DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

October 4 2018

Review

- Logistic regression computes directly
  - $\mathrm{P}[Y = 1|X = x]$
  - Assume sigmoid function for hypothesis
  - Trained with Gradient Descent
- LDA uses Bayes Theorem to estimate
  - $\mathrm{P}[Y = k|X = x] = \frac{\mathrm{P}[X = x|Y = k]\mathrm{P}[Y=k]}{\mathrm{P}[X=x]}$
  - Estimates priors from data
  - Assume feature density is Gaussian
- Both are linear classifiers
  - Linear decision boundary (hyperplane)

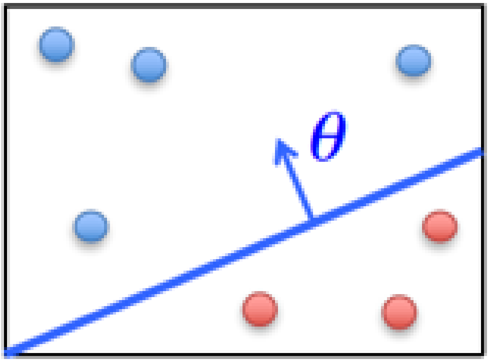
Linear models

- Perceptron

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

- Logistic regression

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



- LDA

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

Outline

- Evaluating classifiers
  - ROC curves, AUC metric
- Feature selection
  - Wrapper
  - Filter
  - Embedded methods
- Decision trees
  - Information gain
  - ID3 algorithm

Confusion Matrix

- Given a dataset of P positive instances and N negative instances:

Actual Class \ Predicted Class	Yes	No
Yes	TP	FN
No	FP	TN

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

- Imagine using classifier to identify positive cases (i.e., for information retrieval)

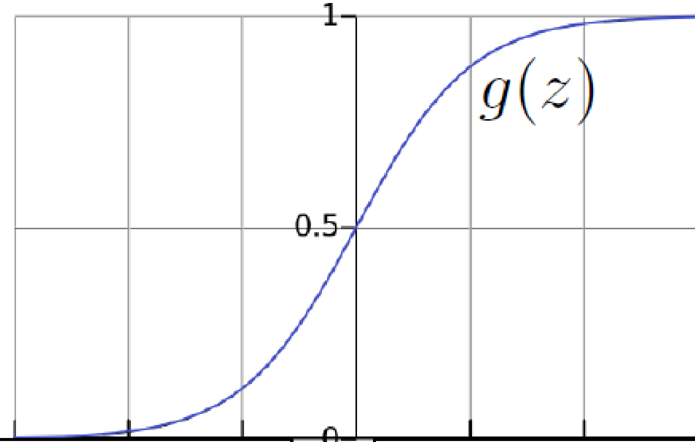
$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

$$\text{F1 score} = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}\right)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

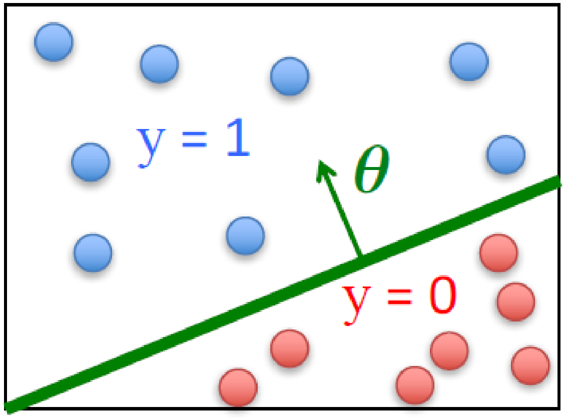


$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$ should be large negative values for negative instances

$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$ should be large positive values for positive instances

Probabilistic model $h_{\theta(x)} = \mathrm{P}[y = 1 | x; \theta]$

- Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
- Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Classifiers can be tuned

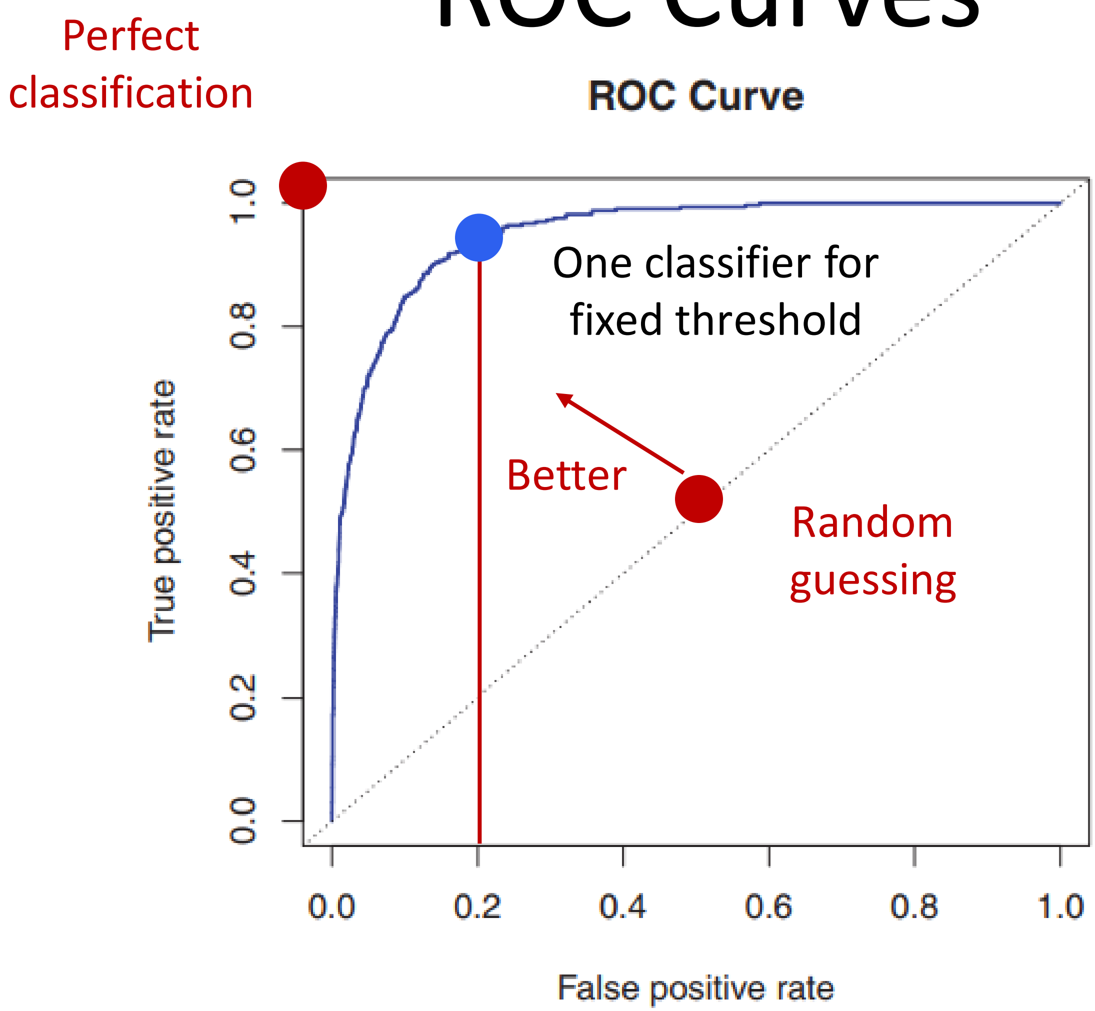
- Logistic regression sets by default the threshold at 0.5 for classifying positive and negative instances
- Some applications have strict constraints on false positives (or other metrics)
  - Example: very low false positives in security (spam)
- Solution: choose different threshold

