

DS 4400

Machine Learning and Data Mining I

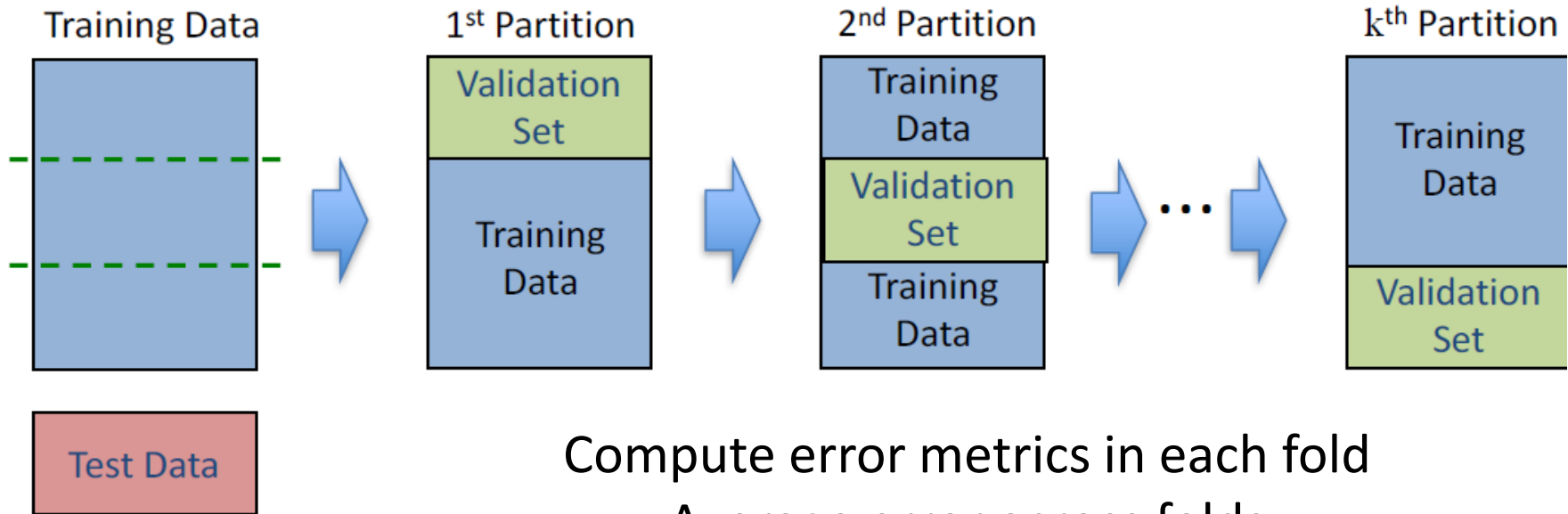
Alina Oprea
Associate Professor, CCIS
Northeastern University

February 7 2019

Outline

- Linear Discriminant Analysis (LDA)
- Lab (logistic regression, LDA, kNN)
- Feature selection
 - Wrapper
 - Filter
 - Embedded methods
- Decision trees
 - Information Gain

Cross Validation



- CV can be used for
 - Hyper-parameter selection
 - Comparing different models and features
- **1. k-fold Cross-Validation**
 - Split data into k partitions of equal size

LDA

- Classify to one of k classes
- Logistic regression computes directly
 - $P[Y = 1|X = x]$
 - Assume sigmoid function
- LDA uses Bayes Theorem to estimate it
 - $P[Y = k|X = x] = \frac{P[X = x|Y = k]P[Y=k]}{P[X=x]}$
 - Let $\pi_k = P[Y = k]$ be the prior probability of class k and $f_k(x) = P[X = x|Y = k]$

LDA

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}.$$

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Assumption: $\sigma_1 = \dots \sigma_k = \sigma$

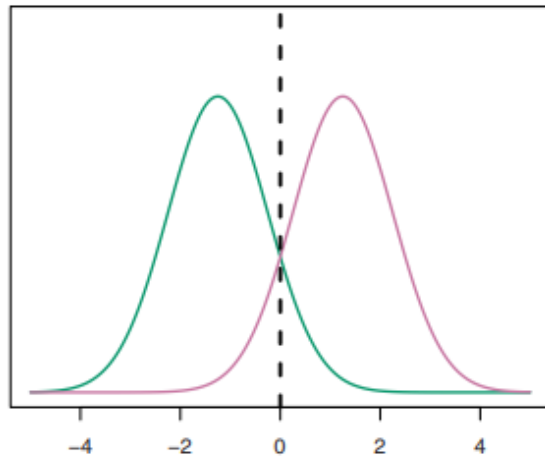
LDA decision boundary

Pick class k to maximize

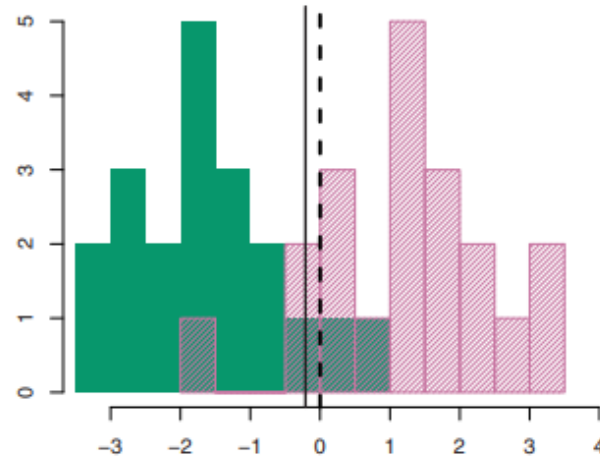
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2, \pi_1 = \pi_2$

Classify as class 1 if $x > \frac{\mu_1 + \mu_2}{2\sigma}$



True decision boundary



Estimated decision boundary

LDA in practice

Given training data $(x^{(i)}, y^{(i)}), i = 1, \dots, n, y^{(i)} \in \{1, \dots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x^{(i)}$$
$$\hat{\sigma}^2 = \frac{1}{n - K} \sum_{k=1}^K \sum_{i:y_i=k} (x^{(i)} - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k / n.$$

Given testing point x , predict k that maximizes:

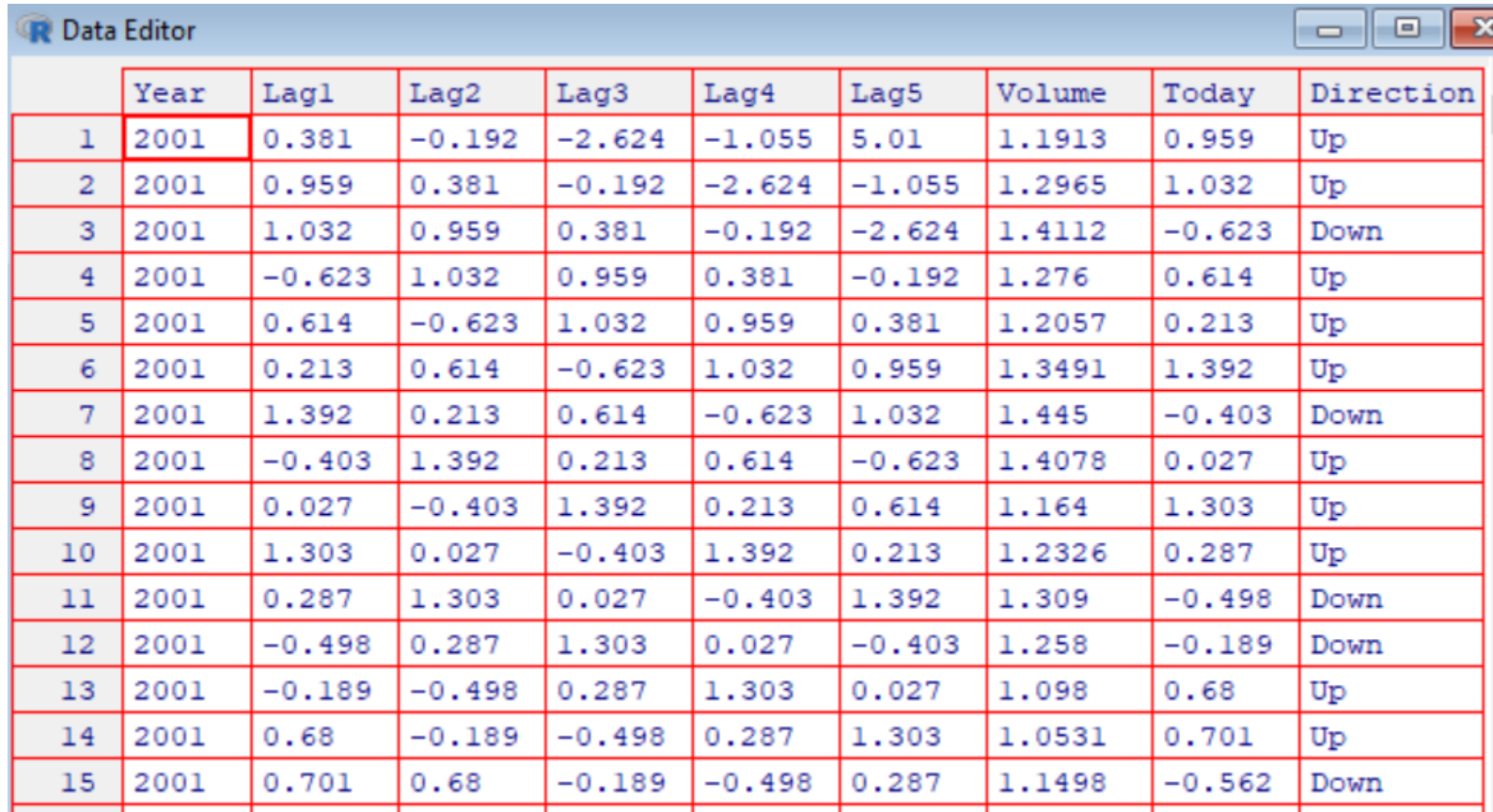
$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

