Classification & Regression

Definitions

- Unsupervised Learning:
  - Given items X, automatically discover the structure, representations, etc.
  - Ex.: Clustering
- Supervised Learning:
  - Given the value of an input vector X and $c(x)$, predict c on future unseen $x's$.
  - Ex.: Regression, classification

Supervised Learning

- Regression
  - Given the value of an input X, the output Y belongs to the set of real values R. Goal is to predict output accurately for new input.
- Classification
  - The predictions or outputs, $c(x)$ are categorical while x can take any set of values (real or categorical). The goal is select correct class for a new instance.
- Time series prediction
  - Data is in the form of a moving time series. The goal is to perform classification/regression on future time series based on data known so far.

Regression

- Predictive technique where the target variable to be estimated is continuous.
  - Applications:
    - Predicting the stock market index based on economic factors
    - Forecasting precipitation based on characteristics of the jet stream
    - Projecting a company's revenue based on the amount spent for advertisement

Regression

- Let D denote a dataset containing N observations,

$$D = \{(x_i, y_i) | i = 1,2, \dots, N\}$$

  - Each x_i corresponds to the set of attributes of the i-th observation. These are called **explanatory variables** and can be discrete or continuous.
  - y_i corresponds to the **target variable**.

Definition. Regression is the task of learning a target function f that maps each attribute set x into a continuous-valued output y.

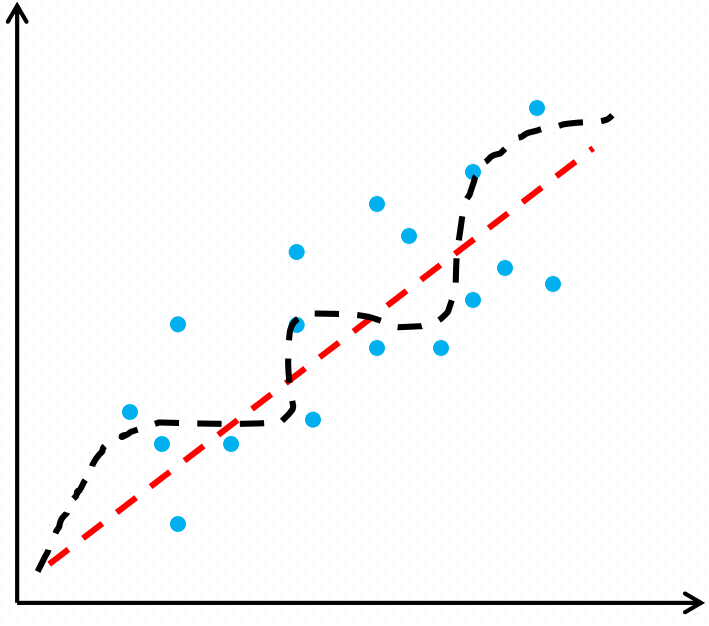
Error function

- The goal of linear regression is to find a target function that can minimize the error, which can be measured as the sum of absolute or squared error.

$$Absolute\ Error = \sum_i |y_i - f(x_i)|$$

$$Squared\ Error = \sum_i \left(y_i - f(x_i)\right)^2$$

Simple Linear Regression



Given a set of points (x_i, y_i) on a scatterplot

Find the best-fit line $f(x_i) = \boldsymbol{w_0} + \boldsymbol{w_1} x_i$

Such that $\text{SSE} = \sum_i (y_i - f(x_i))^2$ is minimized

Simple Linear Regression

- To find the regression parameters w_0 and w_1, we apply the **method of least squares**, which attempts to minimize the SSE

$$SSE = \sum_{i=1}^{N} [y_i - f(x_i)]^2 = \sum_{i=1}^{N} [y_i - w_0 - w_1 x_i]^2$$

Simple Linear Regression

- This optimization problem can be solved by taking the partial derivatives of E with respect to w_0 and w_1, setting them to 0 and solving the system of linear equations.

