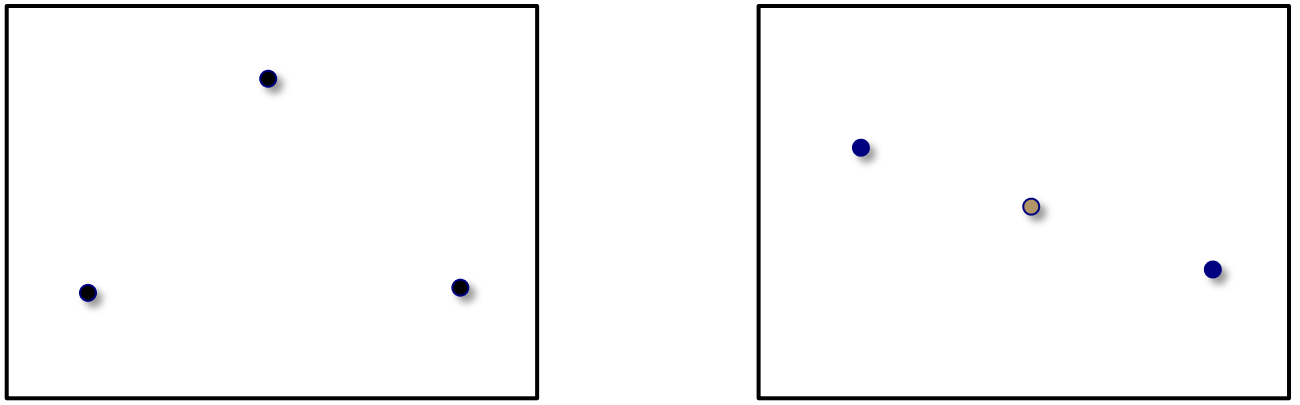


$$m_{\mathcal{H}}(n) = 2^n$$

If $\mathcal{H}$ can generate all possible dichotomies on $\mathbf{x}_1, \ldots, \mathbf{x}_n$, then we say that $\mathcal{H}$ ***shatters*** $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Example 4: Linear classifiers

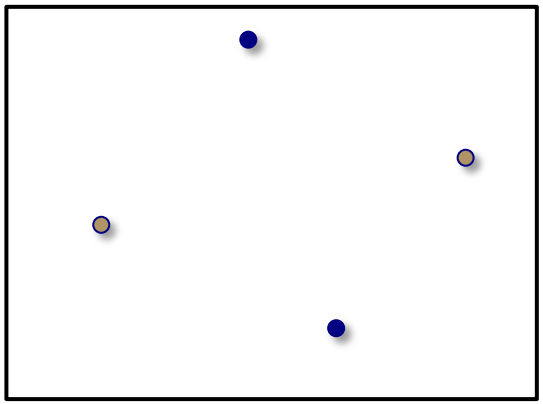
Candidate functions: $h : \mathbb{R}^2 \to \{-1, +1\}$ such that $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$ for some $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$

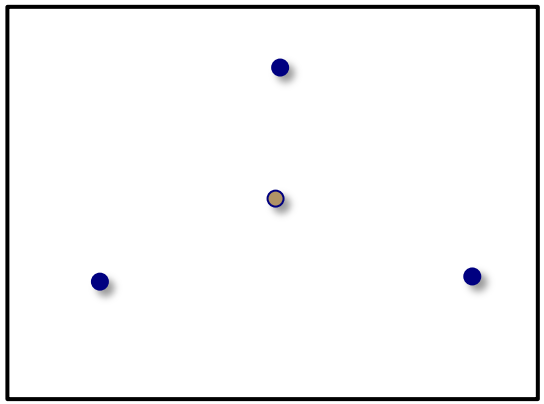


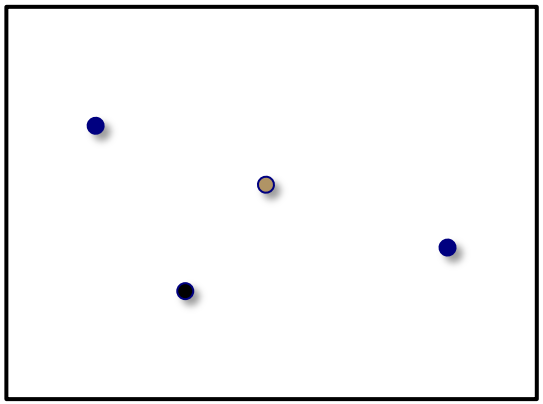
$$m_{\mathcal{H}}(3) = 2^3$$

Example 4: Linear classifiers

Candidate functions: $h : \mathbb{R}^2 \to \{-1, +1\}$ such that $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$ for some $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$







$$m_{\mathcal{H}}(4) = 14$$

Recap: Example growth functions

- Positive rays: $m_{\mathcal{H}}(n) = n + 1$
- Positive intervals: $m_{\mathcal{H}}(n) = \frac{1}{2}n^2 + \frac{1}{2}n + 1$
- Convex sets: $m_{\mathcal{H}}(n) = 2^n$
- Linear classifiers in $\mathbb{R}^2$: $m_{\mathcal{H}}(1) = 2$

$$m_{\mathcal{H}}(2) = 4$$
$$m_{\mathcal{H}}(3) = 8$$
$$m_{\mathcal{H}}(4) = 14$$
$$m_{\mathcal{H}}(n) = ?$$

Back to the big picture

Recall

$$\mathbb{P}\left[\left|\widehat{R}_n(h^*) - R(h^*)\right| > \epsilon\right] \leq 2me^{-2\epsilon^2 n}$$

Another way to express this is that if you pick a δ, then we can guarantee that with probability at least $1 - \delta$

$$R(h^*) \leq \widehat{R}_n(h^*) + \sqrt{\frac{1}{2n}\log\frac{2m}{\delta}}$$

What happens if we replace m with $m_{\mathcal{H}}(n)$?

- If $m_{\mathcal{H}}(n) = 2^n$, $\sqrt{\frac{1}{2n}\log\frac{2m_{\mathcal{H}}(n)}{\delta}}$ is a constant
- If $m_{\mathcal{H}}(n)$ is a polynomial in n, $\sqrt{\frac{1}{2n}\log\frac{2m_{\mathcal{H}}(n)}{\delta}}$ decays like $\sqrt{\frac{\log n}{n}}$

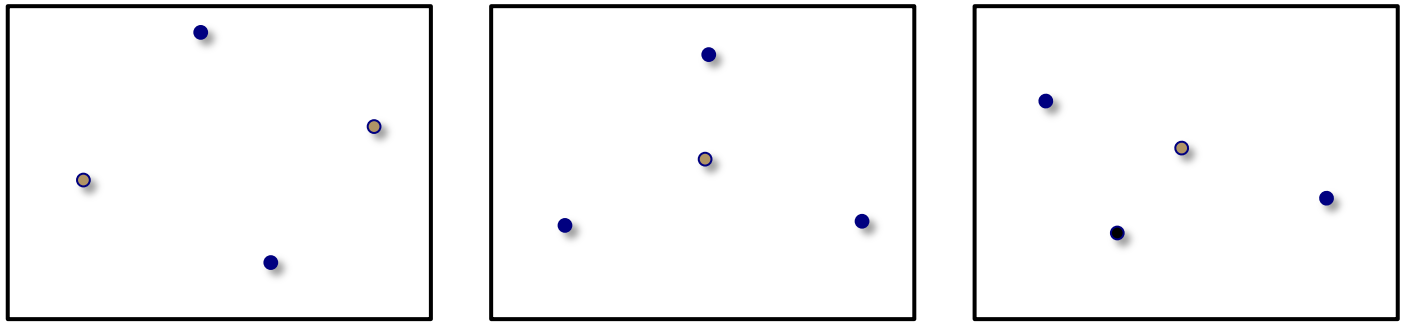
When is learning possible?

Assuming that we will indeed be allowed to substitute $m_{\mathcal{H}}(n)$ for m, we can argue that for a given set of hypotheses $\mathcal{H}$, learning is possible provided that $m_{\mathcal{H}}(n)$ is a polynomial

Key idea: *Break points*

If no data set of size k can be shattered by $\mathcal{H}$, then k is a ***break point*** for $\mathcal{H}$

$$m_{\mathcal{H}}(k) < 2^k$$



If k is a break point, then so is any $k' > k$

Examples

- Positive rays: $m_{\mathcal{H}}(n) = n + 1$
  - break point: $k = 2$

- Positive intervals: $m_{\mathcal{H}}(n) = \frac{1}{2}n^2 + \frac{1}{2}n + 1$
  - break point: $k = 3$

- Convex sets: $m_{\mathcal{H}}(n) = 2^n$
  - break point: $k = \infty$
- Linear classifiers in $\mathbb{R}^2$: $m_{\mathcal{H}}(3) = 8$

  $m_{\mathcal{H}}(4) = 14$
  - break point: $k = 4$

So what?

If there exists any break point, then $m_{\mathcal{H}}(n)$ is polynomial in n

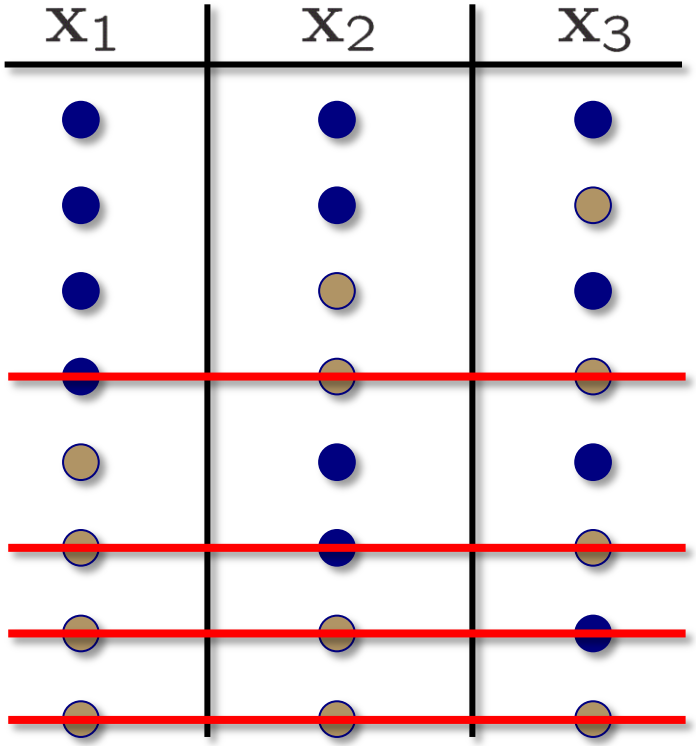
Also, if there are no break points, then $m_{\mathcal{H}}(n) = 2^n$

As soon as we have a single break point, this starts eliminating tons of dichotomies

Puzzle

You are given a hypothesis set which has a break point of 2

How many dichotomies can you get on 3 data points?



Bounding the growth function

We want to show that $m_{\mathcal{H}}(n)$ is polynomial in n

We will show that $m_{\mathcal{H}}(n) \leq$ ***some*** polynomial

Our approach will center around

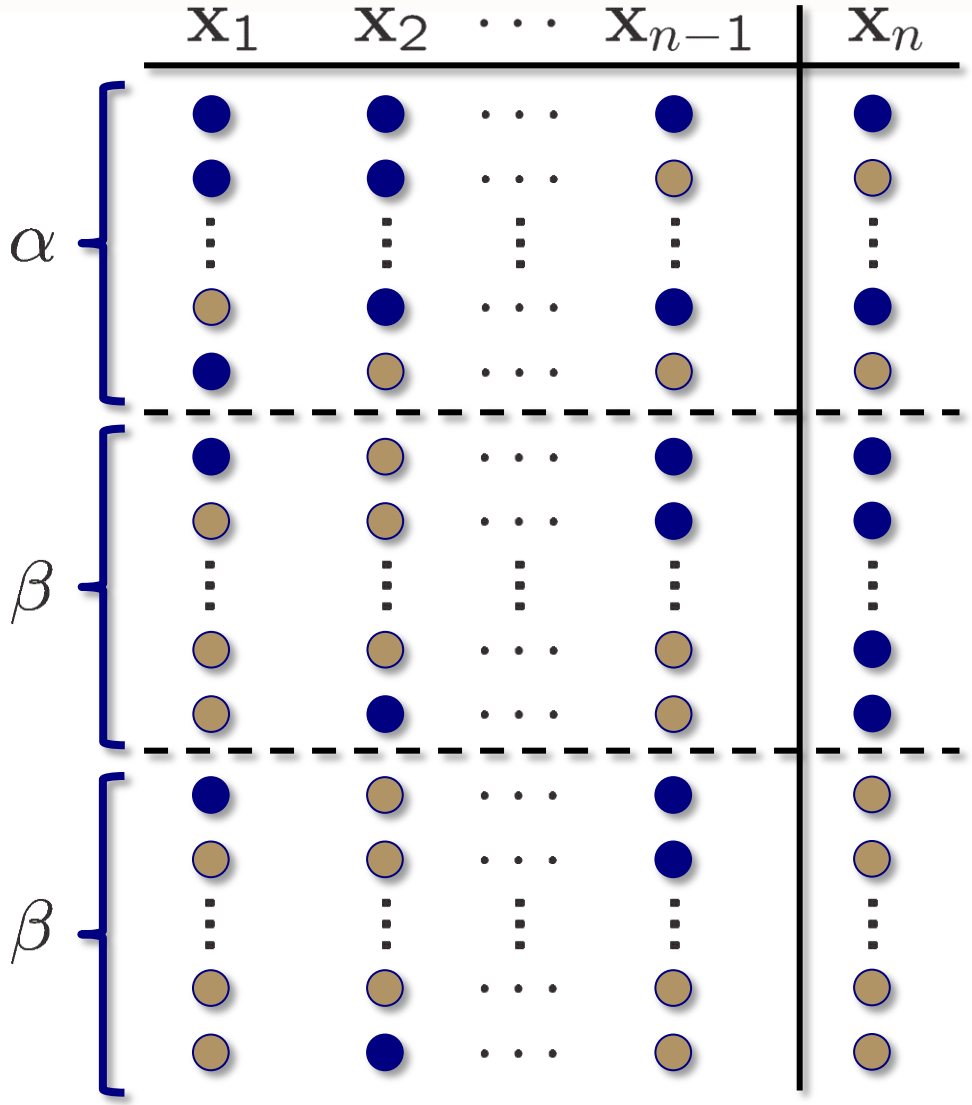
$$B(n,k) := \text{maximum number of dichotomies on } n \text{ points, given a break point of } k$$

$B(n,k)$ is a purely combinatorial quantity

By definition, $m_{\mathcal{H}}(n) \leq B(n,k)$

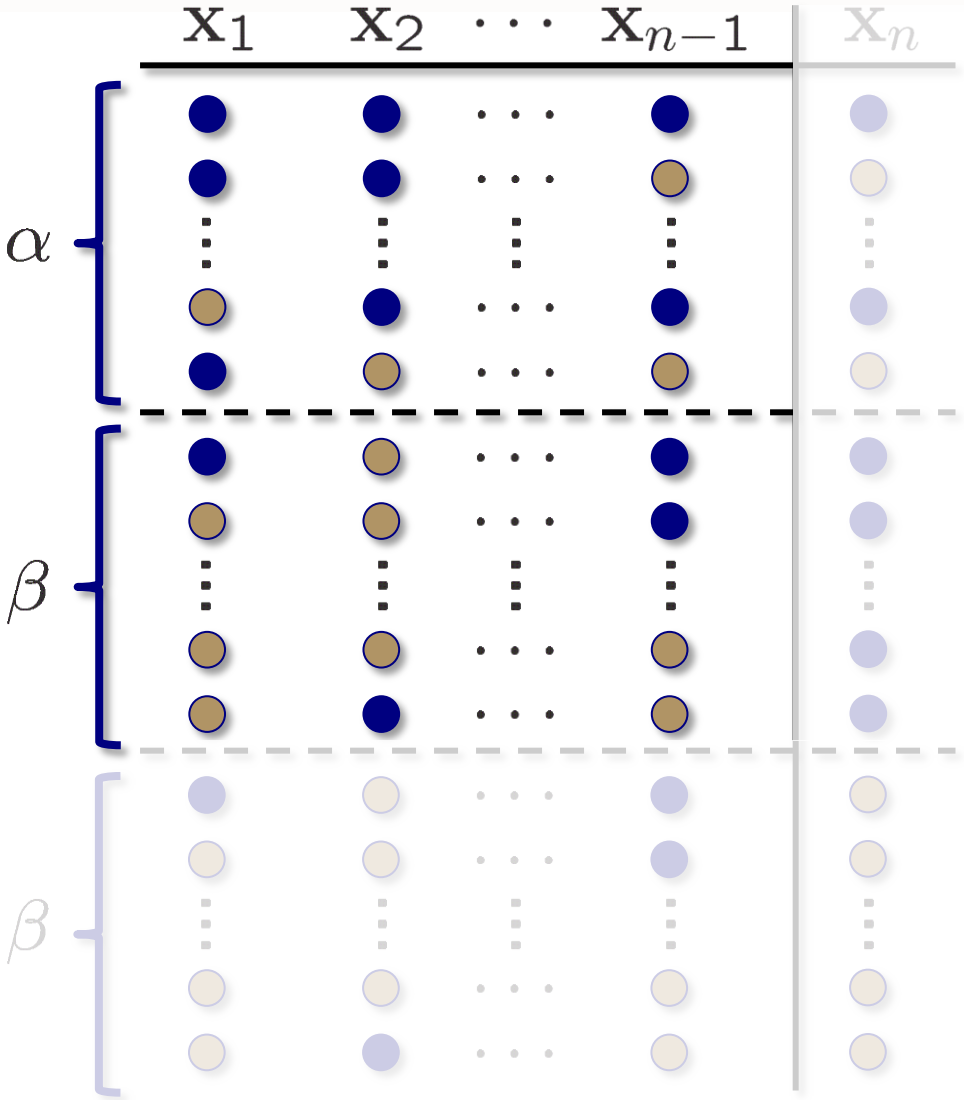
Bounding $B(n,k)$ recursively

$$B(n,k) = \alpha + 2\beta$$



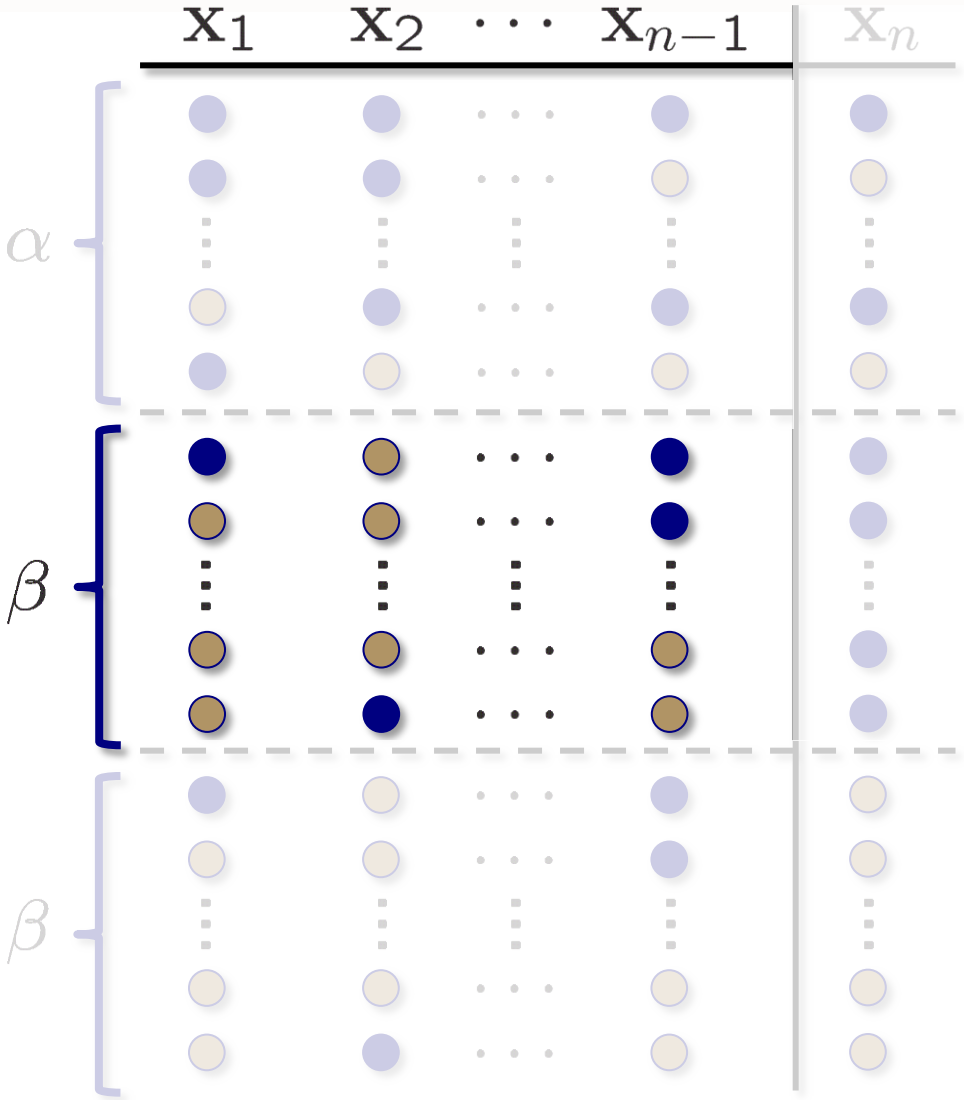
Estimating α and β

$$\alpha + \beta \leq B(n-1, k)$$



Estimating β by itself

$$\beta \leq B(n-1, k-1)$$



Putting it all together

$$B(n,k) = \alpha + 2\beta$$

$$\alpha + \beta \leq B(n-1,k)$$

$$\beta \leq B(n-1,k-1)$$

$$B(n,k) \leq B(n-1,k) + B(n-1,k-1)$$

Lecture 7: VC-bound

Last time: Generalization bound

Recall

$$\mathbb{P}\left[\left|\widehat{R}_n(h^*) - R(h^*)\right| > \epsilon\right] \leq 2me^{-2\epsilon^2 n}$$