LDA vs Logistic Regression

- Logistic regression computes directly $\Pr[Y = 1|X = x]$ by assuming sigmoid function
 - Uses Maximum Likelihood Estimation
 - Discriminative Model
- LDA uses Bayes Theorem to estimate it
 - Estimates mean, co-variance, and prior from training data
 - Generative model
 - Assumes Gaussian distribution for $f_k(x) = \Pr[X = x|Y = k]$
- Which one is better?
 - LDA can be sensitive to outliers
 - LDA works well for Gaussian distribution
 - Logistic regression is more complex to solve, but more expressive

Lab

```
> library(ISLR)  
> fix(Smarket)
```



The screenshot shows the R Data Editor window with a table of market data for the year 2001. The table has 10 columns: Year, Lag1, Lag2, Lag3, Lag4, Lag5, Volume, Today, and Direction. The data is presented in 15 rows, with the first row highlighted in red. The 'Year' column contains the value '2001' for all rows. The 'Direction' column contains 'Up' or 'Down' for each row.

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
1	2001	0.381	-0.192	-2.624	-1.055	5.01	1.1913	0.959	Up
2	2001	0.959	0.381	-0.192	-2.624	-1.055	1.2965	1.032	Up
3	2001	1.032	0.959	0.381	-0.192	-2.624	1.4112	-0.623	Down
4	2001	-0.623	1.032	0.959	0.381	-0.192	1.276	0.614	Up
5	2001	0.614	-0.623	1.032	0.959	0.381	1.2057	0.213	Up
6	2001	0.213	0.614	-0.623	1.032	0.959	1.3491	1.392	Up
7	2001	1.392	0.213	0.614	-0.623	1.032	1.445	-0.403	Down
8	2001	-0.403	1.392	0.213	0.614	-0.623	1.4078	0.027	Up
9	2001	0.027	-0.403	1.392	0.213	0.614	1.164	1.303	Up
10	2001	1.303	0.027	-0.403	1.392	0.213	1.2326	0.287	Up
11	2001	0.287	1.303	0.027	-0.403	1.392	1.309	-0.498	Down
12	2001	-0.498	0.287	1.303	0.027	-0.403	1.258	-0.189	Down
13	2001	-0.189	-0.498	0.287	1.303	0.027	1.098	0.68	Up
14	2001	0.68	-0.189	-0.498	0.287	1.303	1.0531	0.701	Up
15	2001	0.701	0.68	-0.189	-0.498	0.287	1.1498	-0.562	Down

Lab Logistic Regression

Train on data before 2005

```
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252  9
> Direction.2005=Direction[!train]
> glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket, family=binomial, subset=train)
> summary(glm.fits)
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
     Volume, family = binomial, data = Smarket, subset = train)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.302  -1.190   1.079   1.160   1.350
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.191213  0.333690   0.573   0.567
Lag1        -0.054178  0.051785  -1.046   0.295
Lag2        -0.045805  0.051797  -0.884   0.377
Lag3         0.007200  0.051644   0.139   0.889
Lag4         0.006441  0.051706   0.125   0.901
Lag5        -0.004223  0.051138  -0.083   0.934
Volume      -0.116257  0.239618  -0.485   0.628
```

Lab Logistic Regression

Test on data in 2005

```
>
> glm.probs=predict(glm.fits,Smarket.2005,type="response")
> glm.pred=rep("Down",nrow(Smarket.2005))
> glm.pred[glm.probs>.5]="Up"
> head(Smarket.2005)
      Year  Lag1  Lag2  Lag3  Lag4  Lag5 Volume  Today Direction
999  2005 -0.134  0.008 -0.007  0.715 -0.431  0.7869 -0.812   Down
1000 2005 -0.812 -0.134  0.008 -0.007  0.715  1.5108 -1.167   Down
1001 2005 -1.167 -0.812 -0.134  0.008 -0.007  1.7210 -0.363   Down
1002 2005 -0.363 -1.167 -0.812 -0.134  0.008  1.7389  0.351    Up
1003 2005  0.351 -0.363 -1.167 -0.812 -0.134  1.5691 -0.143   Down
1004 2005 -0.143  0.351 -0.363 -1.167 -0.812  1.4779  0.342    Up
> head(glm.probs)
      999      1000      1001      1002      1003      1004
0.5282195 0.5156688 0.5226521 0.5138543 0.4983345 0.5010912
> head(glm.pred)
[1] "Up"  "Up"  "Up"  "Up"  "Down" "Up"
> table(glm.pred,Direction.2005)
      Direction.2005
glm.pred Down Up
      Down   77 97
      Up    34 44
> mean(glm.pred==Direction.2005)
[1] 0.4801587
```

Lab LDA

```
<
> library(MASS)
> lda.fit=lda(Direction~Lag1+Lag2,data=Smarket,subset=train)
> lda.fit
Call:
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
      Down      Up
0.491984 0.508016

Group means:
      Lag1      Lag2
Down 0.04279022 0.03389409
Up   -0.03954635 -0.03132544

Coefficients of linear discriminants:
      LD1
Lag1 -0.6420190
Lag2 -0.5135293
> lda.pred=predict(lda.fit, Smarket.2005)
> lda.class=lda.pred$class
> table(lda.class,Direction.2005)
      Direction.2005
lda.class Down  Up
      Down   35  35
      Up    76 106
> mean(lda.class==Direction.2005)
[1] 0.5595238
<
```

Lab kNN

```
>
> library(class)
> train.X=cbind(Lag1,Lag2) [train,]
> test.X=cbind(Lag1,Lag2) [!train,]
> train.Direction=Direction[train]
> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Direction,k=1)
> table(knn.pred,Direction.2005)
      Direction.2005
knn.pred Down Up
      Down   43 58
      Up    68 83
> mean(knn.pred==Direction.2005)
[1] 0.5
> knn.pred=knn(train.X,test.X,train.Direction,k=3)
> table(knn.pred,Direction.2005)
      Direction.2005
knn.pred Down Up
      Down   48 54
      Up    63 87
> mean(knn.pred==Direction.2005)
[1] 0.5357143
> knn.pred=knn(train.X,test.X,train.Direction,k=7)
> table(knn.pred,Direction.2005)
      Direction.2005
knn.pred Down Up
      Down   41 65
      Up    70 76
> mean(knn.pred==Direction.2005)
[1] 0.4642857
```