Probabilistic model $h_{\theta(x)} = \mathrm{P}[y = 1 | x; \theta]$

- Predict y = 1 if $h_\theta(x) \geq$ T
- Predict y = 0 if $h_\theta(x) <$ T

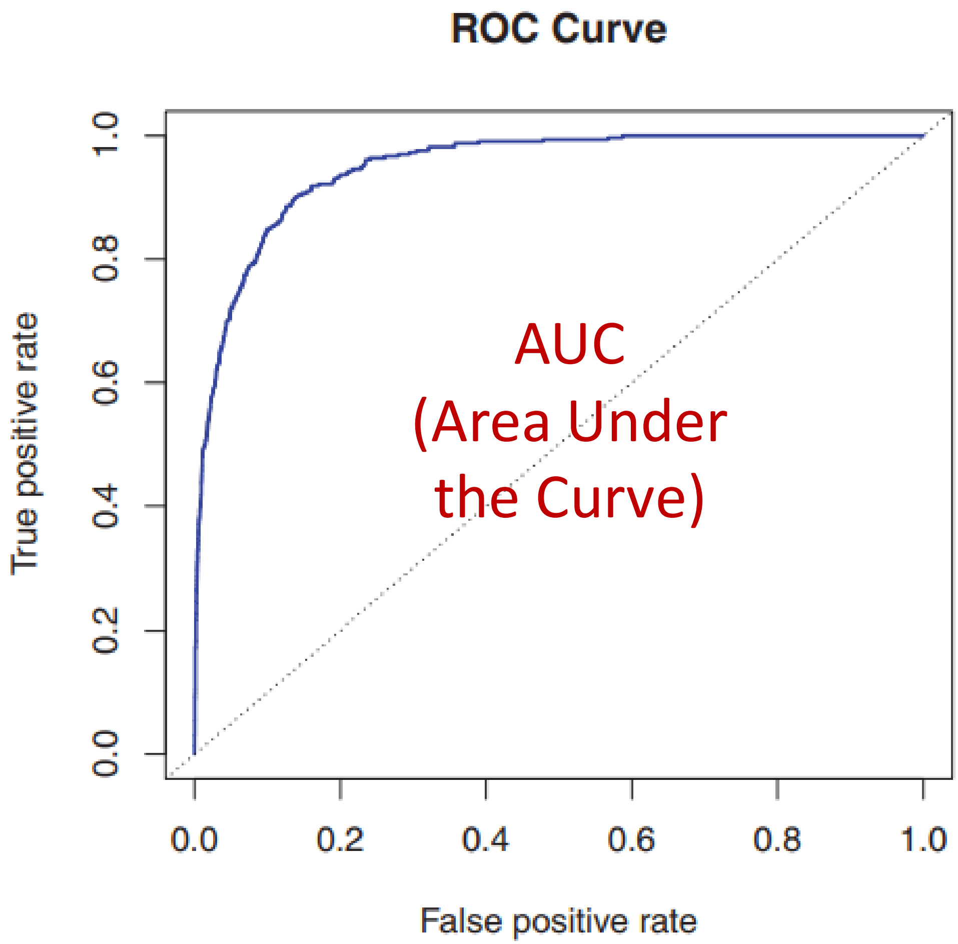
Higher T, lower FP
Lower T, lower FN

ROC Curves



- Receiver Operating Characteristic (ROC)
- Determine operating point (e.g., by fixing false positive rate)

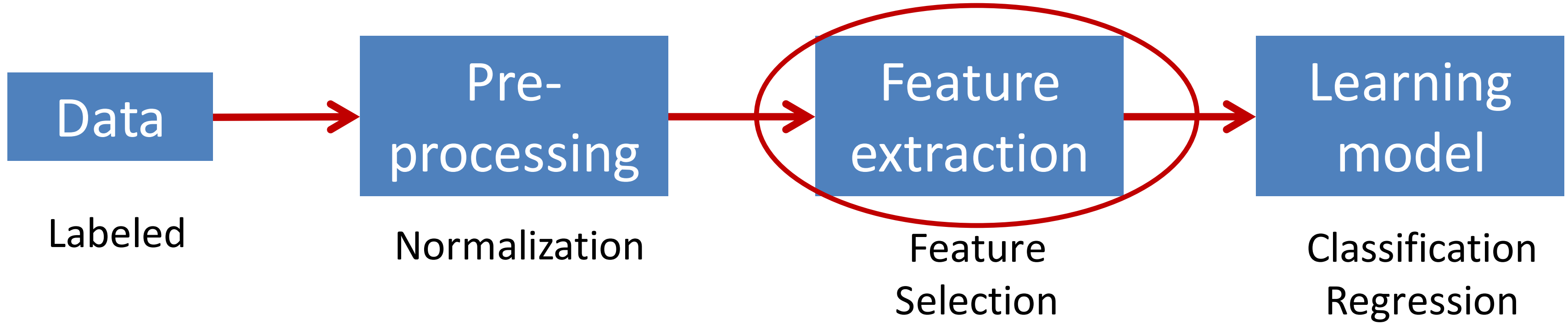
ROC Curves



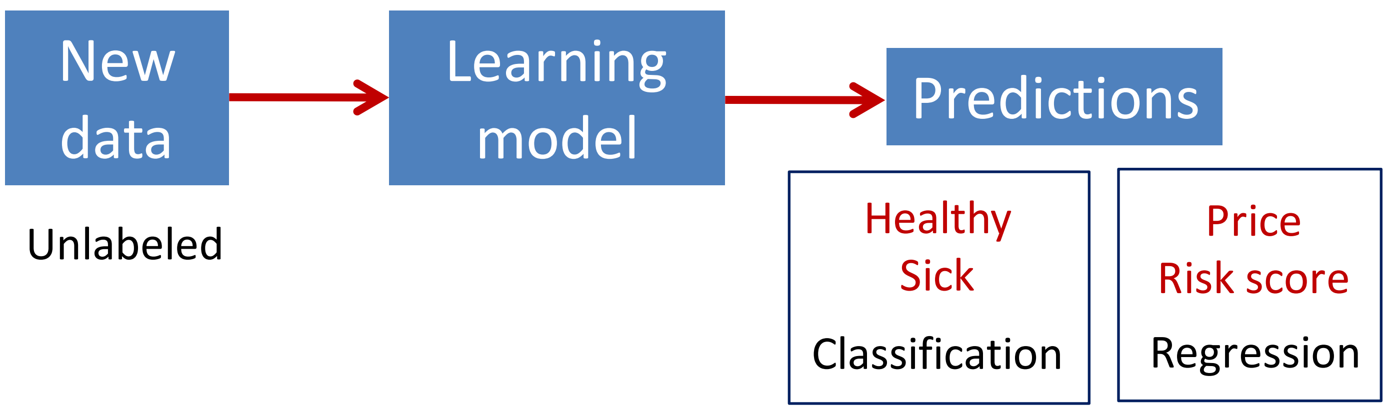
- Another useful metric: Area Under the Curve (AUC)
- The closest to 1, the better!

