

CSC 580 Principles of Machine Learning

05 Practical Considerations

Jason Pacheco

Department of Computer Science



*slides credit: built upon CSC 580 lecture slides by Chicheng Zhang & Kwang-Sung Jun

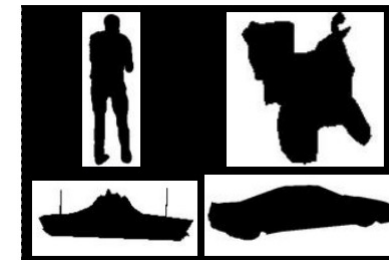
The role of feature in supervised learning

The importance of good feature representation

- Pixel representation:
 - represent an image as a $w \times h \times 3$ dimensional vector
 - treat all coordinates in the same role
 - throw away all locality information in the image



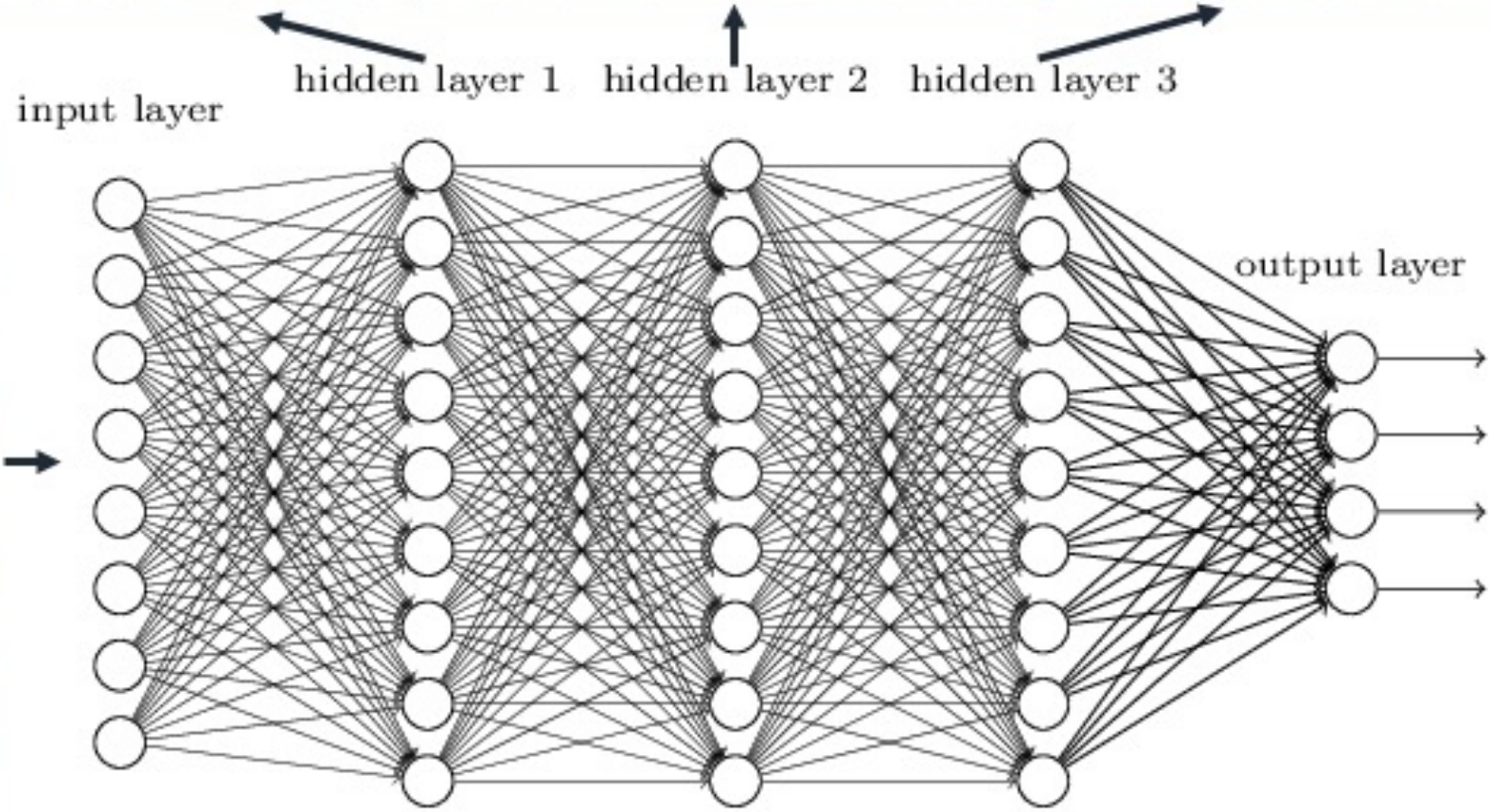
- Shape representation:
 - represent a colored image with a $w \times h$ black-white image

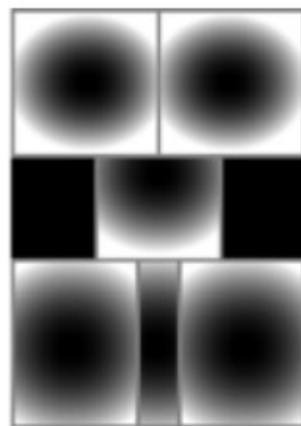
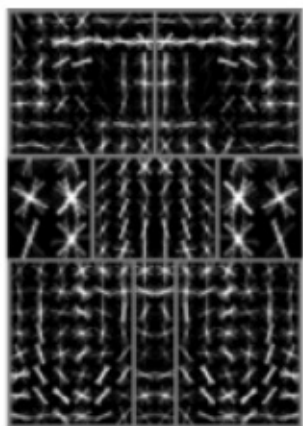
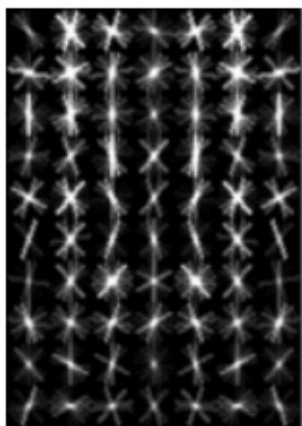
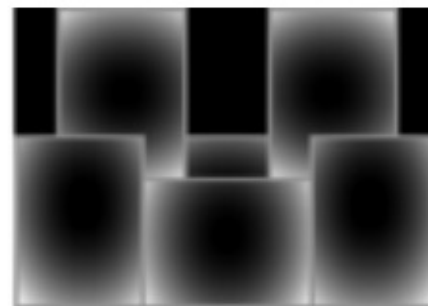
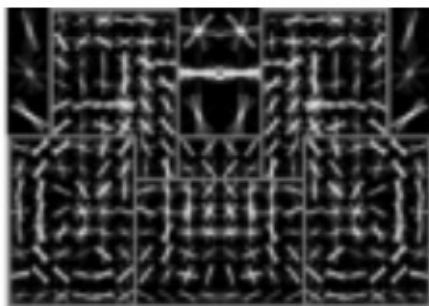
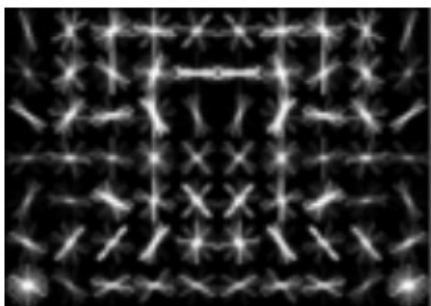
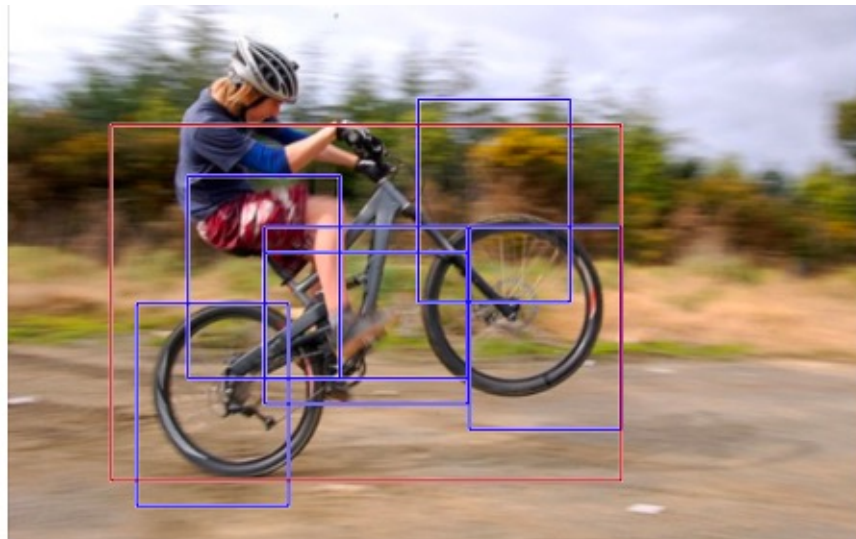
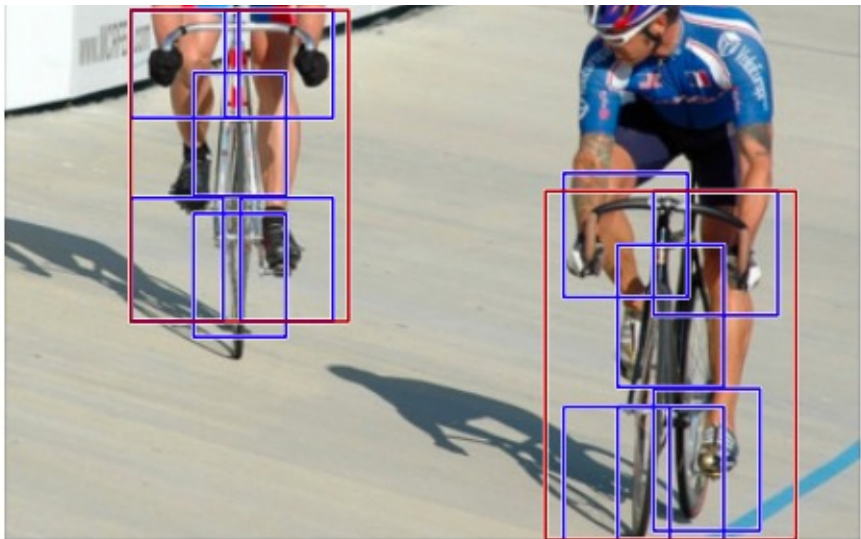


- Bag-of-words representation:

	free	offer	lecture	cs	Spam?
Email 1	2	1	0	0	+1
Email 2	0	1	3	1	-1

Deep neural networks learn hierarchical feature representations





Irrelevant and redundant features

- Irrelevant features
 - y is independent of f
 - y = Road walkability, f = duck activities in the pond



- If #features is large and #examples is small \Rightarrow spurious correlation between some feature & label

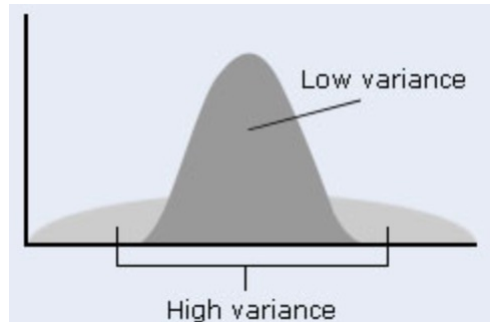
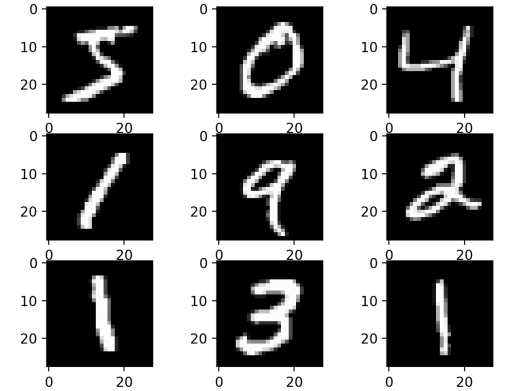
- Redundant features
 - Given f_1 , y is (nearly) independent of f_2
- Learning decision trees implicitly handles these two issues
- How about nearest neighbors / Perceptron?



Feature pruning

- Removing features that are not very useful for prediction
 - E.g. text classification with bag-of-words representation, remove words that appear $\leq K$ docs
 - E.g. digit classification, remove pixels with low variance

$$\mu_f = \frac{1}{N} \sum_{i=1}^N x_{i,f} \quad \sigma_f^2 = \frac{1}{N} \sum_{i=1}^N (x_{i,f} - \mu_f)^2$$



Example: Prostate Cancer Dataset

Term	LS	Ridge	Lasso
Intercept	2.465	2.452	2.468
lcavol	0.680	0.420	0.533
lweight	0.263	0.238	0.169
age	-0.141	-0.046	
lbph	0.210	0.162	0.002
svi	0.305	0.227	0.094
lcp	-0.288	0.000	
gleason	-0.021	0.040	
pgg45	0.267	0.133	

Best LASSO model learns to ignore several features (age, lcp, gleason, pgg45).

Wait...Is **age** really not a significant predictor of prostate cancer? What's going on here?

Age is highly correlated with other factors and thus *not significant* in the presence of those factors

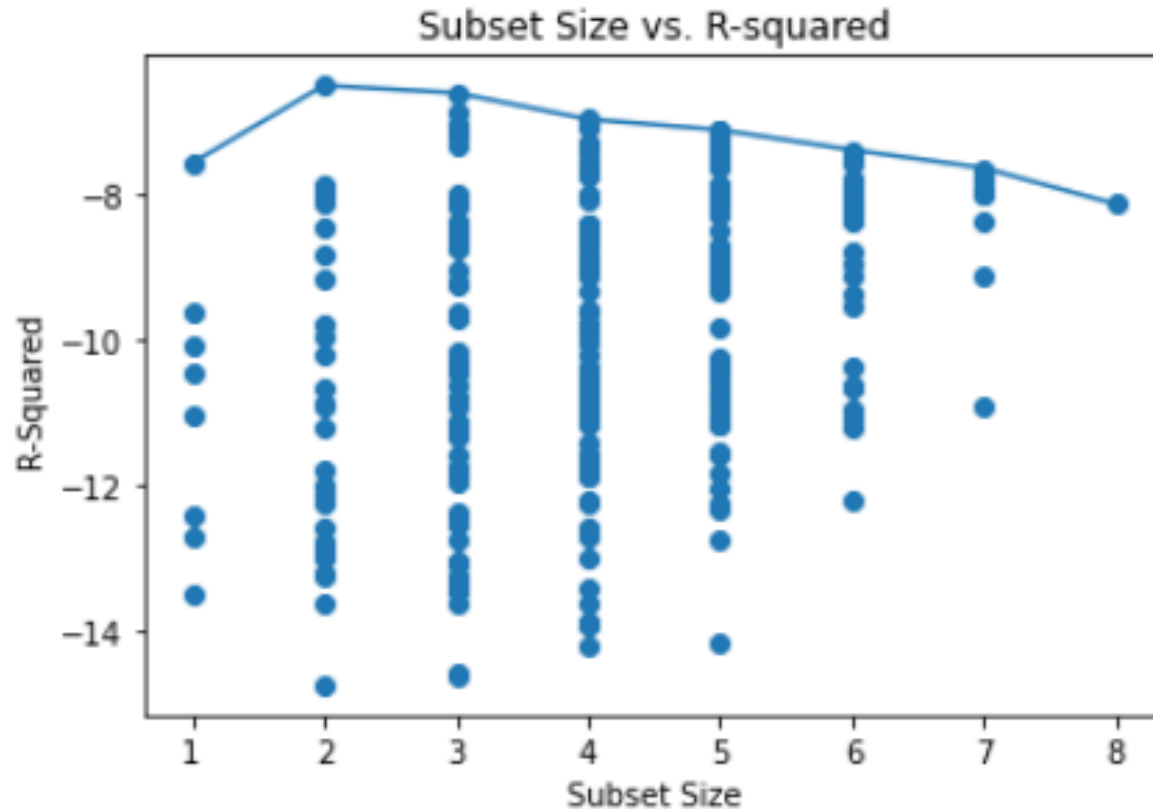
Best-Subset Feature Selection

The optimal strategy for p features looks at models over *all possible combinations* of features,

```
For k in 1, ..., p:  
  subset = Compute all subset of k-features (p-choose-k)  
  For kfeat in subset:  
    model = Train model on kfeat features  
    score = Evaluate model using cross-validation  
Choose the model with best cross-validation score
```

Best-Subset Feature Selection : Prostate Cancer Dataset

Each marker is the cross-val R^2 score of a trained model for a subset of features



Data have 8 features, there are $8\text{-choose-}k$ subsets for each $k=1,\dots,8$ for a total of 255 models

Using 10-fold cross-val requires $10 \times 255 = 2,550$ training runs!

