FSAN/ELEG815: Statistical Learning

Gonzalo R. Arce

Department of Electrical and Computer Engineering
University of Delaware

4. The Learning Problem

Outline of the Course

1. Review of Probability
2. Stationary processes
3. Eigen Analysis, Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)
4. The Learning Problem
5. Training vs Testing
6. Estimation theory: Maximum likelihood and Bayes estimation
7. The Wiener Filter
8. Adaptive Optimization: Steepest descent and the LMS algorithm
9. Least Squares (LS) and Recursive Least Squares (RLS) algorithm
10. Overfitting
11. Regularization: Ridge and Lasso regression models.
12. Neural Networks
13. Matrix Completion

The Learning Problem - Outline

- Example of machine learning
- Components of Learning
- A simple model
- Types of learning
- Puzzle

NETFLIX

Example: Predicting How a Viewer Will Rate a Movie

10 % improvement = 1 millon dollar prize

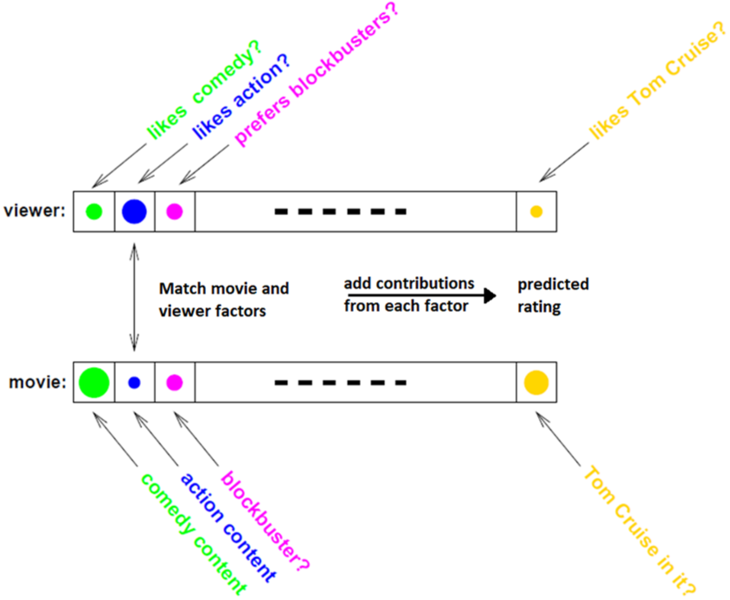
The essence of machine learning:

- A pattern exists.
- We cannot pin it down mathematically.
- We have data on it.

A pattern exists. We don't know it. We have data to learn it.

Movie Rating

- Describe the movie as an array of factors
- Describe each viewer using same factors
- Rating based on match/mismatch
- More factors → better prediction

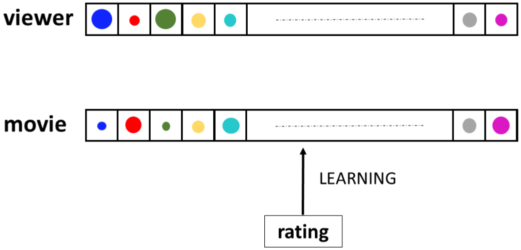


A model for movie rating

The Learning Approach

The learning algorithm does **reverse-engineering** (estimates factors from a given rating).

- Starts with random (meaningless) factors
- Tunes factors to be aligned with a previous rating.
- Does the same for millions of ratings, cycling over and over.
- Eventually the factors are meaningful (complete).



Components of Learning

Metaphor: Credit approval

Applicant information:

age	23 years
gender	male
annual salary	$30,000
years in residence	1 year
years in job	1 year
current debt	$15,000
...	...

Approve credit?

Components of Learning

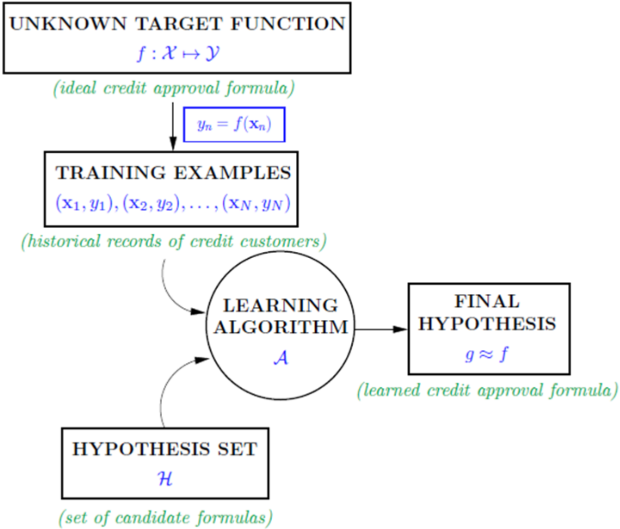
Formalization

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1)$,$(\mathbf{x}_2, y_2)$, $\cdots$,$(\mathbf{x}_N, y_N)$ (*historical records*)

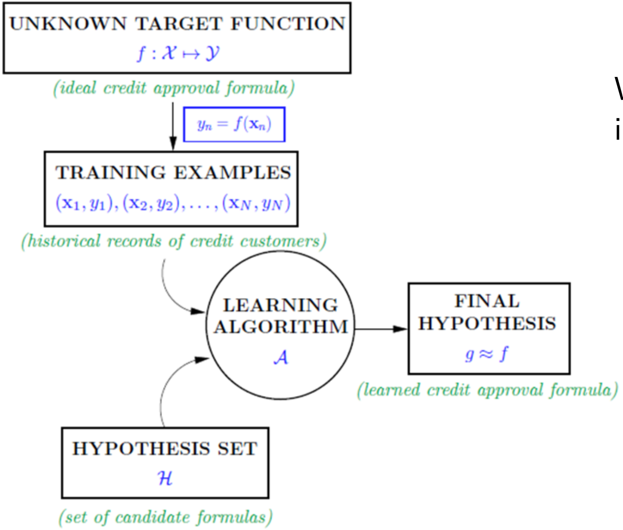
  $\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \rightarrow \mathcal{Y}$ (*formula to be used*)

The goal of learning is: $g \approx f$


UNKNOWN TARGET FUNCTION
$f : \mathcal{X} \mapsto \mathcal{Y}$
(ideal credit approval formula)
$y_n = f(\mathbf{x}_n)$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(learned credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)

- Learning algorithm chooses g from $\mathcal{H}$ that best matches the target f on the *training* examples.
- Hopefully, g will match f on new costumers.


UNKNOWN TARGET FUNCTION
$f : \mathcal{X} \mapsto \mathcal{Y}$
(ideal credit approval formula)
$y_n = f(\mathbf{x}_n)$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(learned credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)

Why choose from a hypothesis set instead of anything?

- No downside (always choose from: Linear formulas, Support Vector Machines, Neural Networks, all of them...).
- Upside: It will allow us to tell how well we can learn.

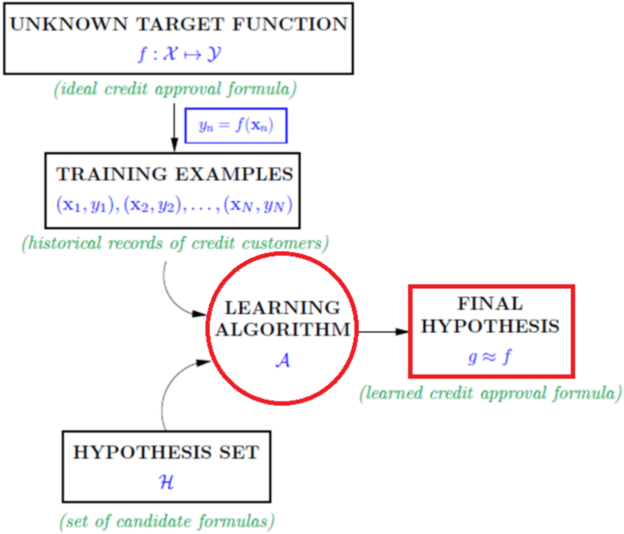
Solution Components

Two solution components of the learning problem:

- The Hypothesis Set or Model
  $\mathcal{H} = \{h\} \quad g \in \mathcal{H}$
- The Learning Algorithm

Together, they are referred to as the *learning model*.

You choose them to solve the problem.


UNKNOWN TARGET FUNCTION
$f : \mathcal{X} \mapsto \mathcal{Y}$
(ideal credit approval formula)
$y_n = f(\mathbf{x}_n)$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(learned credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)

Example: Advertising

Data: sales of a product in 200 different markets, with advertising budgets in: TV, radio, and newspapers.
Goal: predict sales on the basis of the three media budgets.