Supervised Learning

Training



Testing



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.

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)

Wrappers: Search Strategy

- With an **exhaustive search**

$$1011100000010001000010000000000100101010$$

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

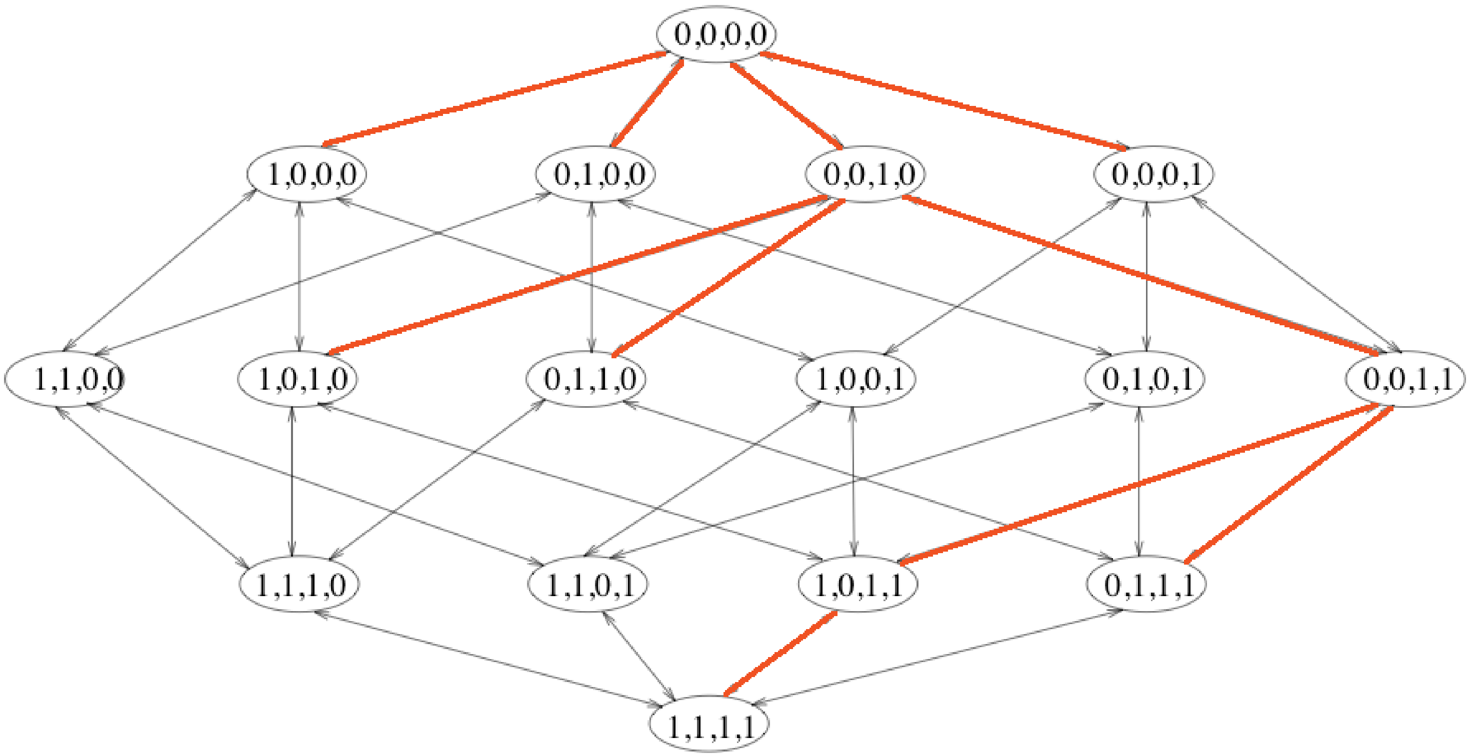
End

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

End

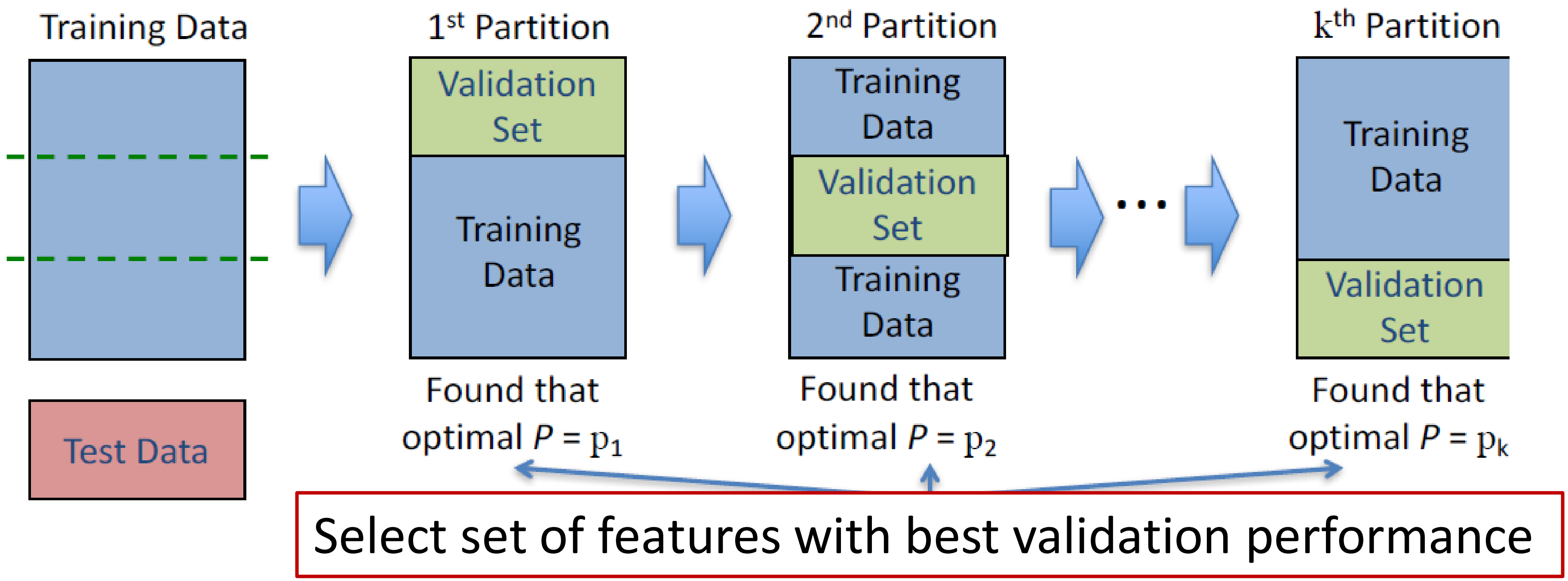
Backward feature selection starts with all features and eliminates backward