Another way to express this is that if you pick a δ, then we can guarantee that with probability at least $1-\delta$

$$R(h^*) \leq \widehat{R}_n(h^*) + \sqrt{\frac{1}{2n}\log\frac{2m}{\delta}}$$

We would like to argue that we can replace m with the ***growth function***

$$m_{\mathcal{H}}(n) := \max_{\mathbf{x}_1,\ldots,\mathbf{x}_n \in \mathcal{X}} |\mathcal{H}(\mathbf{x}_1,\ldots,\mathbf{x}_n)|$$

which we hope will be better behaved for large/infinite $\mathcal{H}$

Example growth functions

- Positive rays: $m_{\mathcal{H}}(n) = n + 1$
- Positive intervals: $m_{\mathcal{H}}(n) = \frac{1}{2}n^2 + \frac{1}{2}n + 1$
- Convex sets: $m_{\mathcal{H}}(n) = 2^n$
- Linear classifiers in $\mathbb{R}^2$: $m_{\mathcal{H}}(1) = 2$

$$m_{\mathcal{H}}(2) = 4$$
$$m_{\mathcal{H}}(3) = 8$$
$$m_{\mathcal{H}}(4) = 14$$
$$m_{\mathcal{H}}(n) = ?$$

Break points

If no data set of size k can be shattered by $\mathcal{H}$, i.e., if $m_{\mathcal{H}}(k) < 2^k$ then k is a ***break point*** for $\mathcal{H}$

If there exists any break point, then $m_{\mathcal{H}}(n)$ is polynomial in n

Examples

- Positive rays: $m_{\mathcal{H}}(n) = n + 1$
  - break point: $k = 2$

- Positive intervals: $m_{\mathcal{H}}(n) = \frac{1}{2}n^2 + \frac{1}{2}n + 1$
  - break point: $k = 3$

- Convex sets: $m_{\mathcal{H}}(n) = 2^n$
  - break point: $k = \infty$
- Linear classifiers in $\mathbb{R}^2$: $m_{\mathcal{H}}(3) = 8$

  $m_{\mathcal{H}}(4) = 14$
  - break point: $k = 4$

Bounding $m_{\mathcal{H}}(n)$ via a break point

We want to show that $m_{\mathcal{H}}(n)$ is polynomial in n

We will show that $m_{\mathcal{H}}(n) \leq$ ***some*** polynomial

Our approach will center around

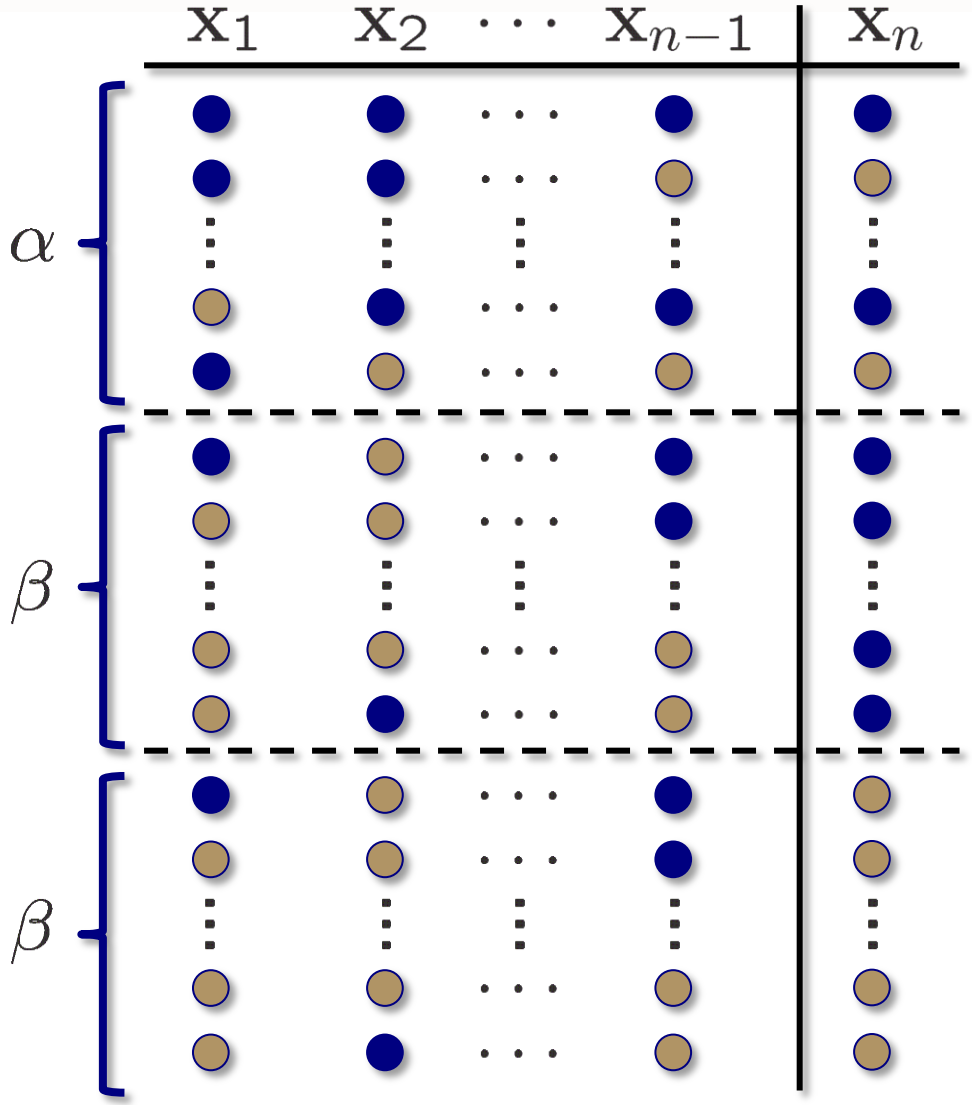
$$B(n,k) := \text{maximum number of dichotomies on } n \text{ points, given a break point of } k$$

$B(n,k)$ is a purely combinatorial quantity

By definition, $m_{\mathcal{H}}(n) \leq B(n,k)$

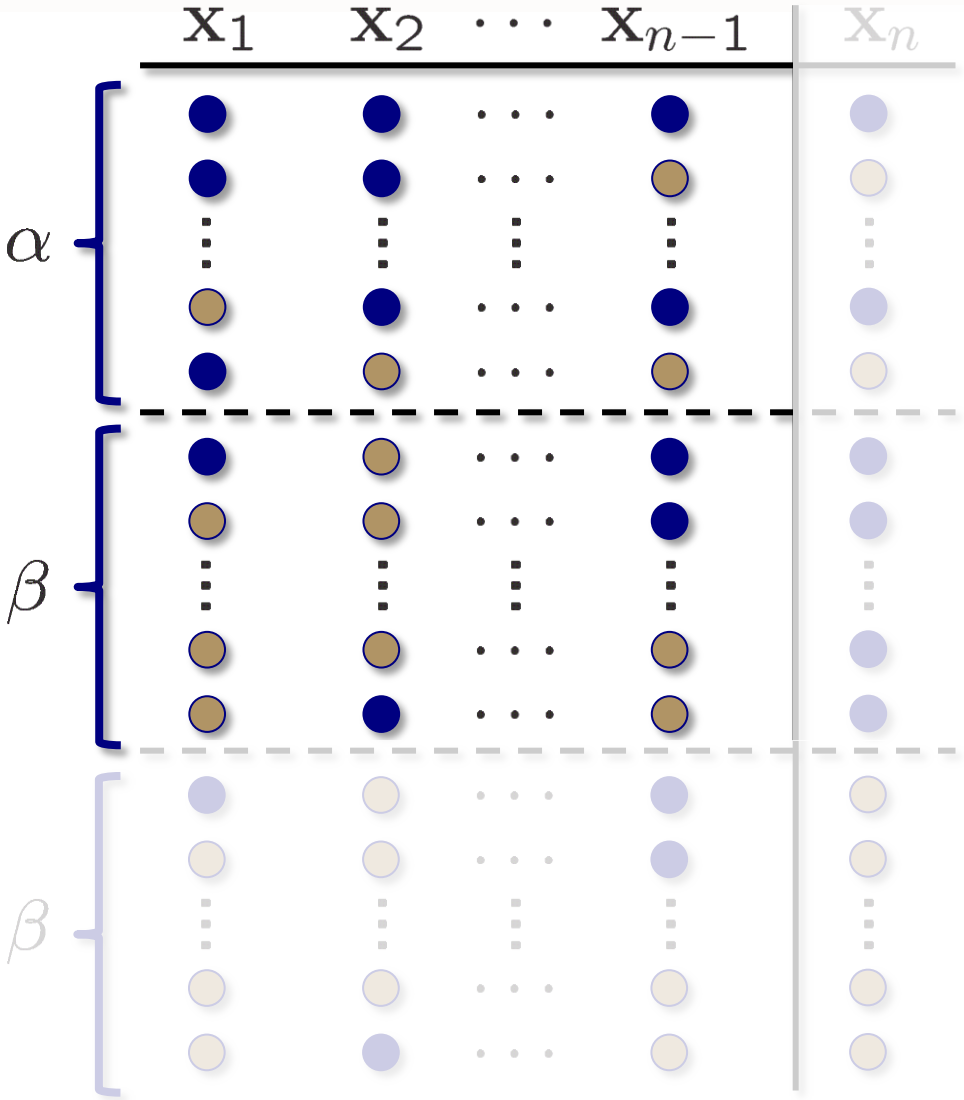
Bounding $B(n,k)$ recursively

$$B(n,k) = \alpha + 2\beta$$



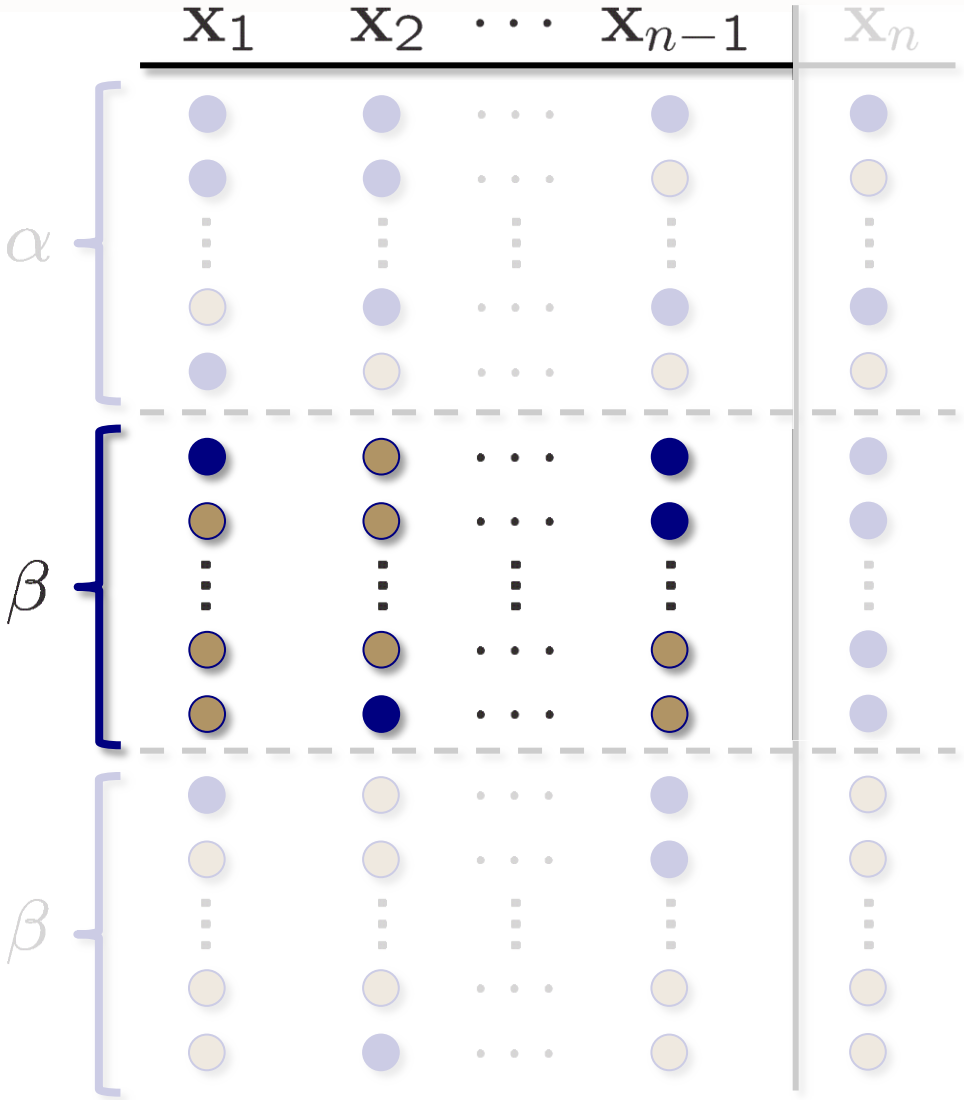
Estimating α and β

$$\alpha + \beta \leq B(n-1, k)$$



Estimating β by itself

$$\beta \leq B(n-1, k-1)$$



Putting it all together

$$B(n,k) = \alpha + 2\beta$$

$$\alpha + \beta \leq B(n-1,k)$$

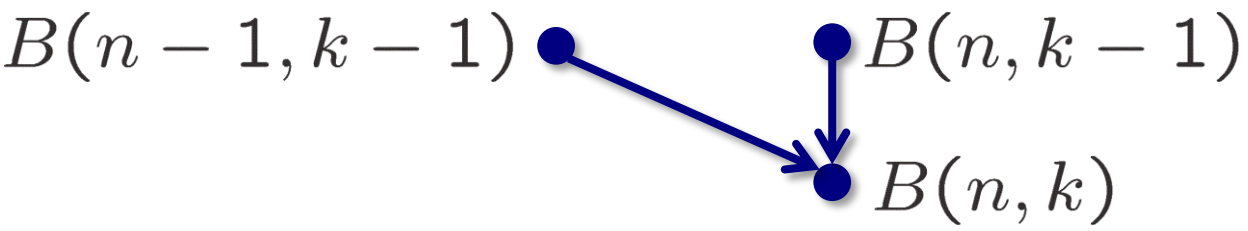
$$\beta \leq B(n-1,k-1)$$

$$B(n,k) \leq B(n-1,k) + B(n-1,k-1)$$

Computing our bound on $B(n,k)$

$$B(n,k) \leq B(n-1,k) + B(n-1,k-1)$$

n \ k	1	2	3	4	5	6	$\cdots$
1	1	2	2	2	2	2	$\cdots$
2	1	3	4	4	4	4	$\cdots$
3	1	4	7	8	8	8	$\cdots$
4	1	5	11	15	16	16	$\cdots$
5	1	6	$\cdots$				
6	1	7	$\cdots$				
$\vdots$	$\vdots$	$\vdots$					



Analytic solution

Theorem

$$B(n,k) \leq \sum_{i=0}^{k-1} \binom{n}{i}$$

In fact, it is actually true that

$$B(n,k) = \sum_{i=0}^{k-1} \binom{n}{i}$$

but all we really need is the upper bound, so that is all we will prove here

Proof

$$B(n,k) \leq \sum_{i=0}^{k-1} \binom{n}{i}$$

Proof by induction

Base case

- $B(n,1) = 1$
- $B(1,k) = \begin{cases} 1 & \text{if } k = 1 \\ 2 & \text{otherwise} \end{cases}$

Proof

Now suppose the inequality is true for $B(n-1,k)$ and $B(n-1,k-1)$

$$B(n,k) \leq \sum_{i=0}^{k-1} \binom{n-1}{i} + \sum_{i=0}^{k-2} \binom{n-1}{i}$$

$$= 1 + \sum_{i=1}^{k-1} \binom{n-1}{i} + \sum_{i=1}^{k-1} \binom{n-1}{i-1}$$

$$= 1 + \sum_{i=1}^{k-1} \left(\binom{n-1}{i} + \binom{n-1}{i-1} \right)$$

$$= 1 + \sum_{i=1}^{k-1} \binom{n}{i} = \sum_{i=0}^{k-1} \binom{n}{i}$$

Examples

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{k-1} \binom{n}{i}$$

- Positive rays: Break point of $k = 2$

$$m_{\mathcal{H}}(n) = n + 1 \leq n + 1$$

- Positive intervals: Break point of $k = 3$

$$m_{\mathcal{H}}(n) = \tfrac{1}{2}n^2 + \tfrac{1}{2}n + 1 \leq \tfrac{1}{2}n^2 + \tfrac{1}{2}n + 1$$

- Linear classifiers in $\mathbb{R}^2$: Break point of $k = 4$

$$m_{\mathcal{H}}(n) = \ ? \ \leq \tfrac{1}{6}n^3 + \tfrac{5}{6}n + 1$$

Bottom line

For a given $\mathcal{H}$, all we need is for a break point to exist

$$m_{\mathcal{H}}(n) \leq \underbrace{\sum_{i=0}^{k-1} \binom{n}{i}}$$

polynomial with dominant term n^{k-1}

All that remains is to argue that we can actually replace $|\mathcal{H}|$ with $m_{\mathcal{H}}(n)$ to obtain an inequality along the lines of

$$R(h^*) \leq \widehat{R}_n(h^*) + \sqrt{\frac{1}{2n} \log \frac{2m_{\mathcal{H}}(n)}{\delta}}$$

VC generalization bound

We won't be able to quite show

$$R(h^*) \leq \widehat{R}_n(h^*) + \sqrt{\frac{1}{2n} \log \frac{2m_{\mathcal{H}}(n)}{\delta}}$$

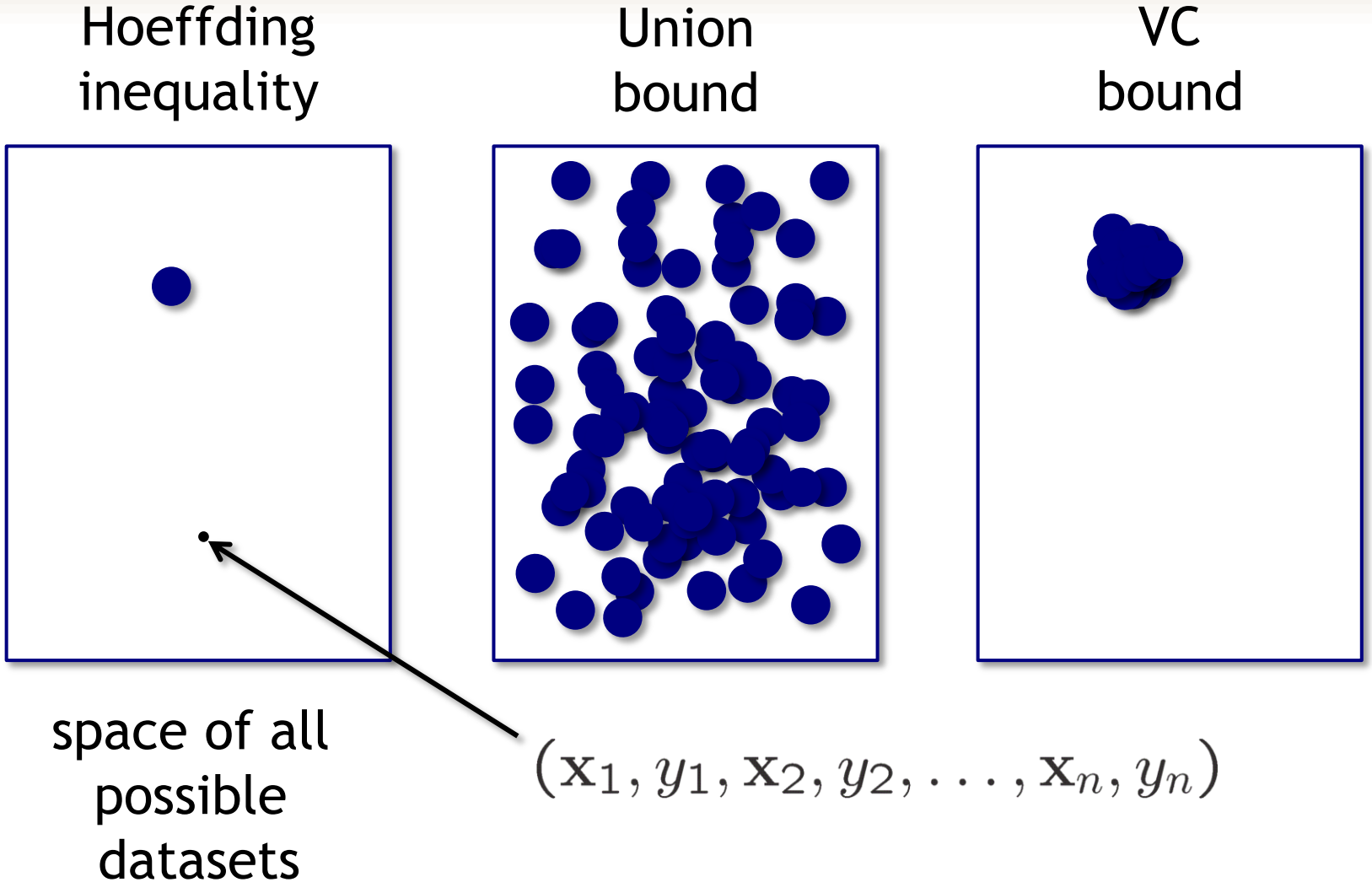
For technical reasons (which we will soon see), we will only be able to show that with probability $\geq 1 - \delta$

$$R(h^*) \leq \widehat{R}_n(h^*) + \sqrt{\frac{8}{n} \log \frac{4m_{\mathcal{H}}(2n)}{\delta}}$$

This is called the ***VC generalization bound***

Named after Vapnik and Chervonenkis, who proved it in 1971

VC bound intuition

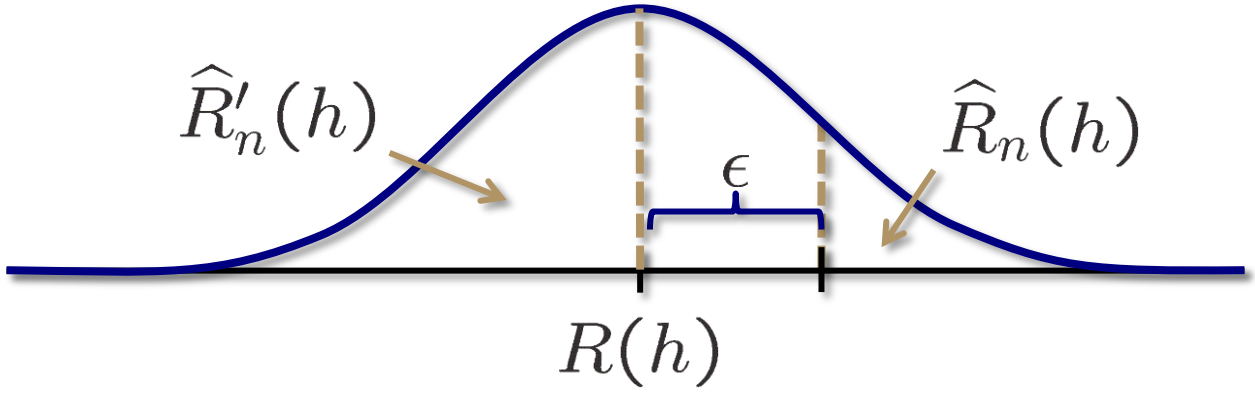


Proof sketch

We aim to get a bound on $\mathbb{P}[|\widehat{R}_n(h) - R(h)| > \epsilon]$ that holds for any $h \in \mathcal{H}$

Perhaps it is not surprising that we can understand $\widehat{R}_n(h)$ using the growth function, but what about $R(h)$?