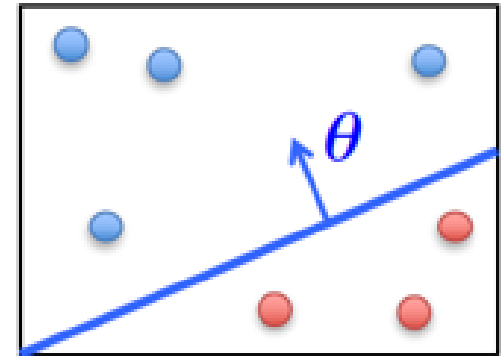
Linear models

- Perceptron

$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^\top \mathbf{x})$$

- Logistic regression

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \mathbf{x}}}$$

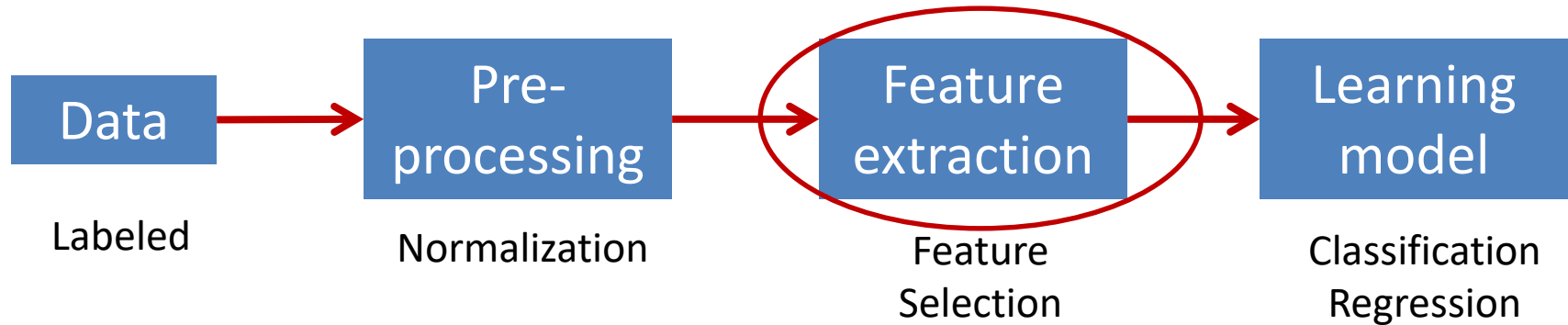


- LDA

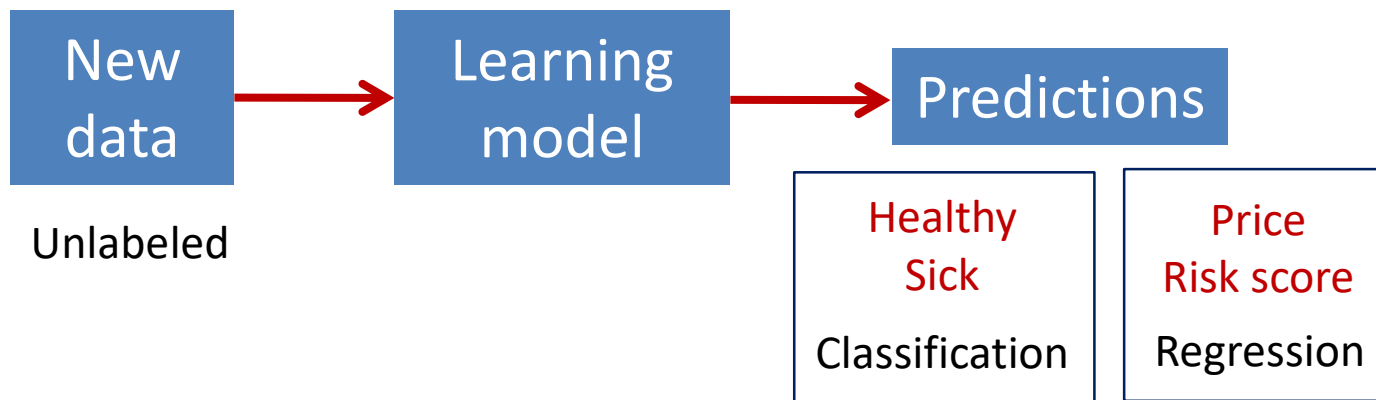
$$\text{Max}_k \delta_k(\mathbf{x}) = \mathbf{x} \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Supervised Learning

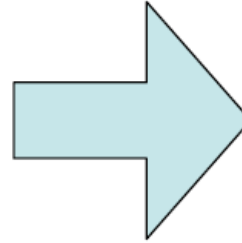
Training



Testing



Example: Email Classification



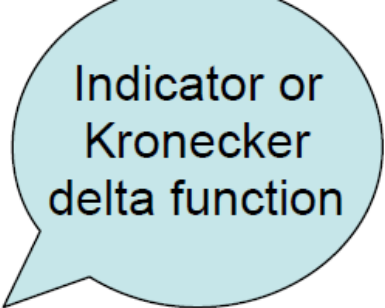
PERSONAL

- Input: a email message
- Output: is the email...
 - spam,
 - work-related,
 - personal, ...

Bag-of-Words

- Input: \mathbf{x} (email-valued)
- Feature vector:

$$f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}, \quad \text{e.g. } f_1(\mathbf{x}) = \begin{cases} 1 & \text{if the email contain: Boston} \\ 0 & \text{otherwise} \end{cases}$$



Indicator or
Kronecker
delta function

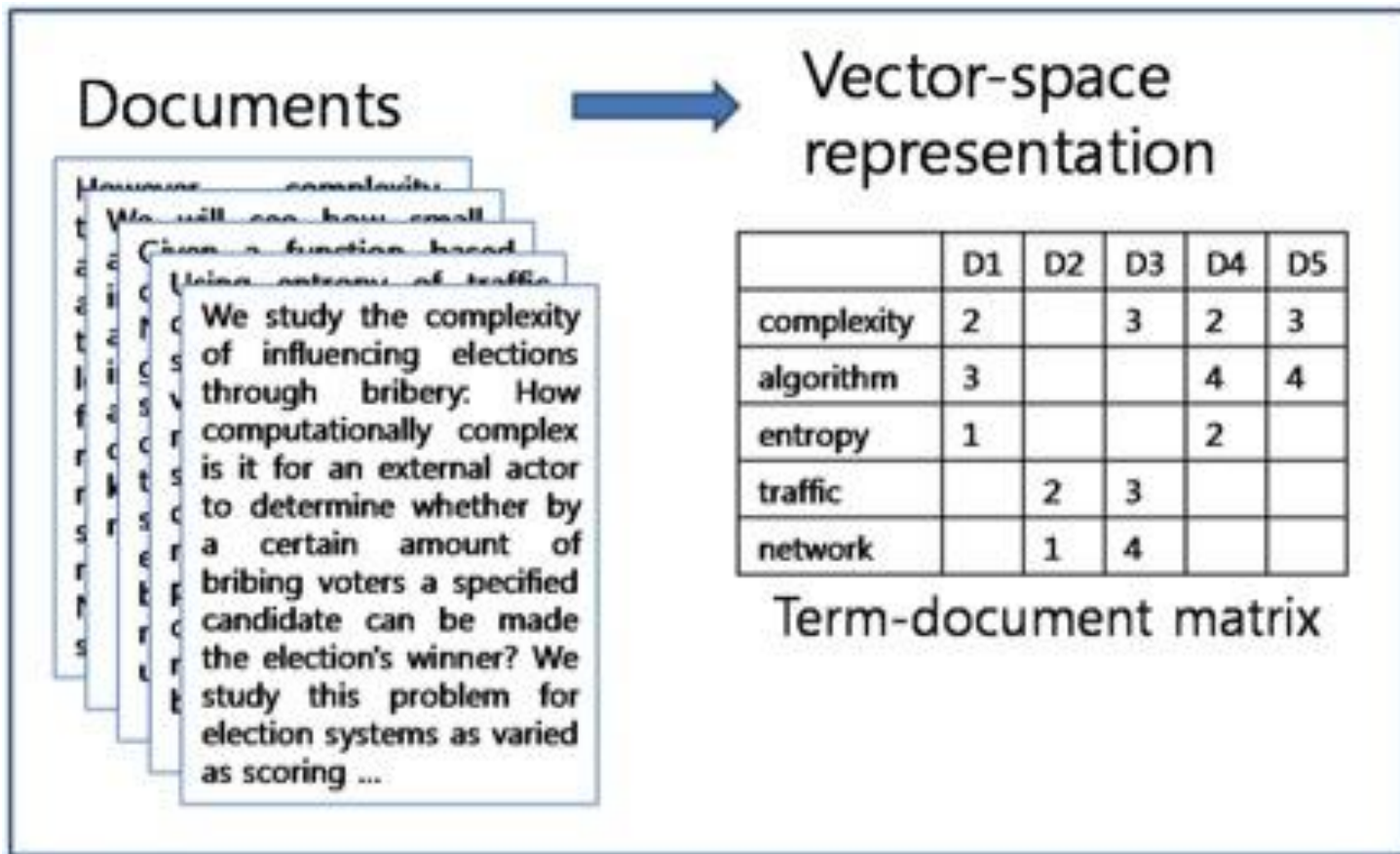
- Learn one weight vector for each class:

$$w_y \in \mathbb{R}^n, \quad y \in \{\text{SPAM, WORK, PERS}\}$$

Could also use frequency

- $f_i(\mathbf{x})$ is the number of times word i appears in \mathbf{x}

Representation



- Large number of words in dictionary (>50,000)
- Very sparse representation (many features set at 0)