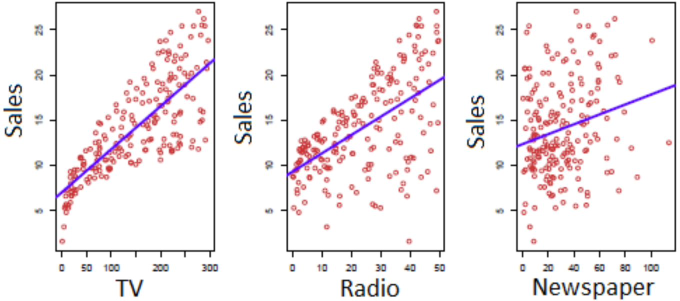
Advertising budgets are input variables:

- x_1: TV budget
- x_2: Radio budget
- x_3: Newspaper budget

The output variable, y: Sales.

Example: Advertising

Display sales as a function of TV, radio, and newspaper budgets. Blue line is a model to predict sales:



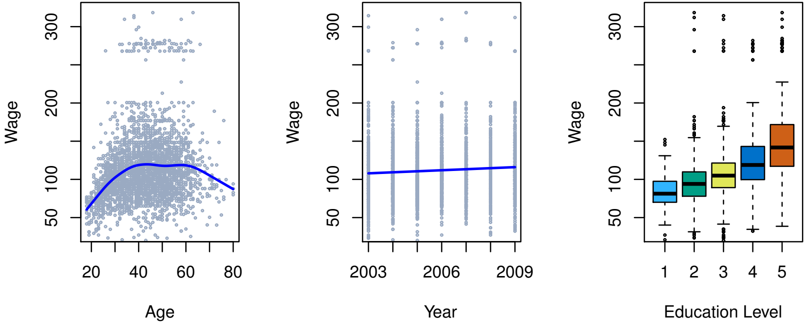
Suppose we observe a response y and d different predictors, $\mathbf{x} = \{x_1, ..., x_d\}$. We can write the form:

$$y = f(\mathbf{x}) + \epsilon$$

where ϵ is a random error term, which is independent of $\mathbf{x}$ and has mean zero.

Example: Wage Data (Atlantic US)

Model wages in relation to education, age, and year.

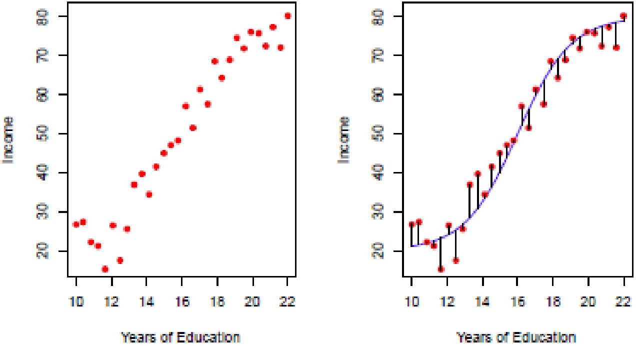


What conclusion do you see from each?
Prediction on wage is based on age, education and year.

Motivation

$$y = f(\mathbf{x}) + \epsilon$$

Example: Plot of income versus years of education for 30 individuals:



Learning estimates f.

Why Estimate f

- Prediction

  Since the error term averages to zero, we can predict y using

  $$\hat{y} = g(\mathbf{x})$$

  where g is our estimate for f, and $\hat{y}$ is our estimate for y.
  The expected value of the squared difference between the predicted and actual value of y:

  $$\begin{aligned} \mathbb{E}[y - \hat{y}]^2 &= \mathbb{E}[f(\mathbf{x}) + \epsilon - g(\mathbf{x})]^2 \\ &= [f(\mathbf{x}) - g(\mathbf{x})]^2 + \text{var}[\epsilon] \end{aligned}$$

  where the first term is called *Reducible error*, and the second term is *Irreducible error*.

Why Estimate f

Inference:

Instead of predicting y, we are interested in the relationship between y and $\mathbf{x}$:

- Which predictors are important?
- What is the relationship between the response and each predictor?
- Can the relationship be summarized by a linear equation or is it more complicated?

Different methods for estimating f may be appropriate:

- Linear models are simple and interpretable, but not always accurate.
- Non-linear approaches provide accurate predictions for y, but less interpretable.

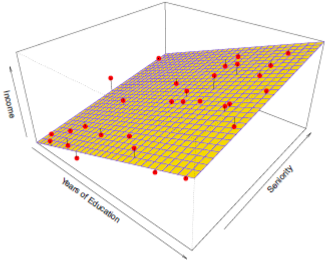
How to Estimate f

- Parametric

  For example:

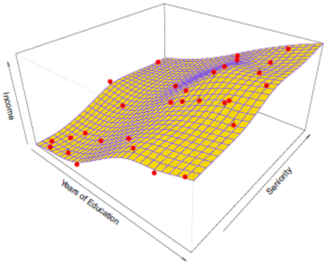
  $$g(\mathbf{x}) = w_0 + w_1 x_1 +, ..., + w_d x_d$$

  Estimates f by estimating a set of parameters.



How to Estimate f

- Non-parametric
  Estimate f getting as close to the data points as possible. Avoids functional form for f.
  For example, a thin-plate spline is used:



Disadvantages:

- Large number of observations are needed.
- Overfitting data.

A Simple Hypothesis Set - The 'Perceptron'

Metaphor: Credit approval

Input: $\mathbf{x} = x_1, \cdots, x_d$ 'attributes of a customer'

$$\text{Approve credit if } \sum_{i=1}^{d} w_i x_i > \text{ threshold}$$

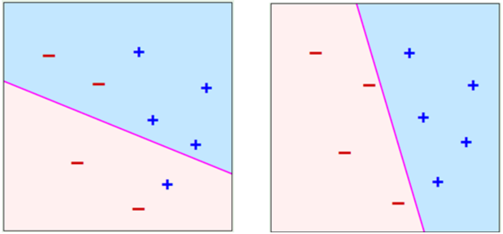
$$\text{Deny credit if } \sum_{i=1}^{d} w_i x_i < \text{ threshold}$$

This linear formula $h \in \mathcal{H}$ can be written as

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{ threshold}\right)$$

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right)$$

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if 'approved'} \\ -1 & \text{if 'deny credit'} \end{cases}$$



(a) Misclassified data (b) Perfectly classified data

Linearly separable data in two-dimensions. Line ($w_1x_1 + w_2x_2 -$ theshold $= 0$) splits the plane into: $+1$ and -1 decision regions.

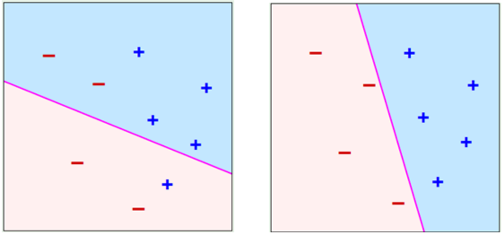
$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + w_0\right)$$

where $w_0 = -$threshold. Introduce an artificial coordinate $x_0 = 1$:

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right)$$

In vector form, the perceptron is

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



(a) Misclassified data (b) Perfectly classified data

Linearly separable data in two-dimensions. Line $(w_0 + w_1x_1 + w_2x_2 = 0)$ splits the plane into: $+1$ and -1 decision region.

The Perceptron Learning Algorithm (PLA)

Simple learning algorithm that determines **w** based on data.

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

Given the training set:

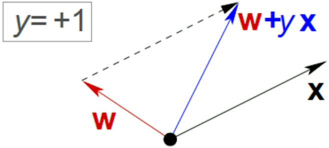
$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

pick a misclassified point:

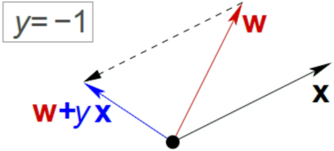
$$\text{sign}(\mathbf{w}^T\mathbf{x}) \neq y_n$$

and update the weight vector:

$$\mathbf{w} \leftarrow \mathbf{w} + y_n\mathbf{x}_n$$



Misclassified point: $\text{sign}(\mathbf{w}^T\mathbf{x}) = -1$



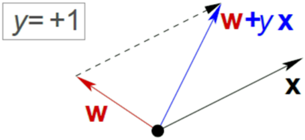
Misclassified point: $\text{sign}(\mathbf{w}^T\mathbf{x}) = 1$

The Perceptron Learning Algorithm (PLA)

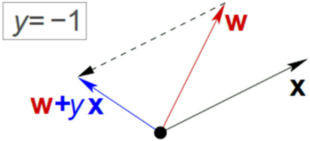
The update adds or subtracts a vector $\mathbf{x}_n$ depending on $y_n \in \{-1, 1\}$

$$\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n$$

making the point more likely to be correctly classified.