Trick: Consider ***two*** datasets!



$$\mathbb{P}[|\widehat{R}_n(h) - R(h)| > \epsilon] \leq 2\mathbb{P}[|\widehat{R}_n(h) - \widehat{R}'_n(h)| > \epsilon]$$

Proof sketch

Lemma 1

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|\widehat{R}_n(h)-R(h)|>\epsilon\right]$$
$$\leq 2\,\mathbb{P}\left[\sup_{h\in\mathcal{H}}|\widehat{R}_n(h)-\widehat{R}'_n(h)|>\frac{\epsilon}{2}\right]$$

Lemma 2

sample of $2n$ observations

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|\widehat{R}_n(h)-\widehat{R}'_n(h)|>\frac{\epsilon}{2}\right]$$
$$\leq m_{\mathcal{H}}(2n)\cdot\sup_{S}\sup_{h\in\mathcal{H}}\mathbb{P}\left[|\widehat{R}_n(h)-\widehat{R}'_n(h)|>\frac{\epsilon}{2}\middle|S\right]$$

Proof sketch

Lemma 3

For ***any*** h and ***any*** S,

$$\mathbb{P}\left[|\widehat{R}_n(h) - \widehat{R}'_n(h)| > \frac{\epsilon}{2}\middle| S\right] \leq 2e^{-\frac{1}{8}\epsilon^2 n}$$

Putting all of this together, we get

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |\widehat{R}_n(h) - R(h)| > \epsilon\right] \leq 2 \cdot m_{\mathcal{H}}(2n) \cdot 2e^{-\frac{1}{8}\epsilon^2 n}$$

Thus, for any $h \in \mathcal{H}$, we have that with probability $\geq 1 - \delta$

$$R(h) \leq \widehat{R}_n(h) + \sqrt{\frac{8}{n}\log\frac{4m_{\mathcal{H}}(2n)}{\delta}}$$

Using the VC bound: The VC dimension

We went to a lot of trouble to show that if k is a break point for $\mathcal{H}$, then

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{k-1} \binom{n}{i} \leq n^{k-1} + 1$$

$$\Rightarrow R(h) \lessapprox \widehat{R}_n(h) + \sqrt{\frac{8(k-1)}{n} \log \frac{8n}{\delta}}$$

The ***VC dimension*** of a hypothesis set $\mathcal{H}$, denoted $d_{\mathsf{VC}}(\mathcal{H})$, is the largest n for which $m_{\mathcal{H}}(n) = 2^n$

- $d_{\mathsf{VC}}(\mathcal{H})$ is the most points that $\mathcal{H}$ can shatter
- $d_{\mathsf{VC}}(\mathcal{H})$ is 1 less than the smallest break point

$$\Rightarrow R(h) \lessapprox \widehat{R}_n(h) + \sqrt{\frac{8d_{\mathsf{VC}}}{n} \log \frac{8n}{\delta}}$$

Lecture 8: VC-Dimension

Recap

For a given $\mathcal{H}$, if we know that k is a ***break point***, meaning that no data set of size k can be ***shattered***, then we know

$$m_{\mathcal{H}}(n) \leq \underbrace{\sum_{i=0}^{k-1} \binom{n}{i}}$$

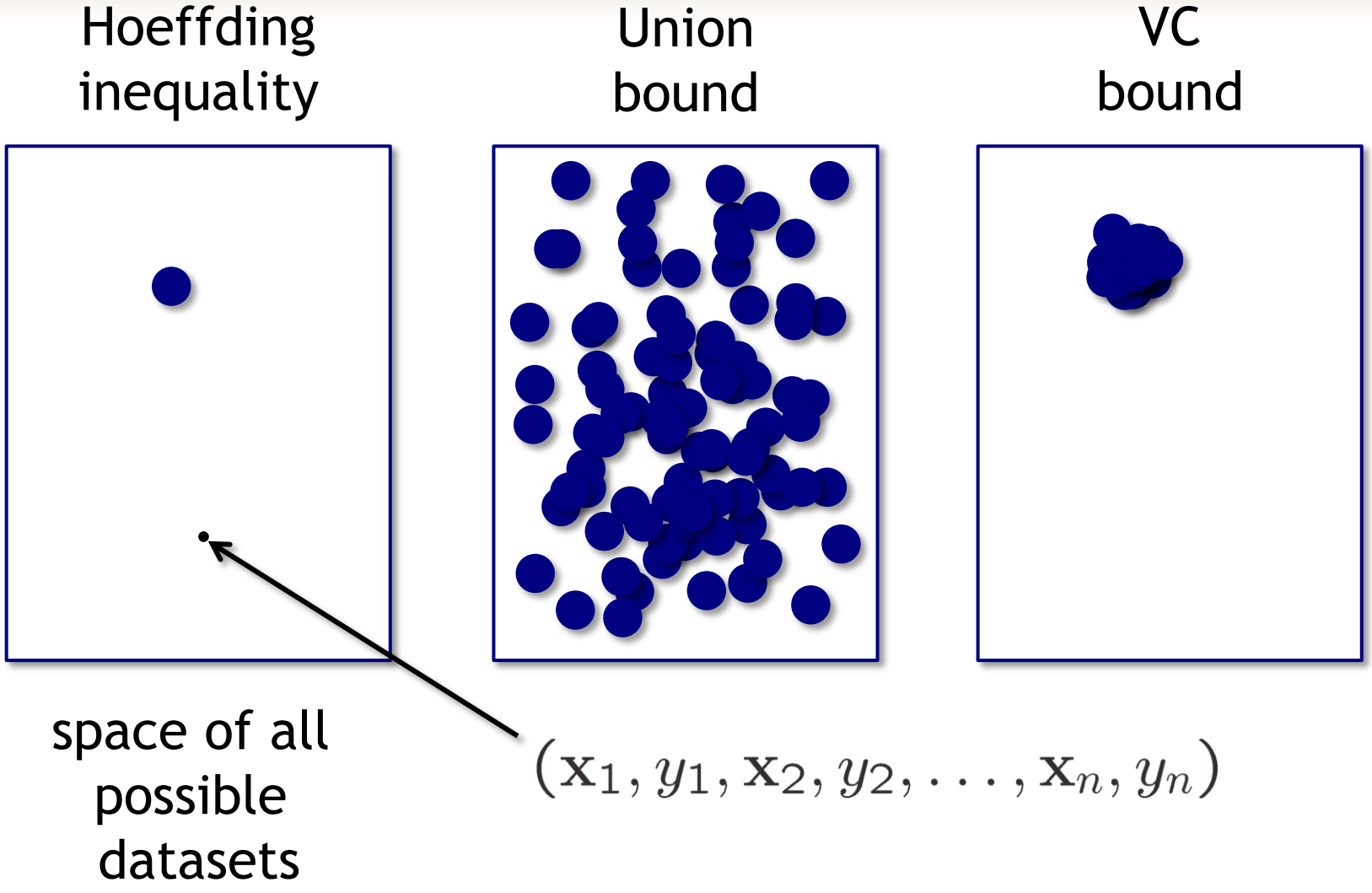
polynomial with dominant term n^{k-1}

VC generalization bound

For any $h \in \mathcal{H}$, with probability $\geq 1 - \delta$

$$R(h) \leq \widehat{R}_n(h) + \sqrt{\frac{8}{n} \log \frac{4m_{\mathcal{H}}(2n)}{\delta}}$$

VC bound intuition

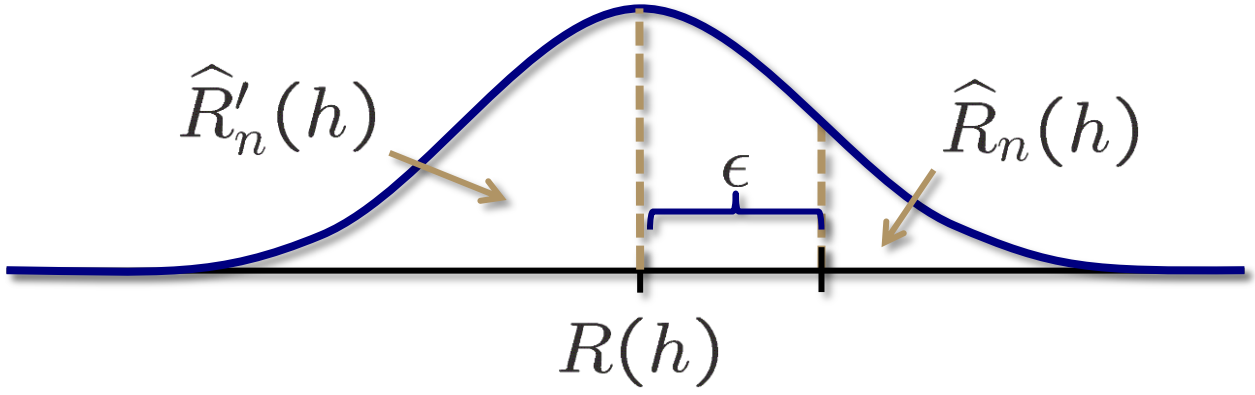


Proof sketch

We aim to get a bound on $\mathbb{P}[|\widehat{R}_n(h) - R(h)| > \epsilon]$ that holds for any $h \in \mathcal{H}$

Perhaps it is not surprising that we can understand $\widehat{R}_n(h)$ using the growth function, but what about $R(h)$?

Trick: Consider ***two*** datasets!



$$\mathbb{P}[|\widehat{R}_n(h) - R(h)| > \epsilon] \leq 2\mathbb{P}[|\widehat{R}_n(h) - \widehat{R}'_n(h)| > \epsilon]$$

Proof sketch

Lemma 1

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|\widehat{R}_n(h)-R(h)|>\epsilon\right]$$
$$\leq 2\,\mathbb{P}\left[\sup_{h\in\mathcal{H}}|\widehat{R}_n(h)-\widehat{R}'_n(h)|>\frac{\epsilon}{2}\right]$$

Lemma 2

sample of $2n$ observations

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|\widehat{R}_n(h)-\widehat{R}'_n(h)|>\frac{\epsilon}{2}\right]$$
$$\leq m_{\mathcal{H}}(2n)\cdot\sup_{S}\sup_{h\in\mathcal{H}}\mathbb{P}\left[|\widehat{R}_n(h)-\widehat{R}'_n(h)|>\frac{\epsilon}{2}\middle|S\right]$$

Proof sketch

Lemma 3

For ***any*** h and ***any*** S,

$$\mathbb{P}\left[|\widehat{R}_n(h) - \widehat{R}'_n(h)| > \frac{\epsilon}{2}\middle| S\right] \leq 2e^{-\frac{1}{8}\epsilon^2 n}$$

Putting all of this together, we get

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |\widehat{R}_n(h) - R(h)| > \epsilon\right] \leq 2 \cdot m_{\mathcal{H}}(2n) \cdot 2e^{-\frac{1}{8}\epsilon^2 n}$$

Thus, for any $h \in \mathcal{H}$, we have that with probability $\geq 1 - \delta$

$$R(h) \leq \widehat{R}_n(h) + \sqrt{\frac{8}{n}\log\frac{4m_{\mathcal{H}}(2n)}{\delta}}$$

Using the VC bound: The VC dimension

We went to a lot of trouble to show that if k is a break point for $\mathcal{H}$, then

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{k-1} \binom{n}{i} \leq n^{k-1} + 1$$

$$\Rightarrow R(h) \lessapprox \widehat{R}_n(h) + \sqrt{\frac{8(k-1)}{n} \log \frac{8n}{\delta}}$$

The ***VC dimension*** of a hypothesis set $\mathcal{H}$, denoted $d_{\mathsf{VC}}(\mathcal{H})$, is the largest n for which $m_{\mathcal{H}}(n) = 2^n$

- $d_{\mathsf{VC}}(\mathcal{H})$ is the most points that $\mathcal{H}$ can shatter
- $d_{\mathsf{VC}}(\mathcal{H})$ is 1 less than the smallest break point

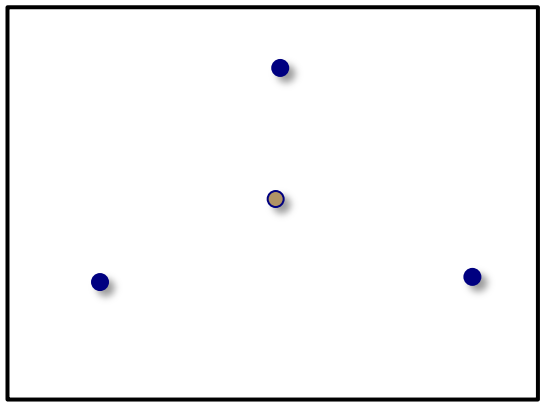
$$\Rightarrow R(h) \lessapprox \widehat{R}_n(h) + \sqrt{\frac{8d_{\mathsf{VC}}}{n} \log \frac{8n}{\delta}}$$

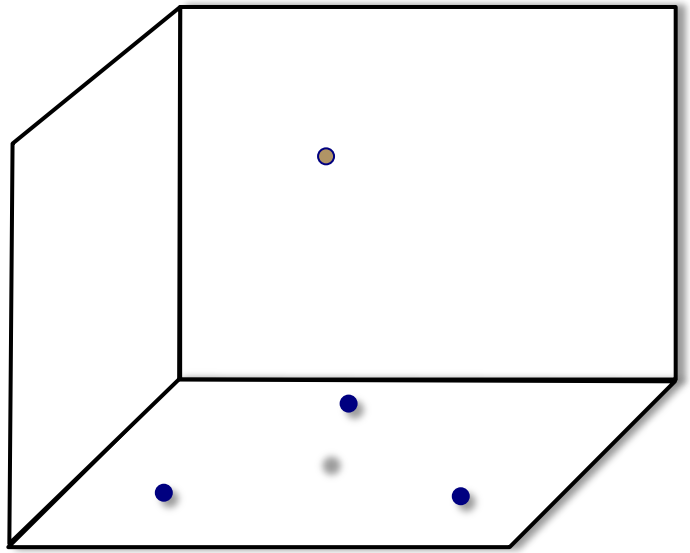
Examples

- Positive rays:
$$d_{VC} = 1$$
- Positive intervals:
$$d_{VC} = 2$$
- Convex sets:
$$d_{VC} = \infty$$
- Linear classifiers in $\mathbb{R}^2$:
$$d_{VC} = 3$$

VC dimension of general linear classifiers

For $d = 2$, $d_{VC} = 3$





In general $d_{VC} = d + 1$

We will prove this by showing that $d_{VC} \leq d + 1$ and $d_{VC} \geq d + 1$

One direction

Lets first show that there exists a set of $d+1$ points in $\mathbb{R}^d$ that are shattered

$$\mathbf{X} = \underbrace{\begin{bmatrix} -\widetilde{\mathbf{x}}_1^T- \\ -\widetilde{\mathbf{x}}_2^T- \\ \vdots \\ -\widetilde{\mathbf{x}}_{d+1}^T- \end{bmatrix}}_{d+1} = \begin{bmatrix} 1 & 0 & 0 & & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & & 0 \\ & \vdots & & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

One can show that $\mathbf{X}$ is invertible

Can we shatter this data set?

For any $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{d+1} \end{bmatrix} = \begin{bmatrix} \pm 1 \\ \pm 1 \\ \vdots \\ \pm 1 \end{bmatrix}$, can we find a vector

satisfying $\text{sign}(\mathbf{X}\mathbf{w}) = \mathbf{y}$?

Easy! Just make $\boldsymbol{\theta} = \mathbf{X}^{-1}\mathbf{y}$ and we have

$$\text{sign}(\mathbf{X}\mathbf{w}) = \text{sign}(\mathbf{y}) = \mathbf{y}$$

We can shatter a set of $d + 1$ points

What does this prove?

a) $d_{VC} = d + 1$

b) $d_{VC} \geq d + 1$ 

c) $d_{VC} \leq d + 1$

d) None of the above

To finish the proof

In order to show that $d_{VC} \leq d + 1$, we need to show

a) There are $d + 1$ points we cannot shatter

b) There are $d + 2$ points we cannot shatter

c) We cannot shatter any set of $d + 1$ points

d) We cannot shatter any set of $d + 2$ points 

The other direction

Take any $d + 2$ points $\mathbf{x}_1, \ldots, \mathbf{x}_{d+2}$

More points than dimensions, so there must be some j for which

$$\mathbf{x}_j = \sum_{i \neq j} \alpha_i \mathbf{x}_i$$

where not all $\alpha_i = 0$

Consider the dichotomy where the $\mathbf{x}_i$ with $\alpha_i \neq 0$ are labeled $y_i = \mathsf{sign}(\alpha_i)$, and $y_j = -1$

No linear classifier can implement such a dichotomy!