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

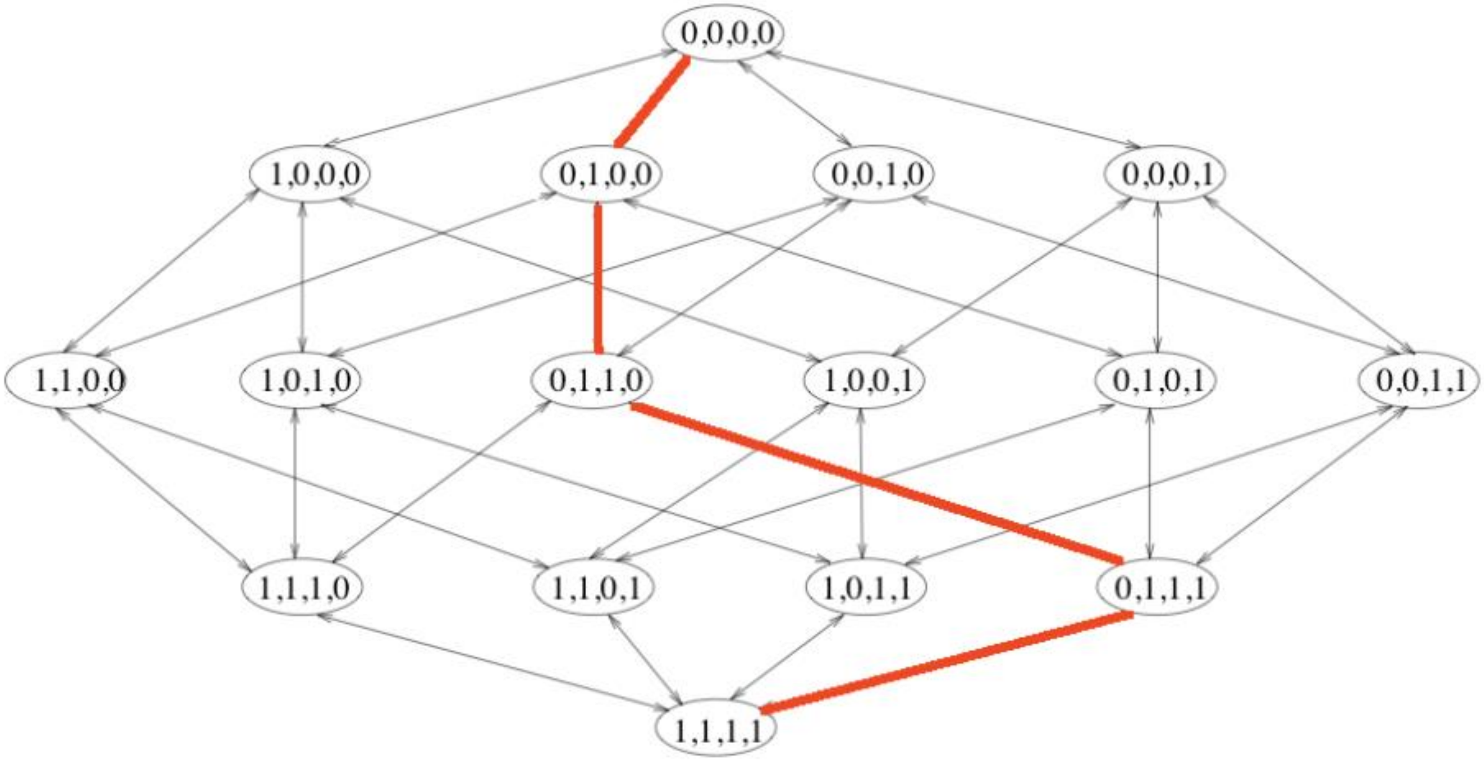
Rank features according to $J(X_f)$

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

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

- 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)

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

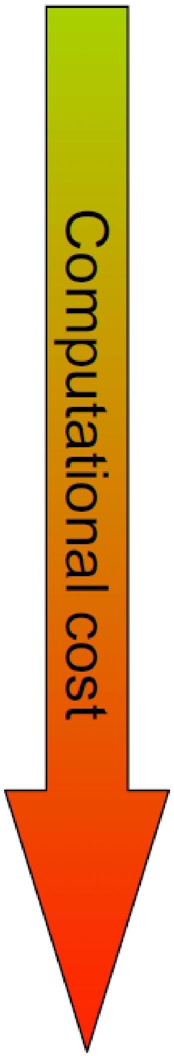
With Error Function:

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

Pros:
- Performs feature selection as part of learning the procedure

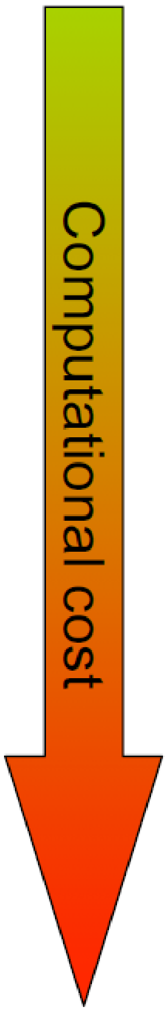
Cons:
- Computationally demanding

Summary: Feature Selection



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

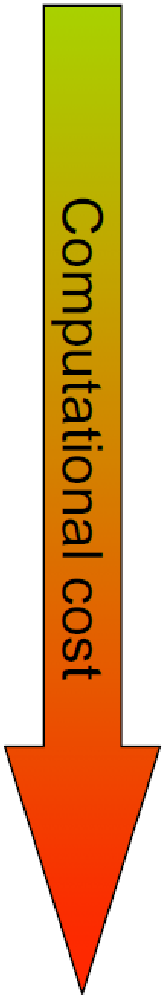
Summary: Feature Selection



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

- Good preprocessing step
- Fails to capture relationship between features

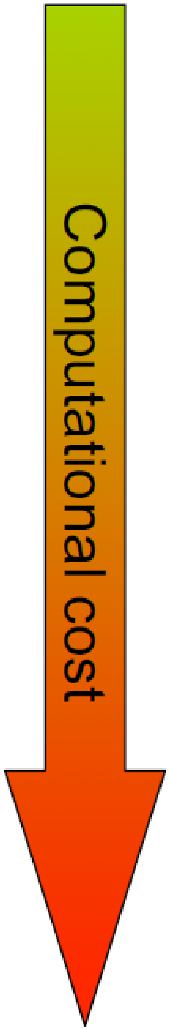
Summary: Feature Selection



- 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

- Evaluating classifiers
  - ROC curves, AUC metric
- Feature selection
  - Wrapper
  - Filter
  - Embedded methods
- Decision trees
  - Information gain
  - ID3 algorithm