$\text{sign}(\mathbf{w}^T\mathbf{x}) = -1$. Iteration: $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}$ such that $\text{sign}(\mathbf{w}^T\mathbf{x}) = 1$



$\text{sign}(\mathbf{w}^T\mathbf{x}) = 1$. Iteration: $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}$ such that $\text{sign}(\mathbf{w}^T\mathbf{x}) = -1$

Iterations of PLA

- One iteration of the PLA:

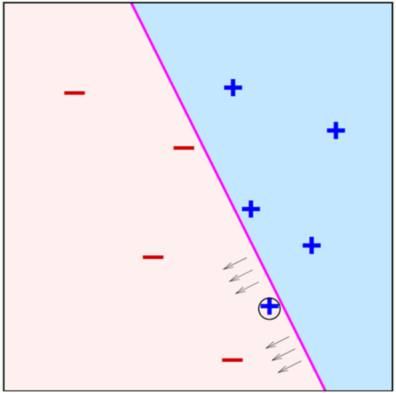
$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y(t)\mathbf{x}(t)$$

  where ($\mathbf{x}$(t),y(t)) is a misclassified training point.

- At iteration $t = 1, 2, 3, \cdots$, pick a misclassified point from

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

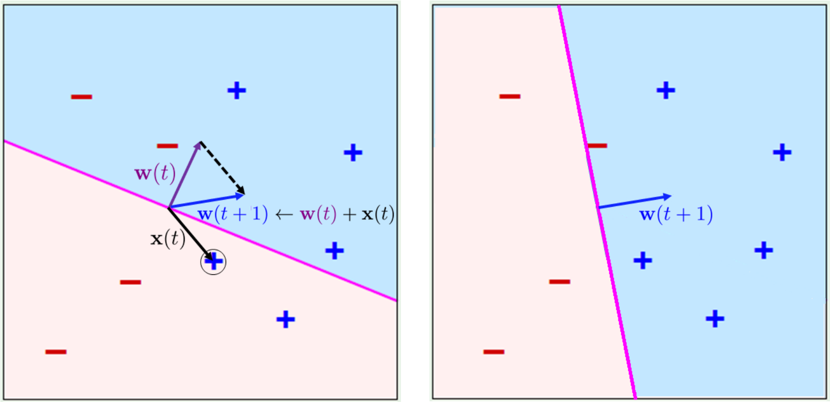
  and run a PLA iteration on it.



Iteration rule moves the boundary ($w_0 + w_1x_1 + w_2x_2 = 0$) to correctly classify the misclassified point ⊕

Example: One iteration of the PLA when the picked misclassified point is $(\mathbf{x}, +1)$

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y(t)\mathbf{x}(t) = \mathbf{w}(t) + \mathbf{x}(t)$$



Iteration rule moves the boundary to correctly classify the misclassified point

Stops when there is no more misclassified points

The Learning Problem - Outline

- Example of machine learning
- Components of learning
- A simple model
- **Types of learning**
- Puzzle

Basic Premise of Learning

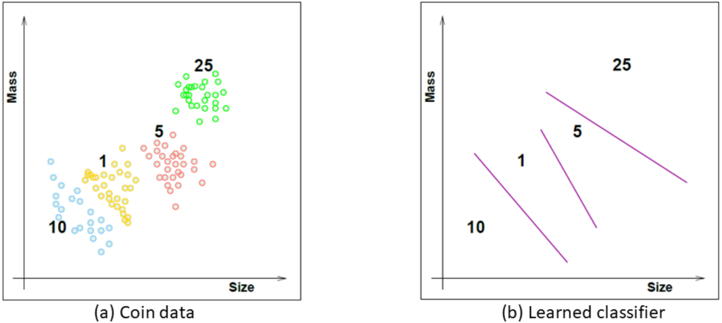
"observations to uncover an underlying process"

broad premise → many variations

- **Supervised Learning**
- **Unsupervised Learning**
- **Reinforcement Learning**

Supervised Learning

Example from vending machines - **coin recognition**



(a) Coin data

(b) Learned classifier

(a) Training data: Mass-Size of pennies, nickles, dimes and quarters (1,5,10, and 25 cents)

(b) A classification rule is learned from the data (Linear Model).

Correct output is specified by colors → Supervised

The Classification Setting

Estimate f based on training $\{(x_0, y_0), \cdots, (x_n, y_n)\}$
Training error rate:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

with $I(y_i \neq \hat{y}_i) = \begin{cases} 1 & \text{if } y_i \neq \hat{y}_i \\ 0 & \text{if } y_i = \hat{y}_i. \end{cases}.$

Here, $\hat{y}_i$ is the predicted class label for the ith observation using $\hat{f}$.
Test error rate:

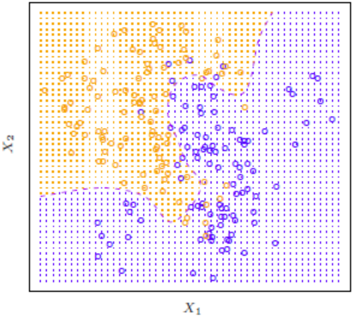
$$Ave(I(y_0 \neq \hat{y}_0))$$

A good classifier is one for which the test error is smallest.

The Bayes Classifier

The test error rate is minimized, on average, by the Bayes classifier that assigns each observation to the most likely class, given its predictor values. That is assign x_0 to class j for which

$$Pr(Y = j|X = x_0) \qquad \text{is largest.}$$



The purple dashed line: *Bayes decision boundary* for two groups.

Bayes classifier produces the lowest possible test error rate

$$1 - E\left(\max_j Pr(Y = j|\mathbf{X})\right)$$

For previous example, this value is 0.1304.

- For real data, the conditional distribution of Y given X is not known, and so computing the Bayes classifier is impossible.
- Bayes classifier serves as an unattainable gold standard.
- Many approaches attempt to estimate the conditional distribution of Y given X.

K-Nearest Neighbors Classifier

Given a positive integer K and a test observation $\mathbf{x}_0$, the KNN classifier:

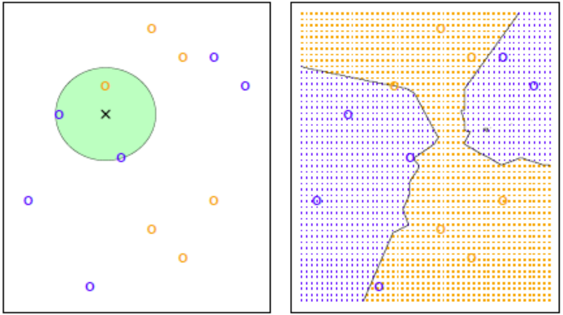
- Identifies the K points in the training data that are closest to $\mathbf{x}_0$, represented by $\mathcal{N}_0$.
- Estimate the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$Pr(Y = j|\mathbf{X} = \mathbf{x}_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = i).$$

- Applies Bayes rule and classifies the test observation $\mathbf{x}_0$ to the class with the largest probability.

K-Nearest Neighbors Classifier

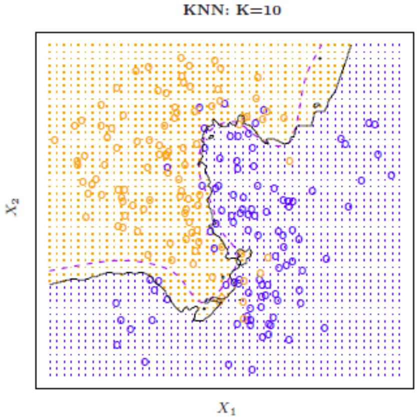
Example for $K = 3$:



Since $Pr(Y = blue|X) = 2/3$, KNN will predict the point to be blue. We do this for all possible value for X_1 and X_2 to form the right plot.

K-Nearest Neighbors Classifier

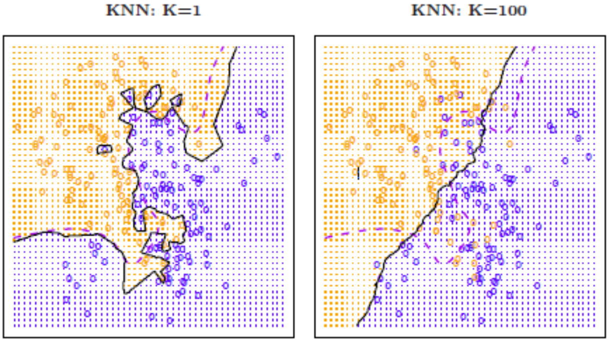
Example with $K = 10$ in 100 observations. Black curve indicates the KNN decision boundary using $K = 10$. The Bayes decision boundary is a purple dashed line.