$$\frac{\partial E}{\partial w_0} = -2 \sum_{i=1}^{N} [y_i - w_1 x_i - w_0] = 0$$

$$\frac{\partial E}{\partial w_1} = -2 \sum_{i=1}^{N} [y_i - w_1 x_i - w_0] x_i = 0$$

Simple Linear Regression

- The previous equations can be summarized by the **normal equation:**

$$\begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

Example

- Consider the set of 10 points:

x	1	2	3	4	4	5	5	6	6	7
y	7	8	9	8	9	11	10	13	14	13

$$\sum_i x_i = 43 \qquad \sum_i x_i^2 = 217$$
$$\sum_i y_i = 102 \qquad \sum_i y_i^2 = 1094$$
$$\sum_i x_i y_i = 476$$

Example

x	1	2	3	4	4	5	5	6	6	7
y	7	8	9	8	9	11	10	13	14	13

$$\sum_i x_i = 43 \quad \sum_i x_i^2 = 217 \quad \sum_i y_i = 102 \quad \sum_i y_i^2 = 1094 \quad \sum_i x_i y_i = 476$$

$$\begin{pmatrix} 10 & 43 \\ 43 & 217 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} 102 \\ 476 \end{pmatrix}$$

$$\begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} 10 & 43 \\ 34 & 217 \end{pmatrix}^{-1} \begin{pmatrix} 102 \\ 476 \end{pmatrix}$$

$$\begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} 5.19 \\ 1.17 \end{pmatrix}$$

$$\boldsymbol{f(x_i) = 1.17x_i + 5.19}$$

Example

x	1	2	3	4	4	5	5	6	6	7
y	7	8	9	8	9	11	10	13	14	13

- A general solution to the normal equations can be expressed as

$$\boldsymbol{w_0 = \bar{y} - w_1 \bar{x}}$$
$$\boldsymbol{w_1 = \frac{\sigma_{xy}}{\sigma_{xx}}}$$

Where $\bar{x}$ and $\bar{y}$ are the average values of x and y and:

$$\sigma_{xy} = \sum_i (x_i - \bar{x})(y_i - \bar{y})$$
$$\sigma_{xx} = \sum_i (x_i - \bar{x})^2$$
$$\sigma_{yy} = \sum_i (y_i - \bar{y})^2$$

Example

x	1	2	3	4	4	5	5	6	6	7
y	7	8	9	8	9	11	10	13	14	13

- A linear model that results in the minimum squared error is then:

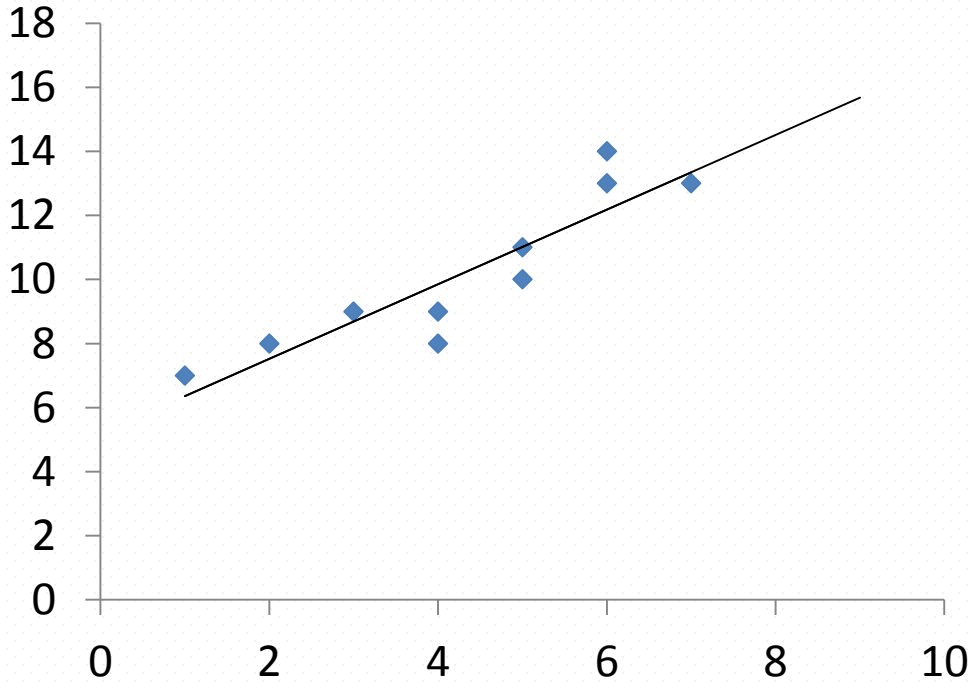
$$\boldsymbol{f(x) = \bar{y} + \frac{\sigma_{xy}}{\sigma_{xx}}(x - \bar{x})}$$