Why not?

$$\mathbf{x}_j = \sum_{i \neq j} \alpha_i \mathbf{x}_i \quad \Rightarrow \quad \boldsymbol{\theta}^T \mathbf{x}_j = \sum_{i \neq j} \alpha_i \boldsymbol{\theta}^T \mathbf{x}_i$$

If $y_i = \text{sign}(\boldsymbol{\theta}^T x_i) = \text{sign}(\alpha_i)$, then $\alpha_i \boldsymbol{\theta}^T \mathbf{x}_i > 0$

This means that $\boldsymbol{\theta}^T \mathbf{x}_j = \sum_{i \neq j} \alpha_i \boldsymbol{\theta}^T \mathbf{x}_i > 0$

Thus $y_j = \text{sign}(\boldsymbol{\theta}^T \mathbf{x}_j) + 1$

Interpreting the VC dimension

We have just shown that for a linear classifier in $\mathbb{R}^d$

$$d_{VC} \geq d + 1$$
$$d_{VC} \leq d + 1$$

$$d_{VC} = d + 1$$

How many parameters does a linear classifier in $\mathbb{R}^d$ have?

$$\mathbf{w} \in \mathbb{R}^d$$
$$b \in \mathbb{R}$$

$$d + 1$$

The usual examples

- Positive rays
  - $d_{VC} = 1$
  - 1 parameter
- Positive intervals
  - $d_{VC} = 2$
  - 2 parameters
- Convex sets
  - $d_{VC} = \infty$
  - as many parameters as you want

Effective number of parameters

Additional parameters do not always contribute additional degrees of freedom

Example

Take the output of a linear classifier, and then feed this into another linear classifier

$$y_i = \text{sign}\left(w'\left(\text{sign}(\boldsymbol{\theta}^T x_i)\right) + b'\right)$$

The parameters w' and b' are totally redundant
(they do not allow us to create any new classifiers/dichotomies)

VC bound in action

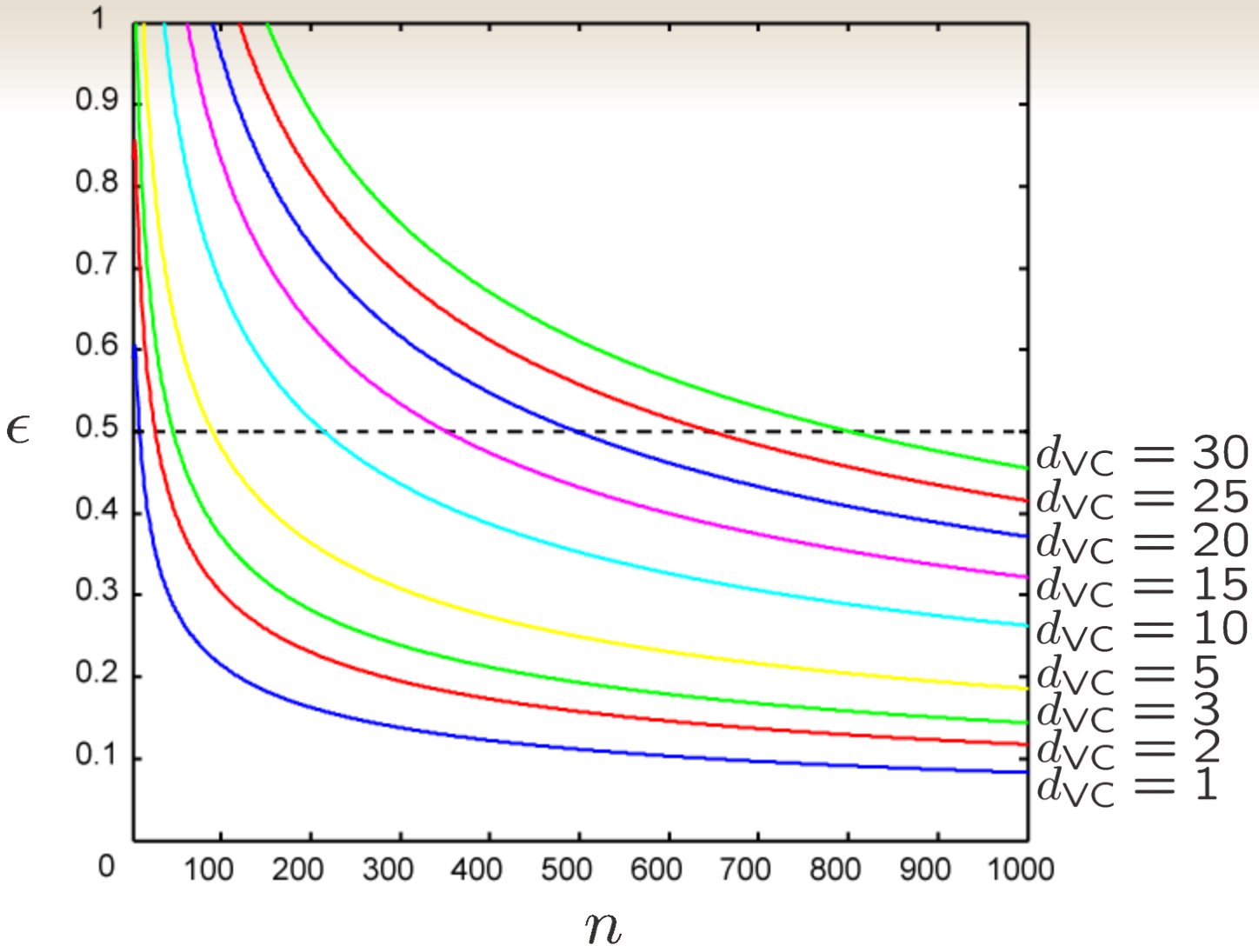
How big does our training set need to be?

$$R(h) \lessapprox \widehat{R}_n(h) + \underbrace{\sqrt{\frac{8d_{VC}}{n} \log \frac{8n}{\delta}}}_{\epsilon}$$

Just to see how this behaves, let's ignore the constants and suppose that

$$\epsilon \sim \sqrt{\frac{d_{VC}}{n} \log n}$$

VC bound in action



RULE OF THUMB: $n \geq 10 d_{\mathrm{VC}}$

Regularization

$$R(h) \lessapprox \widehat{R}_n(h) + \sqrt{\frac{8 d_{VC}}{n} \log \frac{8n}{\delta}}$$

measure of the complexity of the set our hypothesis h came from

The idea of ***regularization*** is to choose a hypothesis by minimizing the sum of the training error and some additional term that penalizes "complexity"

If you can get just as small of a training error with a simpler hypothesis, do it!

Lecture 9: Overfitting, Bias, and Variance

The learning problem

Given a set $\mathcal{H}$, find a function $h \in \mathcal{H}$ that minimizes $R(h)$

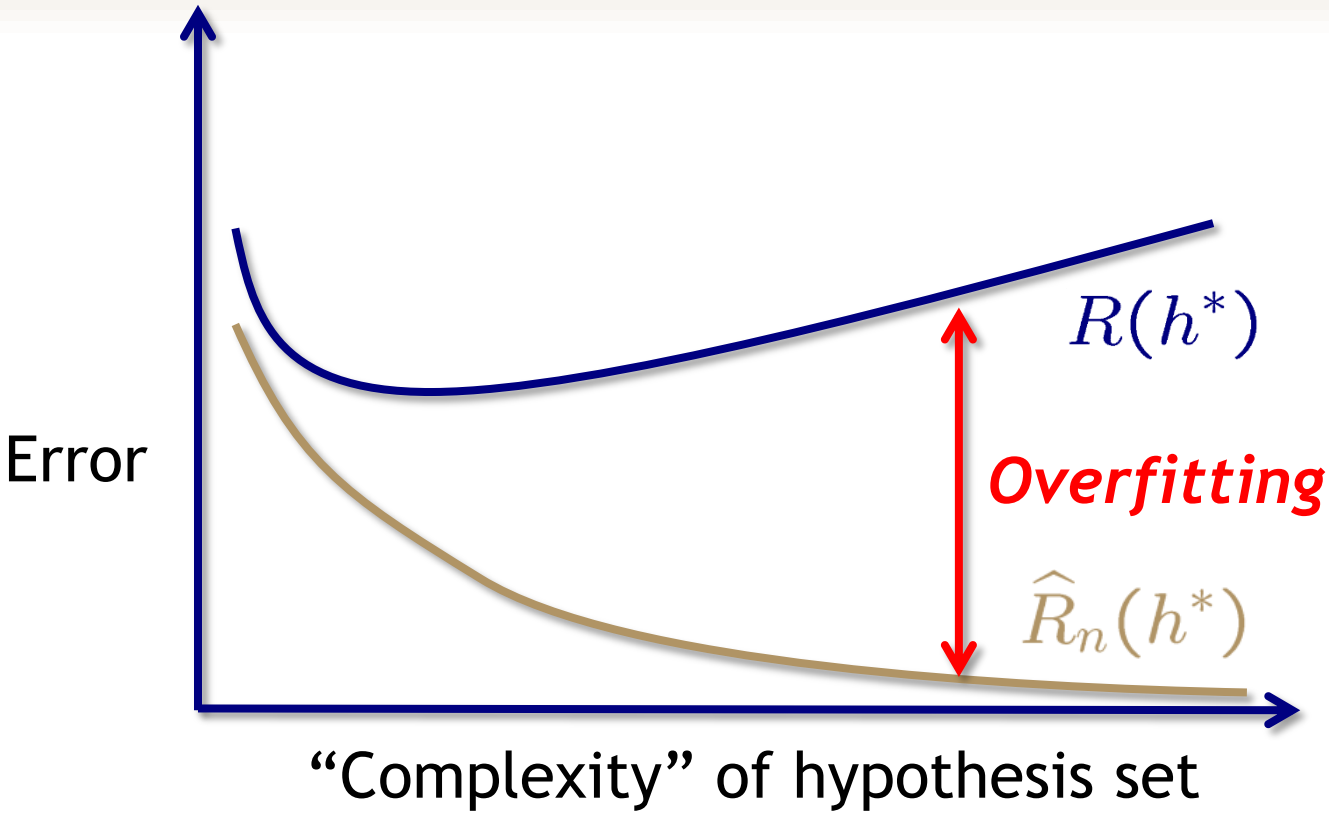
In the case of classification, we can also think of this as trying to find an $h \in \mathcal{H}$ that approximates the Bayes classifier f^*

More complex $\mathcal{H}$ ➡ better chance of ***approximating*** f^*

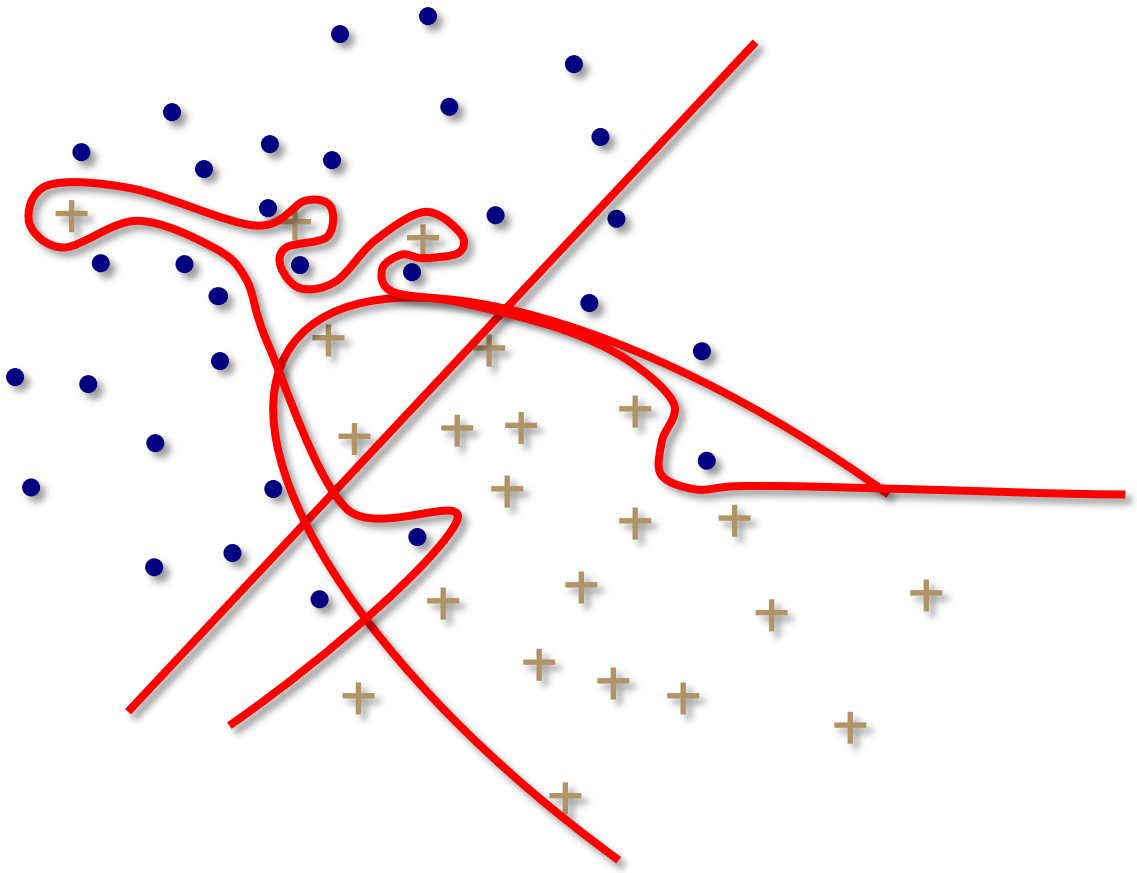
Less complex $\mathcal{H}$ ➡ better chance of ***generalizing*** to new data (out of sample)

Approximation-generalization tradeoff

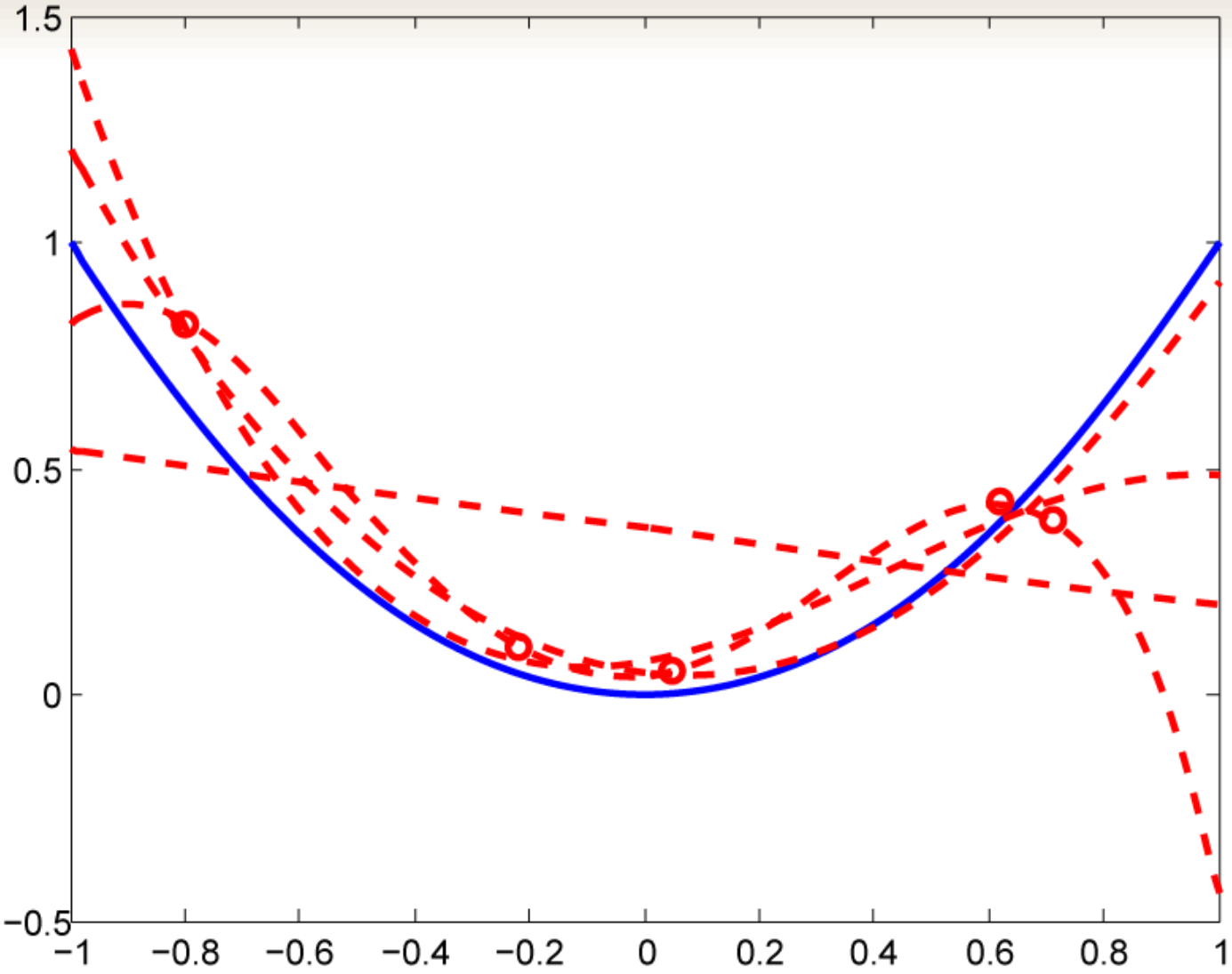
Approximation-generalization tradeoff



Overfitting



Overfitting



Is the problem just noise?

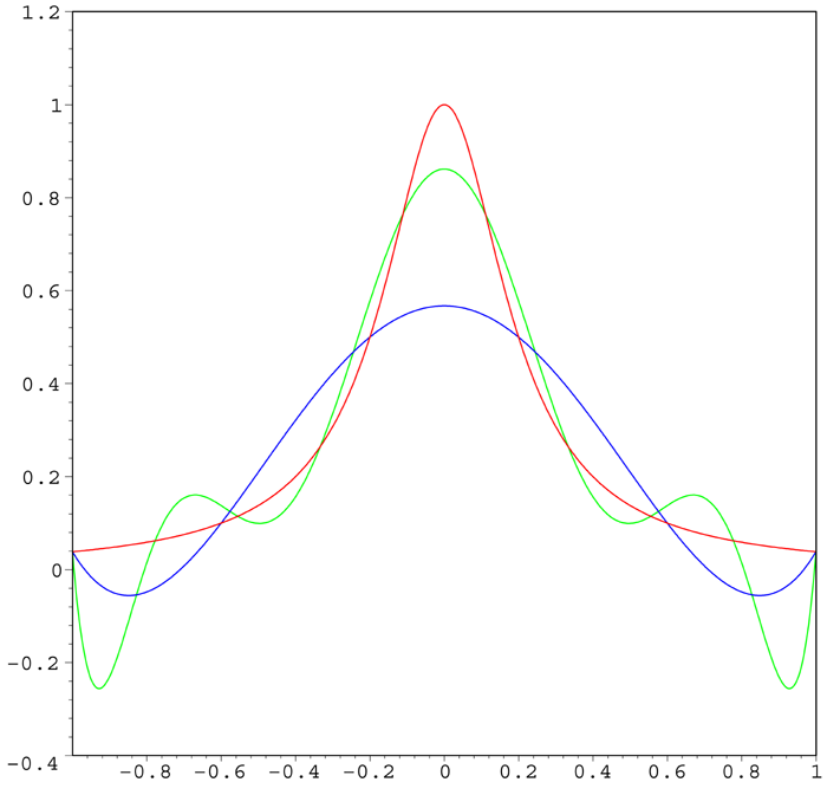
Noise in the observations can make overfitting a big problem

What if there is no noise?

Runge's phenomenon

Take a smooth function

- not exactly polynomial
- well approximated by a polynomial



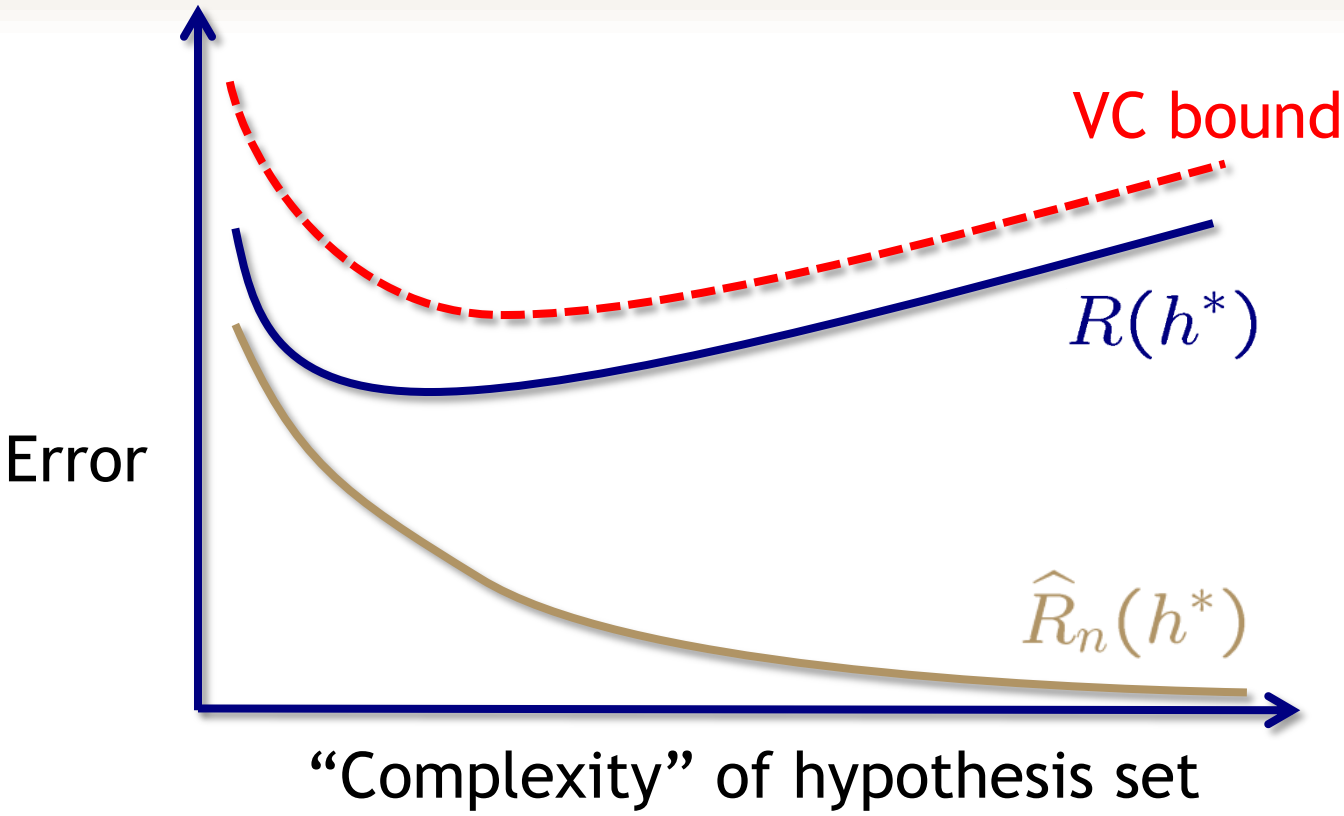
Even in the absence of noise, fitting a higher order polynomial (interpolation) can be incredibly unstable

Quantifying the tradeoff

VC generalization bound

$$R(h) \lessapprox \widehat{R}_n(h) + \epsilon(\mathcal{H}, n)$$

VC generalization bound



Quantifying the tradeoff

VC generalization bound

$$R(h) \lessapprox \widehat{R}_n(h) + \epsilon(\mathcal{H}, n)$$

Alternative approach: Bias-variance decomposition

- ***bias:*** how well can $\mathcal{H}$ approximate f^*
- ***variance:*** how well can we pick a good $h \in \mathcal{H}$

$$R(h) = \text{bias} + \text{variance}$$

Bias-variance decomposition is especially useful because it more easily generalizes to regression