Test error rate for KNN is 0.1363, and Bayes error rate is 0.1304.

K-Nearest Neighbors Classifier

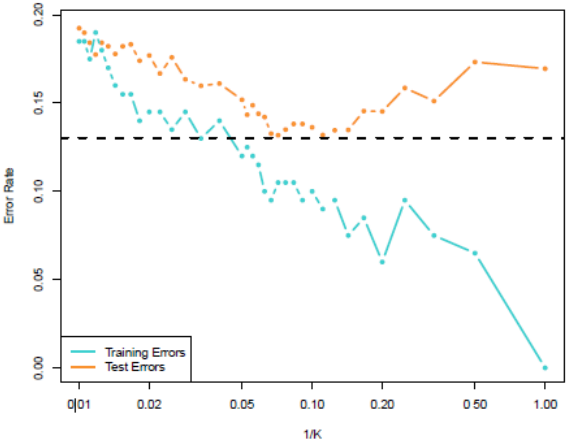
Choice of K



As K grows, KNN becomes less flexible and decision boundary is close to linear: low-variance but high-bias classifier.

K-Nearest Neighbors Classifier

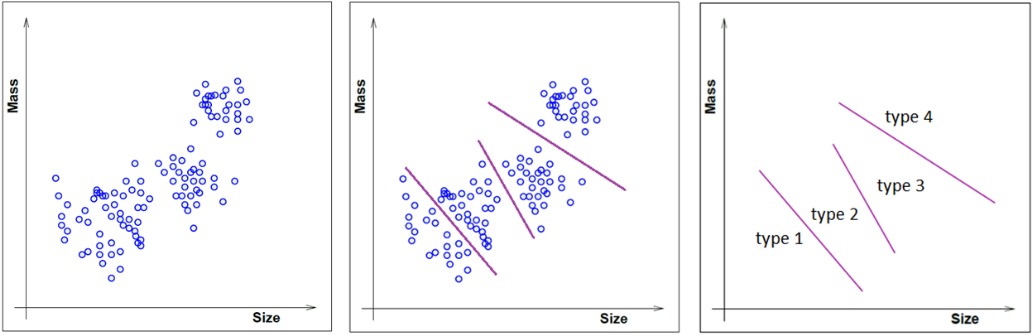
KNN test and training errors as a function of $1/K$. As 1/K increases, the method becomes more flexible.



Observe that U-shape in the test error rate.

Unsupervised Learning

Instead of **(input, correct output)**, we get **(input, ?)**

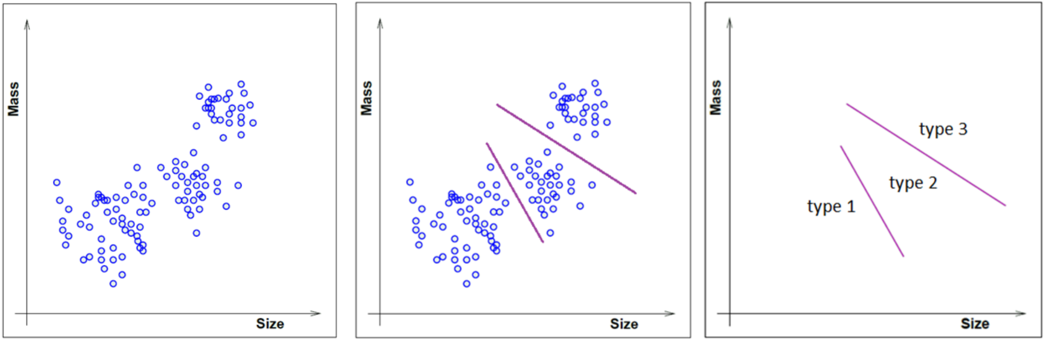


(a) Unlabled Coin Data (b) Clustering (c) Unsupervised learning

Finds patterns and structure in input data

Unsupervised Learning

Instead of **(input, correct output)**, we get **(input, ?)**



(a) Unable Coin Data

(b) Clustering

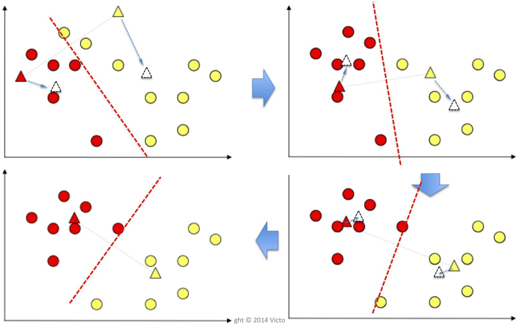
(c) Unsupervised learning

The number of clusters is ambiguous!

Classification using k-means clustering

- Partition of a set $\mathcal{X}$, of observations into a specified number, k, of clusters.
- Assign to the cluster with the nearest mean.
- Iterative procedure.

K-means clustering example



ght © 2014 Victo

Classification using k-means clustering

- Let $\mathbf{Z}$ be the dataset of the form

$$\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{|\mathcal{X}|}\}$$

where $\mathbf{z} \in R^n$

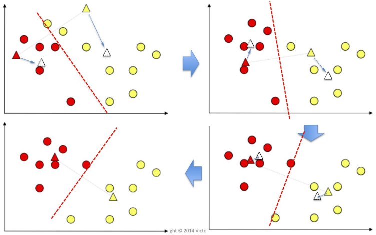
- We want to classify the data into k disjoint sets of the form

$$C = \{C_1, C2, ..., C_k\}$$

such that the criterion of optimality is satisfied

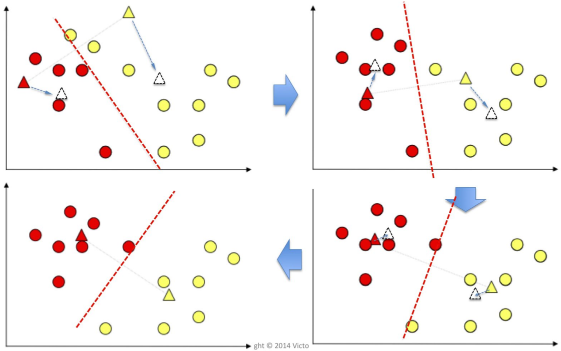
$$\underset{C}{\arg\min} = \left(\sum_{i=1}^{k} \sum_{\mathbf{z} \in C_i} \|\mathbf{z} - \mathbf{m}_i\|^2 \right)$$

K-means clustering example



k-means Algorithm

K-means clustering example



- First, k initial means are generated at random: $\mathbf{m}_i(1)$, $i = 1, 2, ..., k$
- Assign samples to clusters whose mean is the closest.
- Update the clusters' means

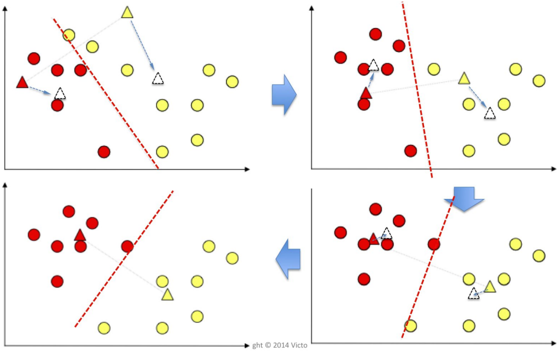
$$\mathbf{m}_i(t) = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \quad i = 1, 2, ..., k$$

where $|C_i|$ is the number of samples in cluster set C_i.

k-means Algorithm

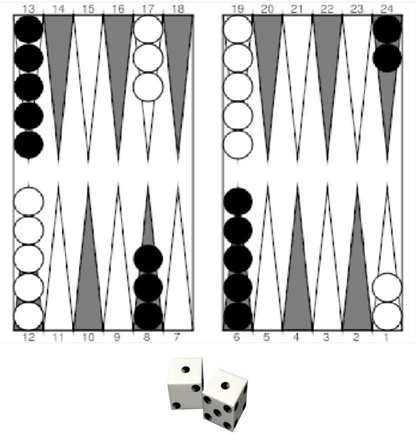
- Compute residual error, E, as the sum of the k Euclidean norms of the differences between the mean vectors in the current and previous steps. Stop if $E \leq T$, where T is a specified threshold.

K-means clustering example



Reinforcement Learning

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

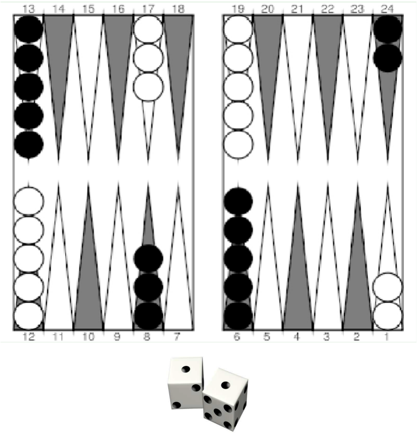


target function → best action given a state

Not a trivial task!