Feature selection

- *Feature Selection*

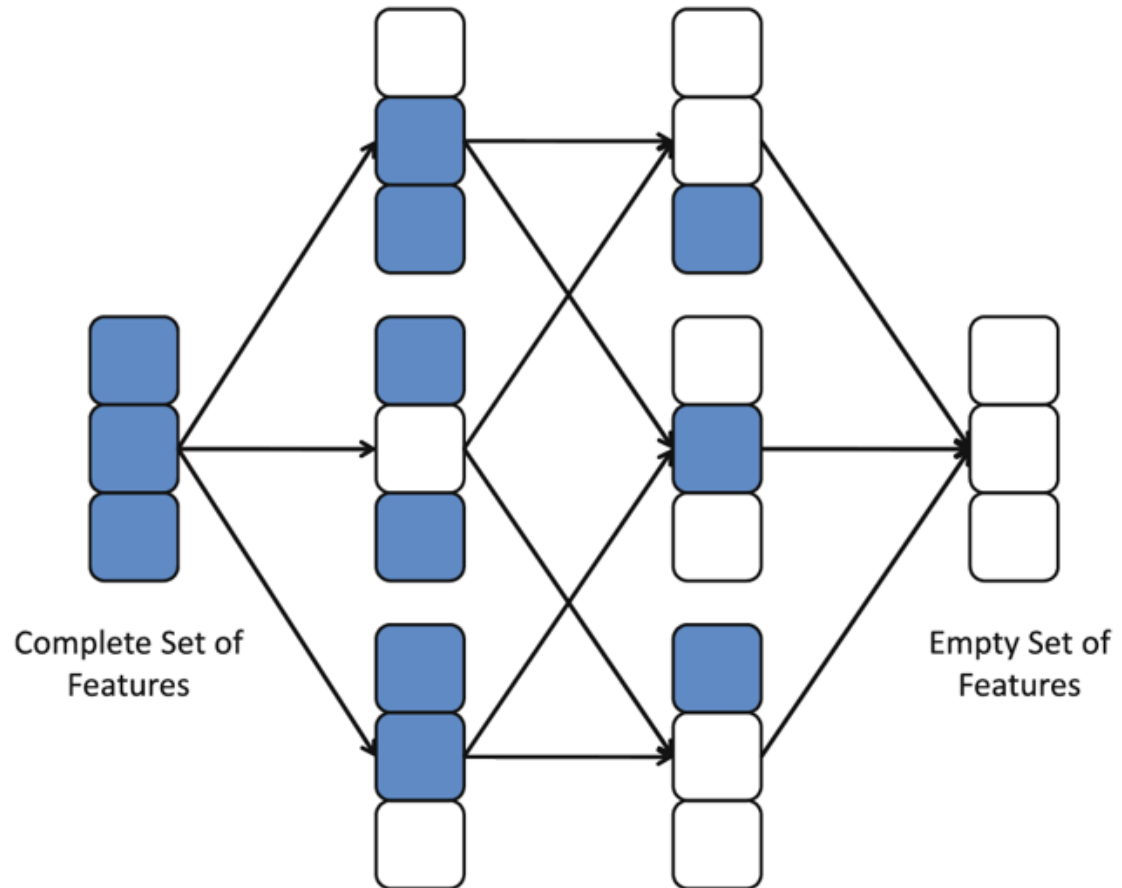
- Process for choosing an optimal subset of features according to a certain criteria

- Why we need Feature Selection:

1. To improve performance (in terms of speed, predictive power, simplicity of the model).
2. To visualize the data for model selection.
3. To reduce dimensionality and remove noise.

Feature Search Space

Search Space:



Exponentially large!

Methods for Feature Selection

- **Wrappers**
 - Select subset of features that gives best prediction accuracy (using cross-validation)
 - Model-specific
- **Filters**
 - Compute some statistical metrics (correlation coefficient, mutual information)
 - Select features with statistics higher than threshold
- **Embedded methods**
 - Feature selection done as part of training
 - Example: Regularization (Lasso, L1 regularization)

Feature Engineering

- Feature engineering is crucial to getting good results
- Strategy: overshoot and regularize
 - Define as many features as you can
 - Use regularization for models that support it
 - Use other feature selection methods (e.g., filters) otherwise
- Do cross-validation to evaluate selected features on multiple runs
- When feature selection is frozen, evaluate on test set

Wrappers: Search Strategy

❖ With an **exhaustive search**

101110000001000100001000000000100101010

With d features $\rightarrow 2^d$ possible feature subsets.

20 features ... 1 million feature sets to check

25 features ... 33.5 million sets

30 features ... 1.1 billion sets

❖ Need for a **search strategy**

- Sequential forward selection
- Recursive backward elimination
- Genetic algorithms
- Simulated annealing
- ...

Wrappers: Sequential Forward Selection

Start with the empty set $S = \emptyset$

While *stopping criteria not met*

For each feature X_f not in S

- Define $S' = S \cup \{X_f\}$
- Train model using the features in S'
- Compute the accuracy on validation set