Bias-variance decomposition

In this treatment, we will assume real-valued observations (i.e., regression) and consider the ***squared error***

$$\mathcal{D} := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \qquad \begin{array}{l} \mathbf{x} \in \mathbb{R}^d \\ y \in \mathbb{R} \end{array}$$

$f : \mathbb{R}^d \to \mathbb{R}$: unknown target function

$h_{\mathcal{D}} : \mathbb{R}^d \to \mathbb{R}$: function in $\mathcal{H}$ we pick using $\mathcal{D}$

$$R(h_{\mathcal{D}}) = \mathbb{E}_X \left[(h_{\mathcal{D}}(X) - f(X))^2 \right]$$

Setting up the decomposition

$$R(h_{\mathcal{D}}) = \mathbb{E}_X \left[(h_{\mathcal{D}}(X) - f(X))^2 \right]$$

$$\mathbb{E}_{\mathcal{D}} \left[R(h_{\mathcal{D}}) \right] = \mathbb{E}_{\mathcal{D}} \left[\mathbb{E}_X \left[(h_{\mathcal{D}}(X) - f(X))^2 \right] \right]$$

$$= \mathbb{E}_X \left[\mathbb{E}_{\mathcal{D}} \left[(h_{\mathcal{D}}(X) - f(X))^2 \right] \right]$$

let's focus on just this term

The average hypothesis

To evaluate

$$\mathbb{E}_{\mathcal{D}}\left[(h_{\mathcal{D}}(X) - f(X))^2\right]$$

we define the *"average hypothesis"*

$$\bar{h}(X) = \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(X)\right]$$

Interpretation

Imagine drawing many data sets $\mathcal{D}_1, \ldots, \mathcal{D}_p$

$$\bar{h}(X) \approx \frac{1}{p}\sum_{i=1}^{p} h_{\mathcal{D}_i}(X)$$

Using the average hypothesis

$$\mathbb{E}_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(X) - f(X)\right)^2\right]$$

$$= \mathbb{E}_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(X) - \bar{h}(X) + \bar{h}(X) - f(X)\right)^2\right]$$

$$= \mathbb{E}_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(X) - \bar{h}(X)\right)^2 + \left(\bar{h}(X) - f(X)\right)^2 \right.$$

$$\left. +2\left(h_{\mathcal{D}}(X) - \bar{h}(X)\right)\left(\bar{h}(X) - f(X)\right)\right]$$

$$= \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(X) - \bar{h}(X)\right)^2\right]}_{\text{variance}(X)} + \underbrace{\left(\bar{h}(X) - f(X)\right)^2}_{\text{bias}(X)}$$

Bias and variance

Plugging this back into our original expression, we get

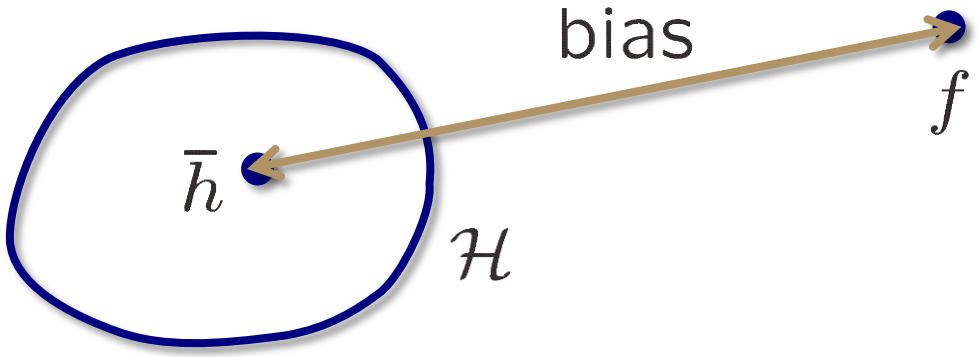
$$\mathbb{E}_{\mathcal{D}}\left[R(h_{\mathcal{D}})\right] = \mathbb{E}_X\left[\mathbb{E}_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(X) - f(X)\right)^2\right]\right]$$

$$= \mathbb{E}_X\left[\text{bias}(X) + \text{variance}(X)\right]$$

$$= \text{bias} + \text{variance}$$

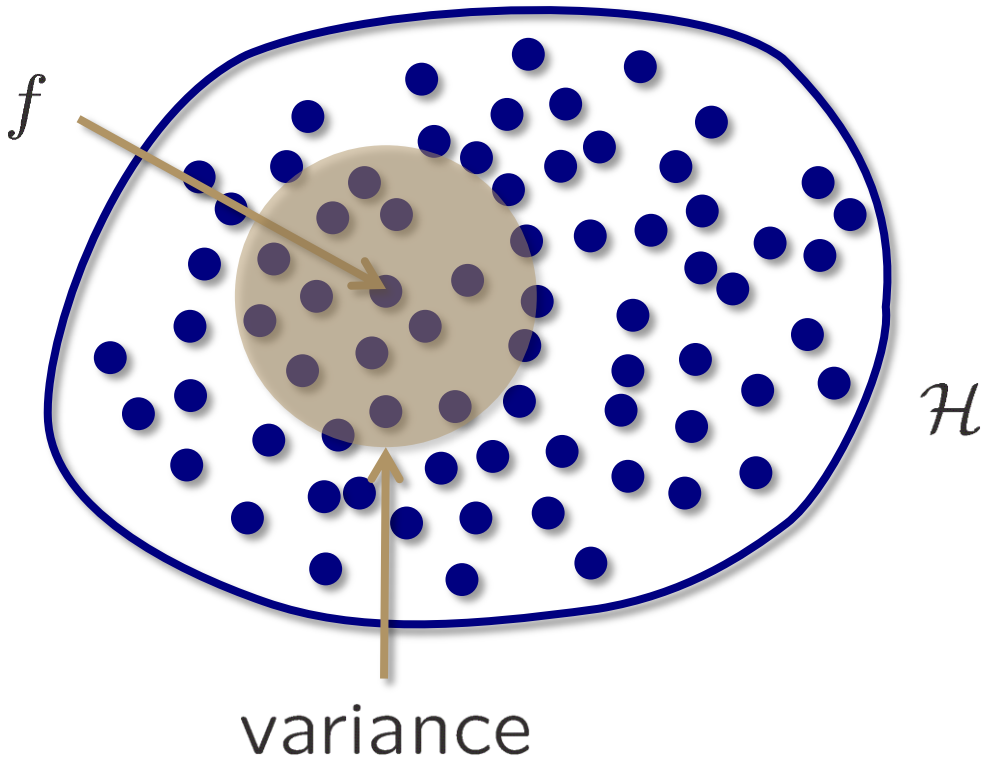
Visualizing the bias

$$\text{bias} = \mathbb{E}_X\left[\left(\bar{h}(X) - f(X)\right)^2\right]$$



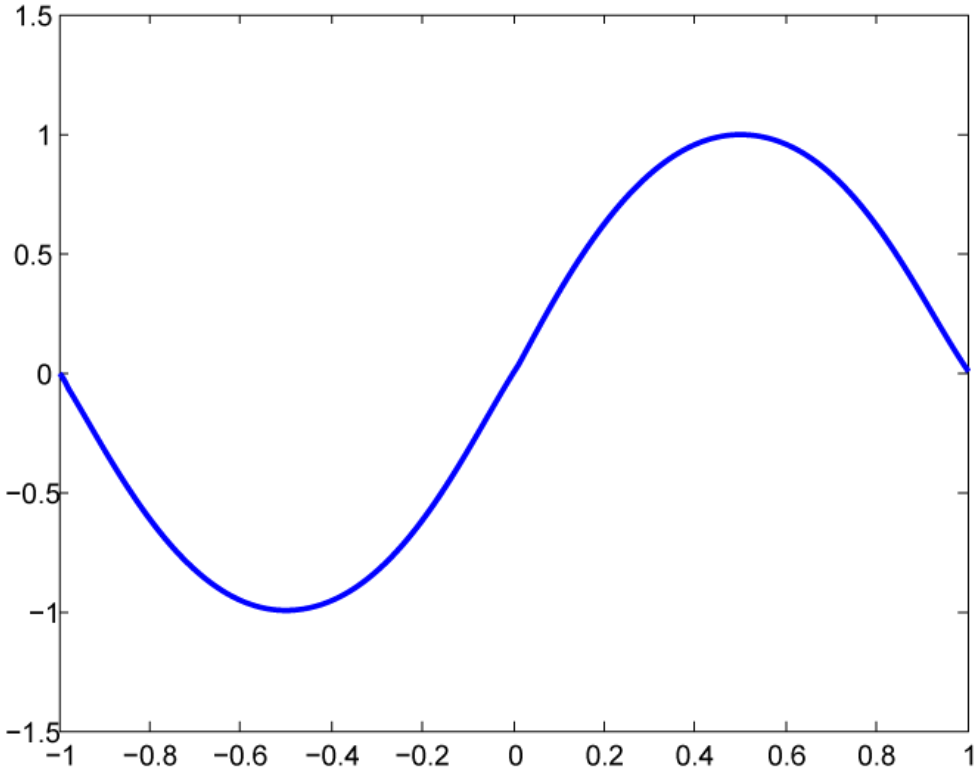
Visualizing the variance

$$\text{variance} = \mathbb{E}_X \left[\mathbb{E}_{\mathcal{D}} \left[\left(h_{\mathcal{D}}(X) - \bar{h}(X) \right)^2 \right] \right]$$



Example: Learning a sine

Suppose $f(x) = \sin(\pi x)$ and we get $n = 2$ training examples



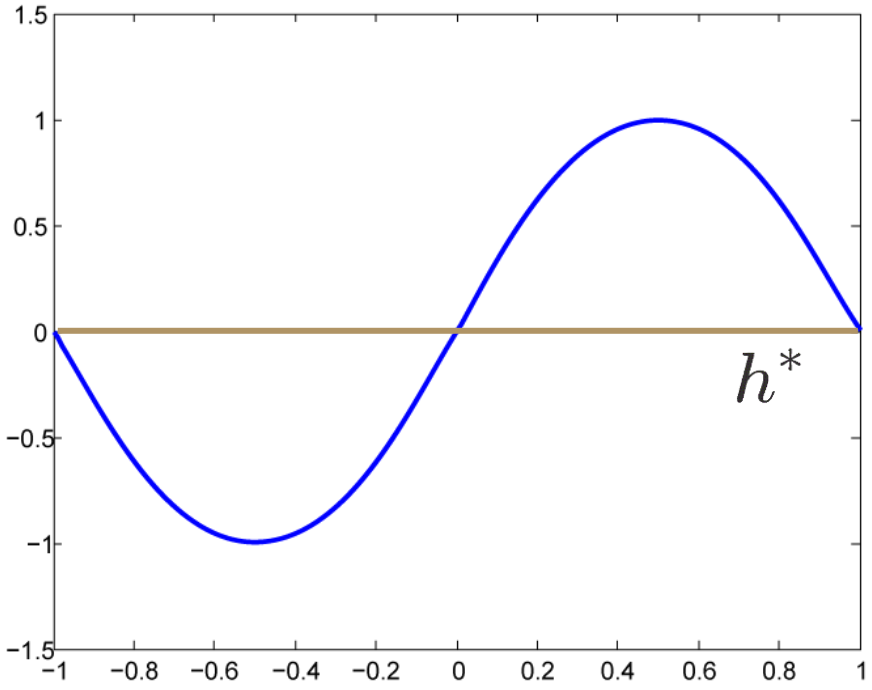
Consider two possible hypothesis sets

- $\mathcal{H}_0 : h(x) = b$
- $\mathcal{H}_1 : h(x) = ax + b$

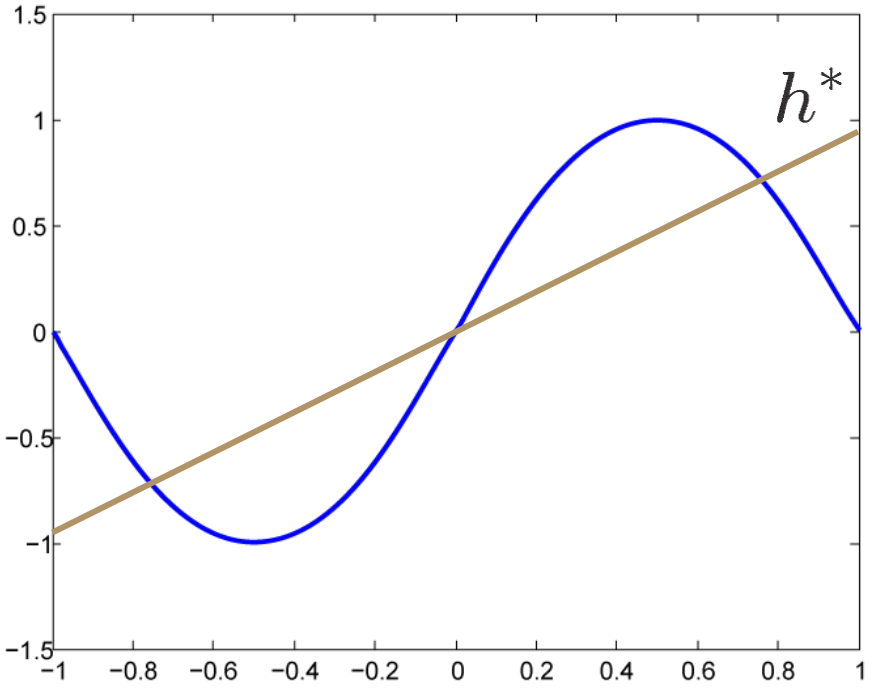
Which one is better?

Approximation

$\mathcal{H}_0$



$\mathcal{H}_1$

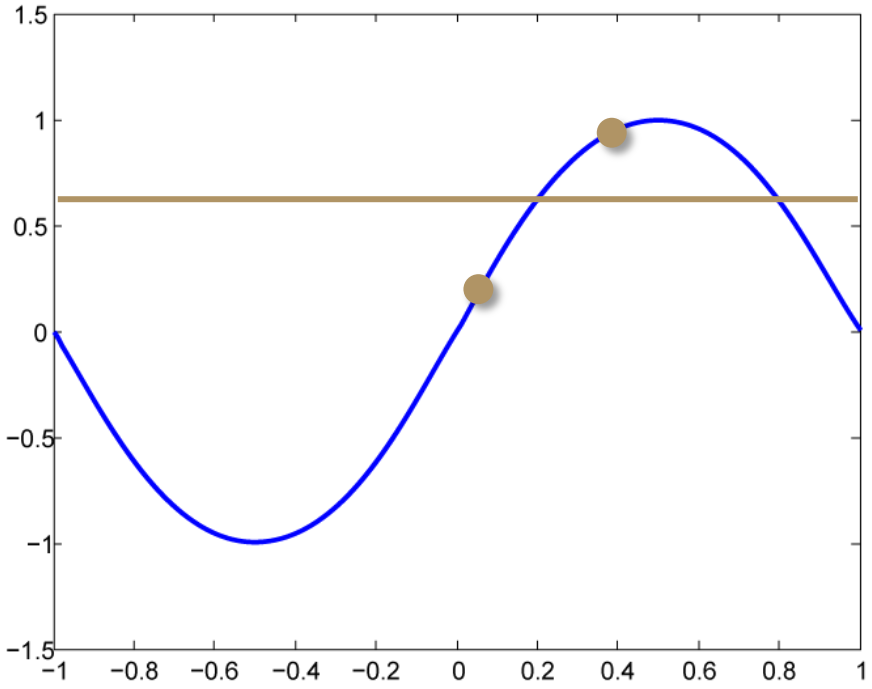


$$R(h^*) = \frac{1}{2}$$

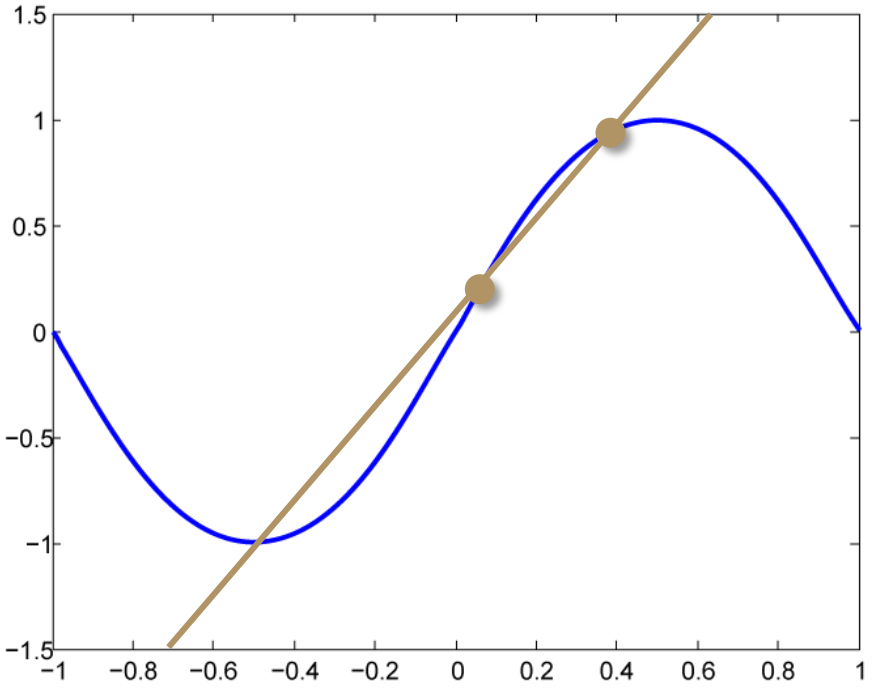
$$R(h^*) = \frac{1}{2} - \frac{3}{\pi^2} \approx 0.196$$

Learning

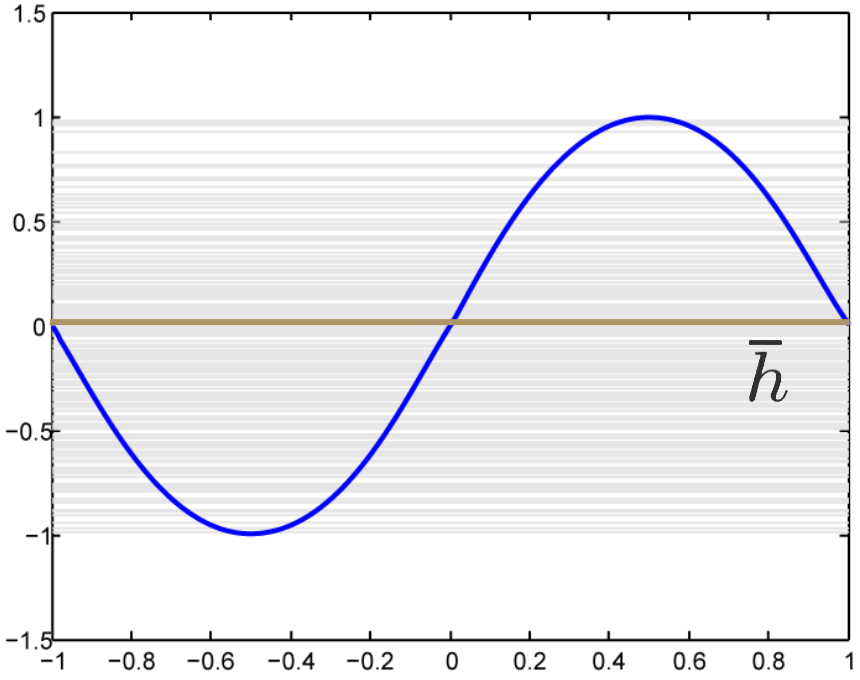
$\mathcal{H}_0$



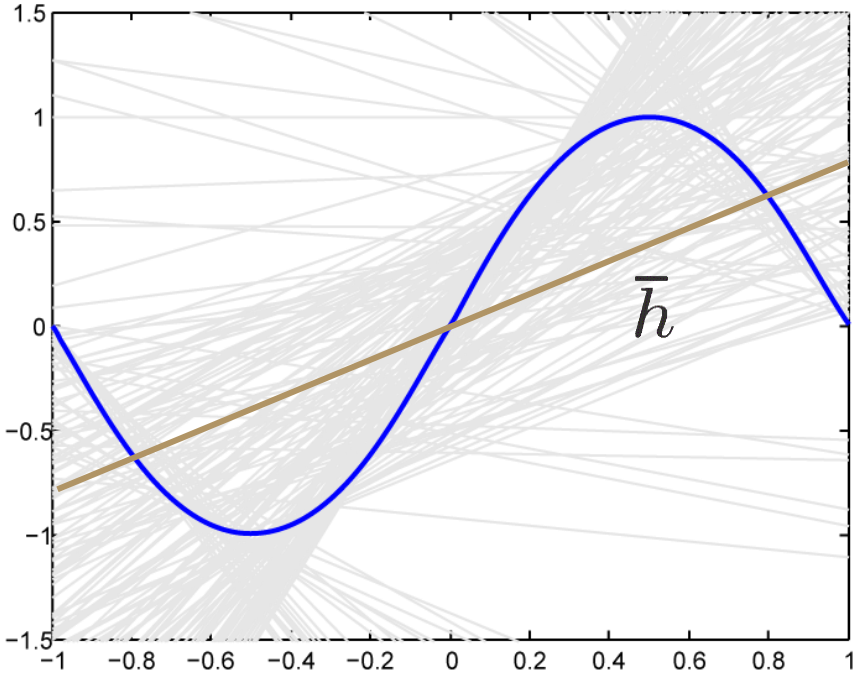
$\mathcal{H}_1$



Average hypothesis for $\mathcal{H}_0$



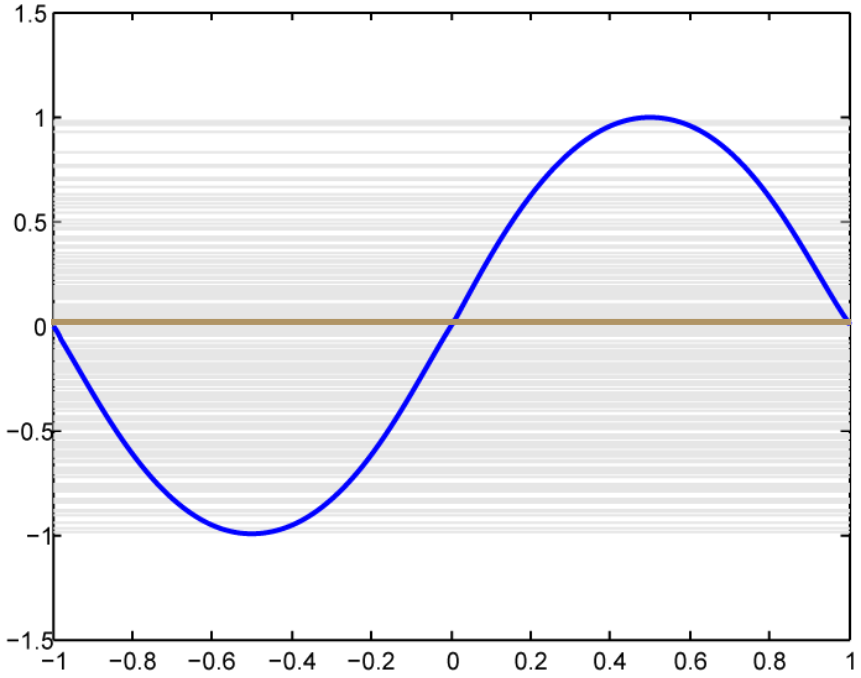
Average hypothesis for $\mathcal{H}_1$



… and the winner is?