- For our example:

$$f(x) = 10.2 + \frac{37.4}{32.1}(x - 4.3)$$
$$f(x) = 10.2 + 1.17(x - 4.3)$$
$$f(x) = 10.2 + 1.17x - 5.01$$
$$\boldsymbol{f(x) = 1.17x + 5.19}$$

Example

x	1	2	3	4	4	5	5	6	6	7
y	7	8	9	8	9	11	10	13	14	13



Evaluating goodness of fit

- In order to measure how well data points fit to our line, we use a method called **R squared (R^2)**
- This value ranges from 0 to 1. It is close to 1 if most variability observed in the target variable can be explained by the regression model

Evaluating goodness of fit

$$R^2 = \frac{SSM}{SST} = \frac{\sum_i [f(x_i) - \bar{y}]^2}{\sum_i [y - \bar{y}]^2}$$

$$R^2 = \frac{\sigma_{xy}^2}{\sigma_{xx}\sigma_{yy}}$$

- As we add more explanatory variables, R^2 increases, so it is typically adjusted as:

$$Adjusted\ R^2 = 1 - \left(\frac{N-1}{N-d}\right)(1 - R^2),$$

Where N is the number of data points and d+1 is the number of parameters of the regression model.

Logistic Regression

- Useful when the target is binary
- Logistic regression or logit regression is a type of probabilistic statistical classification model
- It measures the relationship between the dependent (target) binary variable and the independent explanatory variables

Logistic Regression

- In summary:
  - We have a binary target variable Y, and we want to model the conditional probability $P(Y = 1|X = x)$ as a function $p(x)$ of the explanatory variables x.
  - Any unknown parameters (recall w_0 and w_1) are estimated by maximum likelihood.
  - Can we use linear regression?

Logistic Regression

- Idea 1: Let $p(x)$ be a linear function
  - W are estimating a probability, which must be between 0 and 1
  - Linear functions are unbounded, so this approach doesn't work
- Better idea: Set the odds ratio to a linear function:

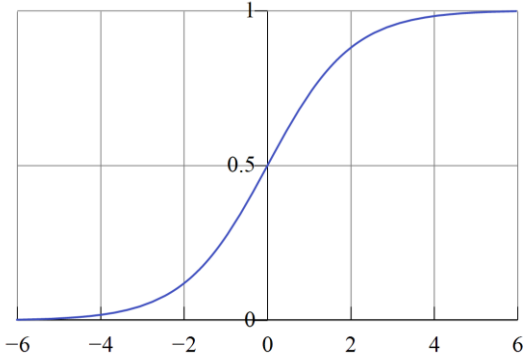
$$\log(odds) = logit(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Solving for p:

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-(\boldsymbol{\beta_0}+\boldsymbol{\beta_1 x})}}$$

  - This is called the logistic (logit) function and it assumes values [0,1]

Logistic Curve



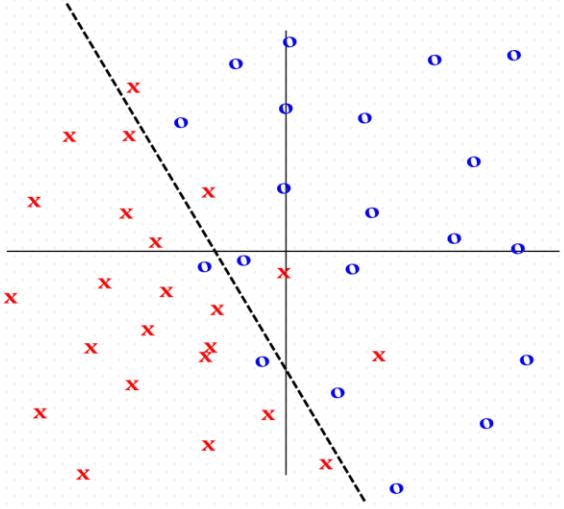
- A sigmoid function that assumes values in the range [0,1]