Feature Selection: Prostate Cancer Dataset

Best subset has highest test accuracy (lowest variance)
with just 2 features

Term	LS	Best Subset	Ridge	Lasso
Intercept	2.465	2.477	2.452	2.468
lcavol	0.680	0.740	0.420	0.533
lweight	0.263	0.316	0.238	0.169
age	-0.141		-0.046	
lbph	0.210		0.162	0.002
svi	0.305		0.227	0.094
lcp	-0.288		0.000	
gleason	-0.021		0.040	
pgg45	0.267		0.133	
Test Error	0.521	0.492	0.492	0.479
Std Error	0.179	0.143	0.165	0.164

Forward Sequential Selection

An efficient method adds the most predictive feature one-by-one

```
featSel = empty
featUnsel = All features
For iter in 1,...,p:
  For kfeat in featUnsel:
    thisFeat = featSel + kfeat
    model = Train model on thisFeat features
    score = Evaluate model using cross-validation
  featSel = featSel + best scoring feature
  featUnsel = featUnsel - best scoring feature
Choose the model with best cross-validation score
```

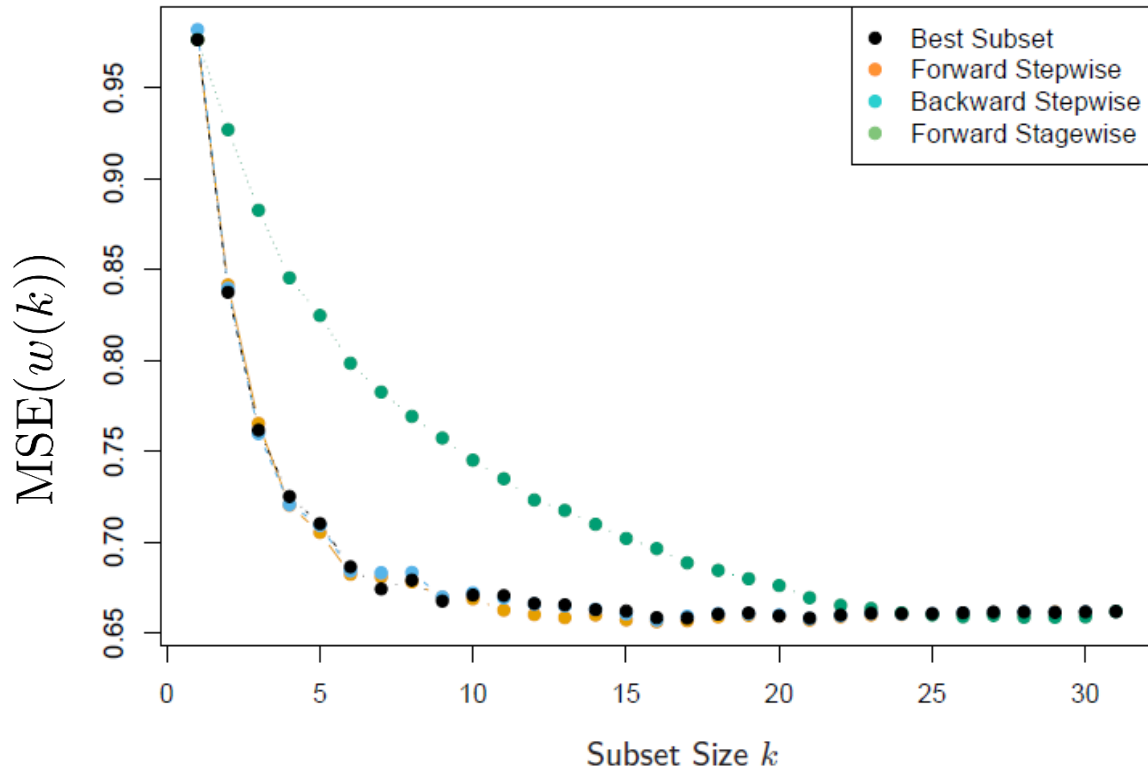
Backward Sequential Selection

Backwards approach starts with *all* features and removes one-by-one

```
featSel = All features
For iter in 1,...,p:
  For kfeat in featSel:
    thisFeat = featSel - kfeat
    model = Train model on thisFeat features
    score = Evaluate model using cross-validation
  featSel = featSel - worst scoring feature
Choose the model with best cross-validation score
```

Comparing Feature Selection Methods

Sequential selection is greedy, but often performs well...



Example Feature selection on synthetic model with $p=30$ features with pairwise correlations (0.85). True feature weights are all zero except for 10 features, with weights drawn from $N(0,6.25)$.

Sequential selection with p features takes $O(p^2)$ time, compared to exponential time for best subset

Sequential feature selection available in Scikit-Learn under:
`feature_selection.SequentialFeatureSelector`

Feature normalization

- Centering:

- $x'_{i,f} = x_{i,f} - \mu_f \Rightarrow \mu'_f = 0$

- Variance scaling:

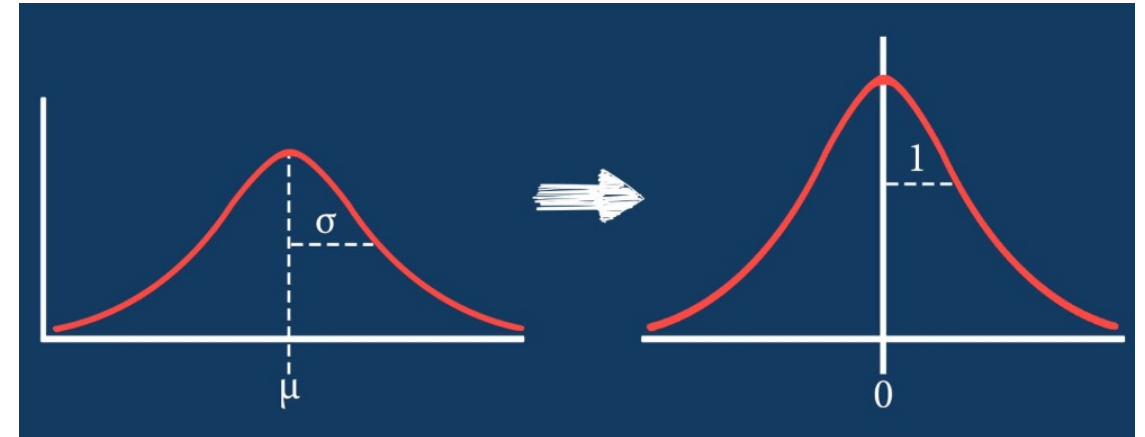
- $x'_{i,f} = x_{i,f}/\sigma_f \Rightarrow (\sigma'_f)^2 = 1$

- Absolute scaling

- $x'_{i,f} = x_{i,f}/r_f$, where $r_f = \max_i |x_{i,f}| \Rightarrow$ range of $x'_{i,f}$'s in $[-1,+1]$

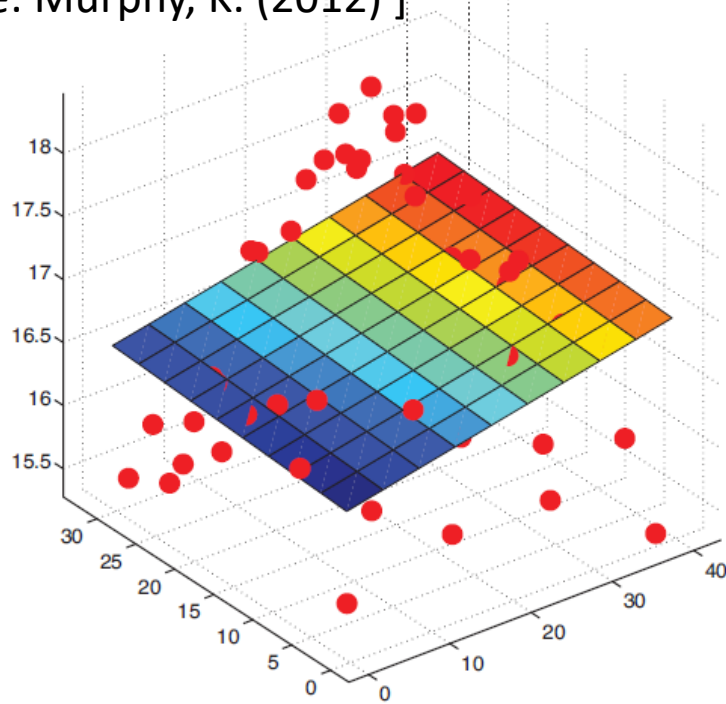
- Same transformation applied to both training set and test data

- Aside: example normalization: $x'_i = \frac{x_i}{\|x_i\|}$ sometimes also can be applied



Linear Models

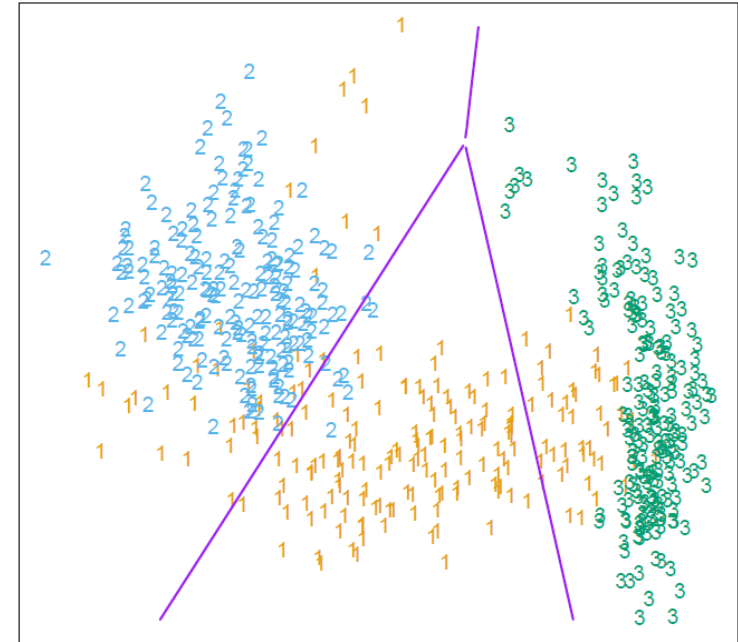
[Image: Murphy, K. (2012)]



Linear Regression Fit a *linear function* to the data,

$$y = w^T x + b$$

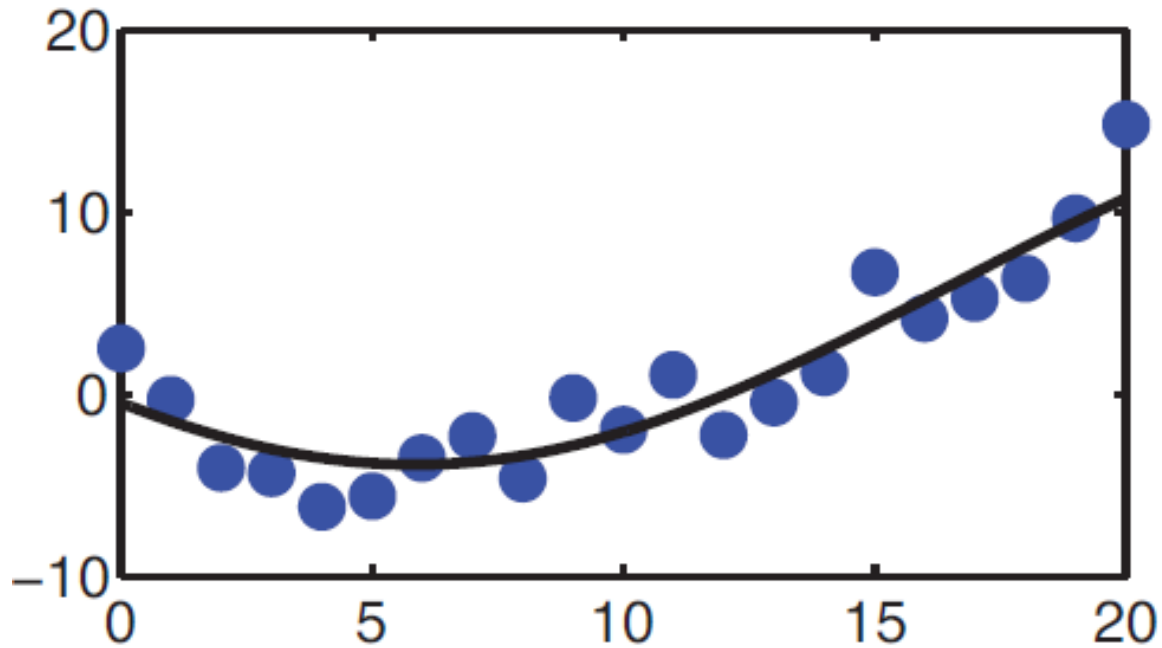
[Image: Hastie et al. (2001)]



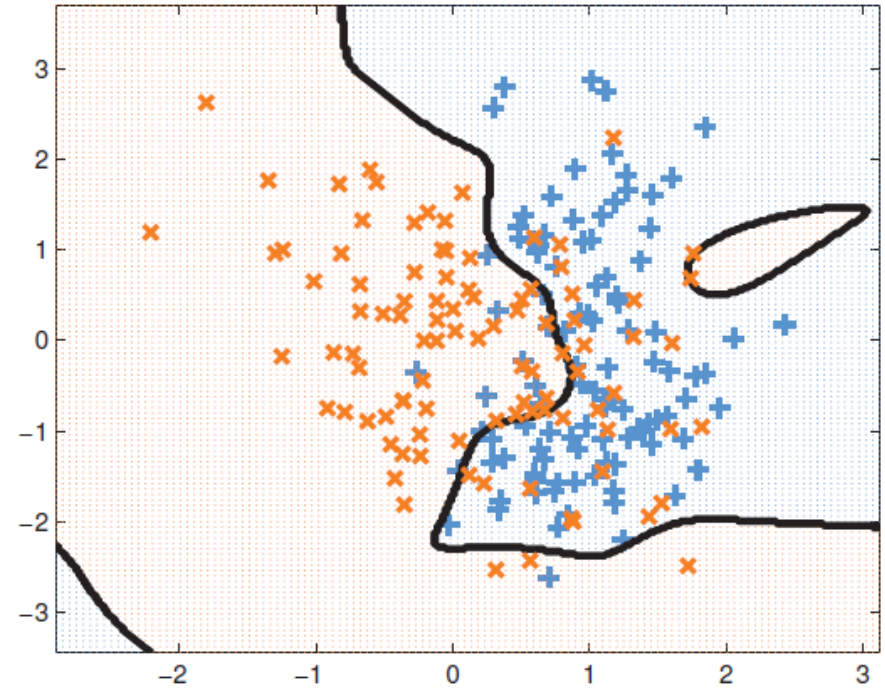
Logistic Regression Learn a decision boundary that is *linear in the data*,

$$\text{logit}(\sigma(w^T x)) = w^T x$$

Nonlinear Data



What if our data are *not* well-described by a linear function?

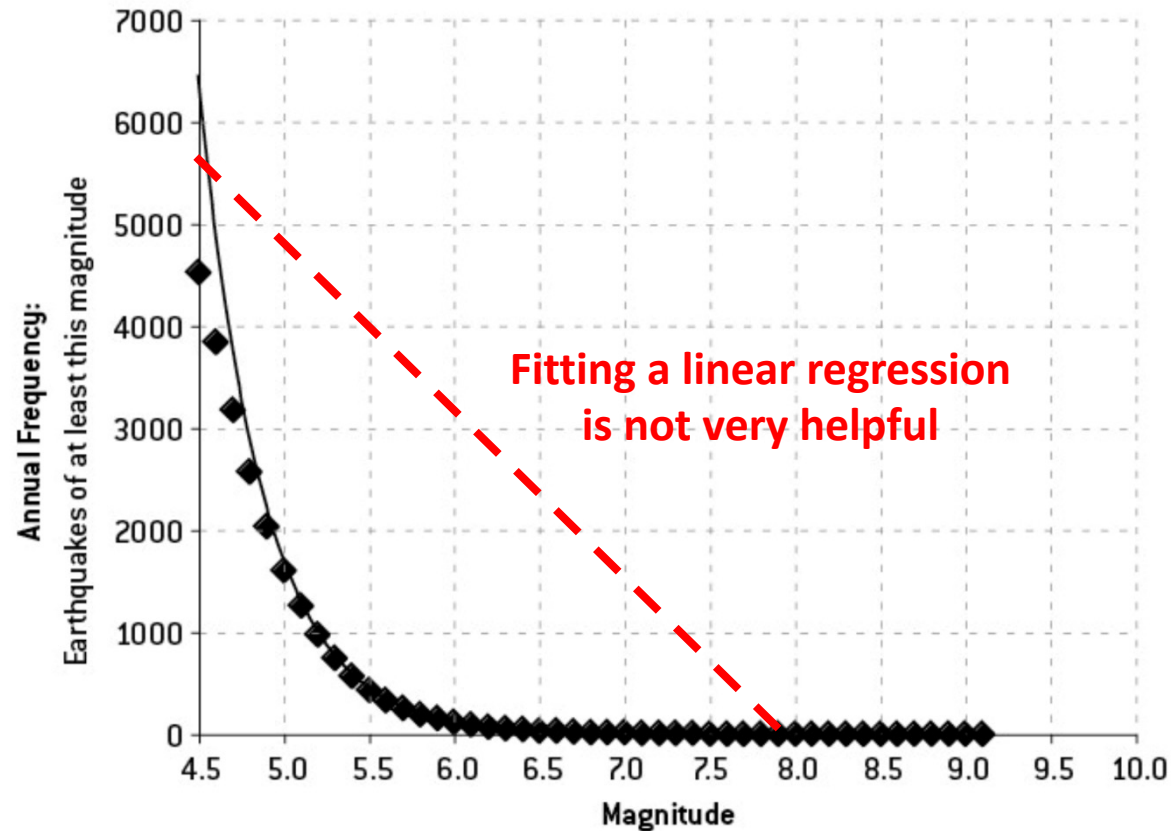


What if classes are *not linearly-separable*?