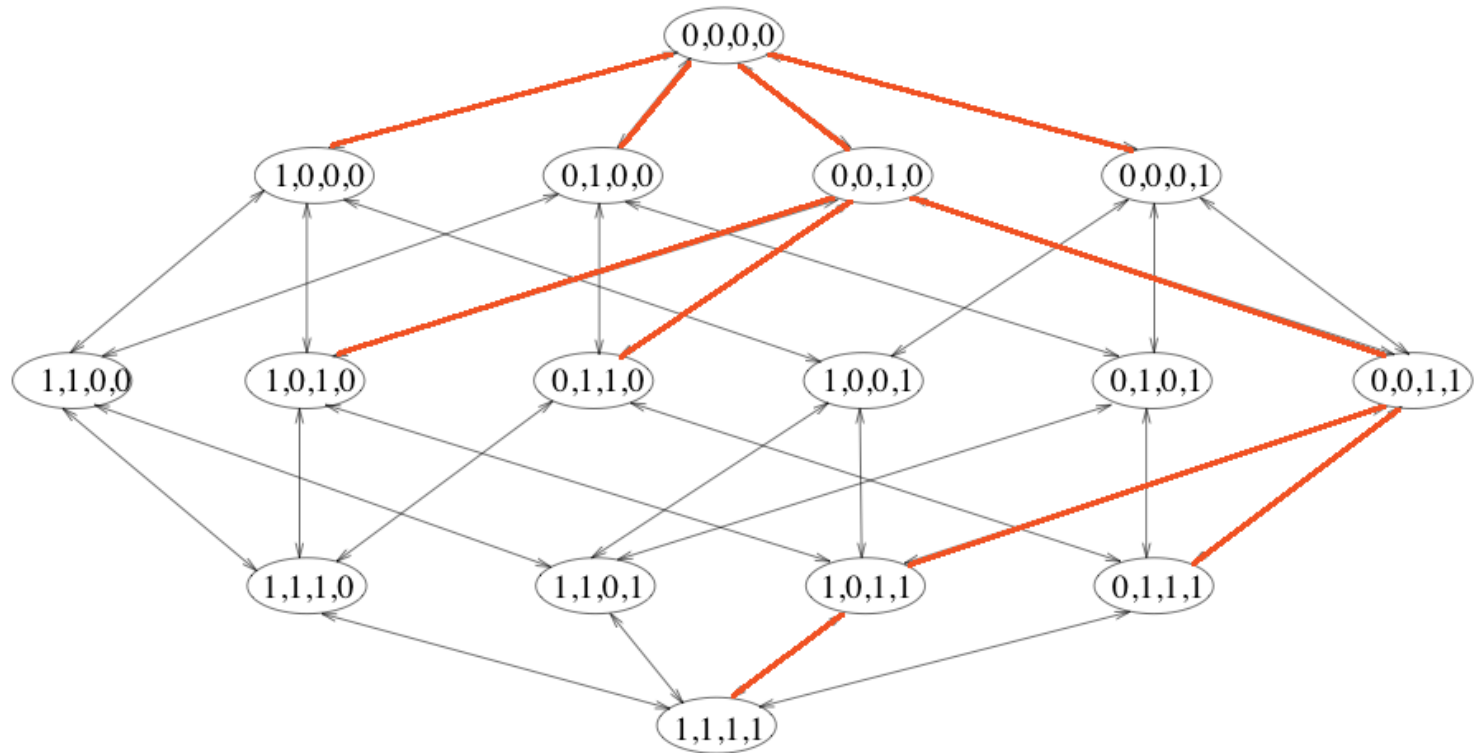
End

$S = S'$ where S' is the feature set with the greatest accuracy

End

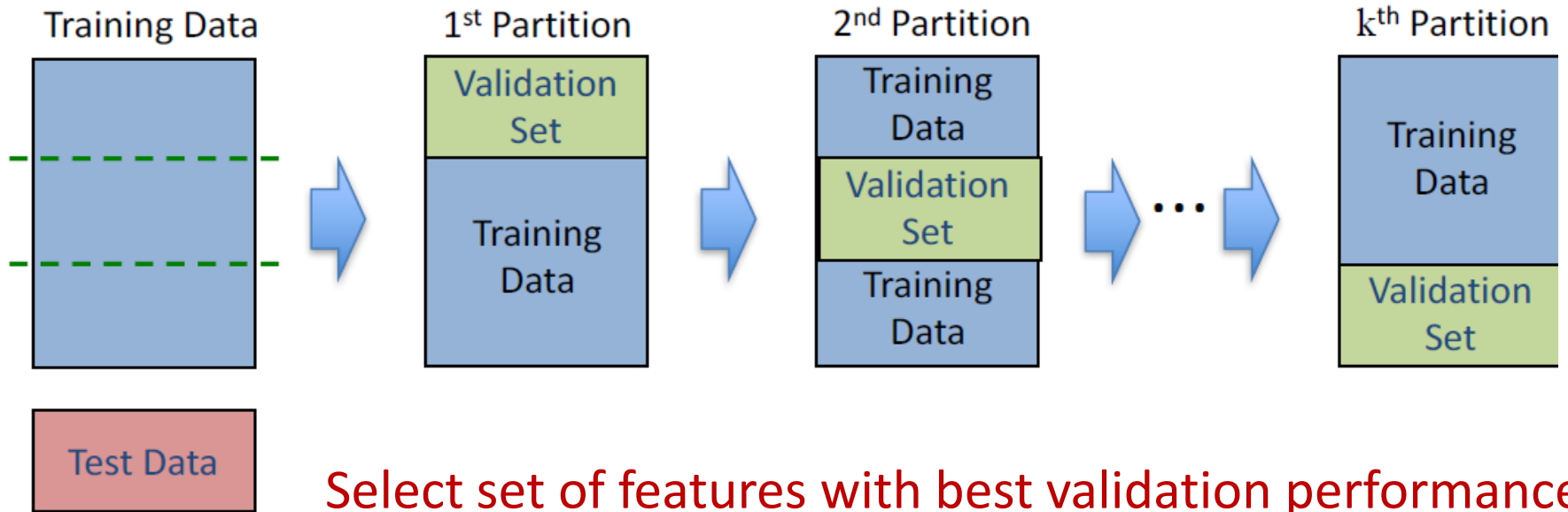
Backward feature selection starts with all features and eliminates backward

Search complexity for sequential forward selection



- Evaluates $\frac{d(d+1)}{2}$ features sets instead of 2^d

Cross Validation



- k-fold CV
 - Split data into k partitions of equal size
- Leave-one-out CV (LOOCV)
 - $k=n$ (validation set only one point)

Filters

Principle: *replace evaluation of model with quick to compute statistics $J(X_f)$*

k	$J(X_k)$
35	0.846
42	0.811
10	0.810
654	0.611
22	0.443
59	0.388
...	...
212	0.09
39	0.05

For each feature X_f

- Compute $J(X_f)$

End

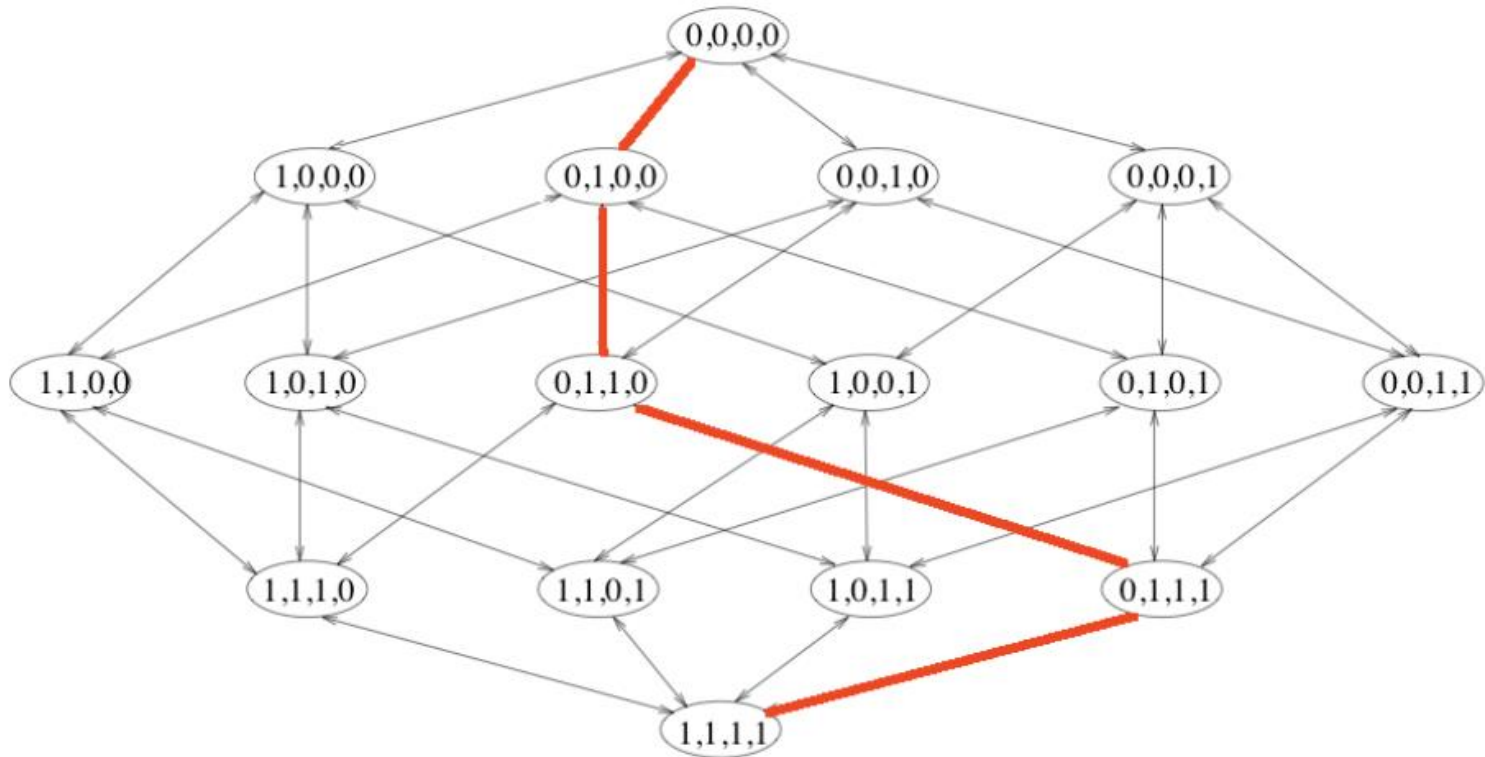
Rank features according to $J(X_f)$

Choose manual cut-off point

Examples of filtering criterion

- The mutual information with the target variable $J(X_f) = I(X_f; Y)$
- The correlation with the target variable
- χ^2 - statistic

Search Complexity for Filter Methods



Pros:

- A lot less expensive!