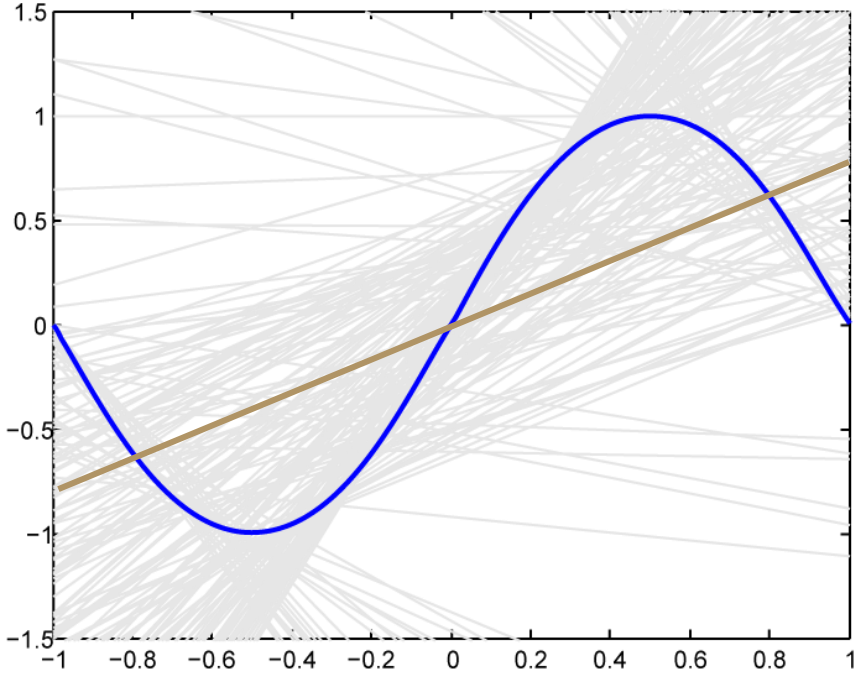
$$\mathbb{E}_{\mathcal{D}}\left[R(h_{\mathcal{D}})\right] = \text{bias} + \text{variance}$$

$\mathcal{H}_0$



$\text{bias} = 0.50$
$\text{variance} = 0.25$

$\mathcal{H}_1$



$\text{bias} \approx 0.21$
$\text{variance} \approx 1.69$

Moral of this story?

VC bound

Keep the "model complexity" small enough compared to n and we can learn ***any*** f

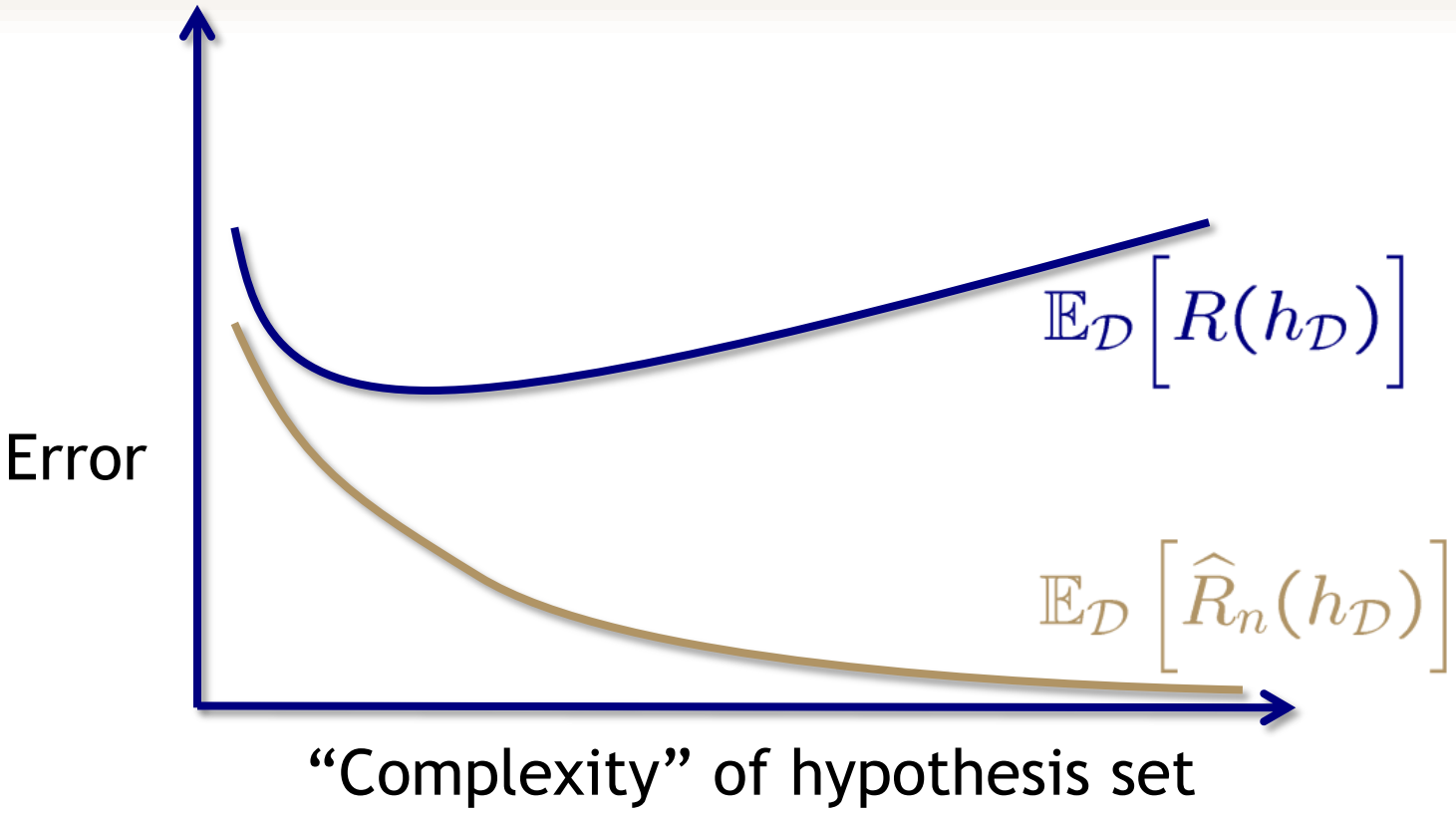
Bias-variance decomposition

For any particular f, we do best by matching the "model complexity" to the "data resources" (not to f)

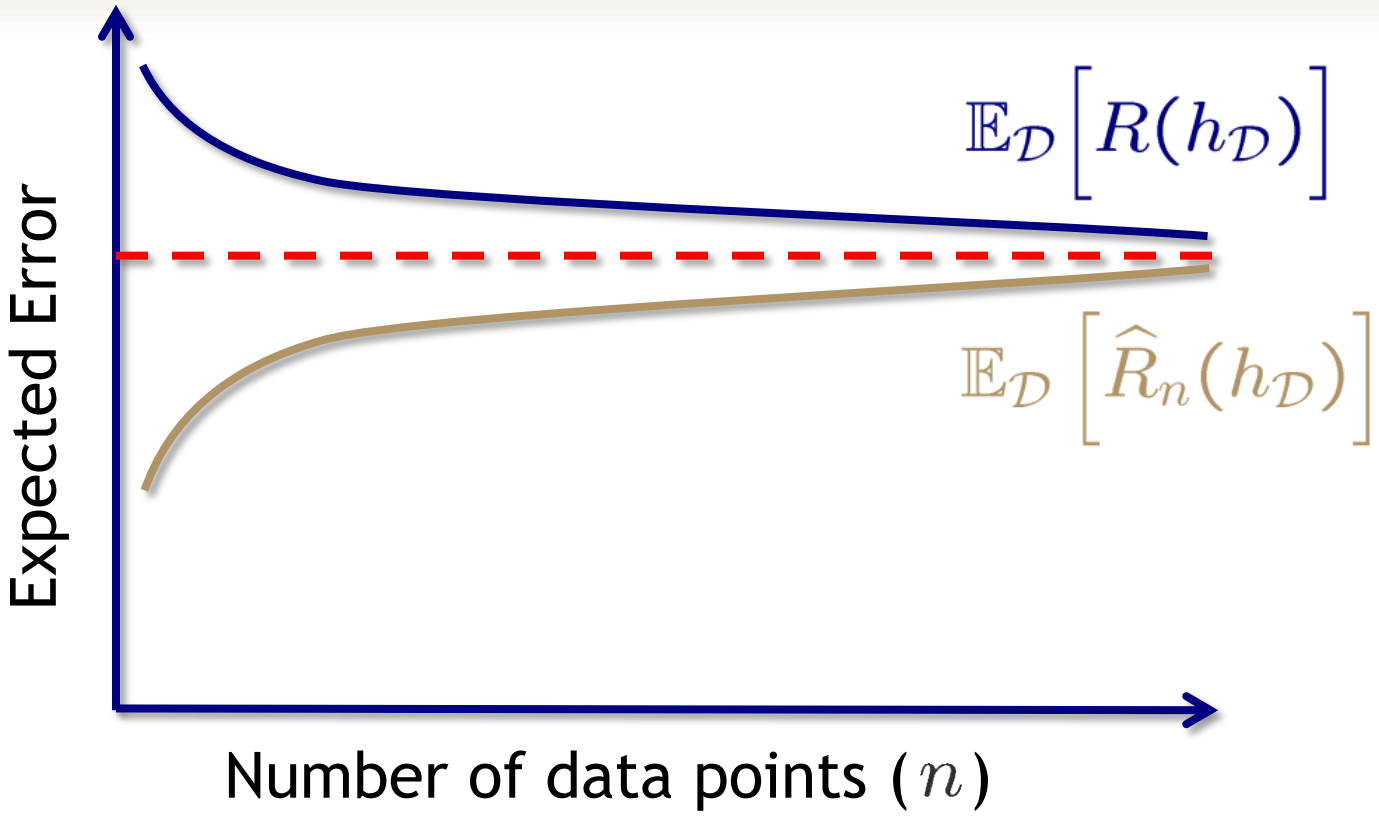
Balance between

- increasing the model complexity to reduce bias
- decreasing the model complexity to reduce variance