Example: Earthquake Prediction

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

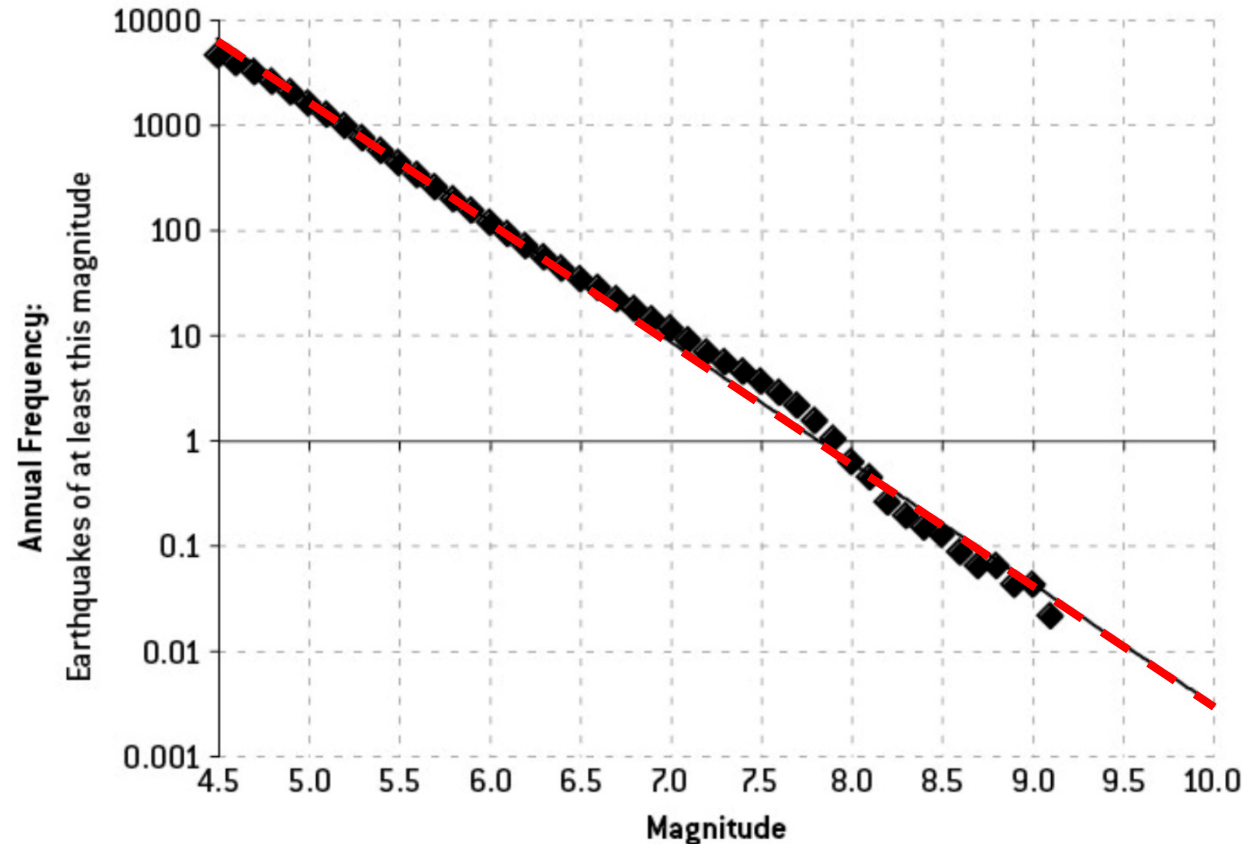
FIGURE 5-3A: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012



Example: Earthquake Prediction

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3B: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012, LOGARITHMIC SCALE

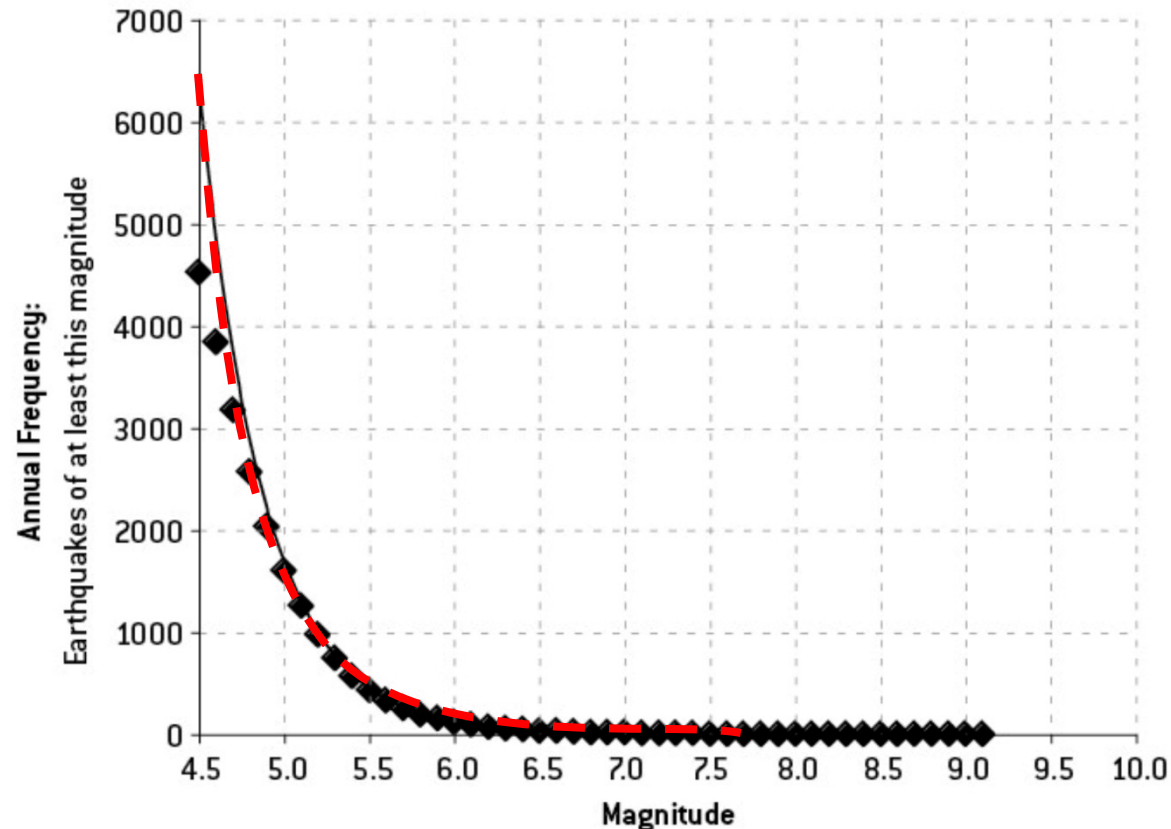


But plotting outputs on a logarithmic scale reveals a strong linear relationship...

Example: Earthquake Prediction

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3A: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012



Idea Instead of fitting ordinary linear regression,

$$y = w^T x$$

First take the logarithm of input values x ,

$$y = w^T \log(x)$$

Basis Functions

- A **basis function** can be any function of the input features X
- Define a set of m basis functions $\phi_1(x), \dots, \phi_m(x)$
- Fit a linear regression model in terms of basis functions,

$$y = \sum_{i=1}^m w_i \phi_i(x) = w^T \phi(x)$$

- Regression model is *linear in the basis transformations*
- Model is *nonlinear in the data X*

Common “All-Purpose” Basis Functions

- Linear basis functions recover the original linear model,

$$\phi_m(x) = x_m$$

Returns m^{th} dimension of X

- Quadratic. $\phi_m(x) = x_j^2$ or $\phi_m(x) = x_j x_k$ capture 2nd order interactions
- An order p polynomial $\phi \rightarrow x_d, x_d^2, \dots, x_d^p$ captures higher-order nonlinearities (but requires $O(d^p)$ parameters)
- Nonlinear transformation of single inputs,

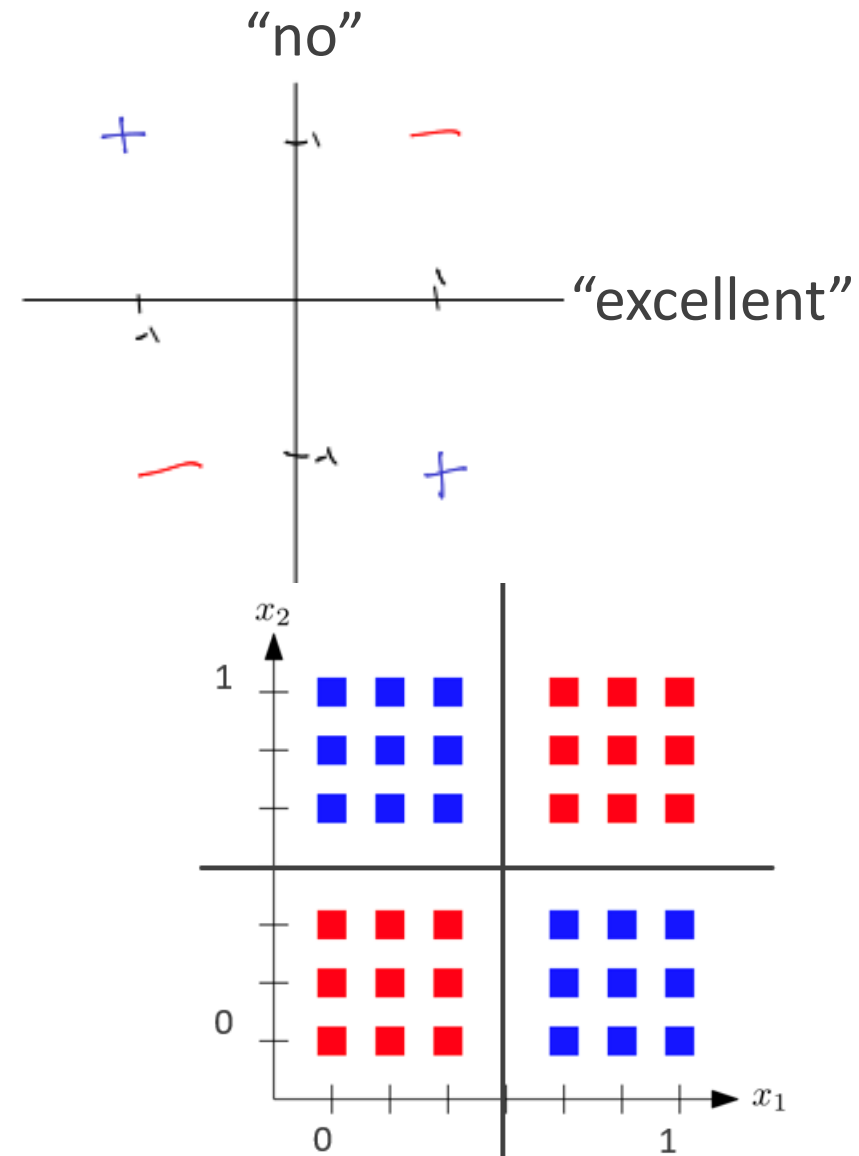
$$\phi \rightarrow (\log(x_j), \sqrt{x_j}, \dots)$$

- An indicator function specifies a region of the input,

$$\phi_m(x) = I(L_m \leq x_k < U_m)$$

Feature transformations

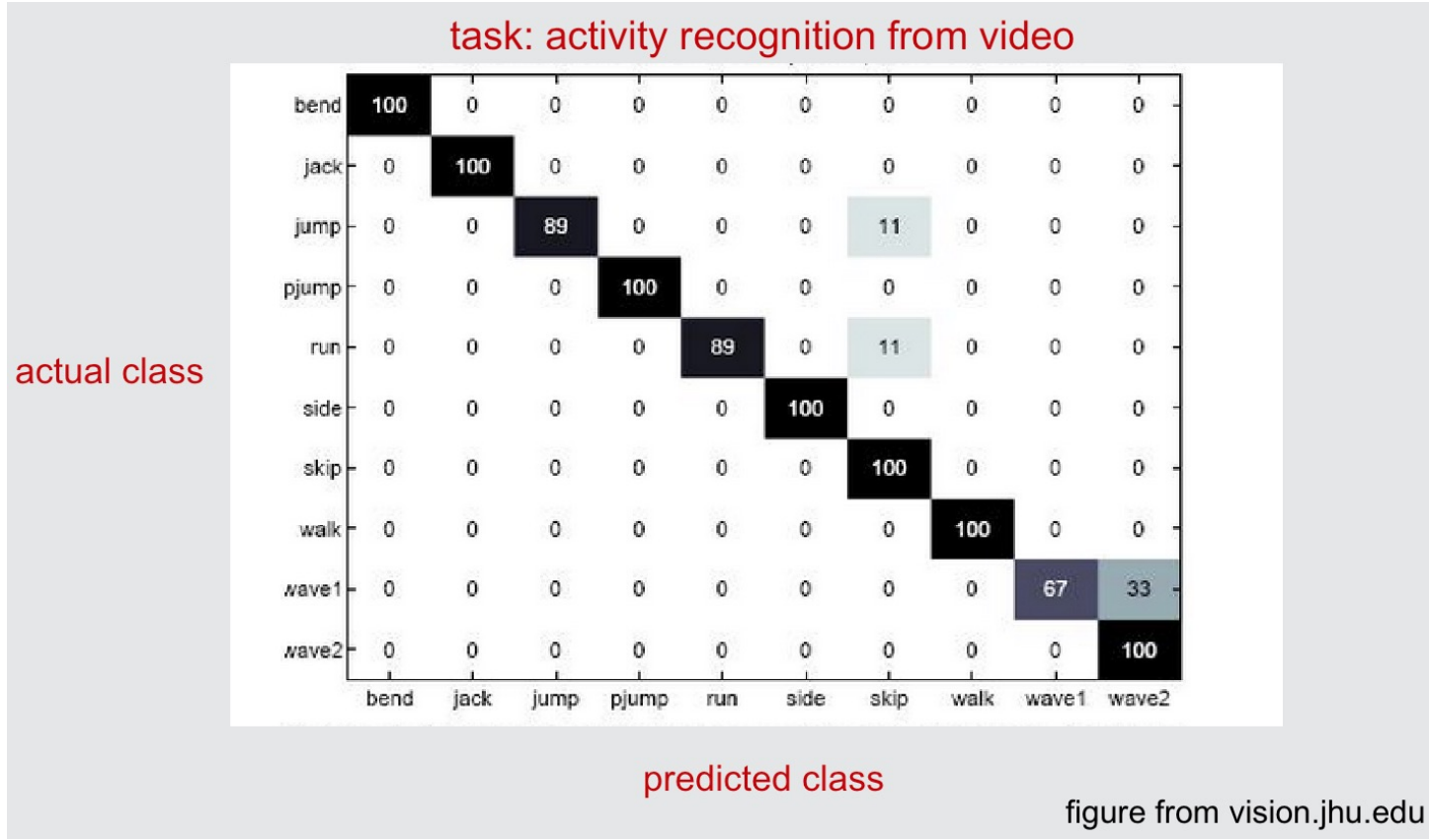
- Combining features into a “meta-feature”, e.g. $x_{\text{no}} \cdot x_{\text{excellent}}$
 - Useful for e.g. Perceptron learners
- In general, $\binom{d}{k}$ meta-features if allowed to combine k features
- Computationally cheaper alternative:
 - train a decision tree, use the meta-feature induced by leaves
- Logarithmic feature transformation
 - $x'_f \leftarrow \log_2(x_f)$ (“excellent” word count: 1- \rightarrow 2 vs. 10- \rightarrow 11)
 - $x'_f \leftarrow \log_2(x_f + 1)$



Classification metrics beyond error rate

Confusion matrix

- E.g. activity recognition
- $P(\hat{y} = \text{skip} \mid y = \text{jump}) = 11\%$



Class imbalance problem



- E.g., 5% pos, 95% negative.
- Baseline: always predict majority class
- Implicit assumption:
misclassifying positive example is more costly than misclassifying negative examples

- Standard ML algorithm aims to find h that minimizes unweighted training error

$$\sum_{i=1}^n I(h(x_i) \neq y_i)$$

- 2 alternatives:
 - Duplicate the minority class to make the positive and negative class balanced
repeat every positive example w times, where $w = P(y = -1)/P(y = +1)$
 - Importance weighted classification: minimize $\sum_{i=1}^n w_i I(h(x_i) \neq y_i)$,
where $w_i = 1$ when $y_i = -1$, $w_i = w$ when $y_i = +1$

New measures of classification performance

- True positive rate (TPR)

$$= \frac{TP}{P} = \frac{P(\hat{y}=+1, y=+1)}{P(y=+1)}$$

(aka recall, sensitivity)

- True negative rate (TNR) = $\frac{TN}{N}$

(specificity)

- False positive rate (FPR) = $\frac{FP}{N}$

- False negative rate (FNR) = $\frac{FN}{P}$

- Precision = $\frac{TP}{P\text{-called}} = \frac{P(\hat{y}=+1, y=+1)}{P(\hat{y}=+1)}$, P - called = TP + FP

The diagram illustrates a confusion matrix for classification performance. The columns represent the actual class (positive and negative), and the rows represent the predicted class (positive and negative). The matrix is divided into four quadrants: true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). The FP and FN cells are labeled as Type I and Type II errors, respectively. Marginal totals are given as P = TP + FN and N = FP + TN.