Reinforcement Learning

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

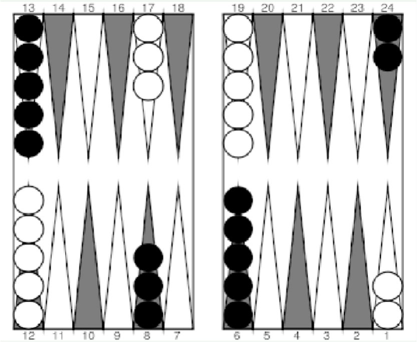


Reinforcement learning...

- Takes some action and reports how well things went.
- Repeats many times.
- Sorts out the information from different examples.

Reinforcement Learning

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



Eventually learns the best line of play.

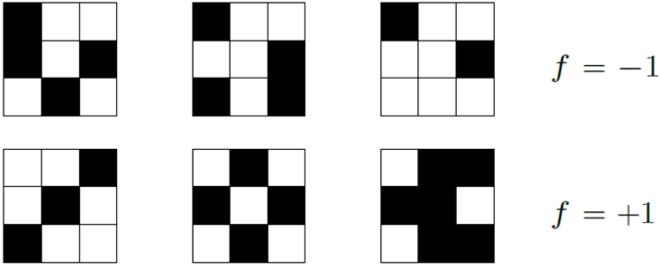
The world champion was a neural network!

A Learning Puzzle

Training examples ($\mathbf{x}$): 9-bit vector represented as a 3×3 black&white array.

First row: $f(\mathbf{x}) = -1$

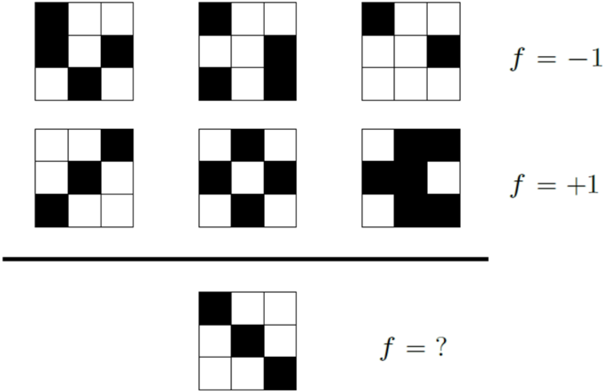
Second row: $f(\mathbf{x}) = +1$



A Learning Puzzle

Task:

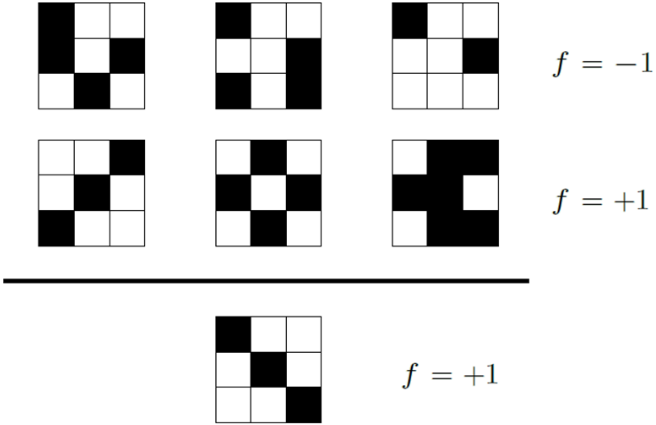
- Learn from data what f is
- Apply f to test input at the bottom. Do you get $+1$ or -1?



A Learning Puzzle

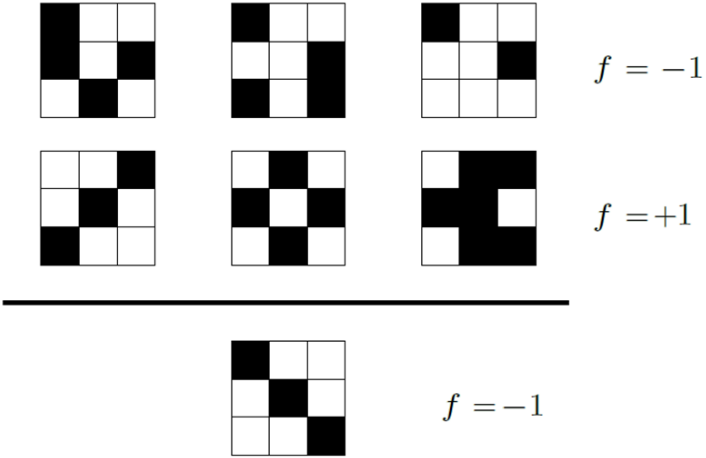
There is more than one possible solution:

- if $f(\mathbf{x}) = +1$ when pattern is symmetric:



A Learning Puzzle

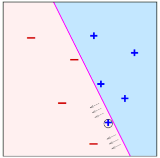
- if $f(\mathbf{x}) = +1$ when the top left square is white:



Review

- **Learning is used when**
  - A pattern exists
  - We cannot pin it down mathematically
  - We have data on it
- **Focus on supervised learning**
  - Unknown target function $y = f(\mathbf{x})$
  - Data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$
  - Learning algorithm picks $g \approx f$ from a hypothesis set $\mathcal{H}$

Example: Perceptron Learning Algorithm



- **Learning an unknown function?**
  - Impossible!. The function can assume any value outside the data we have.
  - So what now?

Feasibility of Learning - Outline

- Probability to the rescue
- Connection to learning
- Connection to *real* learning
- A dilemma and a solution
- Errors and noise

Goal: Showing that we can infer something outside the data set $\mathcal{D}$.

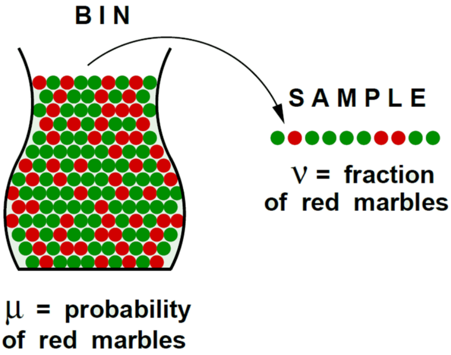
A Related Experiment

- Consider a 'bin' with **red** and **green** marbles.

$$\mathbb{P}[\text{picking a red marble}] = \mu$$

$$\mathbb{P}[\text{picking a green marble}] = 1 - \mu$$

- The value of μ is unknown to us.
- Pick a random sample of N independent marbles
- We know the fraction of red marbles in sample ν



Does ν say anything about μ?

No!

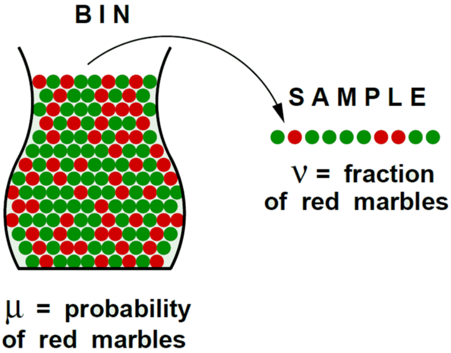
Sample can be mostly green while bin is mostly red.

Although this is possible, it is not probable.

Yes!

Sample frequency ν is likely close to bin frequency μ when N large enough

possible versus **probable**



What does ν say about μ?

In a big sample (large N), ν is probably close to μ (within ϵ) i.e. $\mathbb{P}[|\nu - \mu| > \epsilon]$ is very small.

Formally, we can bound this probability by:

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

Hoeffding's Inequality.

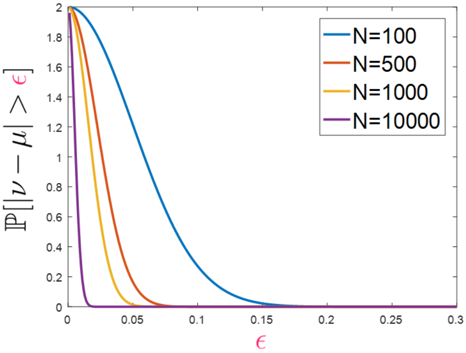
As the sample size N grows, it is exponentially unlikely that ν will deviate from μ by more than our 'tolerance' ϵ.

The statement "$\mu = \nu$" is Probably and Approximately Correct (P.A.C).

What does ν say about μ?

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

Hoeffding's Inequality.