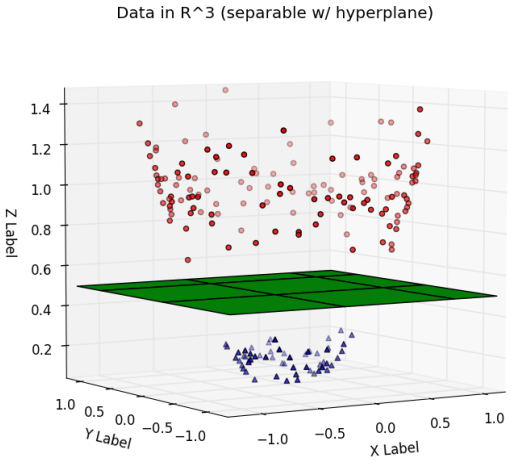
$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

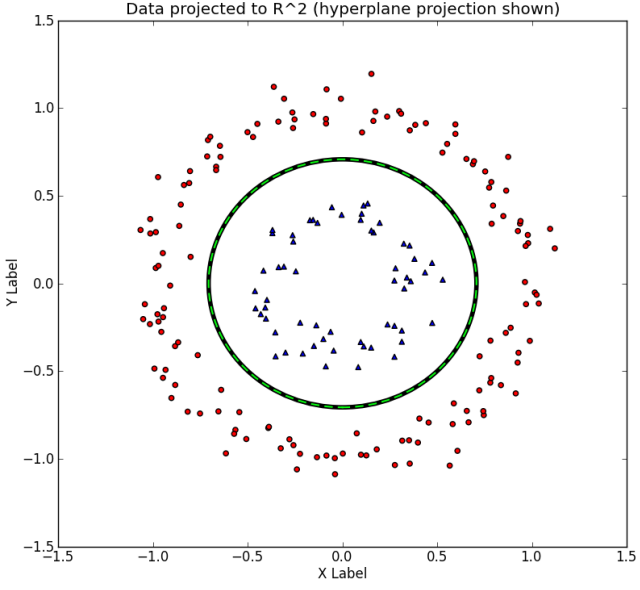
Logistic Regression

- To minimize misclassification rates, we predict:
  - $Y = 1$ when $p(x) \geq 0.5$ and $Y = 0$ when $p(x) < 0.5$
  - So $Y = 1$ when $\beta_0 + \beta_1 x$ is non-negative and 0 otherwise
- Logistic regression gives us a **linear classifier** where the decision boundary separating the two classes is the solution of $\beta_0 + \beta_1 x = 0$
  - A point if we have 1 explanatory variable
  - A line if we have 2
  - A plane if we have 3
  - A disaster if we have more than that

Decision Boundaries







Logistic Regression

- The parameters $\beta_0, \beta_1, \ldots$ are estimated using a technique called Maximum likelihood estimation (MLE)
  - Unlike the least squares methods used for Linear regression, finding a closed form for the coefficients using MLE is not possible. Instead, an iterative process (e.g., Newton's method) is used.
  - This process begins with a tentative solution, revises it slightly to see if it can be improved, and repeats this revision until improvement is minute, at which point the process is said to have converged.

Logistic Regression

- Goodness of fit for logistic regression can't be measured using R^2. Methods used in this case include:
  - Hosmer–Lemeshow test
  - Binary classification performance evaluation
  - Deviance and likelihood ratio tests

Now lets see some regressioning!

Supervised Learning

Regression

Given the value of an input X, the output Y belongs to the set of real values R. The goal is to predict output accurately for a new input.