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP) Type I error
	negative	false negatives (FN) Type II error	true negatives (TN)
		P = TP + FN	N = FP + TN

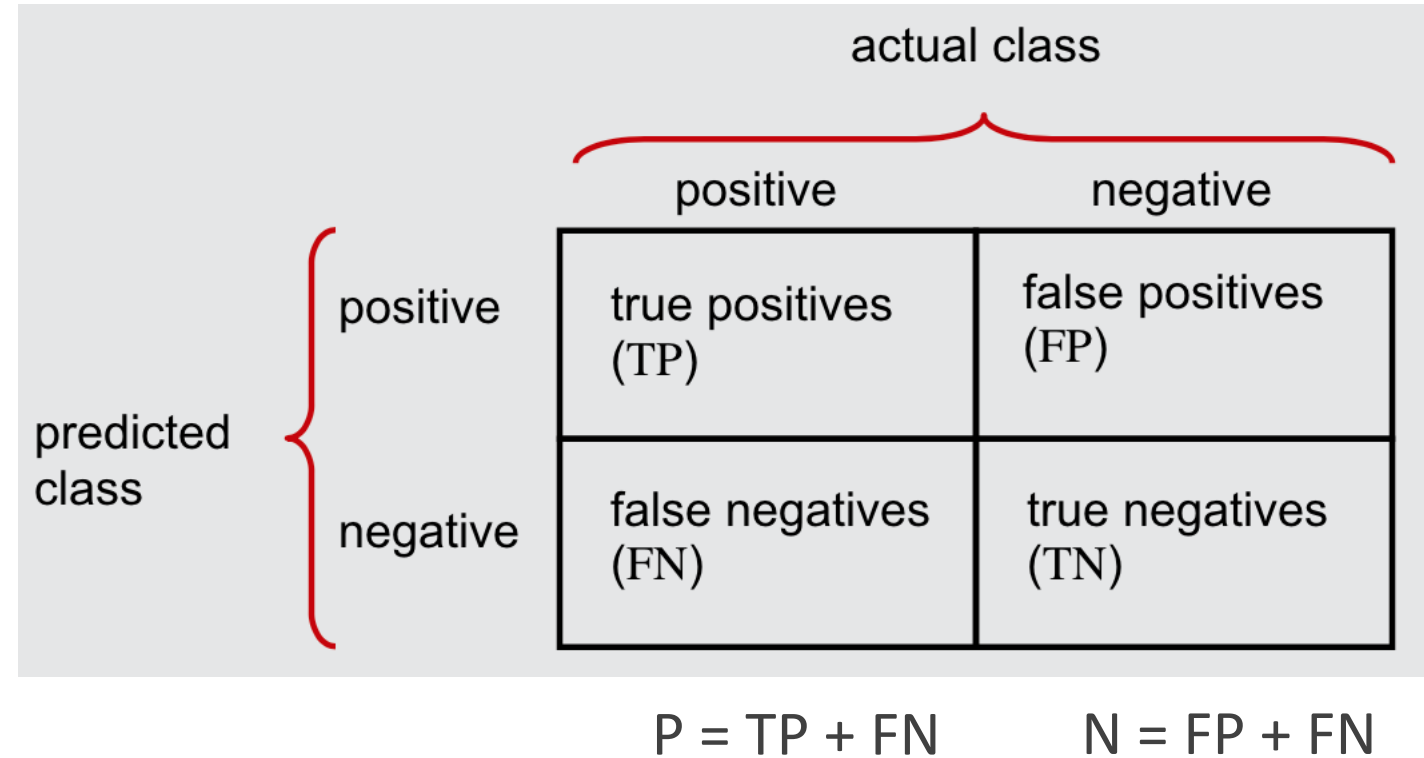
Applications:

- Search engine: precision & recall
- Cancer classification: FNR vs. FPR

Adjusting TP, FP, TN, FN via thresholding

- Decision values (classification scores)

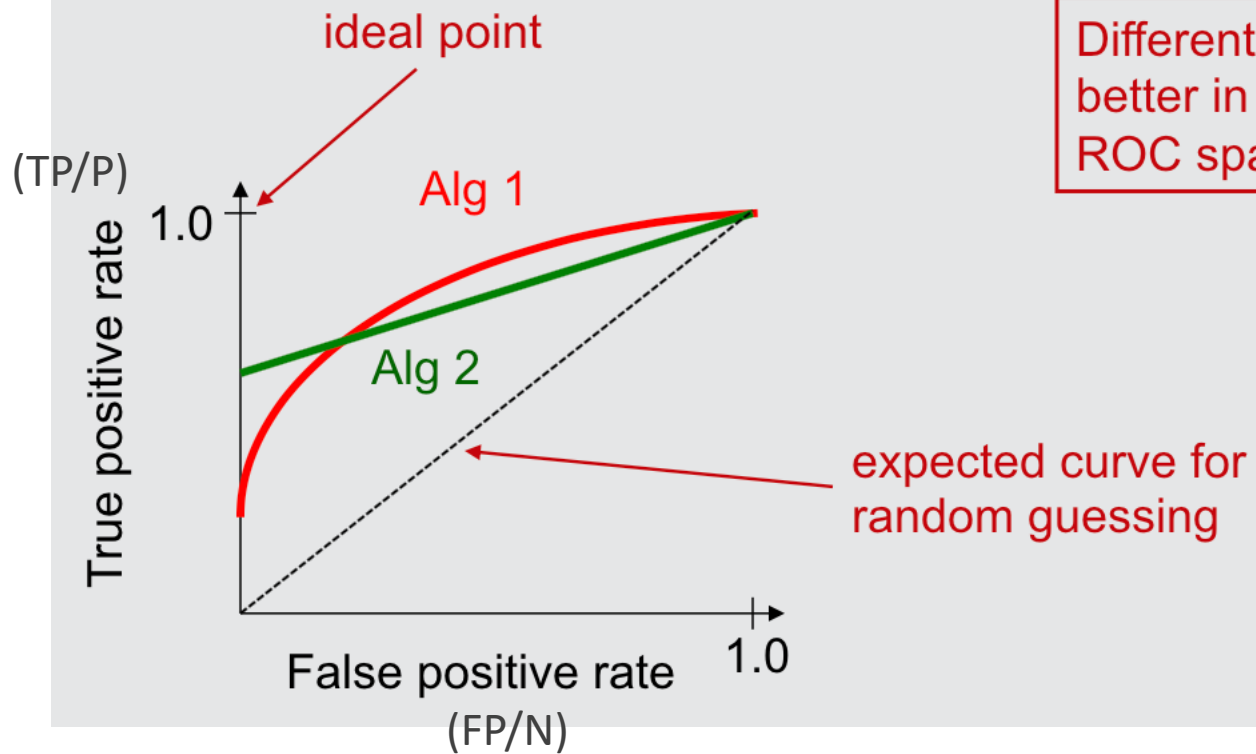
$c(x_i)$	y_i
.99	+
.98	+
.72	-
.51	-
.24	+



- $h_t(x) = I(c(x) \geq t)$
- Choice of threshold t :
 - $t = \infty: h_t \equiv -1 \Rightarrow \text{TPR} = 0, \text{FPR} = 0$
 - $t = 0: h_t \equiv +1 \Rightarrow \text{TPR} = 1, \text{FPR} = 1$

ROC curve

A Receiver Operating Characteristic (ROC) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied

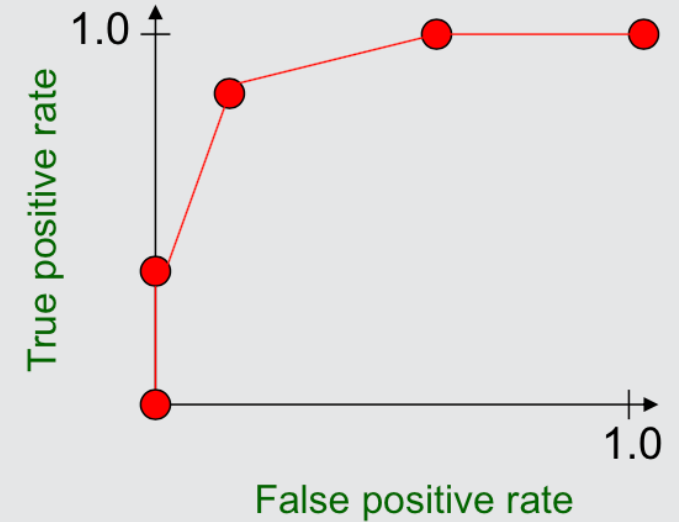


$c(x_i)$	y_i
.99	+
.98	+
.72	+
.51	-
.24	-

ROC curve

- Conceptually, consider every possible threshold, put a dot for each, and connect them.
- Actually, just need to care about when the 'correct class' changes
 - results in staircase shape, but diagonal line can still happen.
- A popular alternative: just plot when going from + to -. (what's shown here)

instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



Calculating ROC curve

let $\left((y^{(1)}, c^{(1)}) \dots (y^{(m)}, c^{(m)}) \right)$ be the test-set instances sorted according to predicted confidence $c^{(i)}$ that each instance is positive

let num_neg, num_pos be the number of negative/positive instances in the test set


$TP = 0, FP = 0$

$last_TP = 0$

for $i = 1$ to m

// find thresholds where there is a pos instance on high side, neg instance on low side

if $(i > 1)$ and $(c^{(i)} \neq c^{(i-1)})$ and $(y^{(i)} == neg)$ and $(TP > last_TP)$

 $FPR = FP / num_neg, TPR = TP / num_pos$

output (FPR, TPR) coordinate

$last_TP = TP$

if $y^{(i)} == pos$

$++TP$

else

$++FP$

$FPR = FP / num_neg, TPR = TP / num_pos$

output (FPR, TPR) coordinate

instance	confidence		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-

ROC curve examples

task: recognizing genomic units called operons

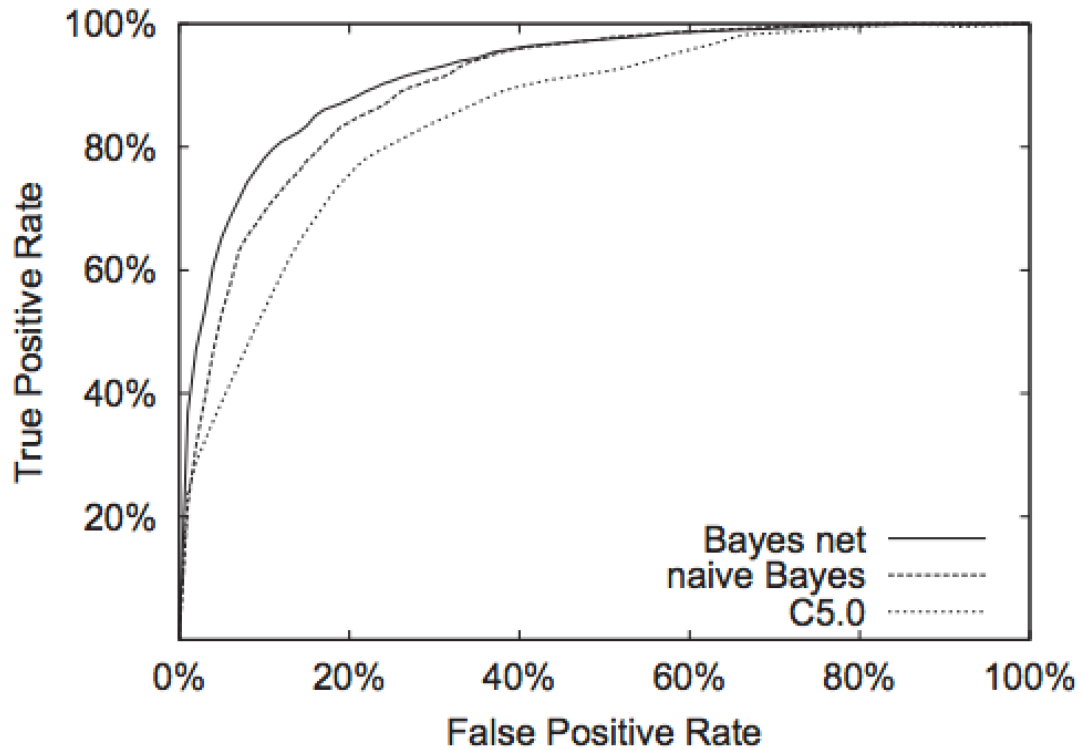
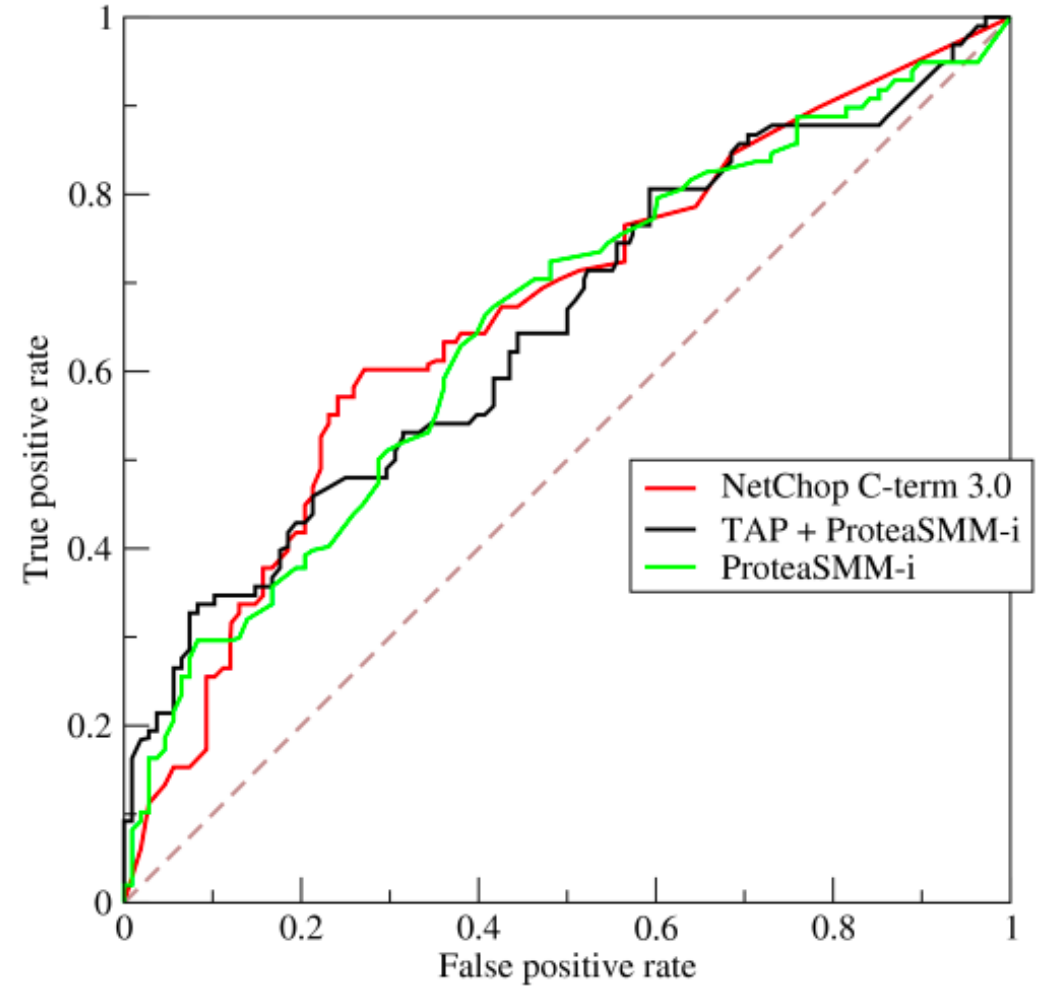


figure from Bockhorst et al., *Bioinformatics* 2003



from Wikipedia

Area under ROC curve

- The boss says “could you just give me one number?”