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

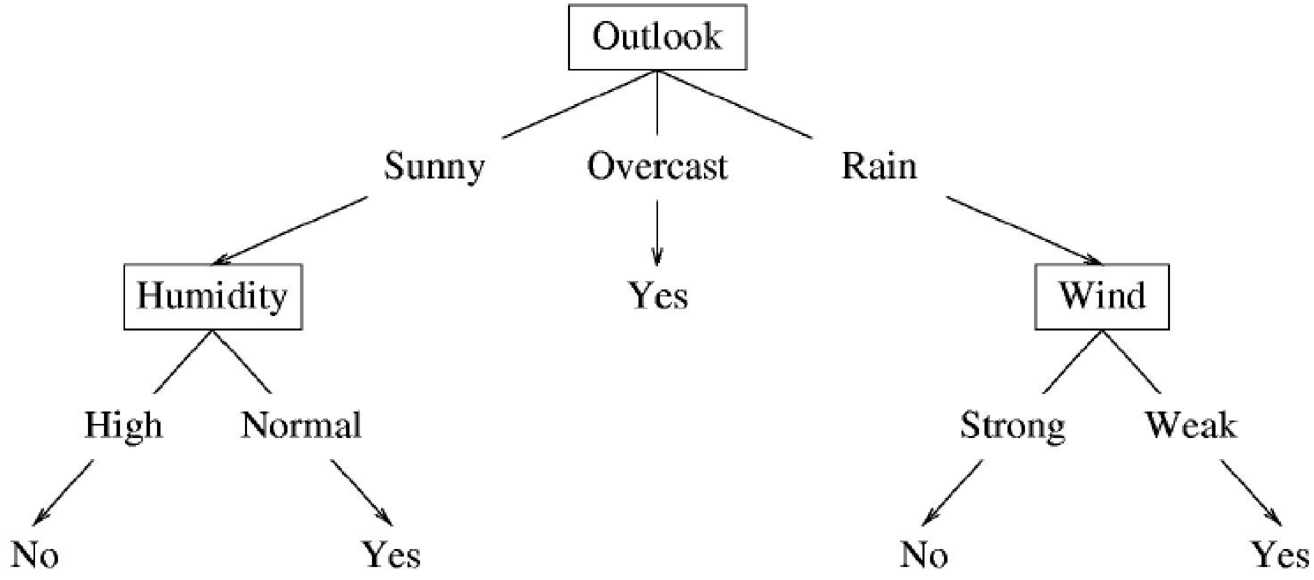
$\langle x^{(i)}, y^{(i)} \rangle$

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

Categorical data

Decision Tree

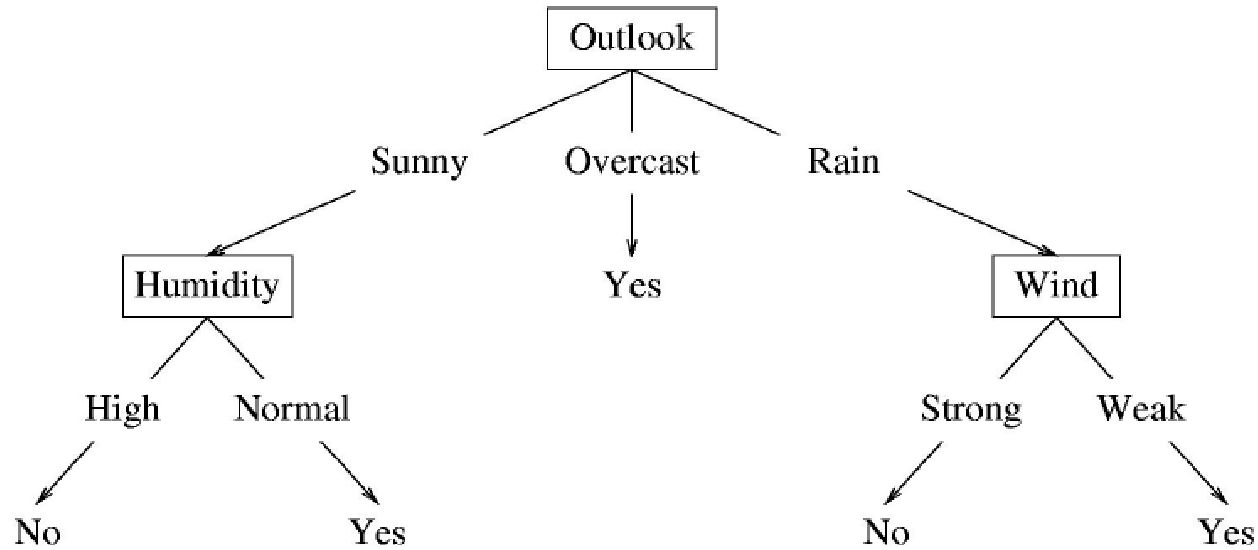
- A possible decision tree for the data:



- 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 $p(Y \mid x \in \text{leaf})$)

Decision Tree

- A possible decision tree for the data:

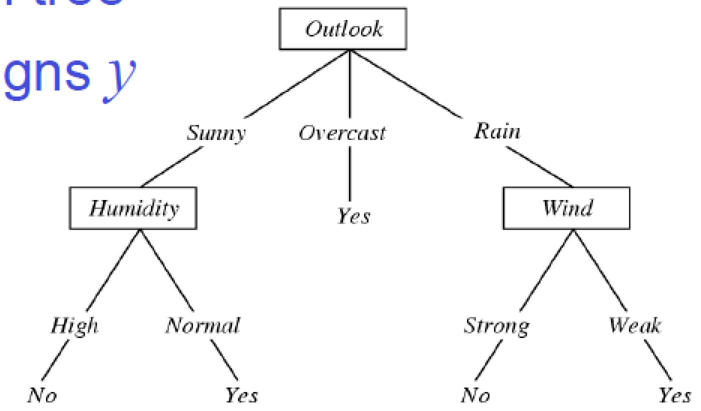


- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- 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

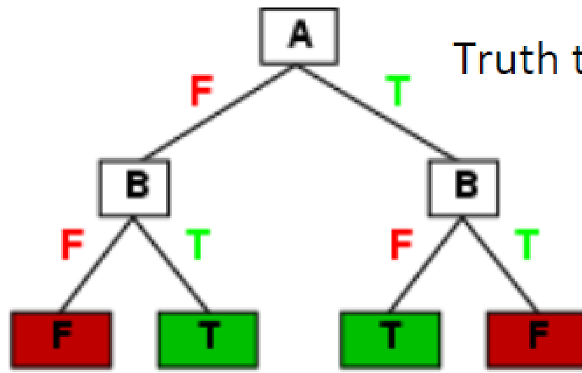


Slide by Tom Mitchell

Expressiveness

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

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Truth table row → path to leaf