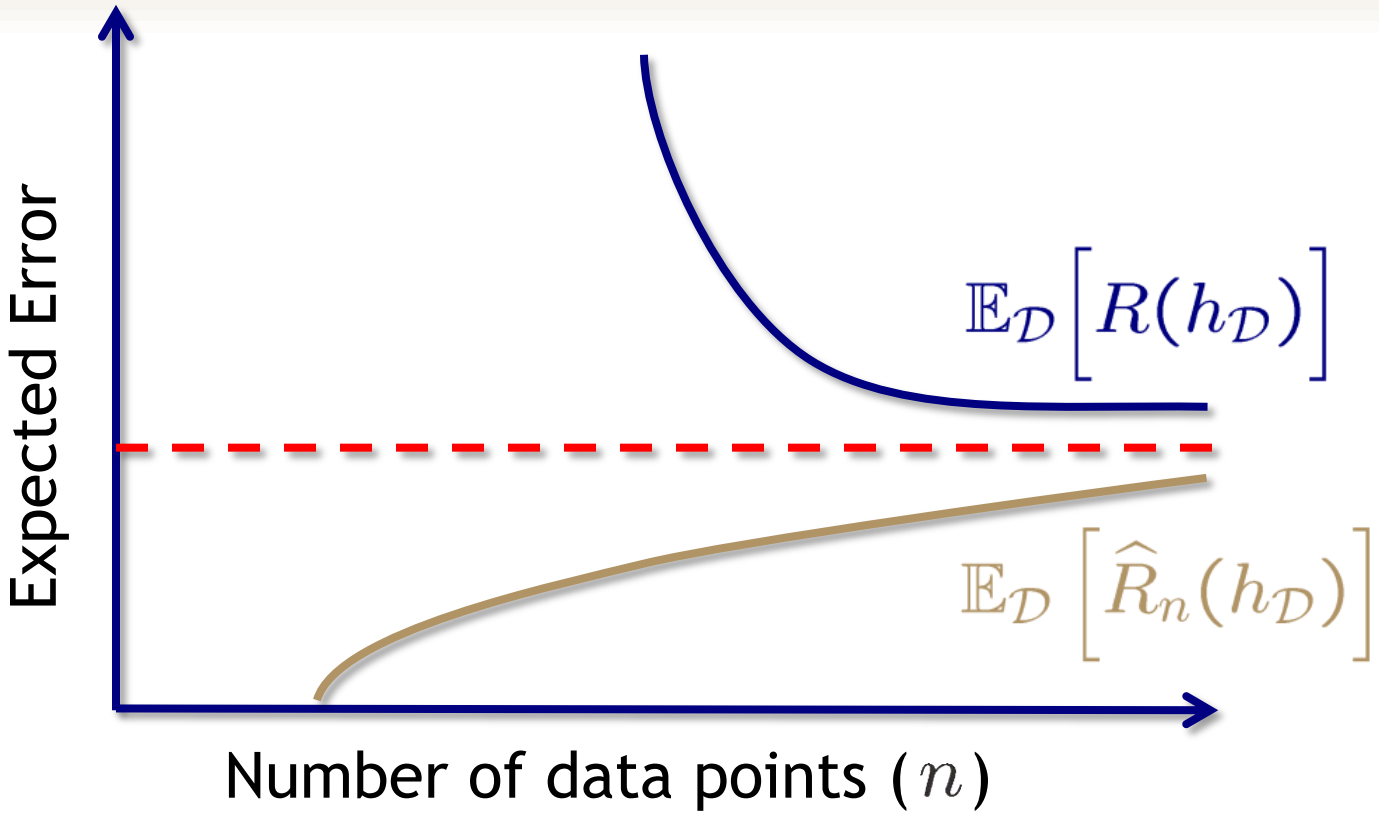
Approximation-generalization tradeoff



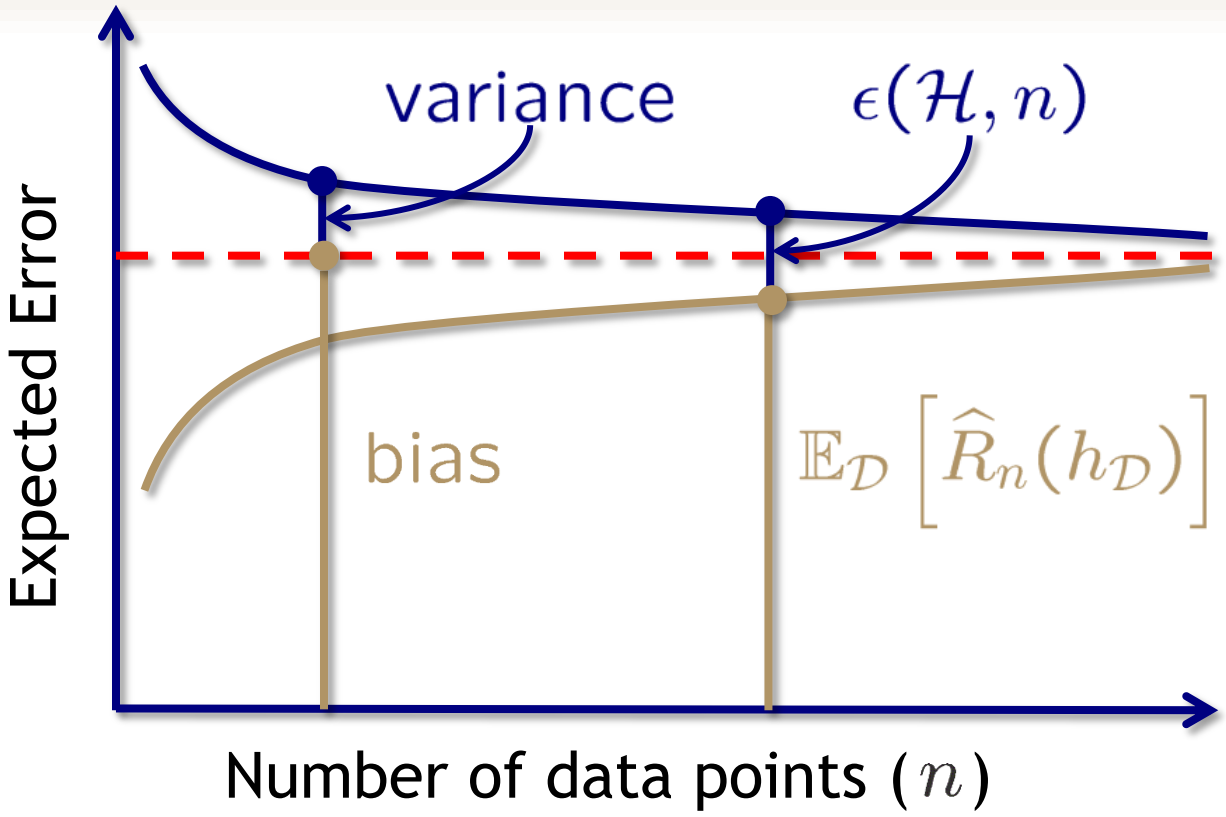
Learning curve – A simple model



Learning curve – A complex model

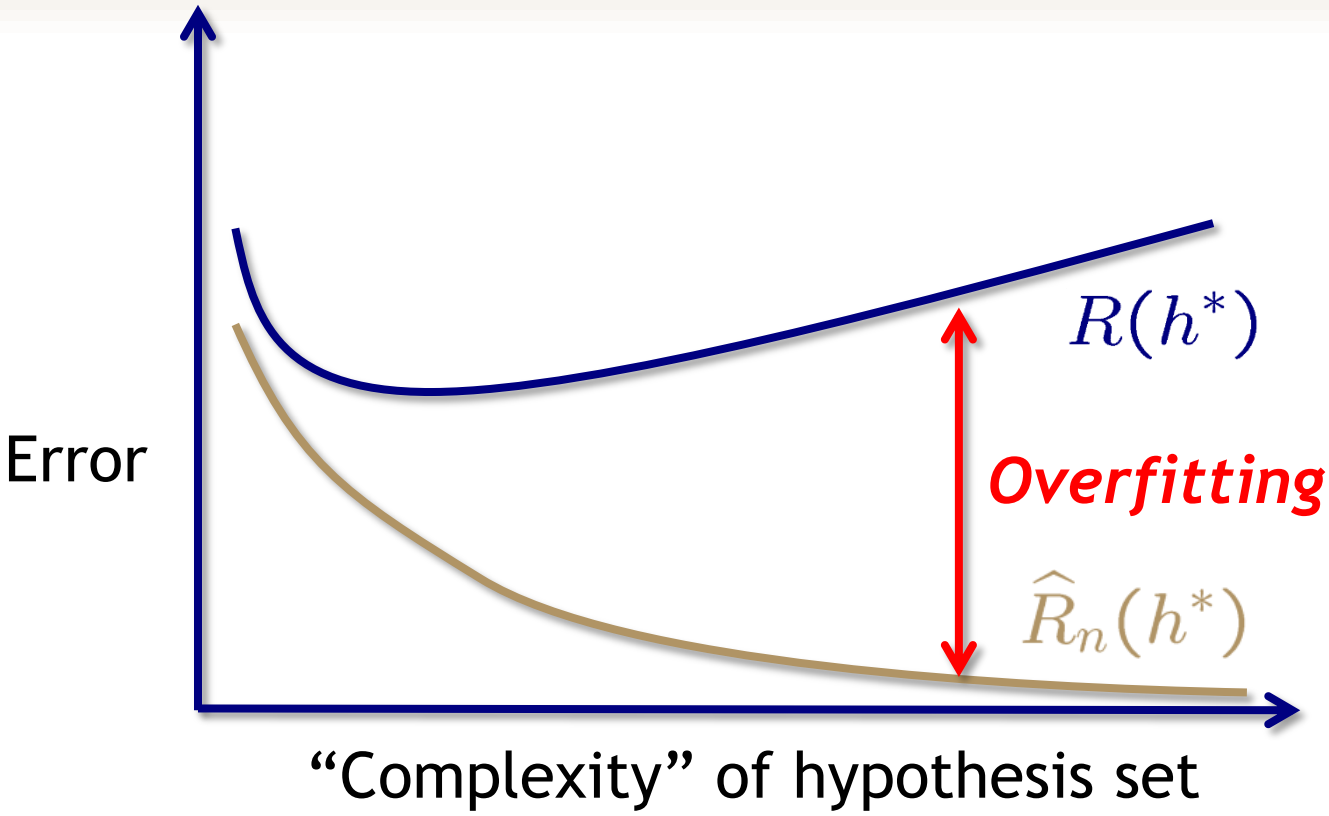


VC versus bias-variance

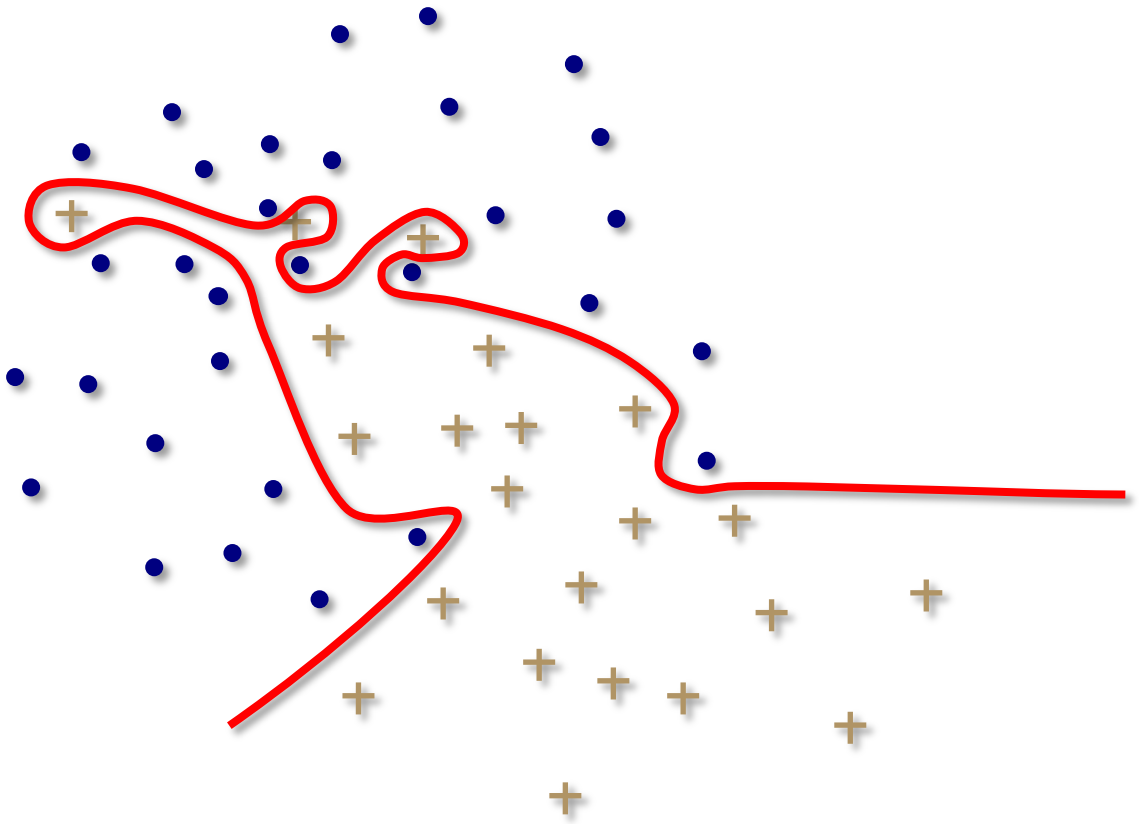


Lecture 10: Regularization

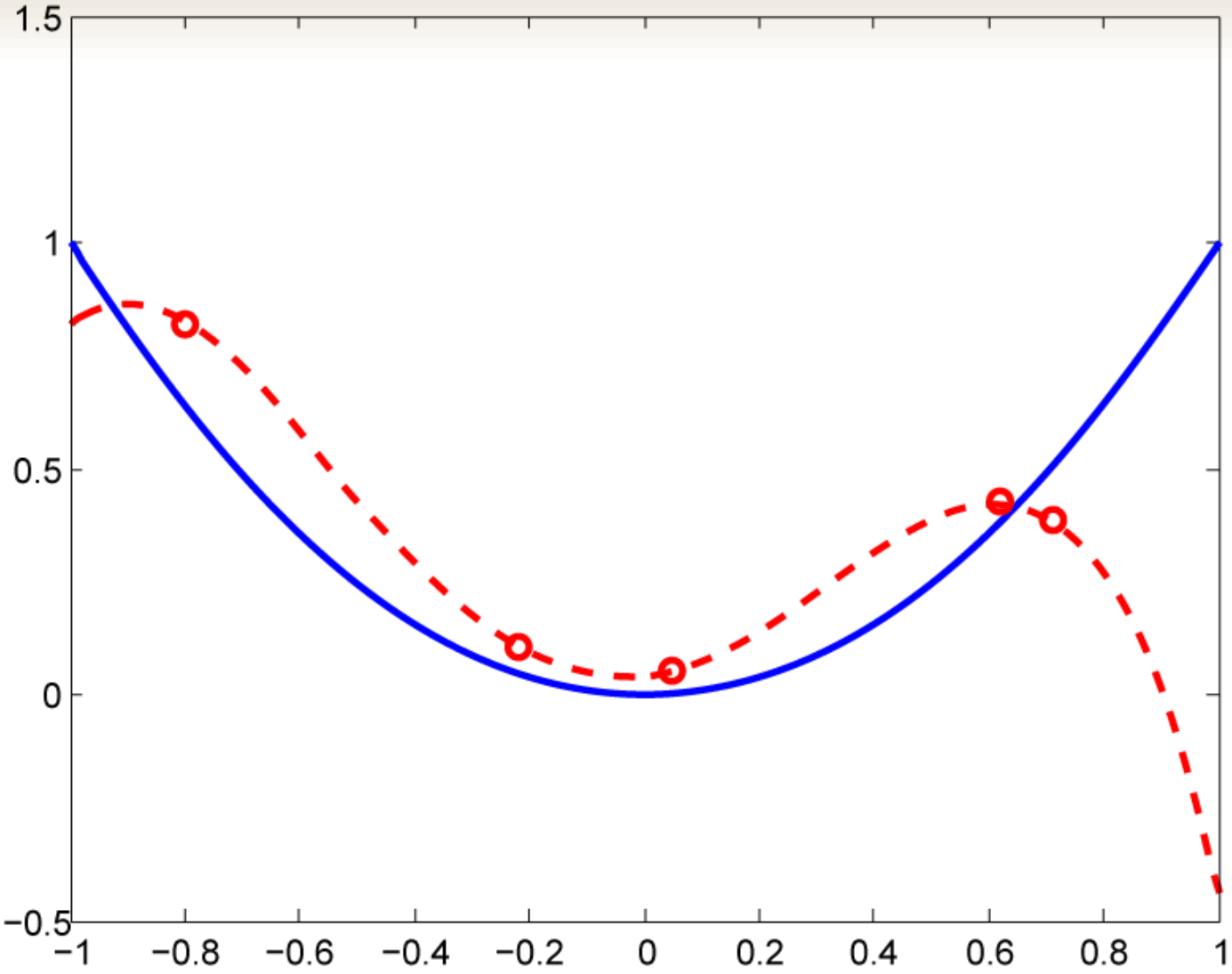
Approximation-generalization tradeoff



Overfitting



Overfitting



Quantifying the tradeoff

VC generalization bound

$$R(h) \leq \widehat{R}_n(h) + \epsilon(\mathcal{H}, n)$$

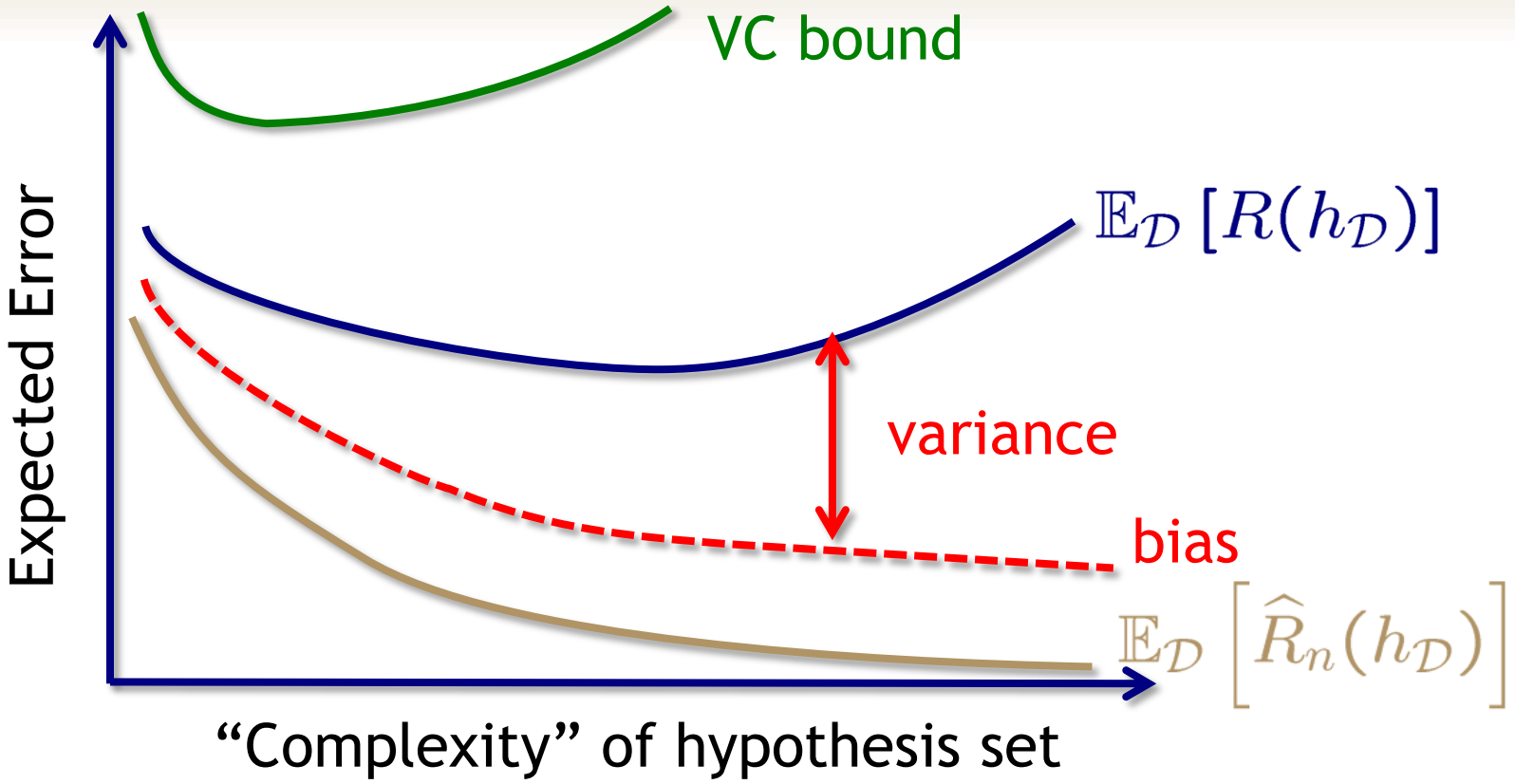
Alternative approach: Bias-variance decomposition

- ***bias:*** how well can $\mathcal{H}$ approximate f^*
- ***variance:*** how well can we pick a good $h \in \mathcal{H}$

$$R(h) = \text{bias} + \text{variance}$$

Bias-variance decomposition is especially useful because it more easily generalizes to regression

VC generalization bound



How to tune the complexity?

The VC analysis tends to suggest an approach called ***empirical risk minimization***

ERM: Simply pick the $h \in \mathcal{H}$ that minimizes $\widehat{R}_n(h)$

VC analysis shows that this works provided that $\mathcal{H}$ is not "too big", so that we can get a good guarantee of the form

$$R(h) \leq \widehat{R}_n(h) + \epsilon(\mathcal{H}, n)$$

One approach to selecting $\mathcal{H}$ would be to consider a wide range of $\mathcal{H}$, find the minimal $\widehat{R}_n(h)$ for each one, and to pick the $\mathcal{H}$ that minimizes the VC bound

Unfortunately, the VC bound is incredibly loose and so this approach tends to be far too conservative

Regularization

The idea of ***regularization*** is to choose a hypothesis by directly minimizing the sum of the training error and some additional term that penalizes "complexity"

$$\widehat{R}_n(h) + r(h, n)$$

This differs from the VC/ERM approach because

- we are essentially considering a much larger $\mathcal{H}$, but then penalizing each $h \in \mathcal{H}$ differently
- our "regularizer" $r(h, n)$ is not limited to be some function of the VC dimension, but can be much more general

Intuition: Promote "simple" hypotheses – if you can get just as small of a training error with a simpler hypothesis, do it!

Regression recap

Recall that in regression we are given training data

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$$

where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$

In linear regression we assume that we are trying to estimate a function of the form

$$f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x} + \beta_0$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$, $\beta_0 \in \mathbb{R}$

Least squares regression: Select $\boldsymbol{\beta}, \beta_0$ to minimize

$$\text{SSE}(\boldsymbol{\beta}, \beta_0) := \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0\right)^2$$

Regression recap

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} 1 & x_1(1) & \cdots & x_1(d) \\ 1 & x_2(1) & \cdots & x_2(d) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n(1) & \cdots & x_n(d) \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \beta_0 \\ \beta(1) \\ \vdots \\ \beta(d) \end{bmatrix}$$

$$\text{SSE}(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0)^2 = \|\boldsymbol{y} - \boldsymbol{A\theta}\|^2$$

Minimizer given by

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \boldsymbol{y}$$

provided that $\boldsymbol{A}^T \boldsymbol{A}$ is ***nonsingular***

Regularization and regression

Overfitting occurs as the number of features d begins to approach the number of observations n

In this regime, we have *too many degrees of freedom*

Idea: penalize candidate solutions for using too many features

One candidate regularizer: $r(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|^2$

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\theta}\|^2 + \lambda\|\boldsymbol{\theta}\|^2$$

$\lambda > 0$ is a "tuning parameter" that controls the tradeoff between fit and complexity

Tikhonov regularization

This is one example of a more general technique called ***Tikhonov regularization***

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{A\theta}\|^2 + \|\boldsymbol{\Gamma\theta}\|^2$$

(Note that λ has been replaced by the matrix $\boldsymbol{\Gamma}$)

Solution: Observe that

$$\begin{aligned}
\|\boldsymbol{y} - \boldsymbol{A\theta}\|^2 + \|\boldsymbol{\Gamma\theta}\|^2 &= (\boldsymbol{y} - \boldsymbol{A\theta})^T(\boldsymbol{y} - \boldsymbol{A\theta}) + \boldsymbol{\theta}^T\boldsymbol{\Gamma}^T\boldsymbol{\Gamma\theta} \\
&= \boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{\theta}^T\boldsymbol{A}^T\boldsymbol{A\theta} - 2\boldsymbol{\theta}^T\boldsymbol{A}^T\boldsymbol{y} \\
&\quad + \boldsymbol{\theta}^T\boldsymbol{\Gamma}^T\boldsymbol{\Gamma\theta} \\
&= \boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{\theta}^T\left(\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{\Gamma}^T\boldsymbol{\Gamma}\right)\boldsymbol{\theta} \\
&\quad - 2\boldsymbol{\theta}^T\boldsymbol{A}^T\boldsymbol{y}
\end{aligned}$$

Tikhonov regularization

$$\frac{\partial}{\partial \boldsymbol{\theta}} \left(\boldsymbol{y}^T \boldsymbol{y} + \boldsymbol{\theta}^T \left(\boldsymbol{A}^T \boldsymbol{A} + \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \right) \boldsymbol{\theta} - 2 \boldsymbol{\theta}^T \boldsymbol{A}^T \boldsymbol{y} \right)$$

$$= 2 \left(\boldsymbol{A}^T \boldsymbol{A} + \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \right) \boldsymbol{\theta} - 2 \boldsymbol{A}^T \boldsymbol{y}$$

Setting this equal to zero and solving for $\boldsymbol{\theta}$ yields

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{A}^T \boldsymbol{A} + \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \right)^{-1} \boldsymbol{A}^T \boldsymbol{y}$$

Suppose $\boldsymbol{\Gamma} = \sqrt{\lambda} \boldsymbol{I}$, then

$$\widehat{\boldsymbol{\theta}} = \left(\underbrace{\boldsymbol{A}^T \boldsymbol{A} + \lambda \boldsymbol{I}} \right)^{-1} \boldsymbol{A}^T \boldsymbol{y}$$

for suitable choice of λ,
always well-conditioned

Another take: Constrained minimization

One can use Lagrange multipliers (KKT conditions) to formally show that

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\theta}\|^2 + \|\boldsymbol{\Gamma}\boldsymbol{\theta}\|^2$$

is formally equivalent to

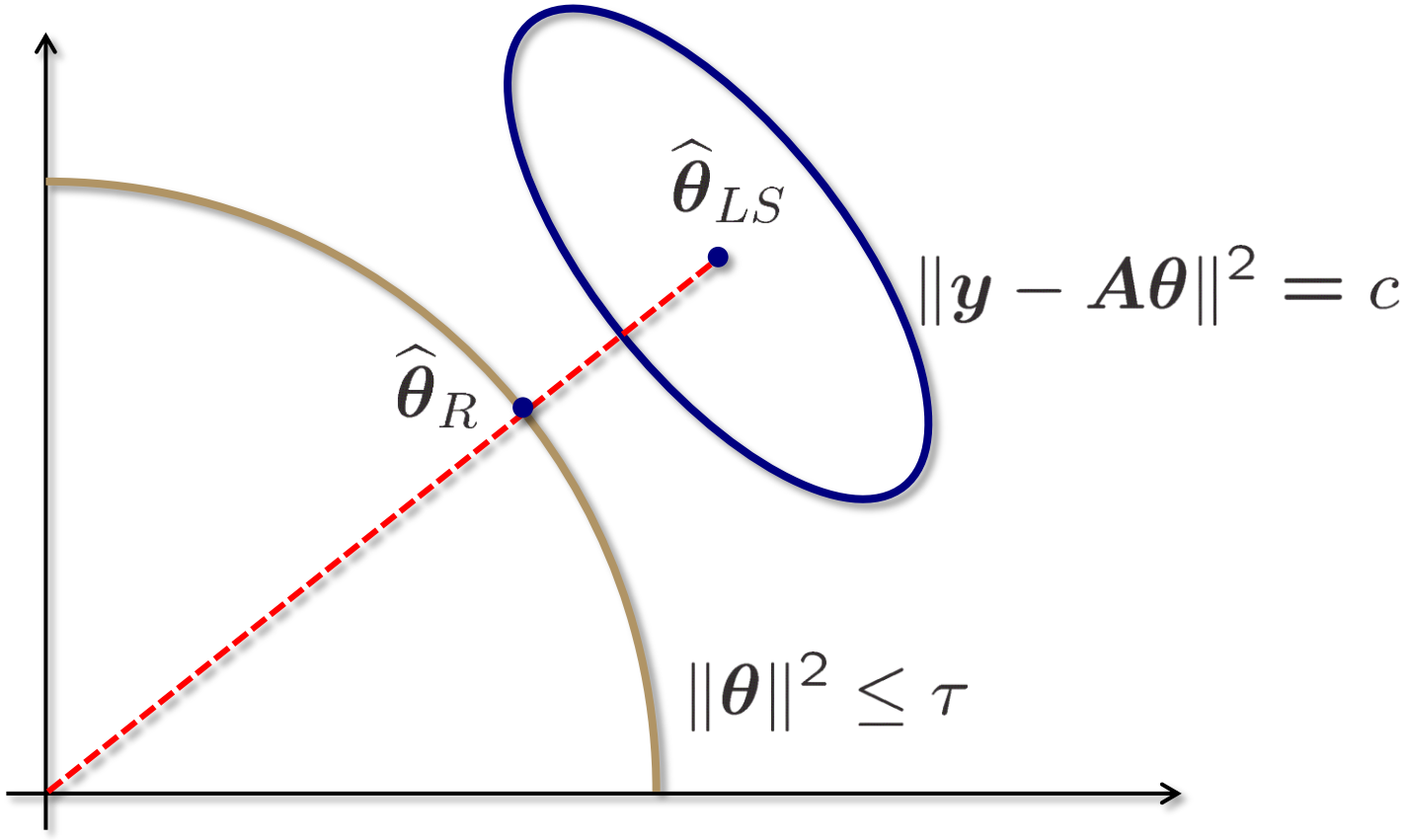
$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\theta}\|^2$$

$$\text{subject to } \|\boldsymbol{\Gamma}\boldsymbol{\theta}\|^2 \leq \tau$$

for a suitable choice of τ

Shrinkage

Assume $\Gamma = I$



Tikhonov regularization is equivalent to shrinking the least squares solution towards the origin

Ridge regression

In the context of regression, Tikhonov regularization has a special name: ***ridge regression***

Ridge regression is essentially exactly what we have been talking about, but in the special case where

$$\Gamma = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{\lambda} & 0 & \cdots & 0 \\ 0 & 0 & \sqrt{\lambda} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{\lambda} \end{bmatrix}$$