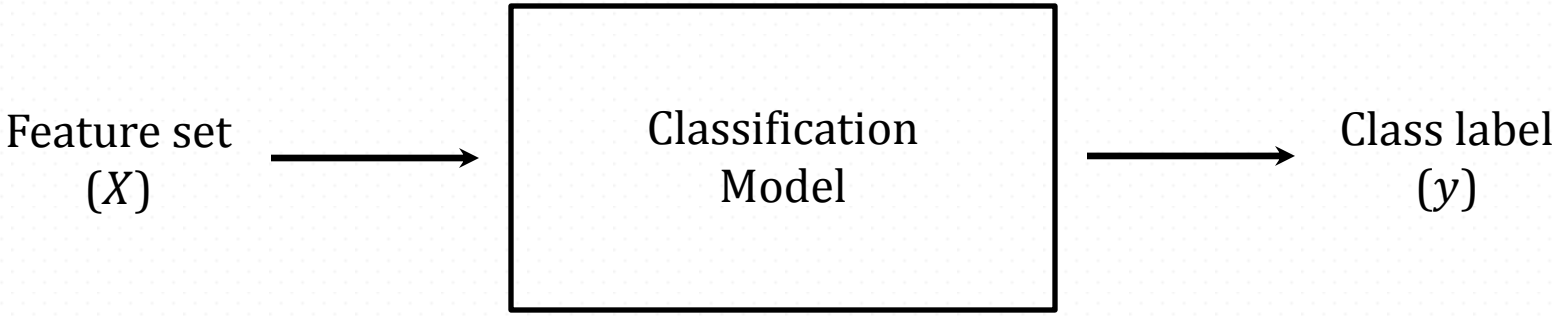
Classification

The predictions or outputs y are categorical, while x can take any set of values (real or categorical). The goal is to select the correct class for a new instance.

Classification

The task of assigning objects to one of several predefined categories.

Classification



Definition of Classification

- The input data for a classification task is a collection of records, each known as an instance, composed of a feature set and a class label.
- Classification, then, is the task of learning a **target function** f that maps each feature set x to one of the predefined class labels y.
- The target function is also known informally as the **classification model**. A classification model is useful for several purposes.

Types of Classification Models

Two purposes of classification models:

1. **Descriptive Modeling**
   A classification model can serve as an explanatory tool to distinguish between objects of different classes.

2. **Predictive Modeling**
   A classification model can also be used to predict the class label of unknown records.

Classification Models: An Example

	Input Features: X			*Class: Y*
	Make	Cylinders	Length	Weight	Style
Instances	Honda	Four	150	1956	Hatchback
	Toyota	Four	167.9	2280	Wagon
	BMW	Six	176.8	2765	Sedan