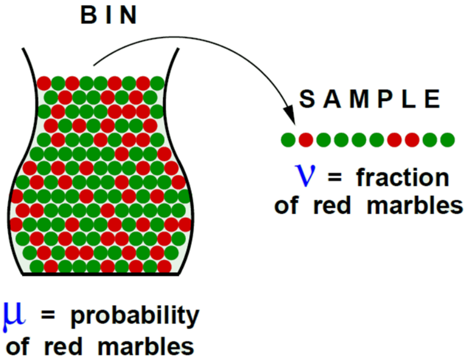


Notice that if ϵ very small (better approximation), larger sample N is needed to make the RHS of Inequality small.

Hoeffding's Inequality

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

- Valid for all N and ϵ
- Bound does not depend on μ
- Tradeoff: N, ϵ, and the bound
- $\nu \approx \mu \implies \mu \approx \nu$
  - ν tends to be close to μ.
  - We infer that μ 'tends' to be close to ν.



Connection to Learning

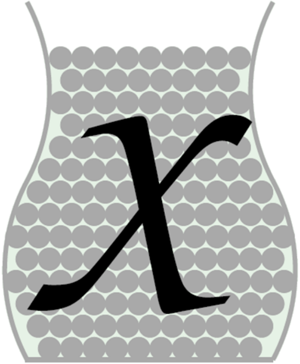
Bin: The unknown is a number μ

Learning: The unknown is a function $f : \mathcal{X} \to \mathcal{Y}$

How does the bin model relate to the learning problem?

Let's assume each marble ● is a point $\mathbf{x} \in \mathcal{X}$

E.g. each marble is a credit card applicant.



Connection to Learning

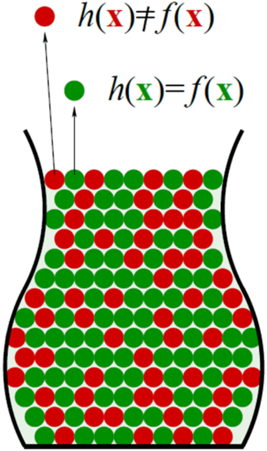
Now, let's relate the colors:

Taking any single hypothesis $h \in \mathcal{H}$ and comparing it to f on each point $\mathbf{x} \in \mathcal{X}$:

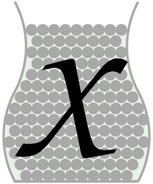
●: Hypothesis got it right $h(\mathbf{x}) = f(\mathbf{x})$

●: Hypothesis got it wrong $h(\mathbf{x}) = f(\mathbf{x})$

Color of each point is **unknown** since f is **unknown** but now there is a probability associated.


$h(\mathbf{x}) \neq f(\mathbf{x})$
$h(\mathbf{x}) = f(\mathbf{x})$

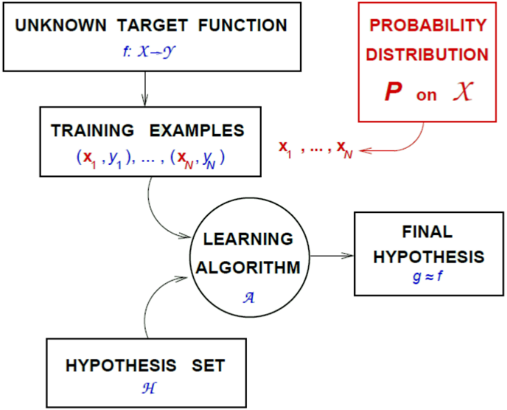
Back to the Learning Diagram



We introduce a new component:

A probability distribution P on $\mathcal{X}$.

- P is any probability distribution.
- No need to know P



UNKNOWN TARGET FUNCTION

$f: \mathcal{X} \to \mathcal{Y}$

PROBABILITY DISTRIBUTION

P on $\mathcal{X}$

TRAINING EXAMPLES

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

$\mathbf{x}_1, \ldots, \mathbf{x}_N$

LEARNING ALGORITHM

$\mathcal{A}$

FINAL HYPOTHESIS

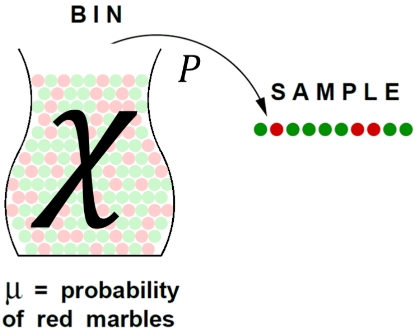
$g \approx f$

HYPOTHESIS SET

$\mathcal{H}$

Connection to Learning

- Pick $\mathbf{x}$ at random according to P
- Each point will be red with some probability μ.
- The space $\mathcal{X}$ behaves like the bin
- Training examples play the role of a sample from the bin
- Color of each sample point will be known to us $\implies$ we know ν
  - $h(\mathbf{x}_n)$ and $f(\mathbf{x}_n)$ are known in the data set $\mathcal{D}$.



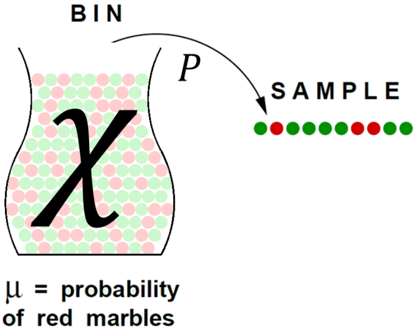
Connection to Learning

- Any P will translate to μ (unknown).

Thus, Hoeffding Inequality can be applied.
Predict outside $\mathcal{D}$, using ν to predict μ.

μ will tell us the error rate h makes in approximating f.

- $\nu \approx 0 \rightarrow h \approx f$ over all $\mathcal{X}$



Are we Done?

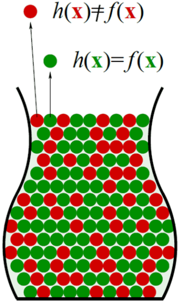
No! h is fixed before we generate the data sets $\mathcal{D}$

For this h, ν generalizes to μ

This is '**verification**' of h, not **learning**

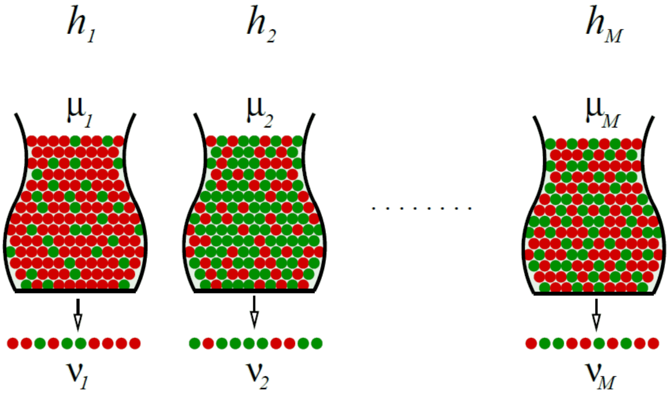
No guarantee ν will be small.

We need to **choose** from multiple h's.


$h(\mathbf{x}) \neq f(\mathbf{x})$
$h(\mathbf{x}) = f(\mathbf{x})$

Multiple Bins

Generalizing the bin model to more than one hypothesis:



Back to the Learning Diagram

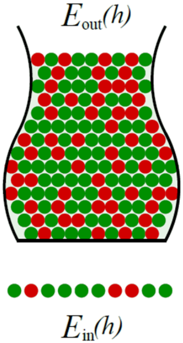
Both μ and ν depend on which hypothesis h

ν is '**in sample error**' denoted by $E_{in}(h)$

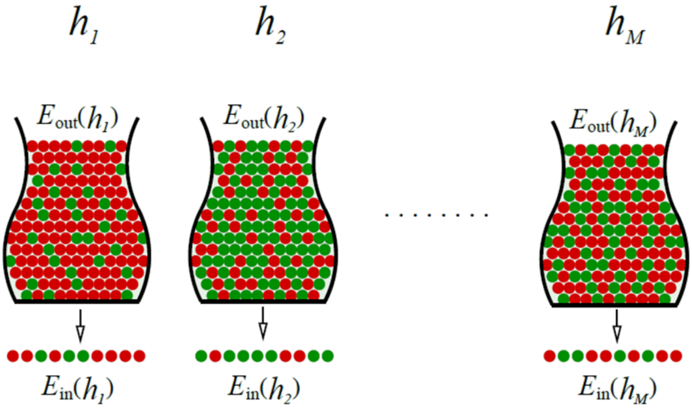
μ is '**out of sample error**' denoted by $E_{out}(h)$

The Hoeffding inequality becomes:

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$



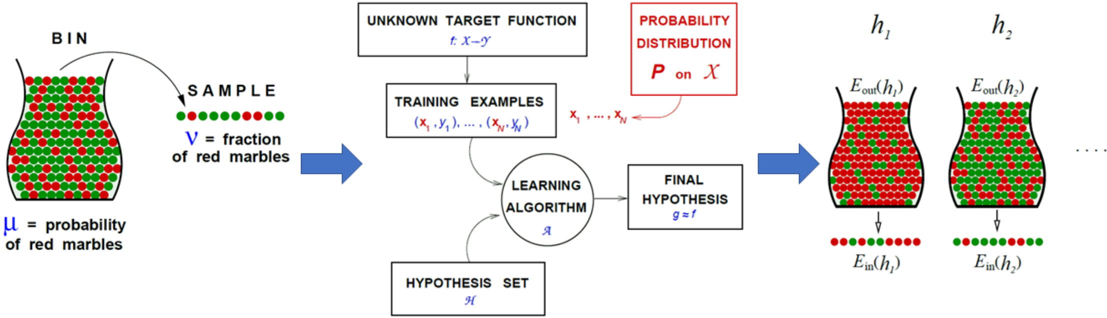
Notation with Multiple Bins



Are we done already?