We are penalizing all coefficients in β equally, but not penalizing the offset β_0

Alternative regularizers

- Akaike information criterion (AIC)
- Bayesian information criterion (BIC)

$$r(\boldsymbol{\theta}) \approx \|\boldsymbol{\theta}\|_0 := |\mathrm{supp}(\boldsymbol{\theta})|$$

- Least absolute shrinkage and selection operator (LASSO)

$$r(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_j |\theta(j)|$$

  - also results in shrinkage, but where all coordinates are shrunk by the same amount
  - promotes sparsity
  - can think of $\|\boldsymbol{\theta}\|_1$ as a more computationally tractable replacement for $\|\boldsymbol{\theta}\|_0$

Regularized logistic regression

Everything we have said so far about least squares regression can be extended to other regression/classification problems

For example, in logistic regression we can replace

$$\min_{\theta} \; -\ell(\boldsymbol{\theta})$$

with

$$\min_{\theta} \; -\ell(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$$

Has a similar interpretation to least squares regularization

- makes the Hessian matrix well conditioned
- super useful when the number of observations is small

Regularization for linear classification

The kinds of regularization we have talked about can also give us a new way to think about designing linear classifiers

Goal (ideal): Find $(\mathbf{w}, b)$ minimizing

$$\frac{1}{n}\sum_{i=1}^{n} 1_{\{y_i(\mathbf{w}^T\mathbf{x}_i+b)<0\}}$$

This is actually much harder than it sounds and is not computationally tractable for large problems

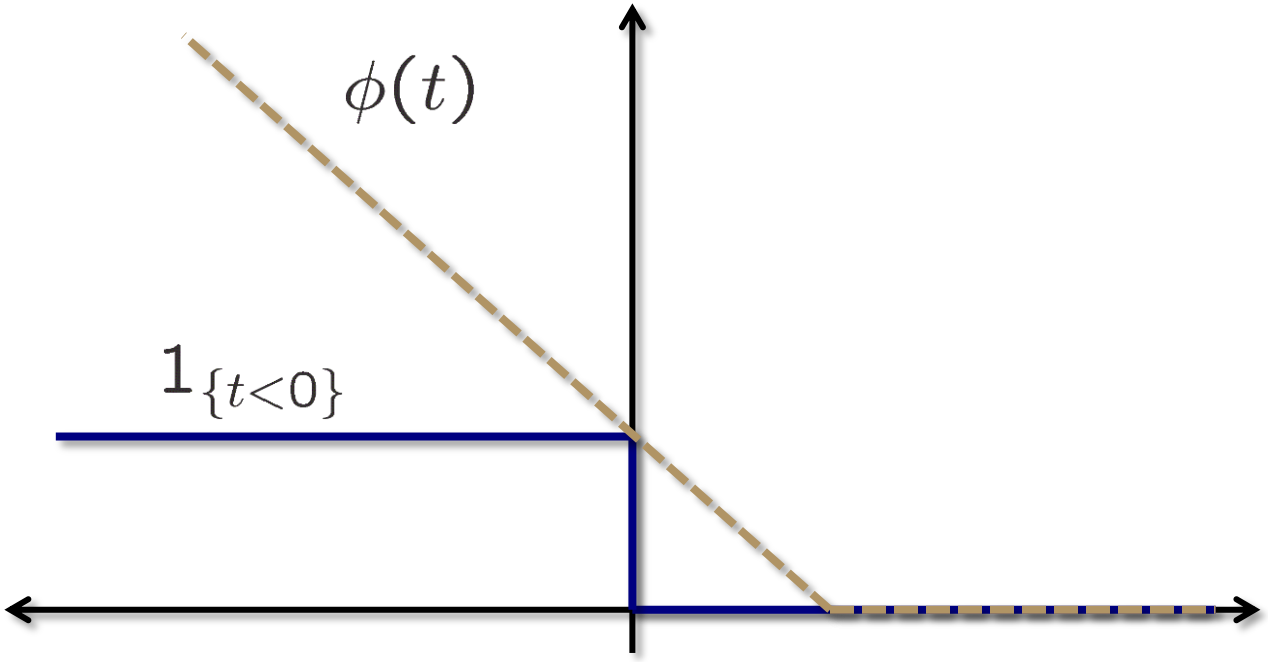
Instead, we can consider replacing this with

$$\frac{1}{n}\sum_{i=1}^{n} \phi(y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

where $\phi(t)$ is some upper bound on $1_{\{t<0\}}$

Hinge loss

Let's take $\phi(t) = \max\{0, 1-t\} =: (1-t)_+$



Adding regularization

Let's try to minimize

$$\frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$$

but to prevent overfitting, let's add a regularization penalty on $\mathbf{w}$

$$\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Soft-margin hyperplane

Compare this to the optimization problem we considered previously for the optimal soft-margin hyperplane:

$$\min_{\mathbf{w},b} \ \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

vs

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \ \ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, n$$

$$\xi_i \geq 0 \quad i = 1, \ldots, n$$

Meet the new boss..

Claim: If $C = \frac{1}{\lambda}$, these two optimization problems are solved by the same $\mathbf{w}, b$

Proof: On board.

Model selection

- The approaches to regularization described today can be made extremely rigorous
- We have theoretical guarantees about when and how well these approaches work under various assumptions
- Analysis is much tighter than for the VC bound
- Unfortunately, these techniques are typically quite sensitive to the choice of the parameter λ

Typically, we must also somehow determine an appropriate value for λ *only using the training data*

The problem of selecting the "free parameters" in a learning algorithm is often called ***model selection***, and is a critical step in most practical learning scenarios

Lecture 11: Kernels

Linear methods for supervised learning

- LDA
- Logistic regression
- Naïve Bayes
- PLA
- Maximum margin hyperplanes
- Soft-margin hyperplanes
- Least squares resgression
- Ridge regression
- …

Nonlinear feature maps

Sometimes linear methods (in both regression and classification) just don't work

One way to create nonlinear estimators or classifiers is to first transform the data via a nonlinear feature map

$$\Phi : \mathbb{R}^d \to \mathbb{R}^p$$

After applying Φ , we can then try applying a linear method to the transformed data $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)$

Regression

In the case of regression, our model becomes

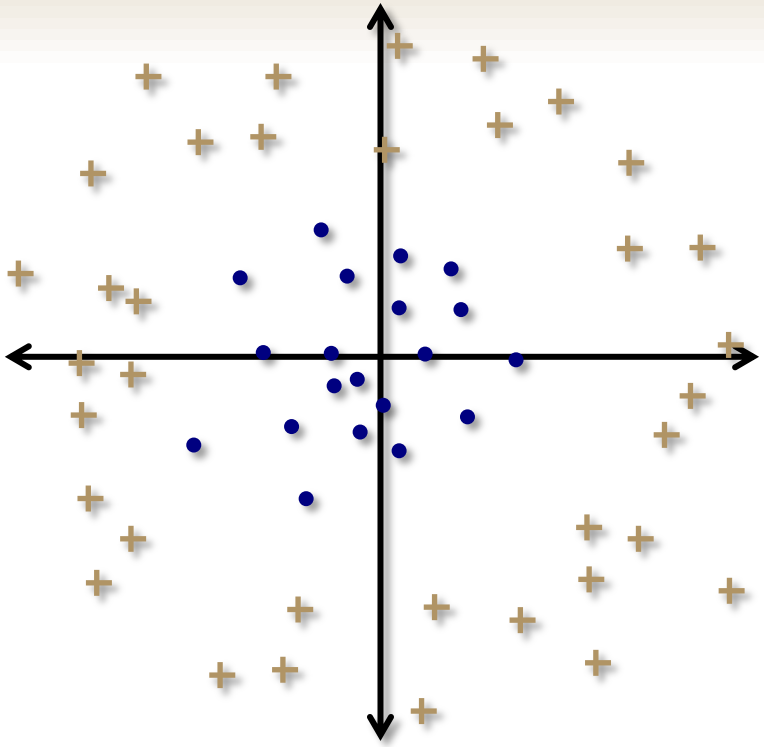
$$f(\mathbf{x}) = \boldsymbol{\beta}^T \Phi(\mathbf{x}) + \beta_0$$

where now $\boldsymbol{\beta} \in \mathbb{R}^p$

Example. Suppose $d = 1$ but $f(x)$ is a cubic polynomial. How do we find a least squares estimate of f from training data?

$$\Phi_k(x) = x^k \quad \Rightarrow \quad \boldsymbol{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

Classification



This data set is not linearly separable

Consider the mapping

$$\Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x(1) \\ x(2) \\ x(1)x(2) \\ x(1)^2 \\ x(2)^2 \end{bmatrix}$$

The dataset ***is*** linearly separable after applying this feature map: $\mathbf{w} = [-1, 0, 0, 0, 1, 1]^T$

Issues with nonlinear feature maps

Suppose we transform our data via

$$\mathbf{x} = \begin{bmatrix} x(1) \\ \vdots \\ x(d) \end{bmatrix} \xrightarrow{\Phi} \Phi(\mathbf{x}) = \begin{bmatrix} \Phi^{(1)}(x) \\ \\ \vdots \\ \\ \Phi^{(p)}(x) \end{bmatrix}$$

where $p \gg d$

- If $p \gtrsim n$, then this can lead to an ill-conditioned problem
  - can be mitigated via regularization
- In addition, however, when p is very large, there can be an increased ***computational*** burden
  - (e.g., inverting a $p \times p$ matrix)

The “kernel trick”

Fortunately, there is a clever way to get around this computational challenge by exploiting two facts:

- Many machine learning algorithms only involve the data through ***inner products***
- For many interesting feature maps Φ, the function

$$k(\mathbf{x}, \mathbf{x}') := \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$$

  has a simple, closed form expression that can be evaluated ***without explicitly calculating*** $\Phi(\mathbf{x})$ and $\Phi(\mathbf{x}')$

Example: Quadratic kernel

Suppose $d = 2$ and consider

$$
\begin{aligned}
k(\mathbf{u}, \mathbf{v}) &= (\mathbf{u}^T \mathbf{v})^2 \\
&= \left([u(1)u(2)] \begin{bmatrix} v(1) \\ v(2) \end{bmatrix} \right)^2 \\
&= (u(1)v(1) + u(2)v(2))^2 \\
&= u(1)^2 v(1)^2 + 2u(1)u(2)v(1)v(2) \\
&\quad + u(2)^2 v(2)^2 \\
&= \langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle
\end{aligned}
$$

$\langle \cdot, \cdot \rangle =$ standard dot product

$$
\Phi(\mathbf{u}) = \begin{bmatrix} u(1)^2 \\ \sqrt{2}u(1)u(2) \\ u(2)^2 \end{bmatrix}
$$

Example: Quadratic kernel

Now suppose d is arbitrary, and

$$
\begin{aligned}
k(\mathbf{u}, \mathbf{v}) &= (\mathbf{u}^T \mathbf{v})^2 \\
&= \left(\sum_{i=1}^{d} u(i) v(i) \right)^2 \\
&= \left(\sum_{i=1}^{d} u(i) v(i) \right) \left(\sum_{j=1}^{d} u(j) v(j) \right) \\
&= \sum_{i=1}^{d} \sum_{j=1}^{d} u(i) v(i) u(j) v(j)
\end{aligned}
$$

Example: Quadratic kernel

$$k(\mathbf{u},\mathbf{v}) = \sum_{i=1}^{d}\sum_{j=1}^{d} u(i)v(i)u(j)v(j) \stackrel{?}{=} \langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle$$

What is Φ and what is the dimension p of the corresponding feature space?

$$\Phi(\mathbf{u}) = \Big[u(1)^2, \ldots, u(d)^2, \ldots \\ \sqrt{2}u(1)u(2), \ldots, \sqrt{2}u(d-1)u(d)\Big]^T$$

$$p = d + \frac{d(d-1)}{2}$$

Example: Cubic kernel

What about $k(\mathbf{u}, \mathbf{v}) = (\mathbf{u}^T\mathbf{v})^3$?

In $\mathbb{R}^2$,

$$\begin{aligned} k(\mathbf{u}, \mathbf{v}) &= (u(1)v(1) + u(2)v(2) + u(3)v(3))^3 \\ &= u(1)^3v(1)^3 + 3u(1)^2u(2)v(1)^2v(2) \\ &\quad + 3u(1)u(2)^2v(1)v(2)^2 + u(2)^3v(2)^3 \\ &= \sum_{i=0}^{3} \binom{3}{i} u(1)^{3-i}u(2)^i \cdot v(1)^{3-i}v(2)^i \end{aligned}$$

$$\Phi(\mathbf{u}) = \left[u(1)^3, \sqrt{3}u(1)^2u(2), \sqrt{3}u(1)u(1)^2, u(2)^3\right]^T$$

Example: Quartic kernel

Polynomial kernels

In general

$$k(\mathbf{u},\mathbf{v}) = (\mathbf{u}^T\mathbf{v})^m$$
$$= \sum_{\substack{\text{partitions} \\ (j_1,\ldots j_m)}} \binom{p}{j_1 \cdots j_m} u(1)^{j_1} v(1)^{j_1} \cdots u(d)^{j_m} v(d)^{j_m}$$

$$\Phi(\mathbf{u}) = \left[\ldots, \sqrt{\binom{p}{j_1 \cdots j_m}} u(1)^{j_1} \cdots u(d)^{j_m}, \ldots\right]^T$$

Or in words, all possible ***monomials*** of degree m

Inner product kernel

Definition. An ***inner product kernel*** is a mapping

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

for which there exists an inner product space $\mathcal{F}$ and a mapping $\Phi : \mathbb{R}^d \to \mathcal{F}$ such that

$$k(\mathbf{u}, \mathbf{v}) = \langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle_{\mathcal{F}}$$

for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$

Given a function $k(\mathbf{u}, \mathbf{v})$, how can we tell when it is an inner product kernel?

- Mercer's theorem
- Positive semidefinite property

Positive semidefinite kernels

We say that $k(\mathbf{u}, \mathbf{v})$ is a ***positive semidefinite*** kernel if

- k is symmetric
- for all n and all $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, the ***Gram matrix*** $\mathbf{K}$ defined by

$$K(i, j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

  is positive semidefinite, i.e., $\mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0$ for all $\mathbf{x}$

Theorem

k is an inner product kernel if and only if k is a positive semidefinite kernel

Proof: See homework.

Examples: Polynomial kernels

Homogeneous polynomial kernel

$$k(\mathbf{u}, \mathbf{v}) = (\mathbf{u}^T \mathbf{v})^m \qquad m = 1, 2, \ldots$$

Inhomogenous polynomial kernel

$$k(\mathbf{u}, \mathbf{v}) = (\mathbf{u}^T \mathbf{v} + c)^m \qquad m = 1, 2, \ldots$$
$$c > 0$$

Φ maps to the set of all monomials of degree $\leq m$

Examples: Gaussian/RBF kernels

Gaussian / Radial basis function (RBF) kernel:

$$k(\mathbf{u}, \mathbf{v}) = (2\pi\sigma^2)^{-d/2} \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right)$$

$$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right)$$

One can show that k is a positive semidefinite kernel, but what is $\mathcal{F}$?

$\mathcal{F}$ is ***infinite dimensional***!

Ridge regression revisited

Ridge regression: Given $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$

$$(\widehat{\boldsymbol{\beta}}, \widehat{\beta}_0) = \arg\min_{\boldsymbol{\beta}, \beta_0} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0)^2 + \lambda \|\boldsymbol{\beta}\|^2$$

Solution: $$\frac{\partial}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0) = 0$$

$$\widehat{\beta}_0 = \frac{1}{n} \sum_{i=1}^{n} y_i - \widehat{\boldsymbol{\beta}}^T \mathbf{x}_i$$

$$= \bar{y} - \widehat{\boldsymbol{\beta}}^T \bar{\mathbf{x}}$$

$$\bar{y} = \frac{1}{n} \sum_i y_i$$

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_i \mathbf{x}_i$$

Ridge regression revisited

Plugging this back in we are left to minimize

$$\sum_{i=1}^{n}(y_i - \bar{y} - \boldsymbol{\beta}^T(\mathbf{x}_i - \bar{\mathbf{x}})^2 + \lambda\|\boldsymbol{\beta}\|^2$$

with respect to $\boldsymbol{\beta}$

$$\widehat{\boldsymbol{\beta}} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\widetilde{\mathbf{y}}$$

$$\mathbf{A} = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix} \qquad \widetilde{\mathbf{y}} = \begin{bmatrix} y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{bmatrix}$$

Kernel ridge regression

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\widetilde{\mathbf{y}}$$

$$\mathbf{A} = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix} \qquad \widetilde{\mathbf{y}} = \begin{bmatrix} y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{bmatrix}$$

$$\widehat{f}(\mathbf{x}) = \bar{y} + \widehat{\beta}^T(\mathbf{x} - \bar{\mathbf{x}})$$

Can we express ridge regression in terms of inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ and $\langle \mathbf{x}_i, \mathbf{x} \rangle$?

Not immediately. $[\boldsymbol{A}^T\boldsymbol{A}](i,j) \neq \mathbf{x}_i^T\mathbf{x}_j$

Woodbury matrix inversion identity

$$(\mathbf{P}+\mathbf{QRS})^{-1}=\mathbf{P}^{-1}-\mathbf{P}^{-1}\mathbf{Q}(\mathbf{R}^{-1}+\mathbf{SP}^{-1}\mathbf{Q}))^{-1}\mathbf{SP}^{-1}$$

$$\mathbf{P}=\lambda\mathbf{I}\quad\mathbf{Q}=\mathbf{A}^T\quad\mathbf{R}=\mathbf{I}\quad\mathbf{S}=\mathbf{A}$$

$$\begin{aligned}(\lambda\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}&=\frac{1}{\lambda}\mathbf{I}-\frac{1}{\lambda}\mathbf{I}\mathbf{A}^T\left(\mathbf{I}+\frac{1}{\lambda}\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\frac{1}{\lambda}\\&=\frac{1}{\lambda}\left[\mathbf{I}-\mathbf{A}^T(\lambda\mathbf{I}+\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\right]\end{aligned}$$

$$(\lambda\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\tilde{\mathbf{y}}=\frac{1}{\lambda}\left[\mathbf{A}^T-\mathbf{A}^T(\lambda\mathbf{I}+\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{A}^T\right]\tilde{\mathbf{y}}$$

Kernelizing ridge regression

$$(\lambda \mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\widetilde{\mathbf{y}} = \frac{1}{\lambda}\left[\mathbf{A}^T - \mathbf{A}^T(\lambda\mathbf{I} + \mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{A}^T\right]\widetilde{\mathbf{y}}$$

$$\begin{aligned} K(i,j) &= (\mathbf{x}_i - \bar{\mathbf{x}})^T(\mathbf{x}_j - \bar{\mathbf{x}}) \\ &= \mathbf{x}_i^T\mathbf{x}_j - \frac{1}{n}\sum_{r=1}^{n}\mathbf{x}_i^T\mathbf{x}_r - \frac{1}{n}\sum_{s=1}^{n}\mathbf{x}_s^T\mathbf{x}_j + \frac{1}{n^2}\sum_{r,s=1}^{n}\mathbf{x}_r^T\mathbf{x}_s \end{aligned}$$

$$(\lambda \mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\widetilde{\mathbf{y}} = \frac{1}{\lambda}\left[\mathbf{A}^T - \mathbf{A}^T(\lambda\mathbf{I} + \mathbf{K})^{-1}\mathbf{K}\right]\widetilde{\mathbf{y}}$$

Kernelizing ridge regression

What about the remaining $\mathbf{A}^T$?

$$\begin{aligned}\widehat{f}(\mathbf{x}) &= \bar{y} + \widehat{\boldsymbol{\beta}}^T(\mathbf{x} - \bar{\mathbf{x}}) \\ &= \bar{\mathbf{y}} + \frac{1}{\lambda}\tilde{\mathbf{y}}^T\left[\mathbf{A} - \mathbf{K}(\lambda\mathbf{I} + \mathbf{K})^{-1}\mathbf{A}\right](\mathbf{x} - \bar{\mathbf{x}}) \\ &= \bar{\mathbf{y}} + \frac{1}{\lambda}\tilde{\mathbf{y}}^T\left[\mathbf{I} - \mathbf{K}(\lambda\mathbf{I} + \mathbf{K})^{-1}\mathbf{I}\right]\mathbf{k}(\mathbf{x})\end{aligned}$$

where $\mathbf{k}(\mathbf{x}) = \begin{bmatrix}(\mathbf{x}_1 - \bar{\mathbf{x}})^T(\mathbf{x} - \bar{\mathbf{x}}) \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T(\mathbf{x} - \bar{\mathbf{x}})\end{bmatrix}$

$$[k(\mathbf{x})](i) = \mathbf{x}_i^T\mathbf{x} - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_i^T\mathbf{x}_j - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}^T\mathbf{x}_j + \frac{1}{n^2}\sum_{j,k=1}^{n}\mathbf{x}_j^T\mathbf{x}_k$$

Homogenous kernel ridge regression

For many kernels, $\Phi(\mathbf{x})$ already contains a constant component, in which case we often omit β_0

Examples

- inhomogenous polynomial kernel
- Gaussian kernel does not seem to require a constant

In this case, the kernel ridge regression solution becomes

$$\widehat{f}(\mathbf{x}) = \mathbf{y}^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$$

where

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Example: Gaussian kernel

$$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right)$$

If we omit β_0, then $\widehat{f}(\mathbf{x}) = \mathbf{y}^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$

$$= \boldsymbol{\alpha}^T\mathbf{k}(\mathbf{x}) = \sum_{i=1}^{n} \alpha(i)k(\mathbf{x}, \mathbf{x}_i)$$