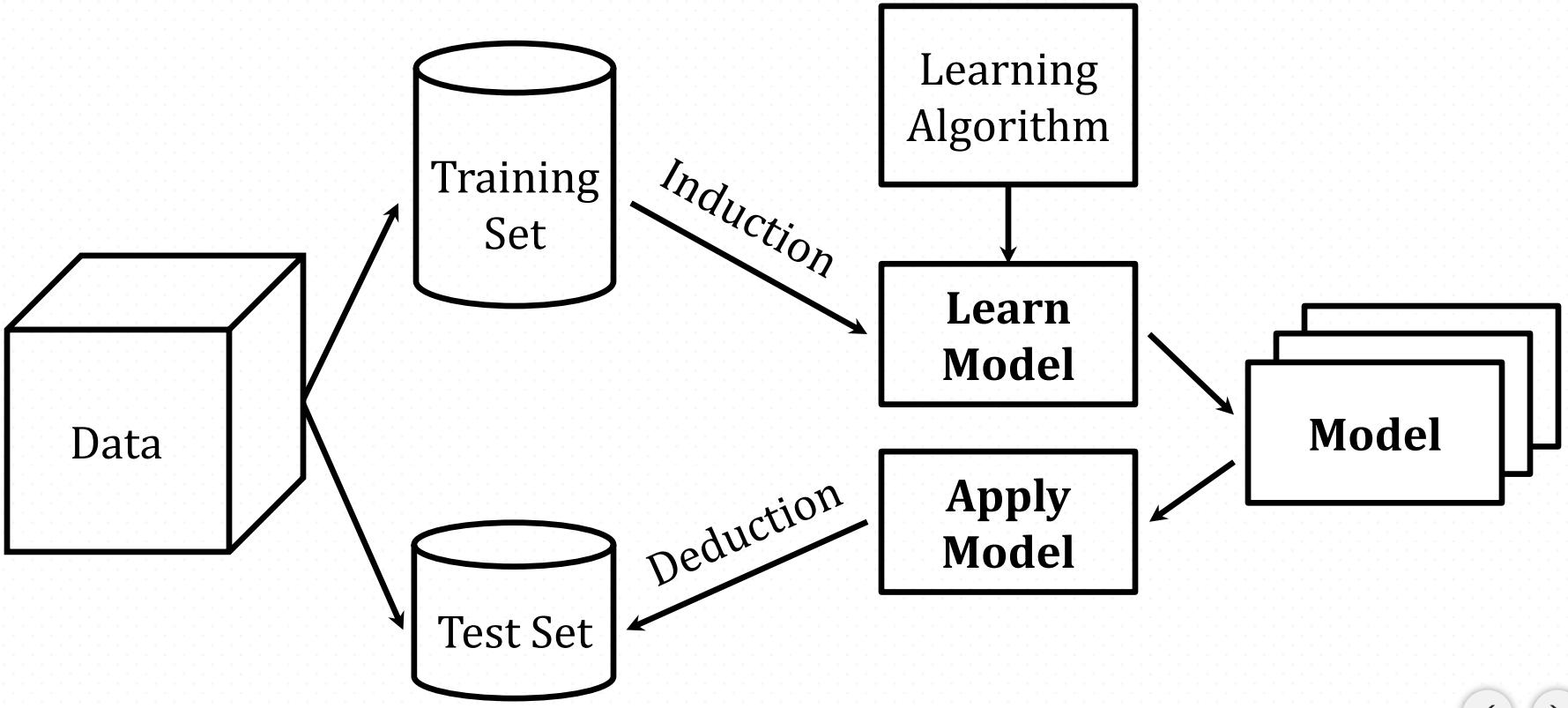
Descriptive modeling: Explain what features define a hatchback, wagon, or sedan.

Predictive modeling: Given another car, determine the style to which it belongs.

Building a Classification Model

- A classification technique (or classifier) is a systematic approach to building classification models from an input dataset.
- Each technique employs a *learning algorithm* to identify a model that best fits the relationship between the features and class of the input data.
- A key objective of the learning algorithm is to build models with good generalization capability.

Building a Classification Model



Types of Learners

- **Eager learners** learn a model that maps the input features to the class label a soon as the training data becomes available.
- **Lazy learners** delay modeling the training data until it is needed to classify the test examples.
  - An example of a lazy leaner is a *rote classifier*, which memorizes the entire training data and performs classification only if a test instance matches one of the training examples exactly.

Rote Learning

Make	Cylinders	Length	Weight	Style
Honda	Four	150	1956	Hatchback
Toyota	Four	167.9	2280	Wagon
BMW	Six	176.8	2765	Sedan

BMW	Six	176.8	2765	**Sedan**

Nearest Neighbor Classifier

The Idea:

Find which training data is closest to the test instance, and classify the test instance as that class.

Problems:

- Computationally expensive.
- Not interpretable.
- Very sensitive to noise/outliers.