- 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

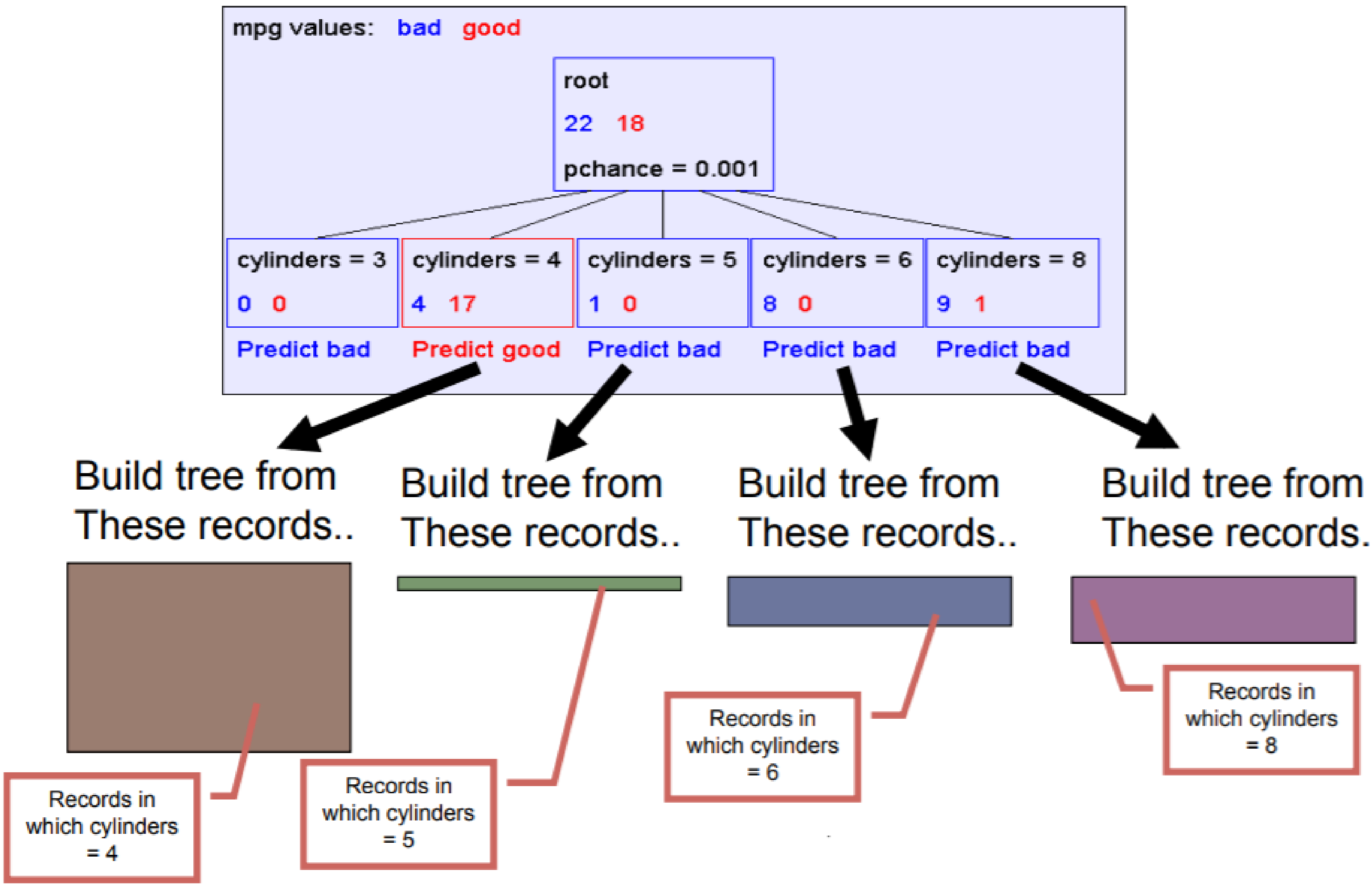
Idea: The simplest consistent explanation is the best