Cons:

- Not model-oriented

Embedded methods: Regularization

Lasso regression

$$J(\theta) = \underbrace{\sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)})^2}_{\text{Squared Residuals}} + \lambda \underbrace{\sum_{j=1}^d |\theta_j|}_{\text{Regularization}}$$

- L1 norm for regularization
- No closed form solution
- Algorithms based on gradient descent or quadratic programming

Embedded methods: Regularization

Principle: the classifier performs feature selection as part of the learning procedure

Example: the **logistic LASSO** (Tibshirani, 1996)

$$f(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x})}} = P(Y = 1 | \mathbf{x})$$

With Error Function:

$$E = - \underbrace{\sum_{i=1}^N \{y_i \log f(\mathbf{x}_i) + (1 - y_i) \log(1 - f(\mathbf{x}_i))\}}_{\text{Cross-entropy error}} + \lambda \underbrace{\sum_{f=1}^d |w_f|}_{\text{Regularizing term}}$$

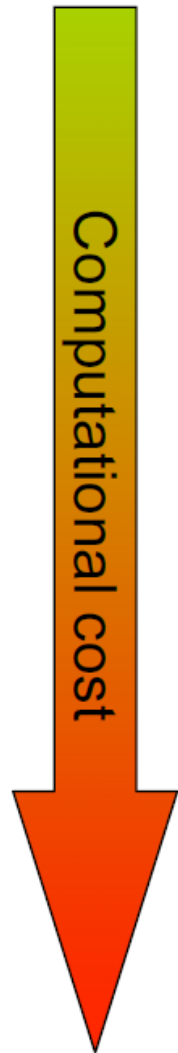
Pros:

- Performs feature selection as part of learning the procedure

Cons:

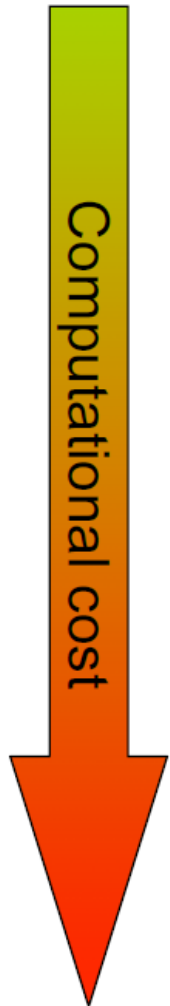
- Computationally demanding

Summary: Feature Selection



- Filtering
- L_1 regularization
(embedded methods)
- Wrappers
 - Forward selection
 - Backward selection
 - Other search
 - Exhaustive

Summary: Feature Selection



- Filtering

- L_1 regularization (embedded methods)

- Wrappers

- Forward selection

- Backward selection

- Other search

- Exhaustive

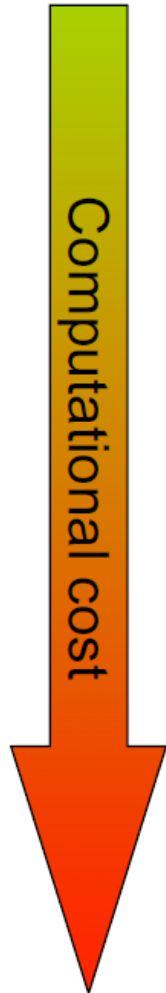
- Good preprocessing step



- Fails to capture relationship between features



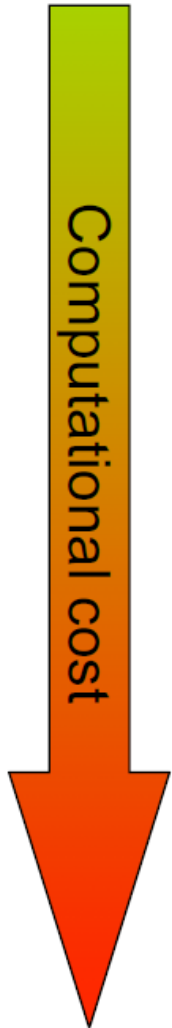
Summary: Feature Selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
 - Forward selection
 - Backward selection
 - Other search
 - Exhaustive

- Can add regularization in optimization objective ✓
- Can be solved with Gradient Descent ✓
- Can be applied to many models (e.g., linear or logistic regression) ✓
- Can not be applied to all methods (e.g., kNN) ✗

Summary: Feature Selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
 - Forward selection
 - Backward selection
 - Other search
 - Exhaustive

- Most directly optimize prediction performance ✓
- Can be very expensive, even with greedy search methods ✗
- Cross-validation is a good objective function to start with

Outline

- Linear Discriminant Analysis (LDA)
- Lab (logistic regression, LDA, kNN)
- Feature selection
 - Wrapper
 - Filter
 - Embedded methods
- Decision trees
 - Information Gain

Sample Dataset

- Columns denote features X_i
- Rows denote labeled instances $\langle x^{(i)}, y^{(i)} \rangle$
- Class label denotes whether a tennis game was played

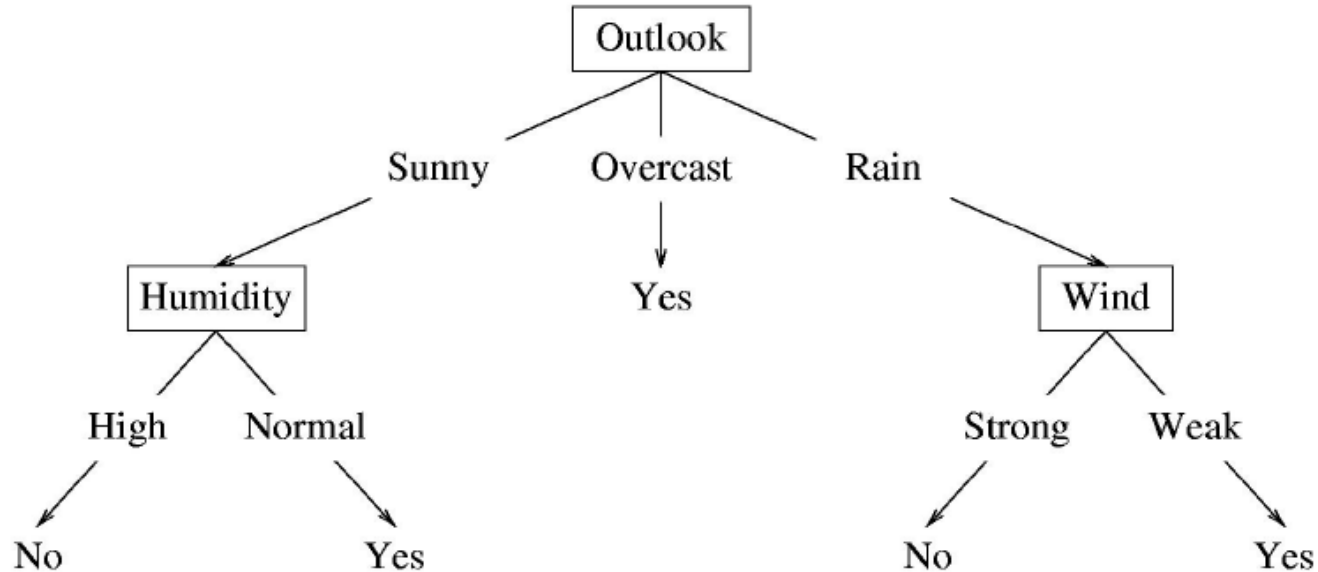
Predictors				Response
Outlook	Temperature	Humidity	Wind	Class
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

$\langle x^{(i)}, y^{(i)} \rangle$

Categorical
data

Decision Tree

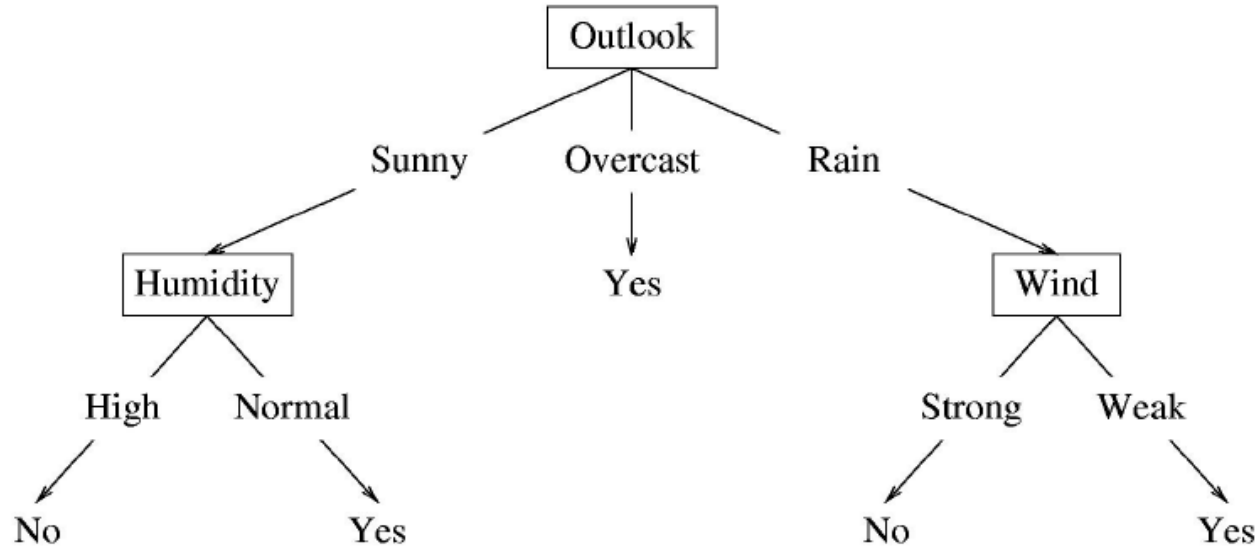
- A possible decision tree for the data:



- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y | x \in \text{leaf})$)

Decision Tree

- A possible decision tree for the data:

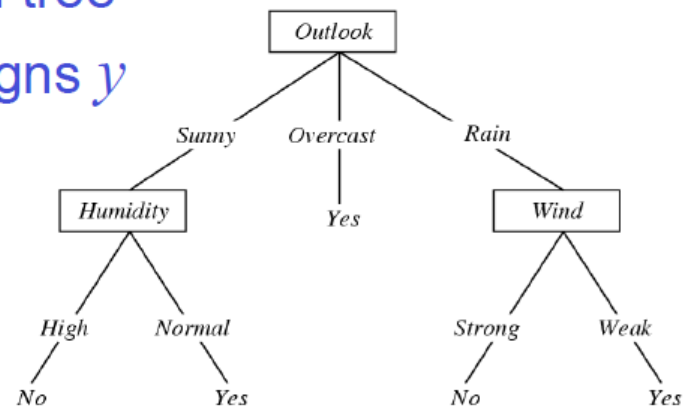


- What prediction would we make for
<outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Decision Tree Learning

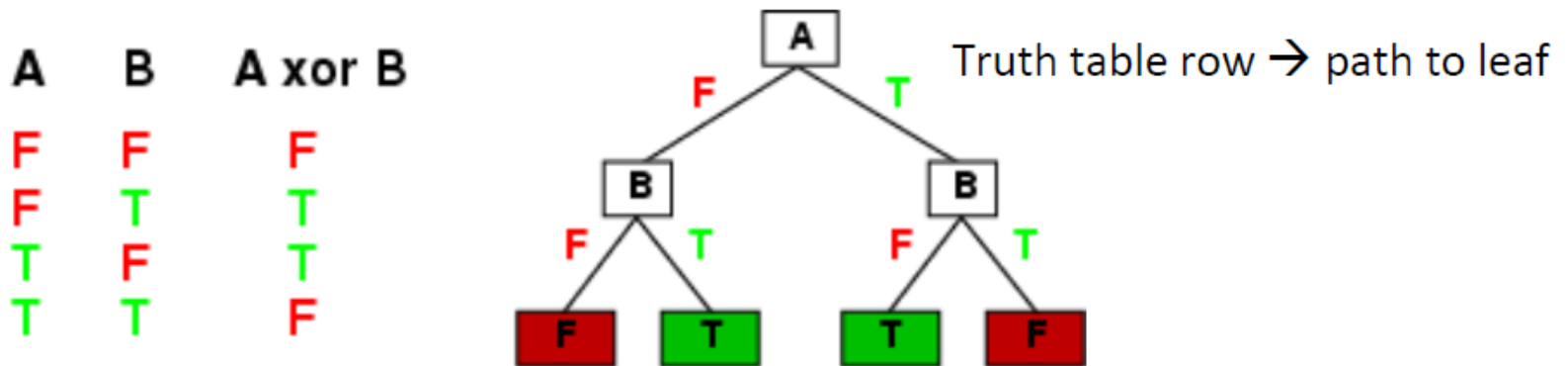
Problem Setting:

- Set of possible instances X
 - each instance x in X is a feature vector
 - e.g., $\langle \text{Humidity}=\text{low}, \text{Wind}=\text{weak}, \text{Outlook}=\text{rain}, \text{Temp}=\text{hot} \rangle$
- Unknown target function $f: X \rightarrow Y$
 - Y is discrete valued
- Set of function hypotheses $H = \{ h \mid h: X \rightarrow Y \}$
 - each hypothesis h is a decision tree
 - trees sorts x to leaf, which assigns y



Expressiveness

- Decision trees can represent any boolean function of the input attributes



- In the worst case, the tree will require exponentially many nodes

XOR cannot be learned with linear classifiers

Occam's Razor

- Principle stated by William of Ockham (1285-1347)
 - “*non sunt multiplicanda entia praeter necessitatem*”
 - entities are not to be multiplied beyond necessity
 - AKA Occam's Razor, Law of Economy, or Law of Parsimony

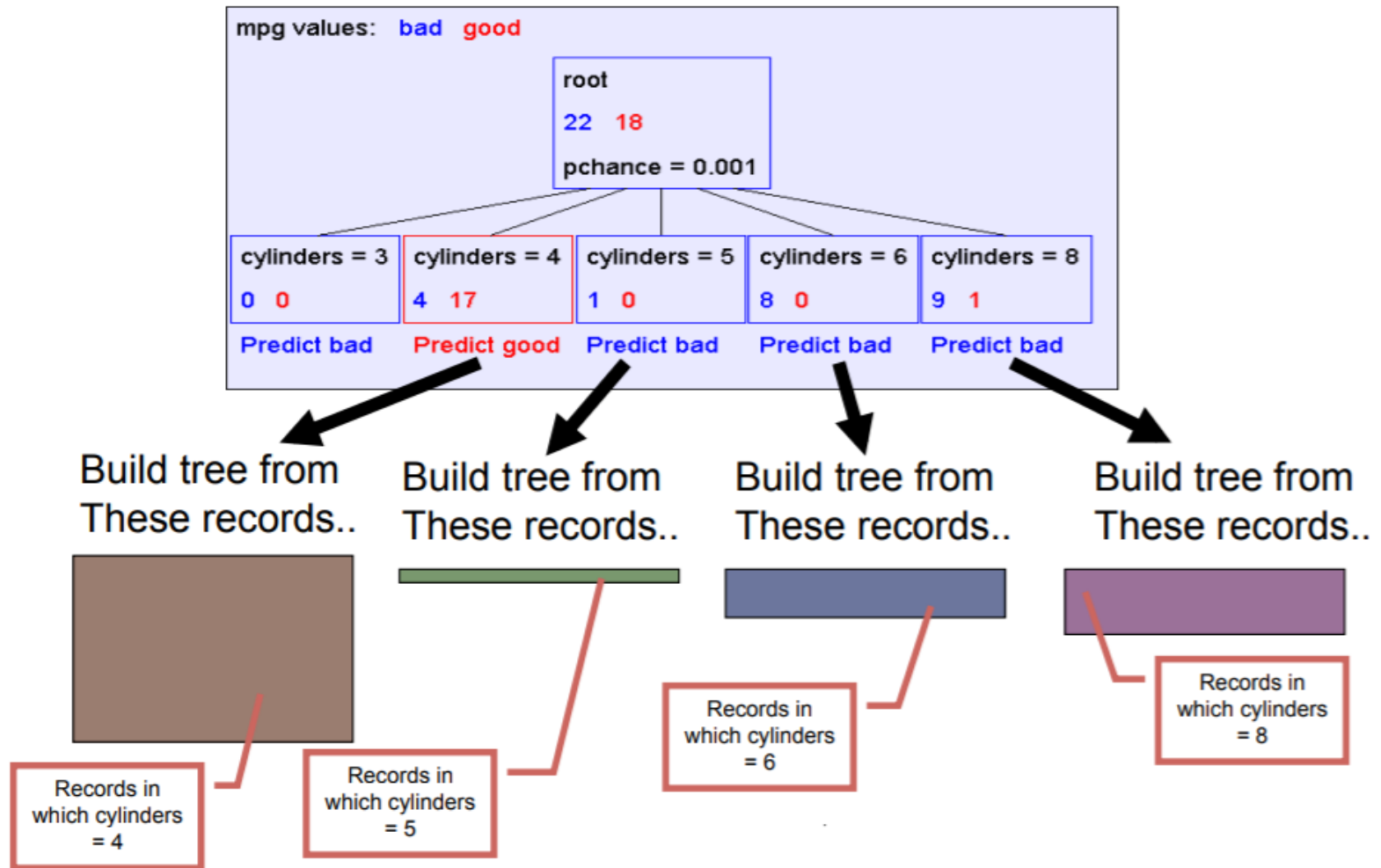
Idea: The simplest consistent explanation is the best

- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
 - Finding the provably smallest decision tree is NP-hard
 - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