Nearest Neighbor Learning

Make	Cylinders	Length	Weight	Style
Honda	Four	150	1956	Hatchback
Toyota	Four	167.9	2280	Wagon
BMW	Six	176.8	2765	Sedan
BMW	Six	174.2	2456	**Sedan**

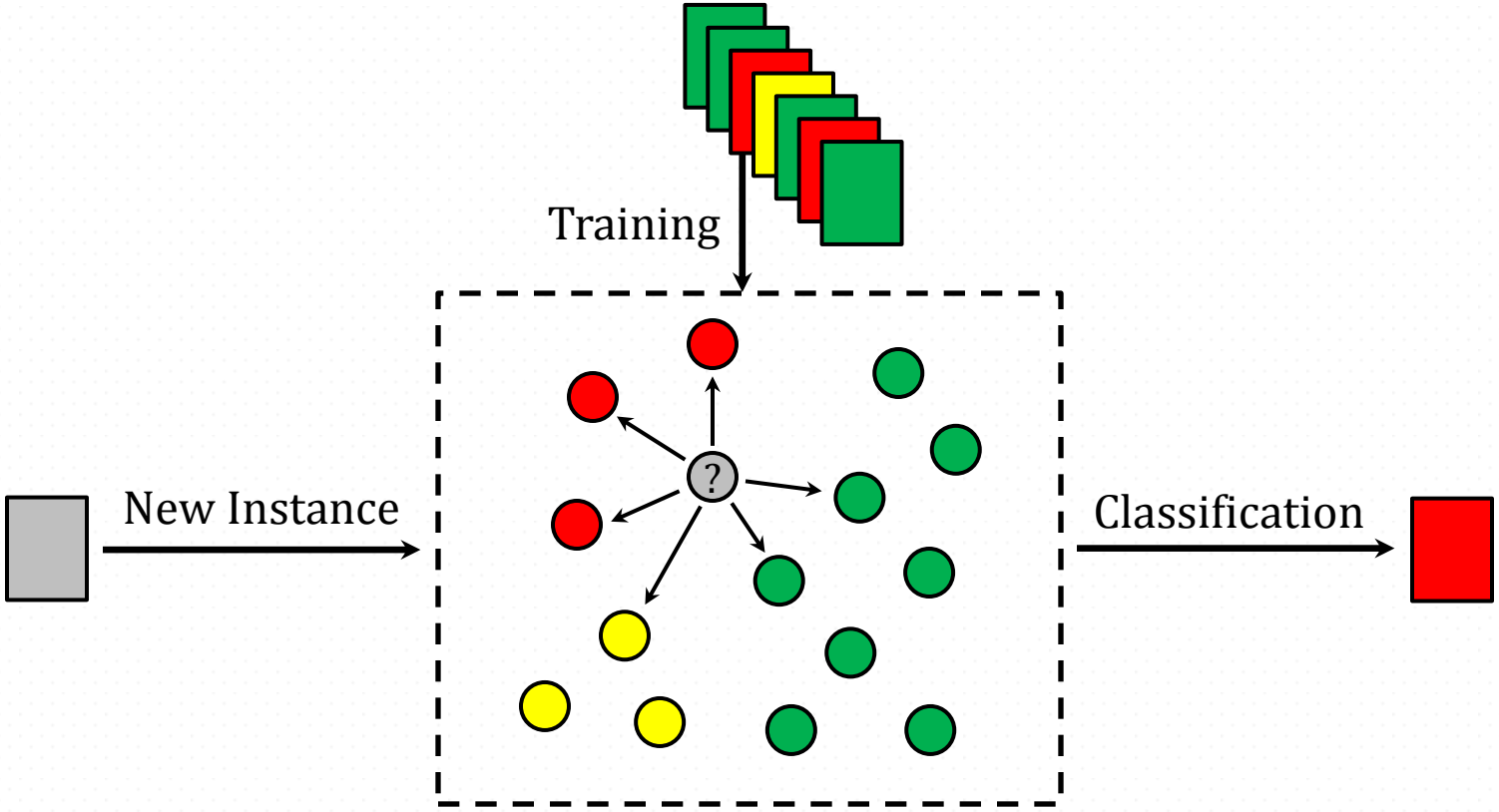
k-Nearest Neighbor (kNN) Classifier

The Idea:

Find the k training instances that are closest to the test instance, and classify the test instance as the majority class of the k nearest training instances.

"If it walks like a duck, quacks like a duck, and looks like a duck, then it's probably a duck."

k-Nearest Neighbor (kNN) Classifier



The kNN Algorithm

The k-nearest neighbor classification algorithm.

1: Let k be the number of nearest neighbors and D be the set of training examples.