- 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

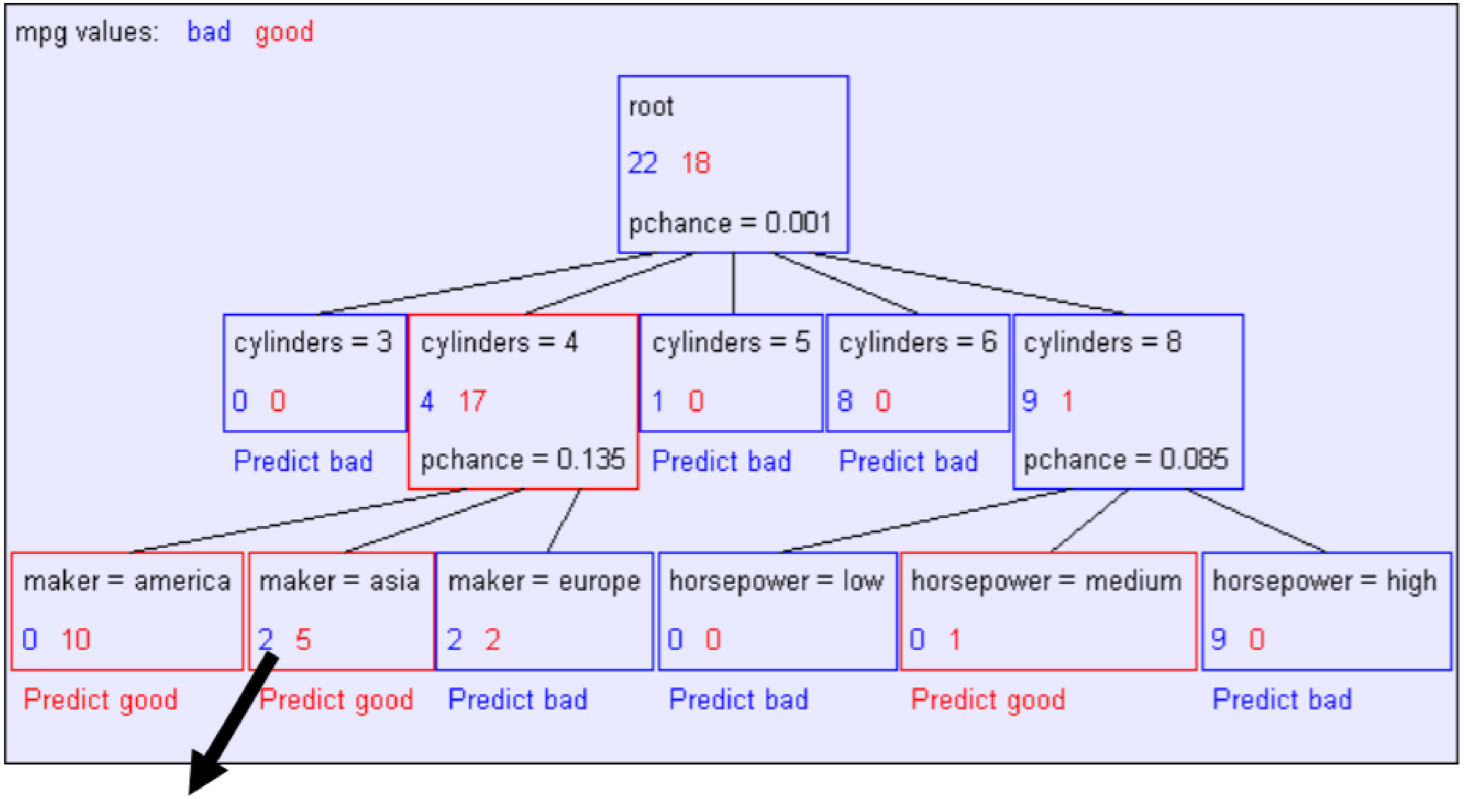
Learning Decision Trees

- 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



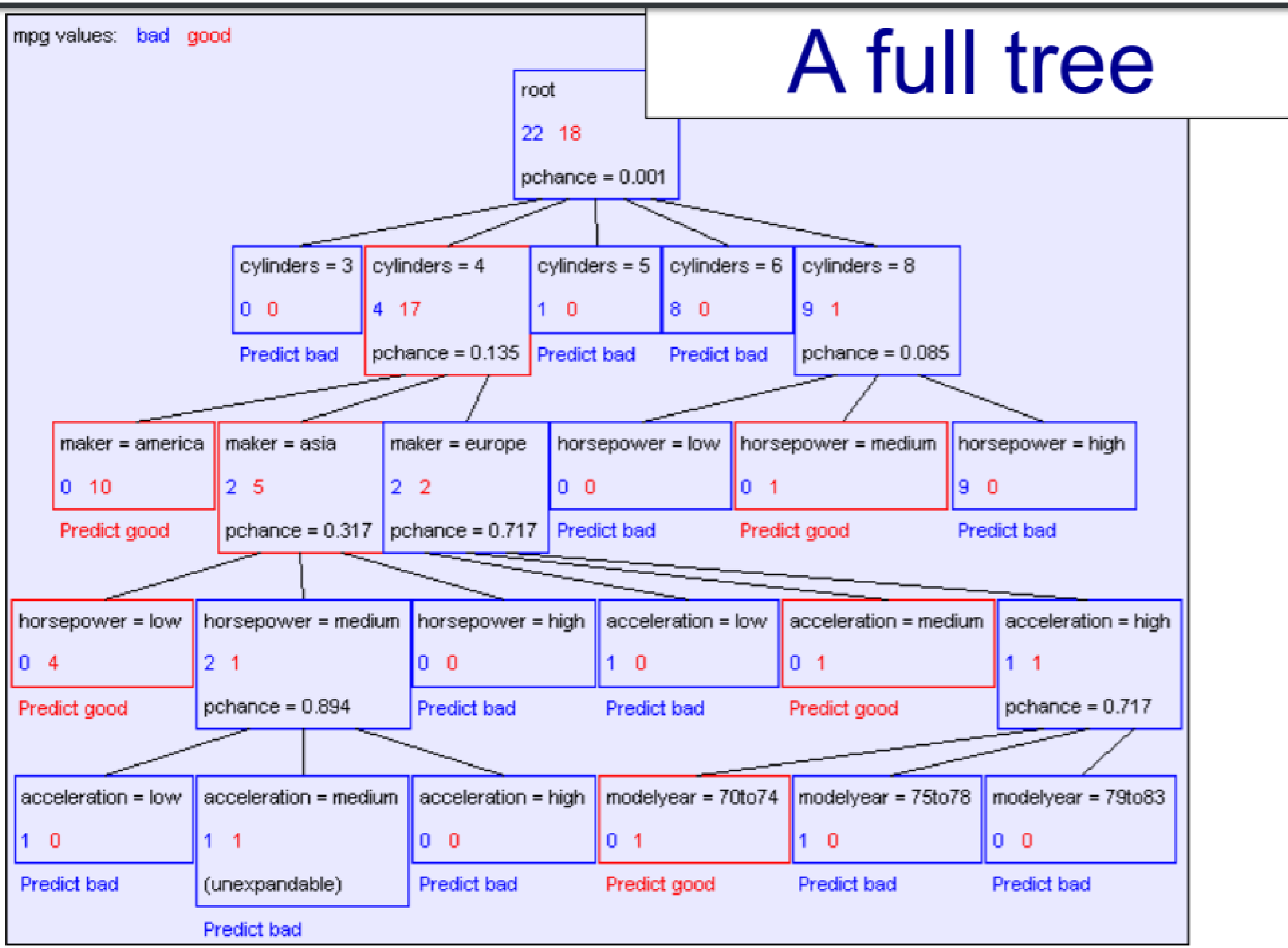
Second Level



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

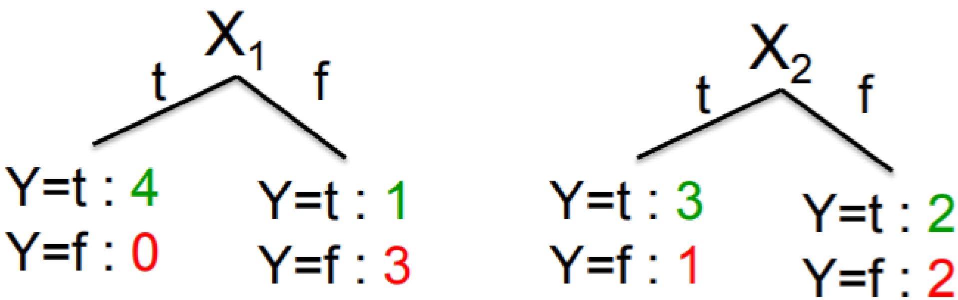
(Similar recursion in the other cases)

Full Tree


A full tree
mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower = low
0 0
Predict bad
horsepower = medium
0 1
Predict good
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
horsepower = high
0 0
Predict bad
acceleration = low
1 0
Predict bad
acceleration = medium
0 1
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad

Splitting

Would we prefer to split on X_1 or X_2?