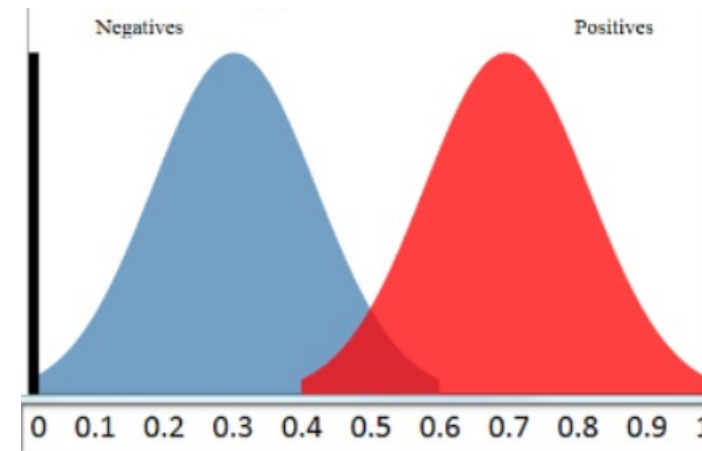
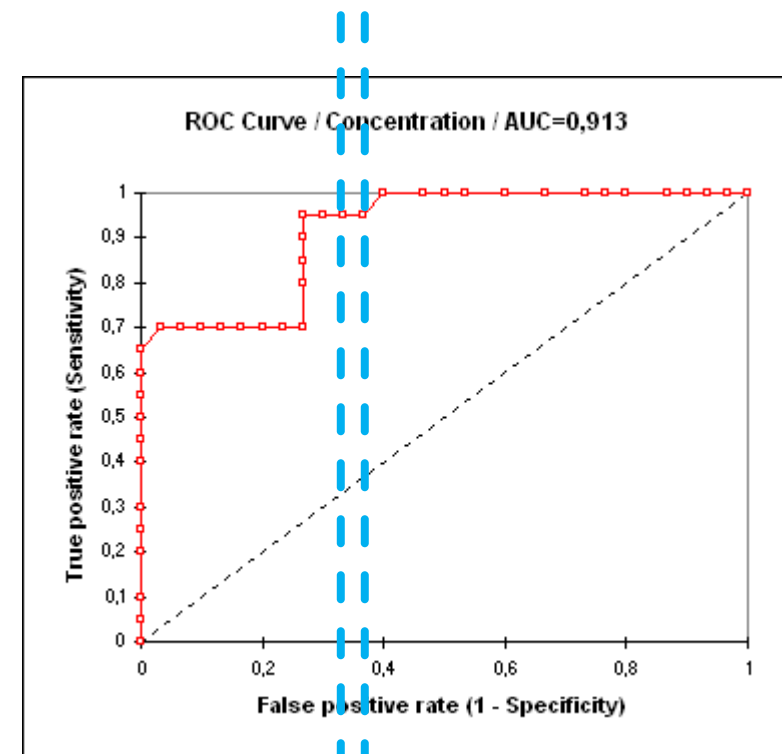
- **AUC**: Area Under the ROC curve:

$$\text{AUC}(c) := \frac{\sum_{(x_-, -1) \in S_-} \sum_{(x_+, +1) \in S_+} I(c(x_+) > c(x_-))}{N_- \cdot N_+}$$

- $c(x)$: decision value of x
- S_- : negative examples, S_+ : positive examples
- Idea: the slice corresponds to x_- has area

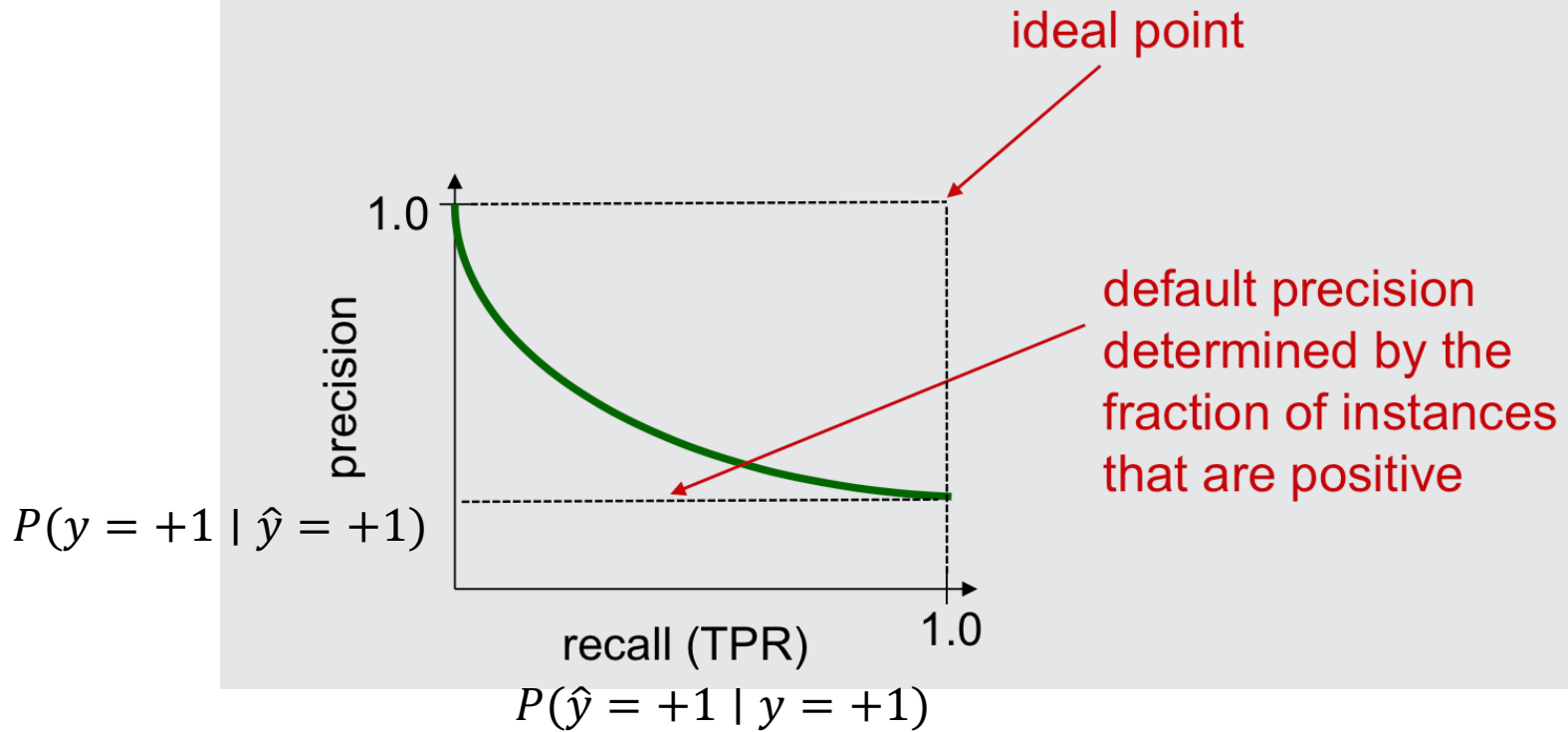
$$\frac{1}{N_-} \cdot \frac{\sum_{(x_+, +1) \in S_+} I(c(x_+) > c(x_-))}{N_+}$$

- Interpretation: “how well does c distinguish between + and -?”



Precision-Recall (PR) curve

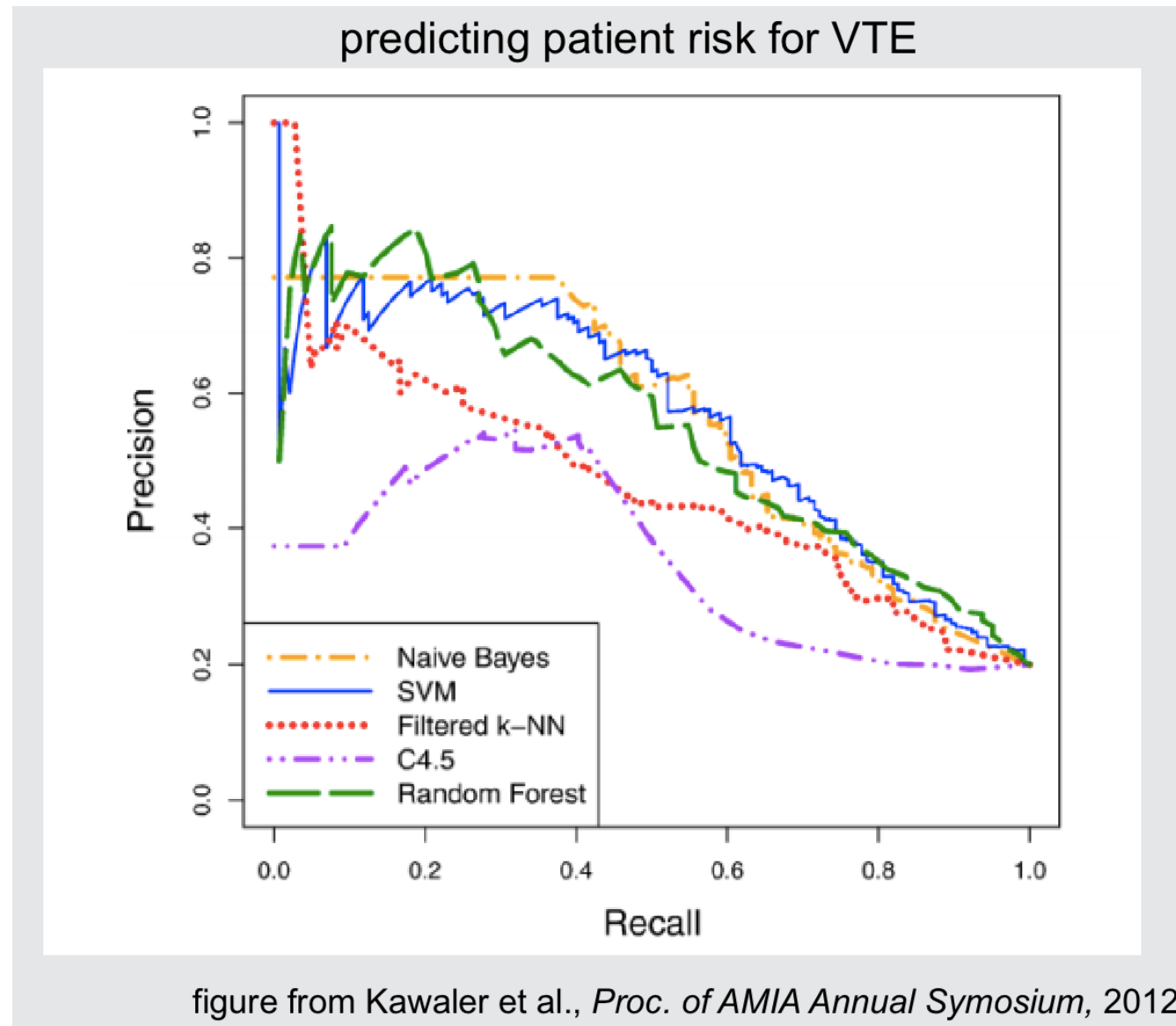
A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied



$c(x_i)$	y_i
.99	+
.98	+
.72	+
.51	-
.24	-

- This is usually a trade-off curve: $t \downarrow \Rightarrow$ recall \uparrow , precision \downarrow

PR-curve example



Summary of precision-recall

- Reporting one number
- Take the harmonic mean: **F1 score**
- Recall: minimum of the two \leq harmonic mean \leq geometric mean \leq arithmetic mean

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}$$

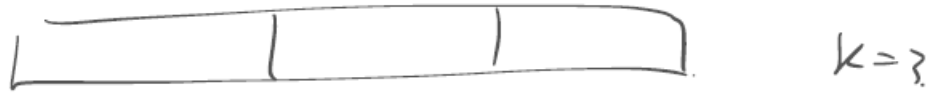
- Emphasizes the smaller measure
 - E.g. recall = 0.1, precision = 0.9 $\Rightarrow F_1 = 0.18$
- Area under PR-curve is also a popular metric

	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.00	0.20	0.26	0.30	0.32	0.33
0.4	0.00	0.26	0.40	0.48	0.53	0.57
0.6	0.00	0.30	0.48	0.60	0.68	0.74
0.8	0.00	0.32	0.53	0.68	0.80	0.88
1.0	0.00	0.33	0.57	0.74	0.88	1.00

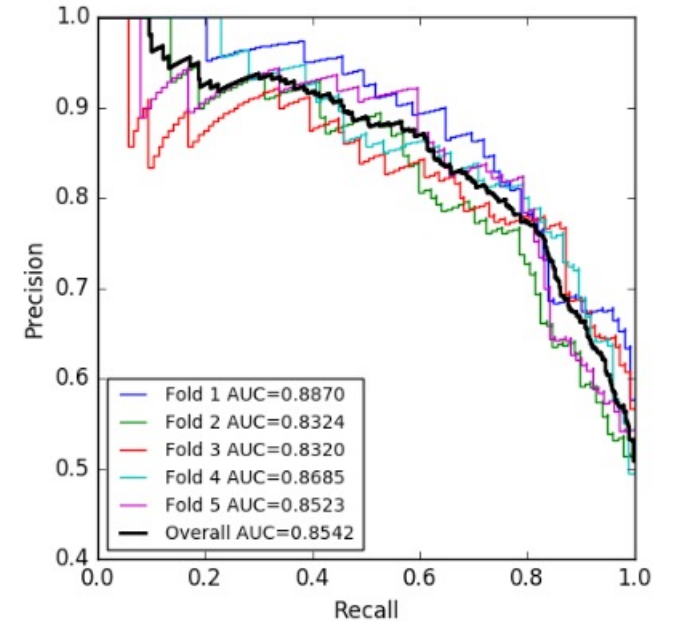
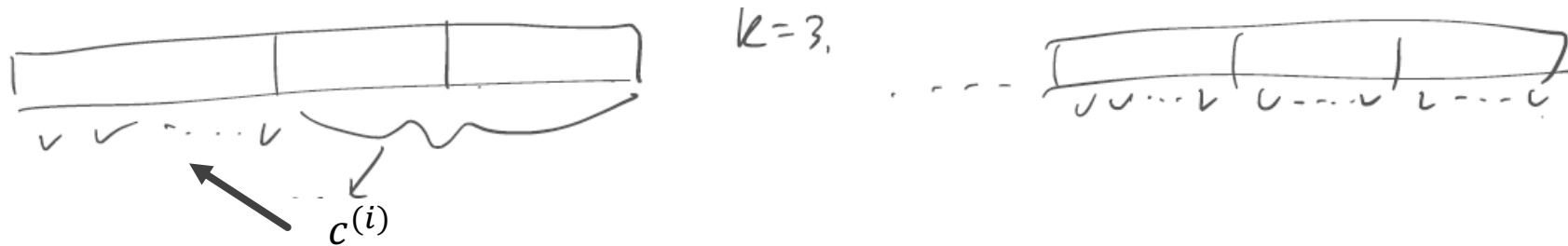
Table 5.2: Table of f-measures when varying precision and recall values.

How to plot ROC/PR curve when training set is small?

- k-fold CV:
 - Obtain k curves and plot them all



- Pooled prediction from k-fold CV.



Model Selection

Motivation: evaluating & comparing ML models

Example

- Your ML model f has test set error = 6.9%
 - Your nemesis, Gabe's, ML model g has test set error = 6.8%
 - How confident are we to conclude that g has smaller generalization error than that of f ?
-
- Intuition: We should be more confident if the test set is larger, less if it's smaller
 - Our uncertainty can be quantified with a *confidence interval*
 - Determining the best model can be done rigorously with *hypothesis testing*

Disclaimer: we only focus on the key ideas (standard stats courses spend ≥ 5 lectures on this)

Confidence Intervals

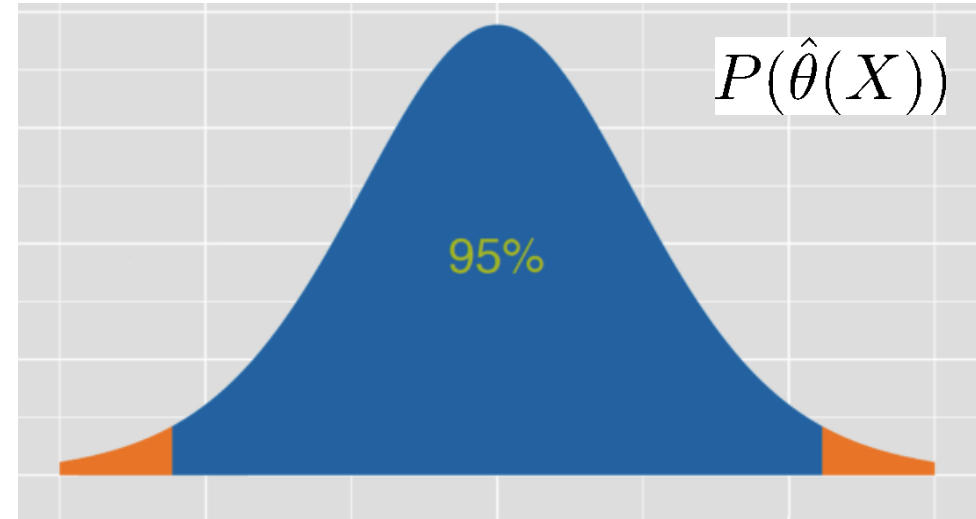
Intuition Find an interval such that we are *pretty sure* it encompasses the true parameter value (e.g. algorithm accuracy).

Given data X_1, \dots, X_n and confidence $\alpha \in (0, 1)$ find interval (a, b) such that,

$$P(\theta \in (a, b)) \geq 1 - \alpha$$

In English the interval (a, b) contains the true parameter value θ with probability **at least** $1 - \alpha$

- Intervals must be computed from data $a(X_1, \dots, X_n)$ and $b(X_1, \dots, X_n)$
- Interval (a, b) is **random**, parameter θ is **not random** (it is fixed)
- Requires that we know the distribution of the estimator $\hat{\theta}$



Warning

Question How should we interpret a confidence interval (e.g. 95%)?

$$P(\theta \in (a(X), b(X))) \geq 0.95$$

Hint Think about what is random and what is not...

This is NOT a probability statement about θ .