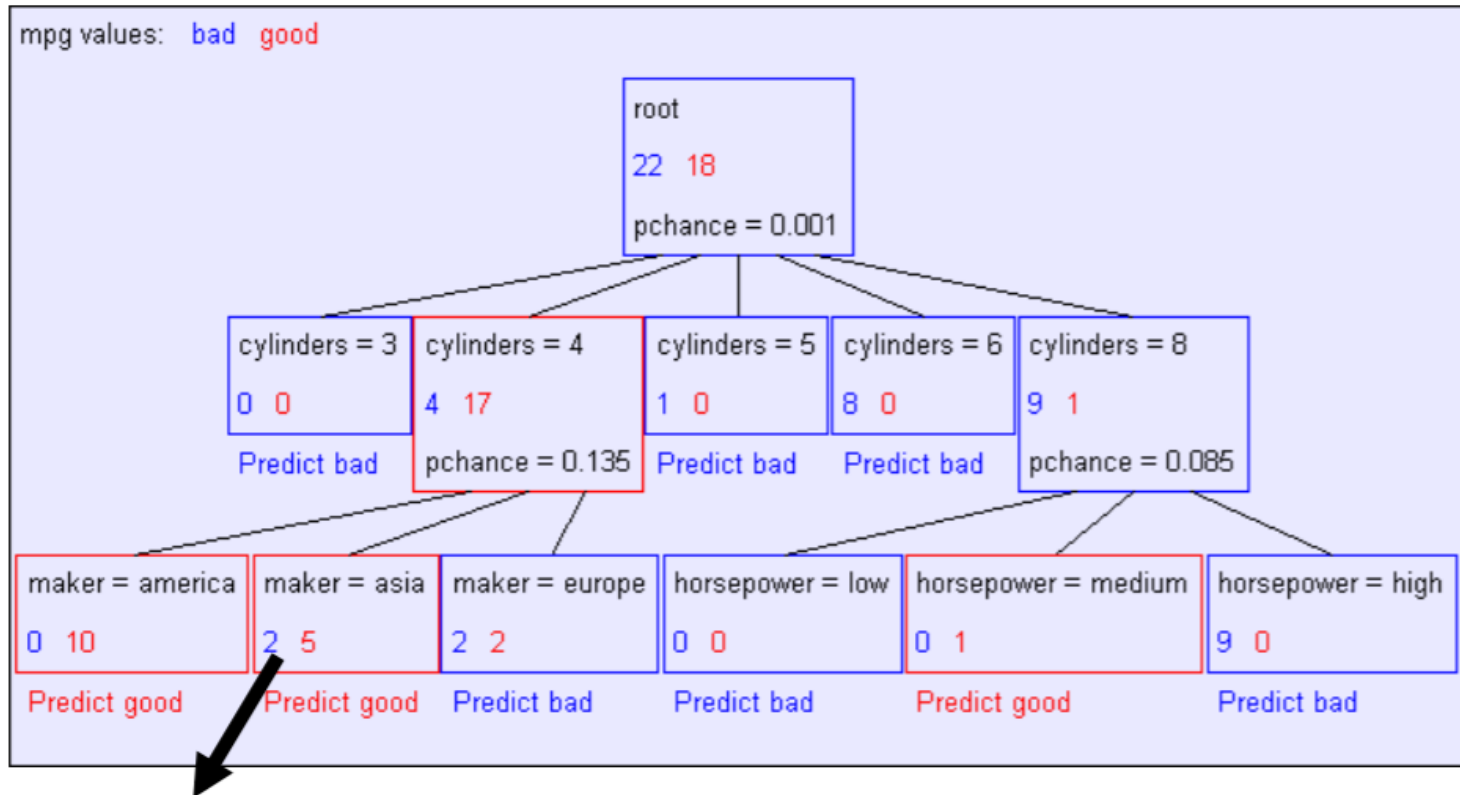
Learning Decision Trees

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]
- Resort to a greedy heuristic:
 - Start from empty decision tree
 - Split on **next best attribute (feature)**
 - Recurse

Key Idea: Use Recursion Greedily



Second Level



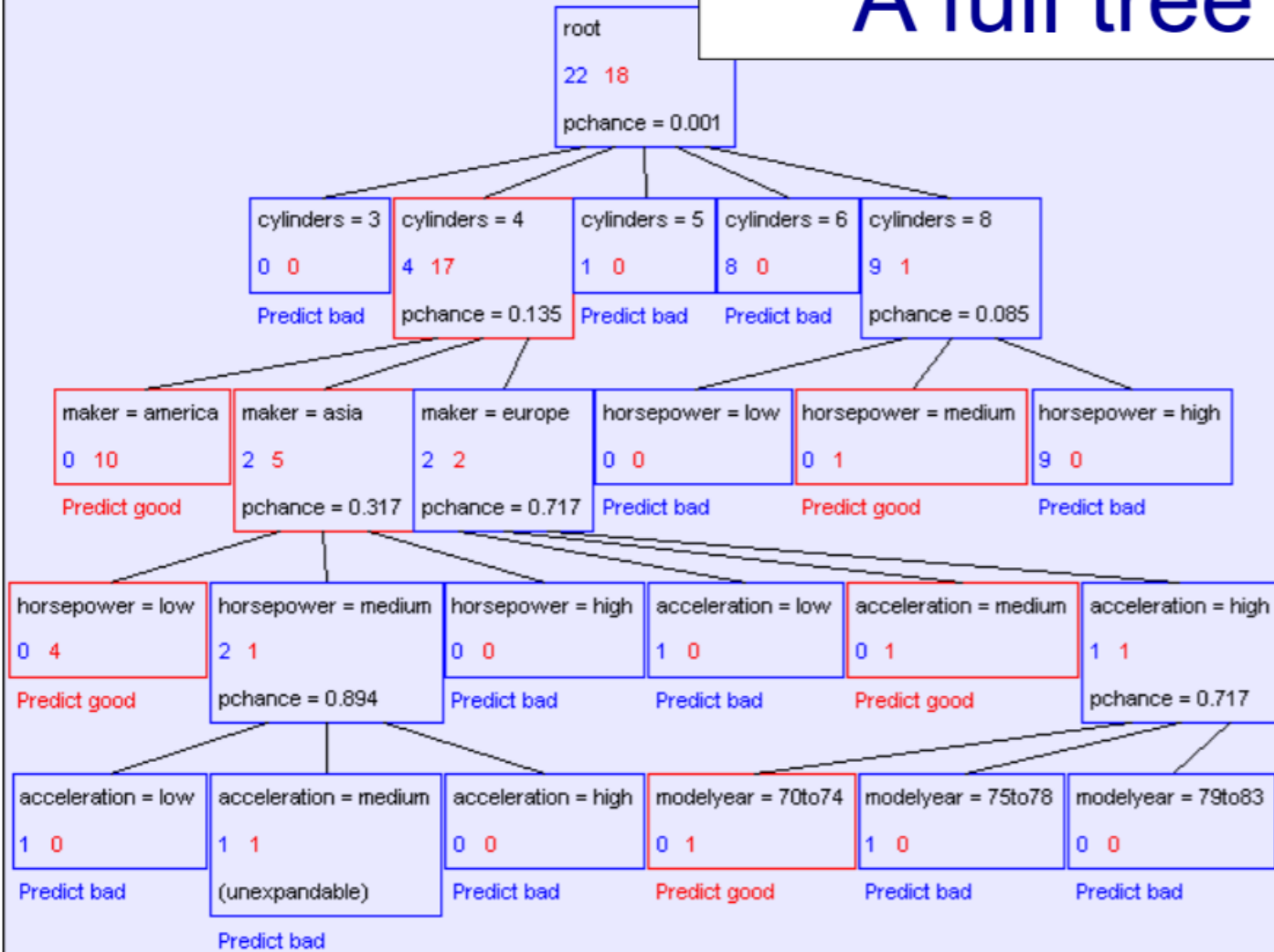
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

Full Tree

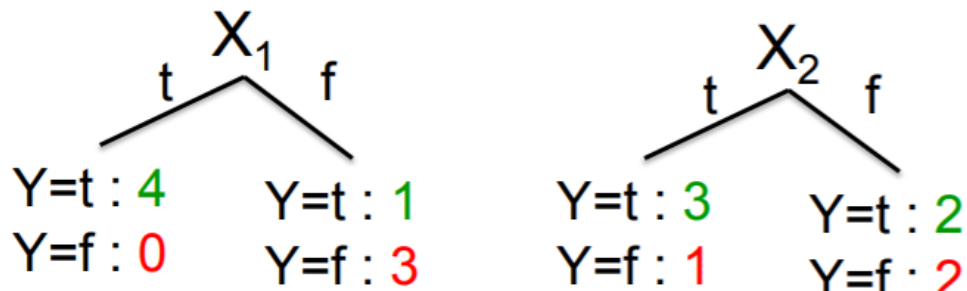
mpg values: bad good

A full tree



Splitting

Would we prefer to split on X_1 or X_2 ?



Idea: use counts at leaves to define probability distributions, so we can measure uncertainty!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Use entropy-based measure (Information Gain)

Acknowledgements

- Slides made using resources from:
 - Andrew Ng
 - Eric Eaton
 - David Sontag
- Thanks!