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

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

Use entropy-based measure (Information Gain)

Transmitting Bits

You are watching a set of independent random samples of X

You see that X has four possible values

P(X=A) = 1/4	P(X=B) = 1/4	P(X=C) = 1/4	P(X=D) = 1/4

So you might see: BAACBADCDADDDA…

You transmit data over a binary serial link. You can encode each reading with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

0100001001001110110011111100…

Use Fewer Bits

Someone tells you that the probabilities are not equal

P(X=A) = 1/2	P(X=B) = 1/4	P(X=C) = 1/8	P(X=D) = 1/8

It's possible…

…to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

Use Fewer Bits

Someone tells you that the probabilities are not equal

P(X=A) = 1/2	P(X=B) = 1/4	P(X=C) = 1/8	P(X=D) = 1/8

It's possible…

…to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

A	0
B	10
C	110
D	111

(This is just one of several ways)

General case

Suppose X can have one of *m* values… $V_1, V_2, \dots V_m$

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution? It's

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \dots - p_m \log_2 p_m$$

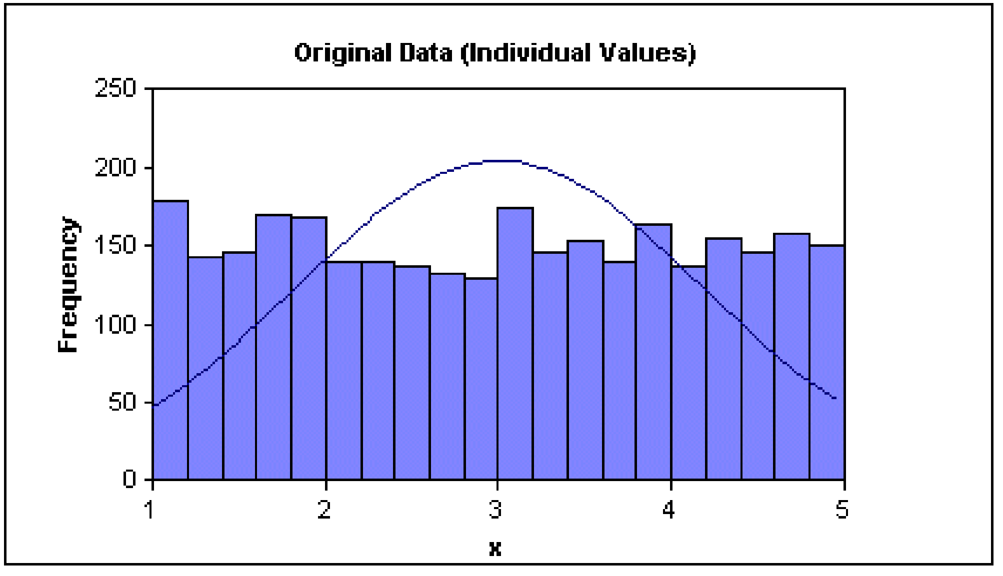
$$= -\sum_{j=1}^{m} p_j \log_2 p_j$$

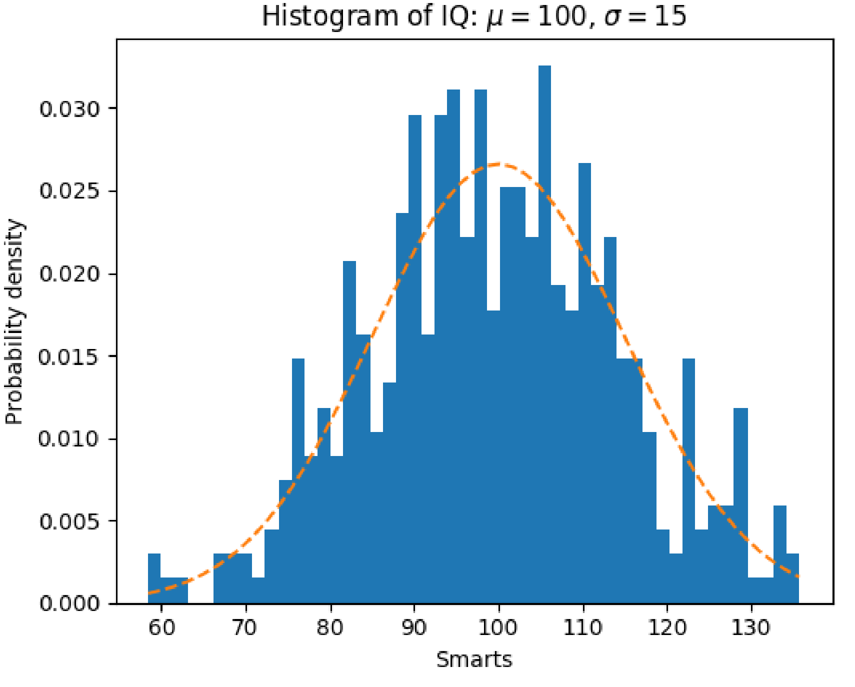
H(X) = The entropy of X

- "High Entropy" means X is from a uniform (boring) distribution
- "Low Entropy" means X is from varied (peaks and valleys) distribution

High/Low Entropy

Which distribution has high entropy?





High

Low

Conditional Entropy

Suppose I'm trying to predict output Y and I have input X

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Let's assume this reflects the true probabilities

E.G. From this data we estimate

- *P(LikeG = Yes) =*
- *P(Major = Math & LikeG = No) =*
- *P(Major = Math) =*
- *P(LikeG = Yes | Major = History) =*

Note:

- $H(X) = 1.5$
- $H(Y) = 1$

Conditional Entropy

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Specific Conditional Entropy:

$H(Y|X=v)$ = **The entropy of** Y **among only those records in which** X **has value** v

Example:

- $H(Y|X=Math) = 1$
- $H(Y|X=History) = 0$
- $H(Y|X=CS) = 0$

Conditional Entropy

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Conditional Entropy:

$H(Y|X)$ = The average specific conditional entropy of Y

= if you choose a record at random what will be the conditional entropy of Y, conditioned on that row's value of X

= Expected number of bits to transmit Y if both sides will know the value of X

$= \Sigma_j Prob(X=v_j)\ H(Y \mid X = v_j)$

Conditional Entropy

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Conditional Entropy:

$H(Y|X)$ = The average conditional entropy of Y

$= \Sigma_j Prob(X=v_j)\ H(Y \mid X = v_j)$

Example:

v_j	$Prob(X=v_j)$	$H(Y \mid X = v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

Information Gain

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Information Gain:

$IG(Y|X)$ = **I must transmit** Y**. How many bits on average would it save me if both ends of the line knew** X**?**

$$IG(Y|X) = H(Y) - H(Y|X)$$

Example:

- **H(Y) = 1**
- **H(Y|X) = 0.5**

Review

- Metrics for evaluating classifiers
  - Accuracy, error, precision, recall, F1 score
  - AUC (area under the ROC curve) measures performance of classifier for different thresholds
- Feature selection methods
  - Filters decide on each feature individually
  - Wrappers select a subset of features by search strategy (fixing model and evaluating with cross-validation)
  - Embedded methods (e.g., regularization) are part of training
- Decision trees are interpretable, non-linear models
  - Greedy algorithm to train Decision Trees
  - Works on categorical and numerical data
  - Node splitting done by highest Information Gain

Acknowledgements

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