Warning

Question *How should we interpret a confidence interval (e.g. 95%)?*

$$P(\theta \in (a(X), b(X))) \geq 0.95$$

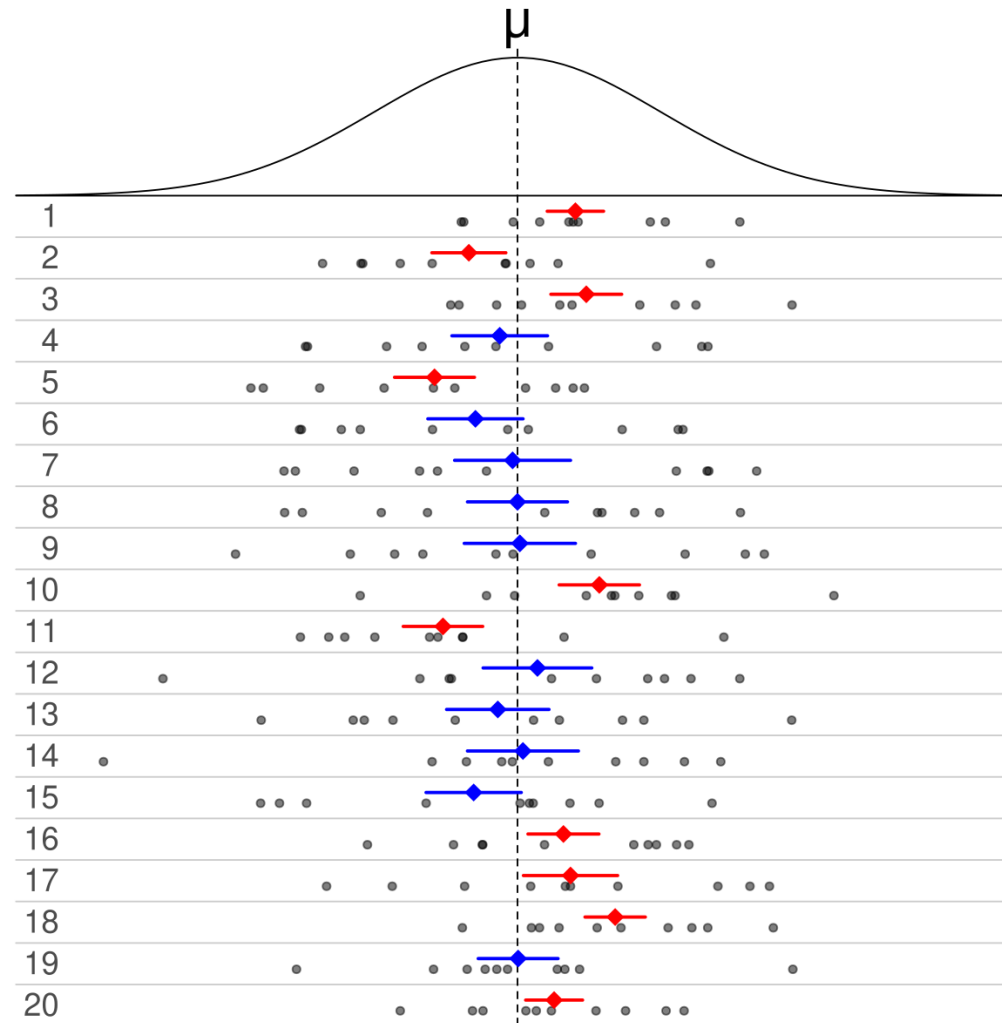
Hint *Think about what is random and what is not...*

Interpretation

On day 1, you collect data and construct a 95 percent confidence interval for a parameter θ_1 . On day 2, you collect new data and construct a 95 percent confidence interval for an unrelated parameter θ_2 . On day 3, you collect new data and construct a 95 percent confidence interval for an unrelated parameter θ_3 . You continue this way constructing confidence intervals for a sequence of unrelated parameters $\theta_1, \theta_2, \dots$. Then 95 percent of your intervals will trap the true parameter value. There is no need to introduce the idea of repeating the same experiment over and over.

Knowledge Check

What is the confidence level of this estimator?



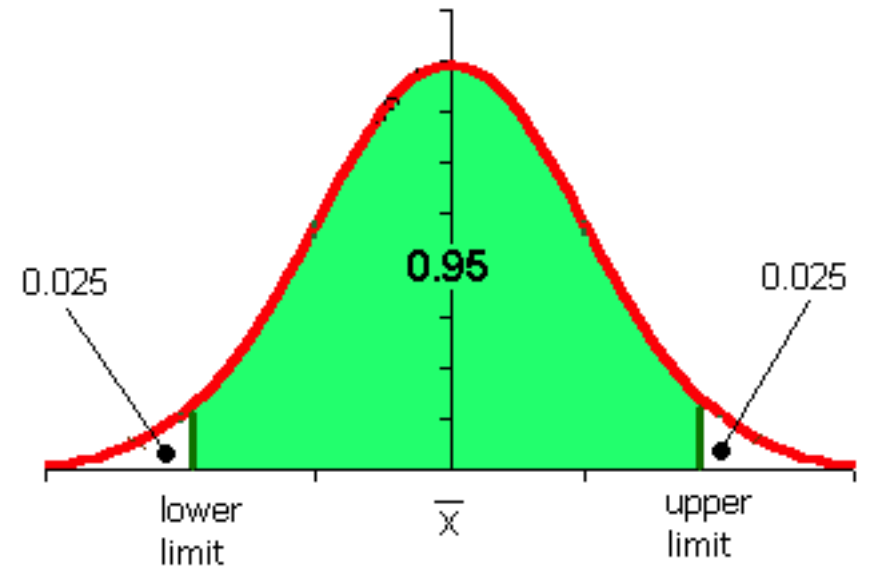
CI construction

A standard recipe:

- Construct an estimator for θ based on S -- call it $\hat{\theta}_S$
- Let $I(S) := [\hat{\theta}_S - w, \hat{\theta}_S + w]$, where w is chosen such that for all θ ,
$$P_{S \sim D_\theta^n}(\theta \in [\hat{\theta}_S - w, \hat{\theta}_S + w]) \geq 1 - \alpha$$
- Sometimes choose $I(S) := [\hat{\theta}_S - w_L, \hat{\theta}_S + w_R]$ with different w_L, w_R 's

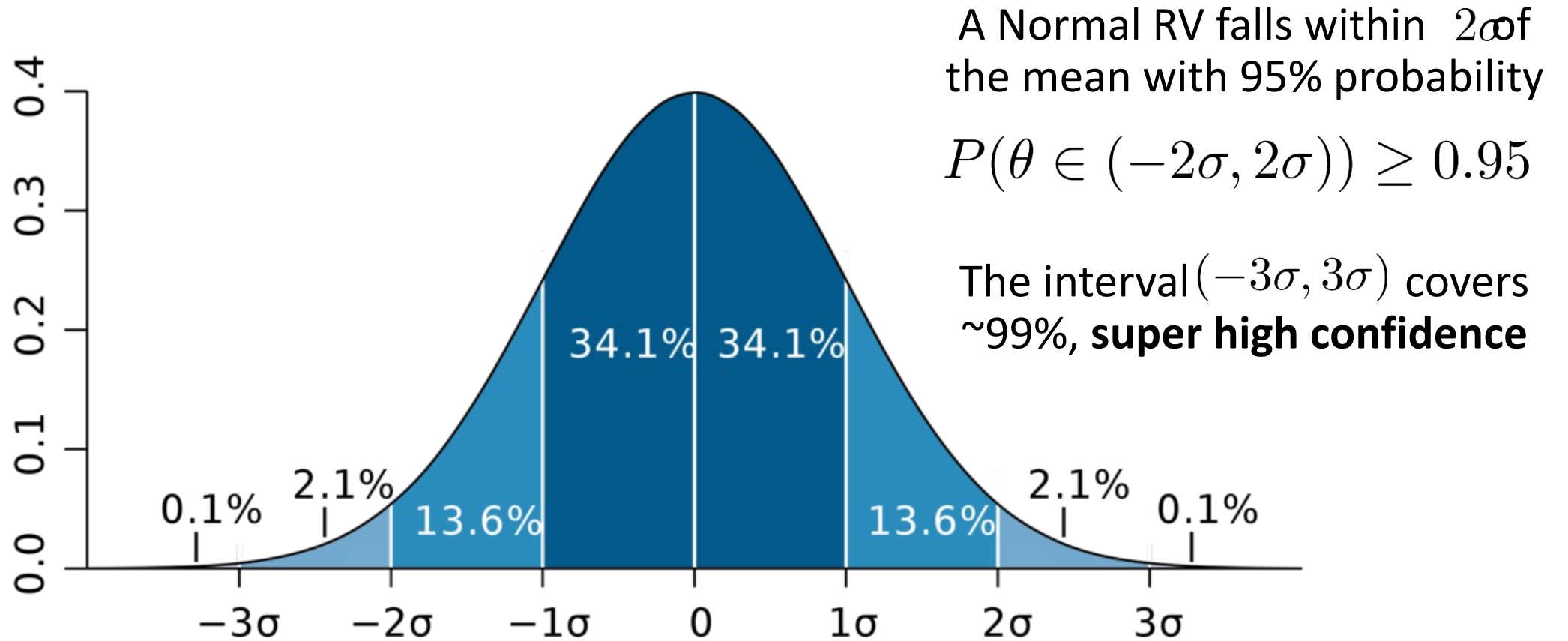
Important example: confidence interval for normal mean

- $D_\mu = N(\mu, 1)$, $S = (X_1, \dots, X_n) \sim D_\mu^n$
- Define $\hat{\mu}_S = \frac{1}{n} \sum_{i=1}^n X_i$ **Known variance**
- $\hat{\mu}_S - \mu \sim N\left(0, \frac{1}{n}\right)$
- How to choose w such that $P(|\hat{\mu}_S - \mu| \leq w) \geq 1 - \alpha$?



Confidence Intervals of the Normal Distribution

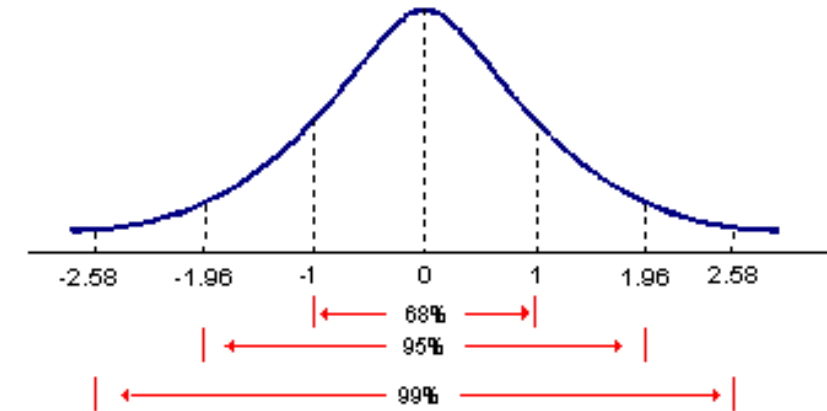
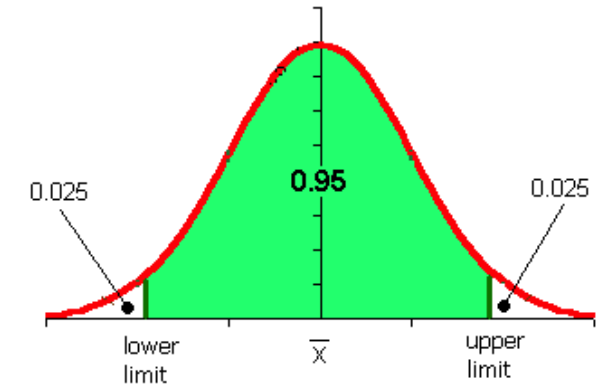
*Many estimators follow a normal distribution with enough data
(central limit theorem)*



For various reasons, 95% has become standard confidence level

CI for normal mean (cont'd)

- $\hat{\mu}_S - \mu \sim N\left(0, \frac{1}{n}\right)$
- How to choose w such that $P(|\hat{\mu}_S - \mu| \leq w) \geq 1 - \alpha$?
- Note: $Z = \sqrt{n} (\hat{\mu}_S - \mu) \sim N(0,1)$ **Central limit theorem**
- Suffices to find z_α such that $P(|Z| \leq z_\alpha) \geq 1 - \alpha$, and let $w = \frac{z_\alpha}{\sqrt{n}}$



- Final $(1 - \alpha)$ -confidence interval construction for μ : $I(S) = \left[\hat{\mu}_S - \frac{z_\alpha}{\sqrt{n}}, \hat{\mu}_S + \frac{z_\alpha}{\sqrt{n}} \right]$
- E.g. 95%-confidence interval for μ : $I(S) = \left[\hat{\mu}_S - \frac{1.96}{\sqrt{n}}, \hat{\mu}_S + \frac{1.96}{\sqrt{n}} \right]$

CI for means of general distributions, *unknown* variance

- Given D_θ with mean parameter θ with *unknown* variance

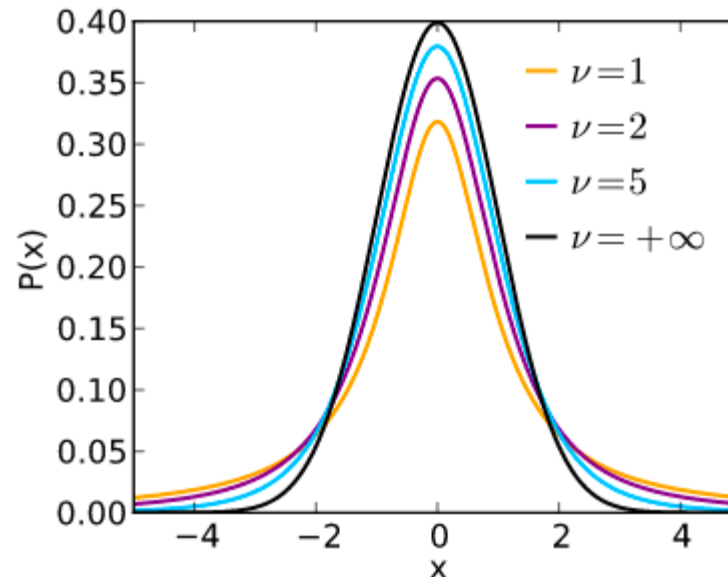
- $\hat{\sigma}_n^2 := \frac{\sum_{i=1}^n (X_i - \hat{\mu}_n)^2}{n-1} \Rightarrow$ unbiased estimator of $\text{var}(D_\theta)$

- *Theorem:* Let $X_1, \dots, X_n \sim N(\mu, \sigma^2)$, and $\hat{\mu}_n := \frac{1}{n} \sum_{i=1}^n X_i$

$$\sqrt{n} \frac{\hat{\mu}_n - \mu}{\hat{\sigma}_n} \sim \text{student-t (mean 0, scale 1, degrees of freedom = } n - 1)$$

- CI: $\left[\hat{\mu}_n \pm \frac{\hat{\sigma}_n \cdot t_\alpha}{\sqrt{n}} \right]$

How do we estimate variance of algorithm performance?



```
import scipy.stats as st
alpha = 0.05
st.t.ppf(1-alpha/2,df=2)
=> 4.302652729911275
```

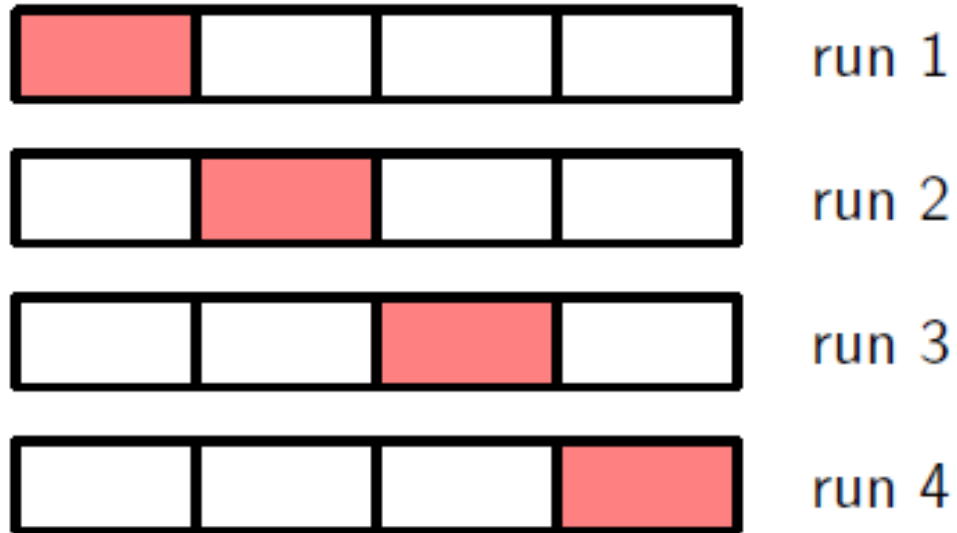
```
st.t.ppf(1-alpha/2,df=5)
=> 2.5705818366147395
```

```
st.t.ppf(1-alpha/2,df=10)
=> 2.2281388519649385
```

```
st.t.ppf(1-alpha/2,df=30)
=> 2.0422724563012373
```

```
st.t.ppf(1-alpha/2,df=100)
=> 1.9839715184496334
```



Cross-Validation



K-fold Cross Validation Partition training data into K “chunks” and for each run select one chunk to be validation data

For each run, fit to training data ($K-1$ chunks) and measure accuracy on validation set. Average model error across all runs. Estimate variance.