No! Hoeffding doesn't apply to multiple bins.

Why doesn't apply?



Coin Analogy

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

Answer: $\approx 0.1\%$

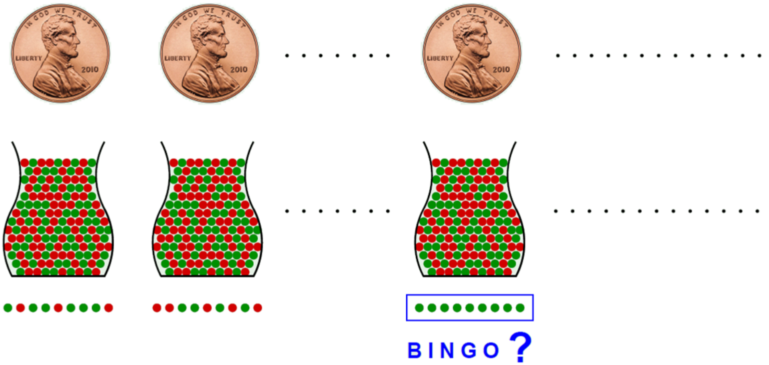
Question: If you toss 1000 fair coins 10 times each, what is the probability that some coin will get 10 heads?

Answer: $\approx 63\%$

In this case the in-sample probability does not approximate the real probability.

Why?

From Coins to Learning



- Fair coin → Bins are half red, half green (μ=1/2).
- It is not BINGO. We tried so hard that eventually happens.

A Simple Solution

$$
\begin{aligned}
\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \quad \leq \mathbb{P}[\quad & |E_{in}(h_1) - E_{out}(h_1)| > \epsilon \\
& \textbf{or} |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\
& \cdots \\
& \textbf{or} |E_{in}(h_M) - E_{out}(h_M)| > \epsilon] \\
\leq \quad & \sum_{m=1}^{M} \mathbb{P}[|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]
\end{aligned}
$$

A Simple Solution

$$
\begin{aligned}
\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \;\leq \mathbb{P}[\; & |E_{in}(h_1) - E_{out}(h_1)| > \epsilon \\
& \textbf{or}\, |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\
& \cdots \\
& \textbf{or}\, |E_{in}(h_M) - E_{out}(h_M)| > \epsilon] \\
\leq \; & \sum_{m=1}^{M} \mathbb{P}[|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]
\end{aligned}
$$

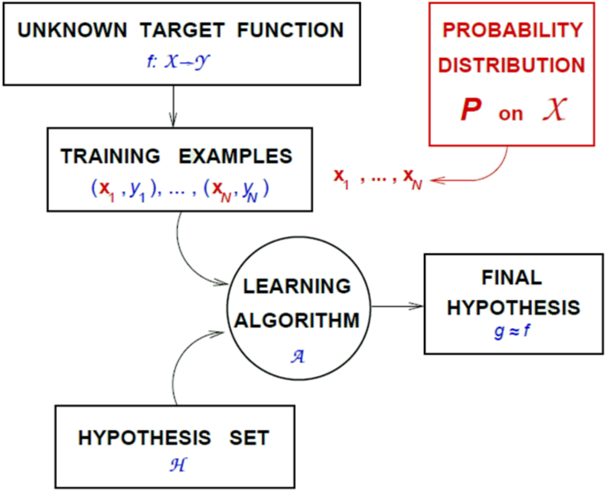
The Final Verdict

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \sum_{m=1}^{M} \mathbb{P}[|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]$$

$$\leq \sum_{m=1}^{M} 2e^{-2\epsilon^2 N}$$

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2{\color{red}M}e^{-2\epsilon^2 N}$$

Remembering the Learning Diagram



UNKNOWN TARGET FUNCTION

$f: \mathcal{X} \to \mathcal{Y}$

PROBABILITY DISTRIBUTION

P on $\mathcal{X}$

TRAINING EXAMPLES

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

$\mathbf{x}_1, \ldots, \mathbf{x}_N$

LEARNING ALGORITHM

$\mathcal{A}$

FINAL HYPOTHESIS

$g \approx f$

HYPOTHESIS SET

$\mathcal{H}$

Error Measures

What does "$h \approx f$" mean?

- To quantify how well g approximates f, we need to define an error measure:

  $$\text{Error} = E(h, f)$$

  where $E(h, f)$ is based on the entirety of h and f.
- Errors on individual input points $\mathbf{x}$ is almost always considered $e(h(\mathbf{x}), f(\mathbf{x}))$. Examples:
  - Squared error:

    $$\mathrm{e}(h(\mathbf{x}), f(\mathbf{x})) = (h(\mathbf{x}) - f(\mathbf{x}))^2$$
  - Binary error:

    $$\mathrm{e}(h(\mathbf{x}), f(\mathbf{x})) = [h(\mathbf{x}) \neq f(\mathbf{x})]$$

What are the criteria for choosing one error measure over another?

From Pointwise to Overall

Overall error $E(h, h)$ = average of pointwise errors $\mathrm{e}(h(\mathbf{x}), f(\mathbf{x}))$.

In-sample error:

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{e}(h(\mathbf{x}_n), f(\mathbf{x}_n))$$

Out-of-sample error:

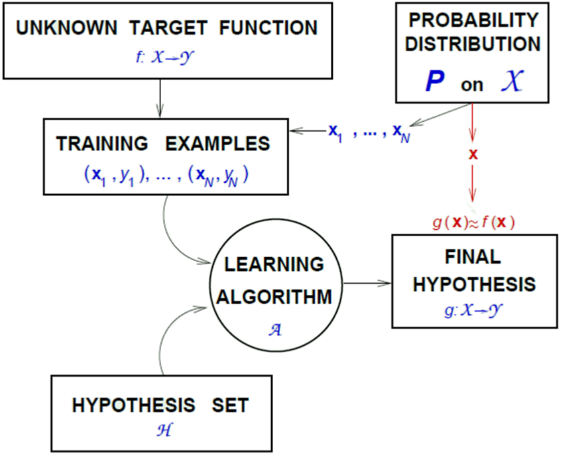
$$E_{out}(h) = \mathbb{E}_x[\mathrm{e}(h(\mathbf{x}), f(\mathbf{x}))]$$

We do not know $E_{out}(h)$ since P on $\mathcal{X}$ is unknown.

The Learning Diagram - with Pointwise Error

- Decide if $g \approx f$ (our goal) based on pointwise error measure on a point $g(\mathbf{x}) \approx f(\mathbf{x})$.
- The point $\mathbf{x}$ comes from the same probability distribution P

When testing, use points drawn from the same probability distribution (requirement to invoke Hoeffding).


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
PROBABILITY DISTRIBUTION
P on $\mathcal{X}$
$\mathbf{x}_1, \dots, \mathbf{x}_N$
$\mathbf{x}$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
$g(\mathbf{x}) \approx f(\mathbf{x})$
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g: \mathcal{X} \to \mathcal{Y}$
HYPOTHESIS SET
$\mathcal{H}$

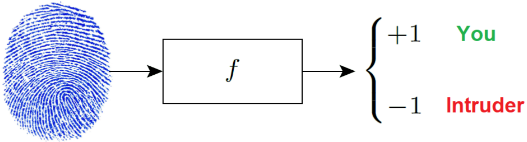
How to Choose the Error Measure?