Algorithm 8 CROSSVALIDATE(*LearningAlgorithm*, *Data*, *K*)

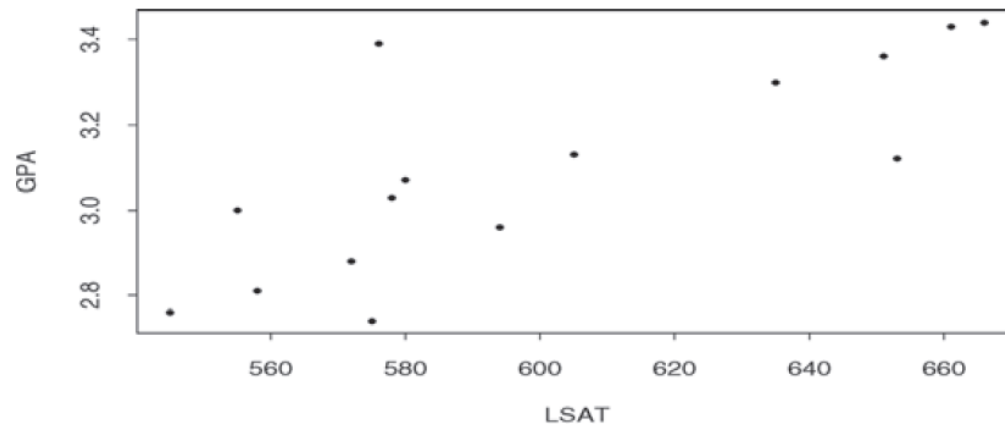
```
1:  $\hat{\epsilon} \leftarrow \infty$  // store lowest error encountered so far
2:  $\hat{\alpha} \leftarrow \text{unknown}$  // store the hyperparameter setting that yielded it
3: for all hyperparameter settings  $\alpha$  do
4:    $err \leftarrow []$  // keep track of the  $K$ -many error estimates
5:   for  $k = 1$  to  $K$  do
6:      $train \leftarrow \{(x_n, y_n) \in Data : n \bmod K \neq k - 1\}$ 
7:      $test \leftarrow \{(x_n, y_n) \in Data : n \bmod K = k - 1\}$  // test every  $K$ th example
8:      $model \leftarrow \text{Run LearningAlgorithm on } train$ 
9:      $err \leftarrow err \oplus \text{error of } model \text{ on } test$  // add current error to list of errors
10:  end for
11:   $avgErr \leftarrow \text{mean of set } err$   Can also estimate variance here
12:  if  $avgErr < \hat{\epsilon}$  then
13:     $\hat{\epsilon} \leftarrow avgErr$  // remember these settings
14:     $\hat{\alpha} \leftarrow \alpha$  // because they're the best so far
15:  end if
16: end for
```

Drawback Need to perform training K times for each model.

Bootstrap Example

Example Suppose we have LSAT scores and GPA for 15 law students and wish to estimate the correlation between LSAT and GPA:

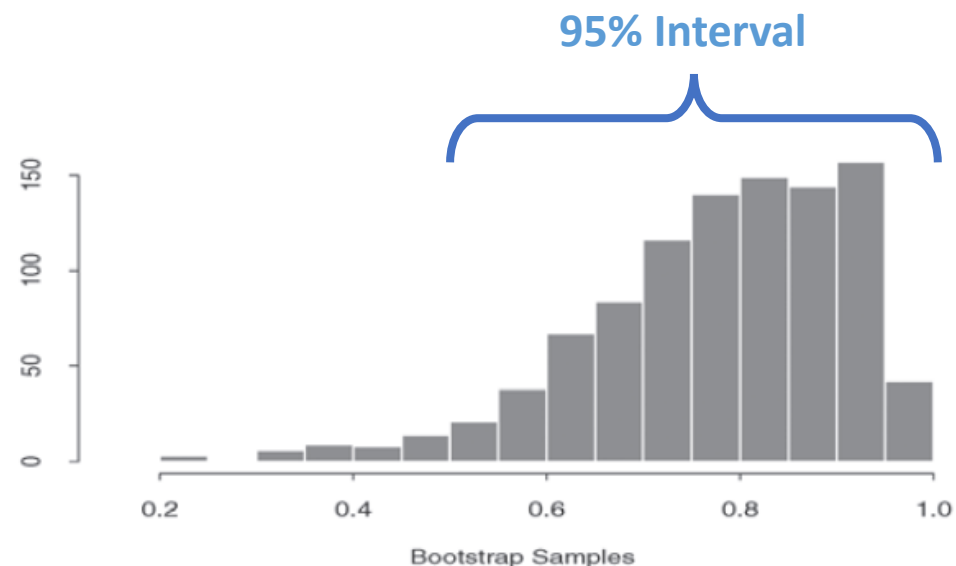
LSAT	576	635	558	578	666	580	555	661
	651	605	653	575	545	572	594	
GPA	3.39	3.30	2.81	3.03	3.44	3.07	3.00	3.43
	3.36	3.13	3.12	2.74	2.76	2.88	3.96	



95% Bootstrap confidence interval from $B=1000$ estimates of the **correlation**,

$$.78 \pm .274 \Rightarrow (.51, 1.00)$$

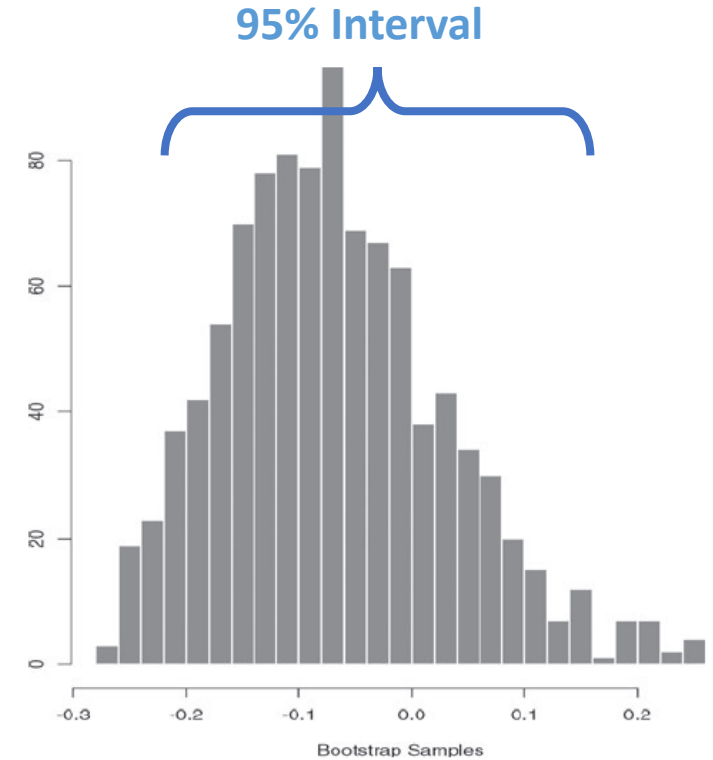
Q Should we trust this confidence interval?
Why or why not?



Bootstrap Example

Eight subjects who used medical patches to infuse a hormone into the blood using three treatments: placebo, old-patch, new-patch

subject	placebo	old	new	old – placebo	new – old
1	9243	17649	16449	8406	-1200
2	9671	12013	14614	2342	2601
3	11792	19979	17274	8187	-2705
4	13357	21816	23798	8459	1982
5	9055	13850	12560	4795	-1290
6	6290	9806	10157	3516	351
7	12412	17208	16570	4796	-638
8	18806	29044	26325	10238	-2719



Estimate whether relative efficacy is the same under new drug,

$$\theta = \frac{\mathbf{E}[\text{new} - \text{old}]}{\mathbf{E}[\text{old} - \text{placebo}]}$$

Bootstrap B=1,000 samples yields 95% confidence interval,

$$\theta \in (-0.24, 0.15)$$

Q Is this more trustworthy than in previous example?

Bootstrapping CI

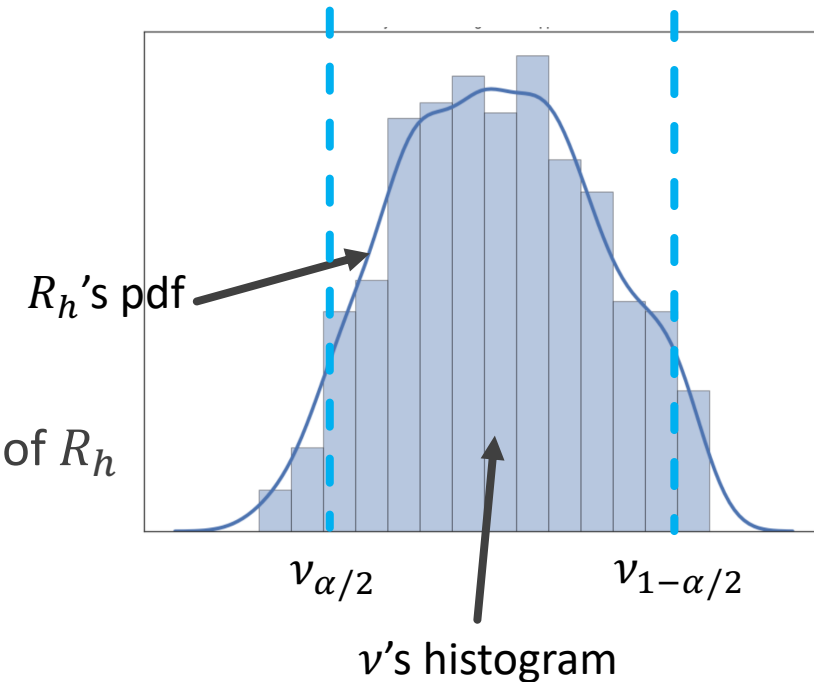
empirical distribution of $\{X_1, \dots, X_n\}$:
 $\frac{1}{n} \sum_{i=1}^n \delta_{X_i}$ where δ_X is a dirac delta function

- Goal: estimate property h of D ($:=h(D)$) using confidence intervals, using sample S (e.g. h =F1 of model f)

- Idea: estimate the distribution of $h(S) - h(D)$, denoted by R_h

by *bootstrapping* (resampling)

- perform n times of “sampling with replacement” from S
- repeat B times (e.g., $B \approx 10^4$) to obtain S_1, \dots, S_B
- take $\nu :=$ **empirical distribution** of $\{h(S_b) - h(S)\}_{b=1}^B$, as the **‘shape’** of R_h



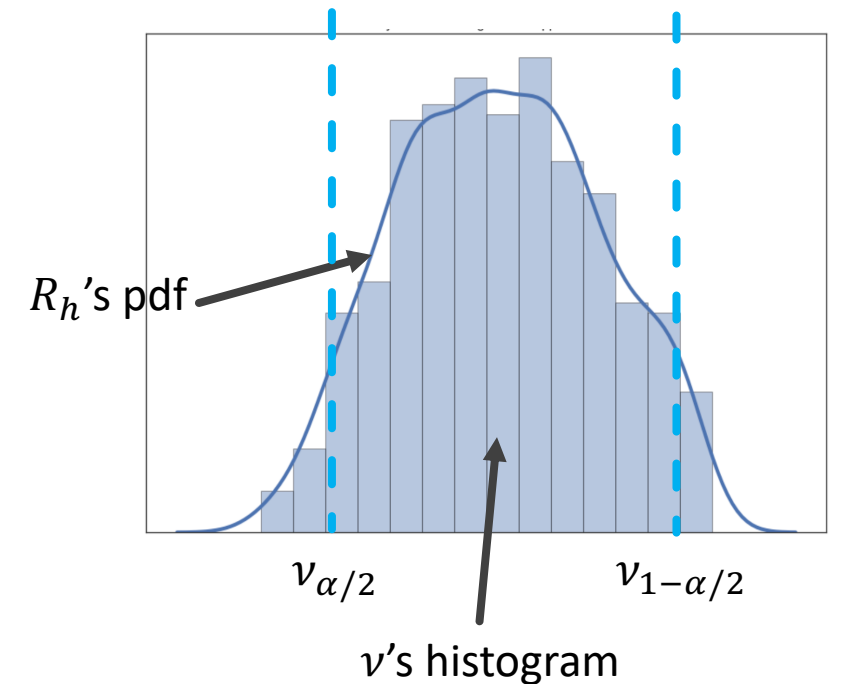
- Assumption: $h(S) - h(D) \sim R_h \approx \text{emp_distribution}[\{h(S_b) - h(S)\}_{b=1}^B]$

Quantile interval: sort values and take top/bottom-quantiles (see next slide)

- With prob. $\approx 1 - \alpha$, $h(S) - h(D) \in [\nu_{\alpha/2}, \nu_{1-\alpha/2}] \Rightarrow I(S) = [h(S) - \nu_{1-\alpha/2}, h(S) - \nu_{\alpha/2}]$

Bootstrapping CI: Implementation

- From bootstrapping, obtain $\{h(S_b) - h(S)\}_{b=1}^B$
- How to calculate its empirical distribution's quantiles?
 - Sort them in increasing order; say $v[0 \dots (B-1)]$
 - $v_{1-\alpha/2} :=$ the top 0.025 (i.e., $v[\text{int}(0.975*B)]$)
 - $v_{\alpha/2} :=$ the bottom 0.025 (i.e., $v[\text{int}(0.025*B)]$)



Hypothesis testing: motivation

- How to claim your new system A is better than existing one B
- Ex1: each test data point => take prediction from A & B => record correct/not
- Ex2: each evaluator => a random keyword is picked, and then both systems pick top 10 relevant documents and rank them => the evaluator provides rating (1-5) for both lists.

Evaluator	1	2	3	4	5	6	...
A	5	2	2	5	4	2	...
B	4	1	1	4	3	1	...

Two-sample hypothesis testing: definition

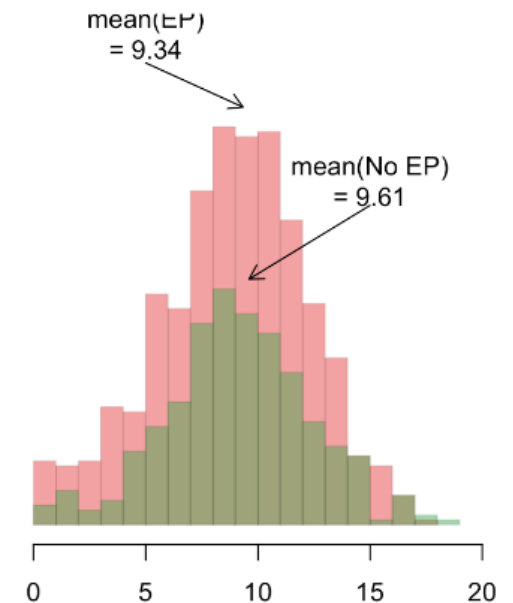
- Given D_θ with parameter θ
- Samples $S_X = (X_1, \dots, X_n)$ and $S_Y = (Y_1, \dots, Y_n)$ drawn iid from distribution D_{θ_X} and D_{θ_Y} , respectively

- Equality test version:

- Null hypothesis $H_0: \theta_X = \theta_Y$
- Alternative hypothesis $H_1: \theta_X \neq \theta_Y$

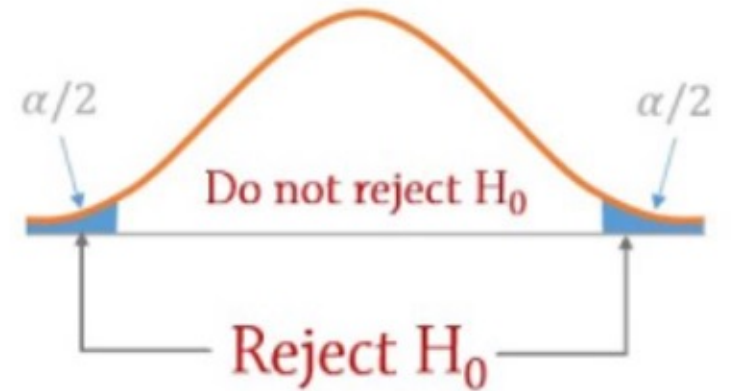
- E.g. $D_\mu = \text{Ber}(\mu)$, $H_0: \mu_X = \mu_Y$

- Similarly, design hypothesis tester T such that the two types of errors are controlled



Paired t-test

- $S_X = (X_1, \dots, X_n)$ and $S_Y = (Y_1, \dots, Y_n)$ drawn iid from distribution $D_{\theta_X} = N(\mu_X, \sigma_X^2)$ and $D_{\theta_Y} = N(\mu_Y, \sigma_Y^2)$, respectively
 - $H_0: \mu_X = \mu_Y$
 - $H_1: \mu_X \neq \mu_Y$
- Let $\delta_i := X_i - Y_i$, for all $i = 1, \dots, n$
- Let $\bar{\delta}_n := \frac{1}{n} \sum_{i=1}^n \delta_i$
- Design hypothesis test T so that $P_{H_0}(T(S) = 0) \geq 1 - \alpha$
- Intuition: reasonable to reject if $|\bar{\delta}_n|$ is large

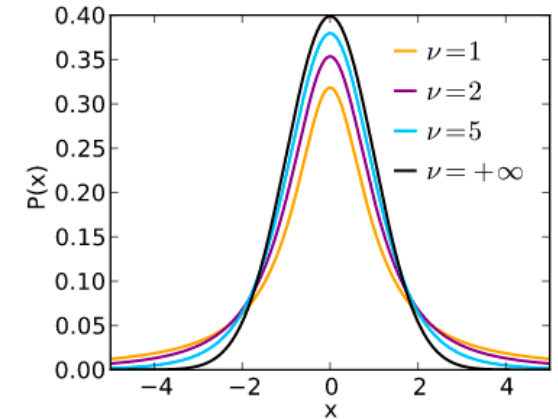


Paired t-test

- Under H_0 , $\delta_i \sim N(0, \sigma^2)$, $i = 1, \dots, n$, where $\sigma^2 = \sigma_X^2 + \sigma_Y^2$

- Recall Thm: Let $\delta_1, \dots, \delta_n \sim N(0, \sigma^2)$, and $\bar{\delta}_n := \frac{1}{n} \sum_{i=1}^n \delta_i$, $\hat{\sigma}_n^2 := \frac{\sum_{i=1}^n (\delta_i - \bar{\delta}_n)^2}{n-1}$

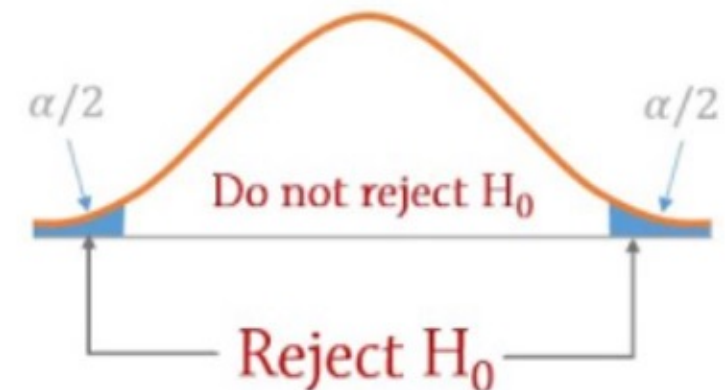
$$Z = \sqrt{n} \frac{\bar{\delta}_n}{\hat{\sigma}_n} \sim \text{student-t (mean 0, scale 1, degrees of freedom = } n - 1)$$



- Let's ask "under H_0 , what is a plausible range of values of Z with failure rate $\alpha = 0.05$?"

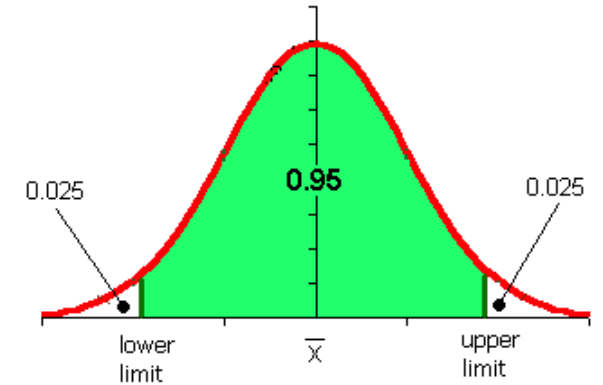
- Find the 0.025, 0.975-quantiles of $Z \Rightarrow t_{0.025}, t_{0.975}$
- Hypothesis tester

$$T(S) = I(Z \notin [t_{0.025}, t_{0.975}]) = I\left(\sqrt{n} \frac{\bar{\delta}_n}{\hat{\sigma}_n} \notin [t_{0.025}, t_{0.975}]\right)$$

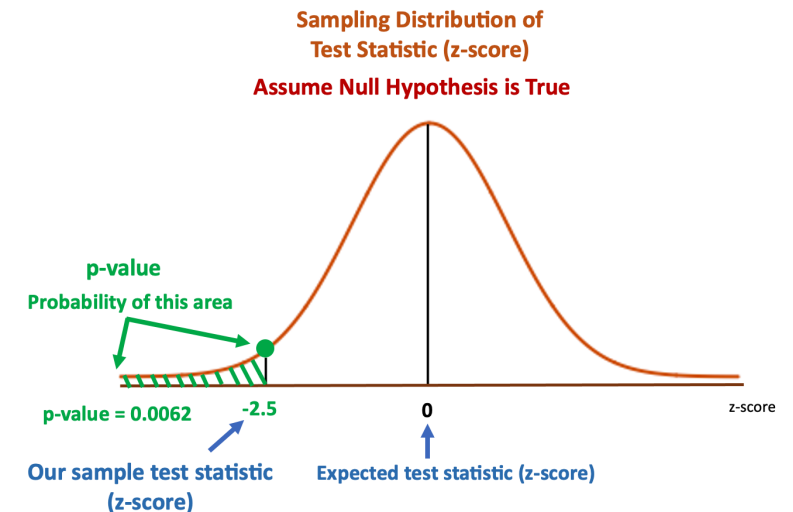


Hypothesis testing: additional remarks

- Confidence intervals can be used for hypothesis testing
 - $S = (X_1, \dots, X_n)$ drawn iid from distribution D_μ
 - $H_0: \mu = 0$
 - $H_1: \mu \neq 0$
 - I is a $(1 - \alpha)$ -CI construction for $\mu \Rightarrow$ hypothesis test $T(S) = I(0 \notin I(S))$ has significance α



- p-value: given dataset S , and a family of hypothesis tests T_α 's with different significance α 's
 p = the smallest α with which you can still reject H_0

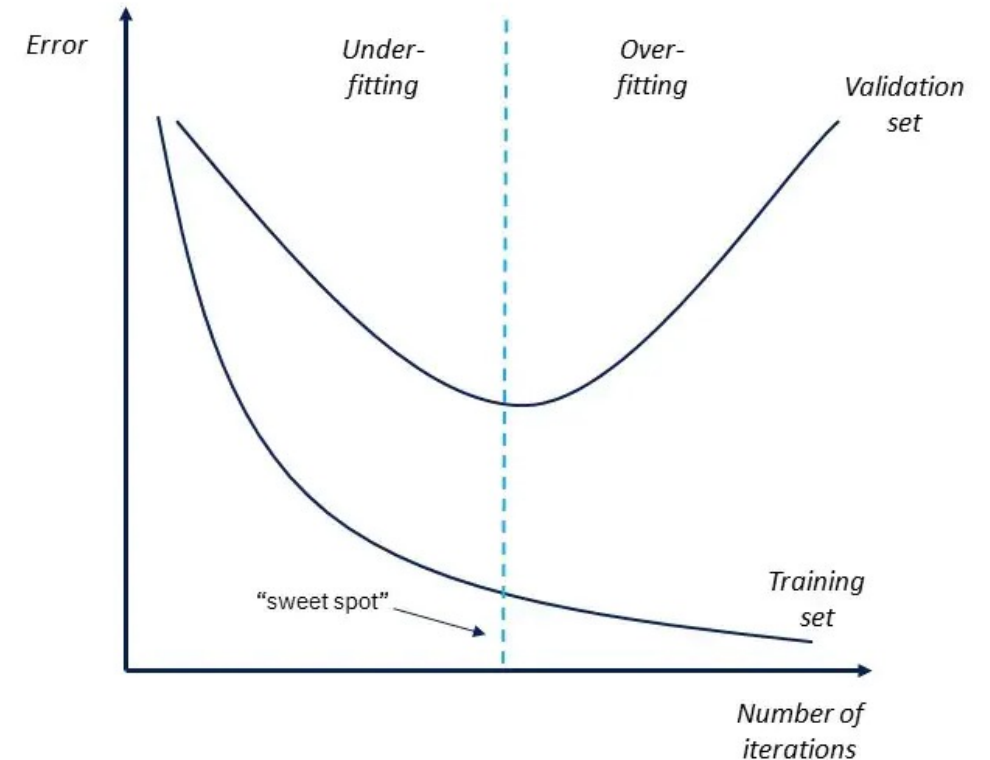


Debugging Learning Algorithms

Debugging Learning Algorithms

Is the problem with generalization to test data?

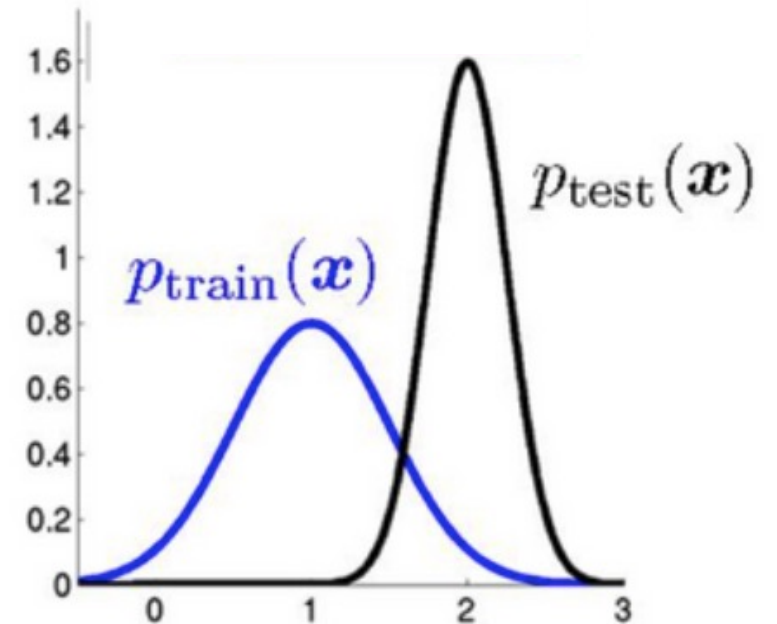
- Is it doing well on training?
- Unrealistic to do *better* on test than on training
- If it does well on training then problem is *generalization*
- Model may be too complicated (overfitting)
- Too many features, not enough training data
- Otherwise, problem may be *representation* : need better features or better data



Debugging Learning Algorithms

Is there a mismatch between training and test?

- Training data may be inadequate
- Do results change with different train / test split?
- If so then test distribution is probably strange
- Otherwise you have other generalization problems...



Debugging Learning Algorithms

Is the learning algorithm implemented correctly?

- Is it optimizing the loss function that you intended?
- Try measuring / visualizing your loss function during training-is it going down?
- Do the data meet your algorithmic assumptions?
- Hand-craft datasets where you know the desired behavior
 - KNN on XOR function
 - Perceptron on data that is trivially linearly-separable ($y=x+1$ and $y=x-1$)
 - Decision tree on axis-aligned data
 - Generally, create dataset that meets assumptions of your algorithm

Debugging Learning Algorithms

Do you have adequate representation?

- Your feature set could be inadequate
- For binary classification try this...
 - Add a feature (maybe call it `CheatingIsFun`)
 - Set value to +1 for positive instances and -1 for negative instances
 - This feature is a *perfect indicator*-problem is now trivially solvable
 - Does your algorithm solve it?
- If your algorithm doesn't get near 0% error then you *may* have a bug! (or more data / less features)
- If it does then you need better features or a different model (e.g. decision tree vs. linear model)

Debugging Learning Algorithms

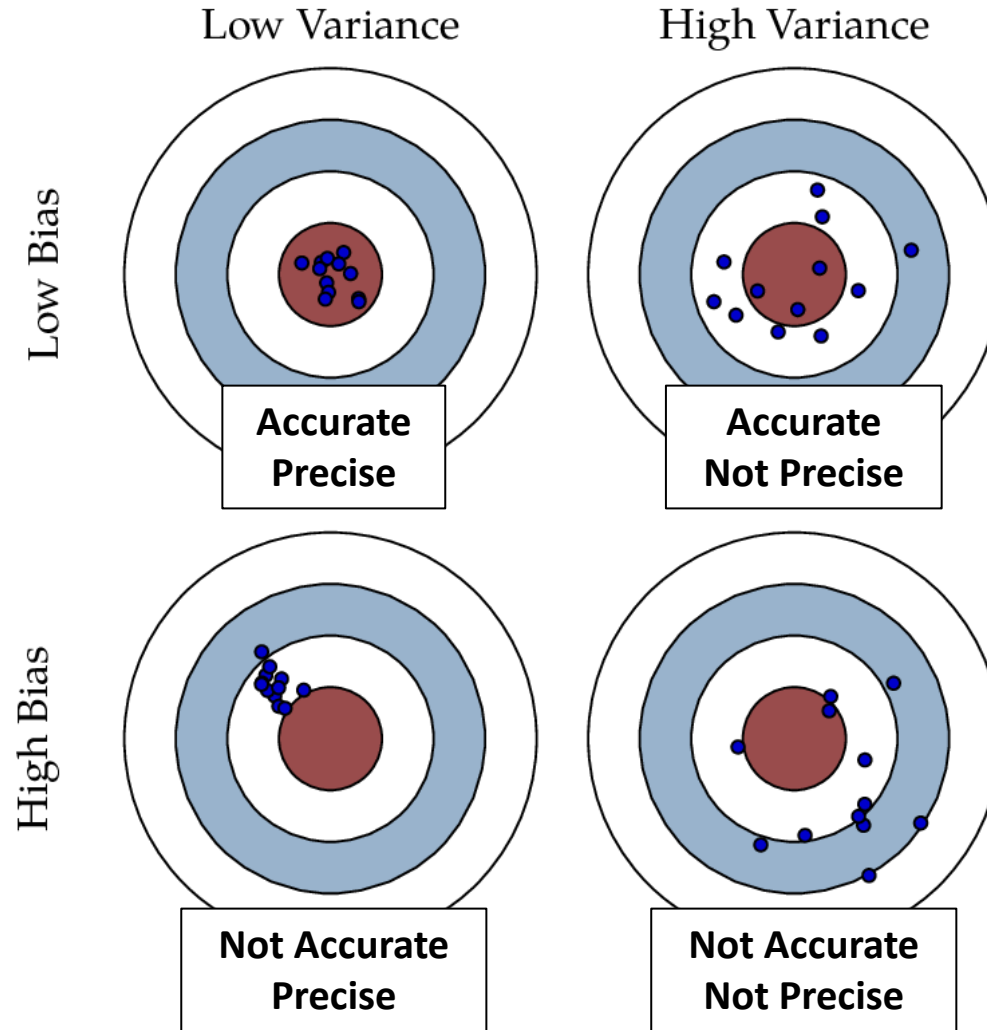
Do you have enough data?

- Always have *at least* as many training data as you have learnable model parameters
- Try training on 80% of your training data
 - Does performance suffer?
 - How much? A lot?
 - If so then getting more data is likely helpful
 - If not then you may be *data saturated*-look elsewhere
- More training data should never lead to *worse* performance (just slower training)

Bias / Variance Tradeoff

Bias-Variance Tradeoff

Suppose an archer takes multiple shots at a target...



Bias-Variance Tradeoff

Is an unbiased estimator “better” than a biased one? It depends...

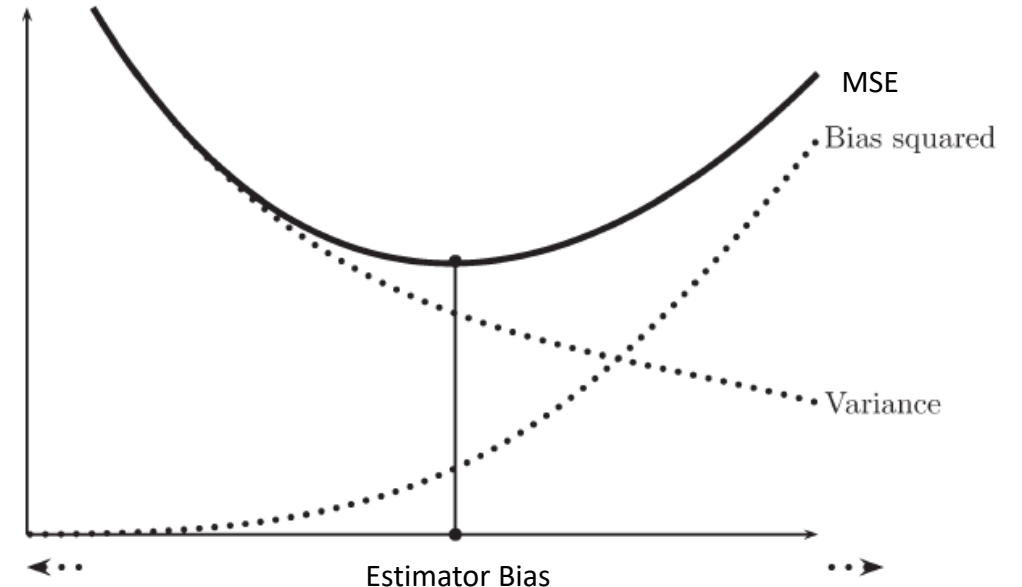
Evaluate the quality of estimate $\hat{\theta}$ using **mean squared error**,

$$\text{MSE}(\hat{\theta}) = \mathbf{E} \left[(\hat{\theta} - \theta)^2 \right] = \text{bias}^2(\hat{\theta}) + \mathbf{Var}(\hat{\theta})$$

- MSE for unbiased estimators is just,

$$\text{MSE}(\hat{\theta}) = \mathbf{Var}(\hat{\theta})$$

- Bias-variance is fundamental tradeoff in statistical estimation
- MSE increases as **square** of bias
- Estimators with small bias (but low variance) can have lower MSE than unbiased estimators



Bias-Variance Decomposition

$$\begin{aligned}\text{MSE}(\hat{\theta}) &= \mathbf{E} \left[(\hat{\theta}(X) - \theta)^2 \right] \\ &= \mathbf{E} \left[\left(\hat{\theta} - \mathbf{E}[\hat{\theta}] + \mathbf{E}[\hat{\theta}] - \theta \right)^2 \right] \\ &= \mathbf{E}[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2] + 2(\mathbf{E}[\hat{\theta}] - \theta)\mathbf{E}[\hat{\theta} - \mathbf{E}[\hat{\theta}]] + \mathbf{E}[(\mathbf{E}[\hat{\theta}] - \theta)^2] \\ &= \mathbf{E}[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2] + \left(\mathbf{E}[\hat{\theta}] - \theta \right)^2 \\ &= \text{Var}(\hat{\theta}) + \text{bias}^2(\hat{\theta})\end{aligned}$$

Other materials

- Bootstrap test: https://ocw.mit.edu/courses/mathematics/18-05-introduction-to-probability-and-statistics-spring-2014/readings/MIT18_05S14_Reading24.pdf
- Permutation test: <https://www.jwilber.me/permutationtest/>
- STAT 566 lecture slides (at UA): <https://www.math.arizona.edu/~jwatkins/stat566s20s.html>

Next lecture (9/19)

- Linear models revisited: classification, regression, loss minimization formulations
- Assigned reading: CIML Chapter 7
- *Note: We are skipping Chapter 6 for now!*