Example: Fingerprint verification:

- Two types of error:
  $false\ accept$ and $false\ reject$

- How do we penalize each type?

		f	
		$+1$	-1
h	$+1$	no error	$false\ accept$
	-1	$false\ reject$	no error



f

$+1$ You

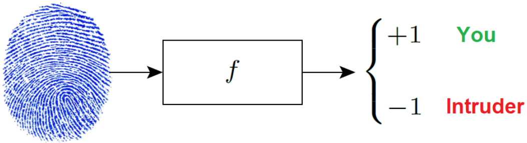
-1 Intruder

The Error Measure - for Supermarkets

Supermarket verifies fingerprint for discounts

- $false\ reject$ is costly; customer gets annoyed.
- $false\ accept$ is minor; gave away a discount and intruder left their fingerprint

		f	
		$+1$	-1
h	$+1$	0	1
	-1	10	0



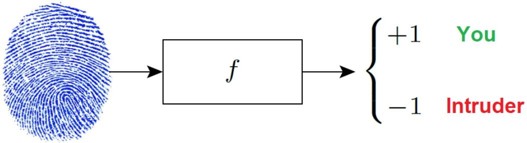
The Error Measure - for the CIA

CIA verifies fingerprint for security

- $false\ accept$ is a disaster.
  Unauthorized person gets access.

- $false\ reject$ can be tolerated.
  You are an employee. Try again.

		f	
		$+1$	-1
h	$+1$	0	1000
	-1	1	0



The Error Measure - Conclusion

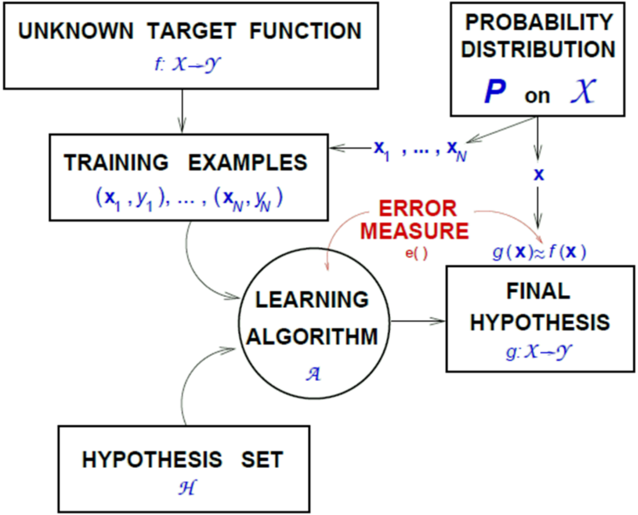
- The error measure should be specified by the user, it is not determined during the learning process.

Not always possible:

- User does not provide any.
- Difficult objective function to optimize. Alternatives:
  - Plausible measures: squared error $\equiv$ Gaussian noise
  - Friendly measures: closed-form solution, convex optimization.

The Learning Diagram - Including Noisy Target

- Quantify how well g approximates f.
- The learning algorithm minimizes the error measure (in-sample error).
- Different error measures may lead to different final hypothesis.


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
PROBABILITY DISTRIBUTION
P on $\mathcal{X}$
$\mathbf{x}_1, \ldots, \mathbf{x}_N$
$\mathbf{x}$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
ERROR MEASURE
e( )
$g(\mathbf{x}) \approx f(\mathbf{x})$
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g: \mathcal{X} \to \mathcal{Y}$
HYPOTHESIS SET
$\mathcal{H}$

Noisy Targets

The ‘target function’ is not always a function.

- The output is not uniquely determined by the input.

Consider the credit-card approval:

age	23 years
gender	male
annual salary	\$30,000
years in residence	1 year
years in job	1 year
current debt	\$15,000
...	...

Two ‘identical’ customers (same $\mathbf{x}$) $\rightarrow$ two different credit behaviors.

Noisy Targets

Instead of $y = f(\mathbf{x})$, we use target distribution:

$$P(y|\mathbf{x})$$

$(\mathbf{x}, y)$ is now generated by the joint distribution:

$$P(\mathbf{x})P(y|\mathbf{x})$$

Noisy target $=$ deterministic target $f(\mathbf{x}) = \mathbb{E}(y|\mathbf{x})$ plus noise $\epsilon = y - f(\mathbf{x})$

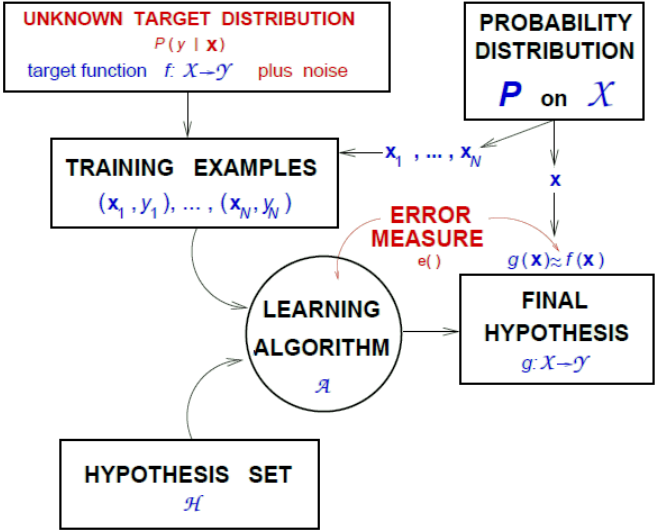
$$y = f(\mathbf{x}) + \epsilon$$

Deterministic target is a special case of noisy target:

$$P(y|\mathbf{x}) \text{ is zero except for } y = f(\mathbf{x})$$

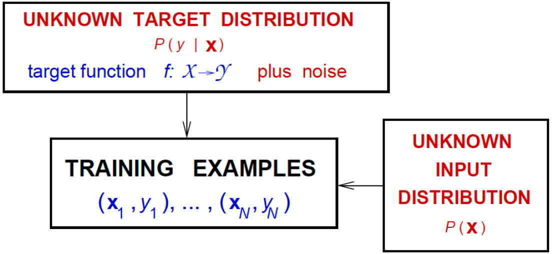
There is no lose of generality.

The Learning Diagram - including noisy target


UNKNOWN TARGET DISTRIBUTION
$P(y \mid \mathbf{x})$
target function $f: \mathcal{X} \to \mathcal{Y}$ plus noise
PROBABILITY DISTRIBUTION
P on $\mathcal{X}$
$\mathbf{x}_1, \ldots, \mathbf{x}_N$
$\mathbf{x}$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
ERROR MEASURE
e( )
$g(\mathbf{x}) \approx f(\mathbf{x})$
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g: \mathcal{X} \to \mathcal{Y}$
HYPOTHESIS SET
$\mathcal{H}$

Distinction between $P(y|\mathbf{x})$ and $P(\mathbf{x})$

- Both convey probabilistic aspects of $\mathbf{x}$ and y.
- The target distribution $P(y|\mathbf{x})$ is what we are trying to learn.
- The input distribution P($\mathbf{x}$) quatifies relative importance of the point $\mathbf{x}$ in gauging how well we have learned.
- Merging $P(\mathbf{x})P(y|\mathbf{x})$ as $P(\mathbf{x}, y)$ mixes the two concepts.


UNKNOWN TARGET DISTRIBUTION
$P(y \mid \mathbf{x})$
target function $f: \mathcal{X} \rightarrow \mathcal{Y}$ plus noise
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
UNKNOWN INPUT DISTRIBUTION
$P(\mathbf{x})$

What we Know so Far

Learning is feasible in a **probabilistic** sense. It is likely that

$$E_{out}(g) \approx E_{in}(g)$$

Is this learning?
We need $g \approx f$, which means

$$E_{out}(g) \approx 0$$

The 2 Questions of Learning

$E_{out}(g) \approx 0$ is achieved through:

$$E_{out}(g) \approx E_{in}(g) \quad \text{and} \quad E_{in}(g) \approx 0$$

Learning is thus split into 2 questions:

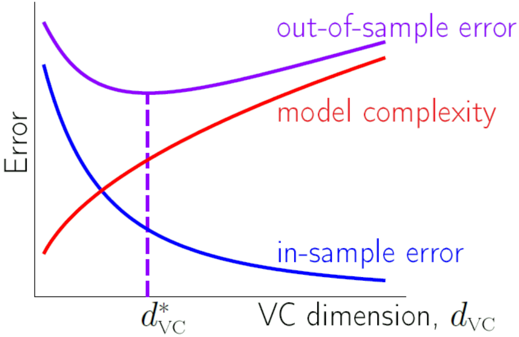
1. Can we make sure that $E_{out}(g)$ is close enough to $E_{in}(g)$?
2. Can we make $E_{in}(g)$ small enough?

What the Theory will Achieve

- Characterizing the feasibility of learning for infinite M
- Characterizing the tradeoff:

Model complexity ↑ E_{in} ↓

Model complexity ↑ $E_{out} - E_{in}$ ↑


out-of-sample error
model complexity
in-sample error
Error
d^*_{VC}
VC dimension, d_{VC}