2: **for** each test example $z = (\mathbf{x}', y')$ **do**

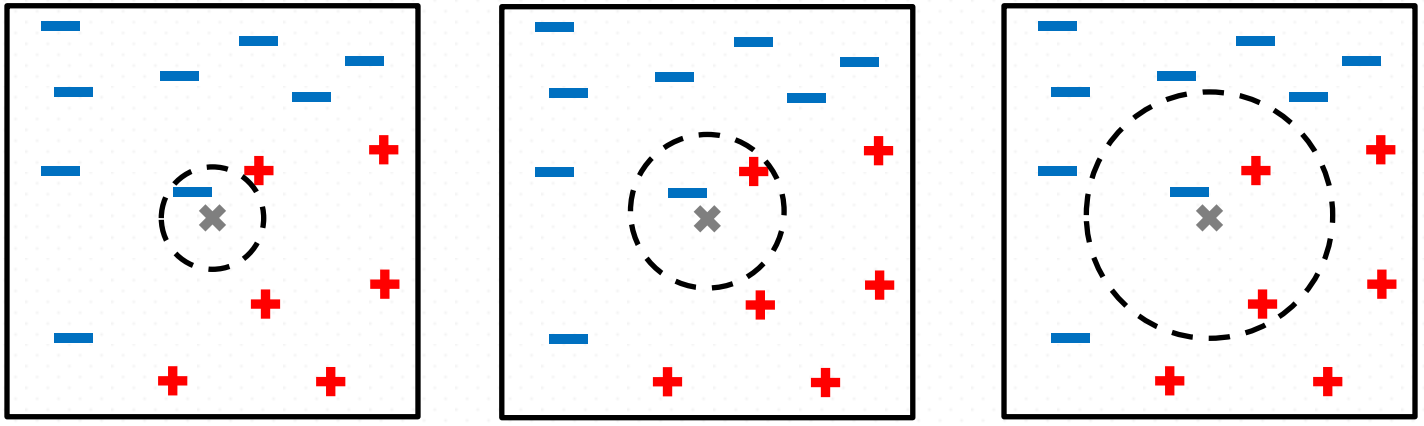
3: Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every example, $(\mathbf{x}, y) \in D$.

4: Select $D_z \subseteq D$, the set of k closest training examples to z.

5: $y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$

6: **end for**

Nearest Neighbors



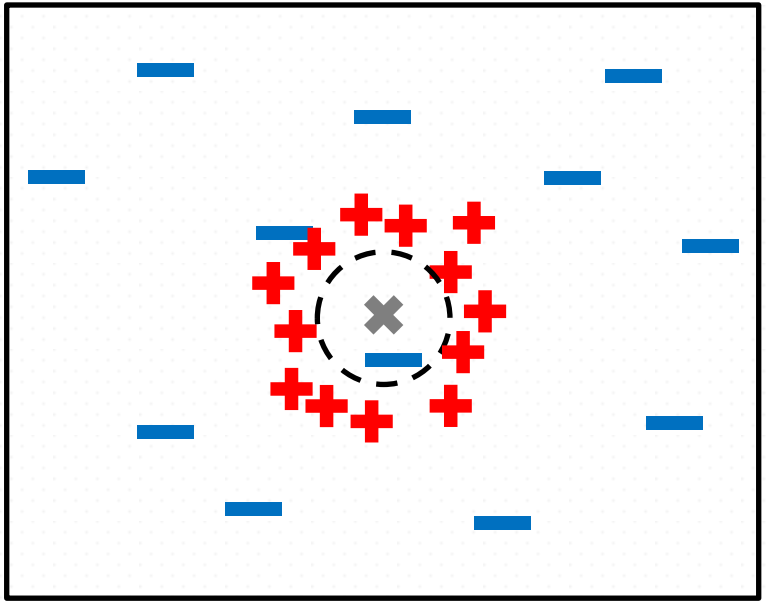
1-nearest neighbor | 2-nearest neighbor | 3-nearest neighbor

The k nearest neighbors of an example x are the data points that have the k smallest distances to x.

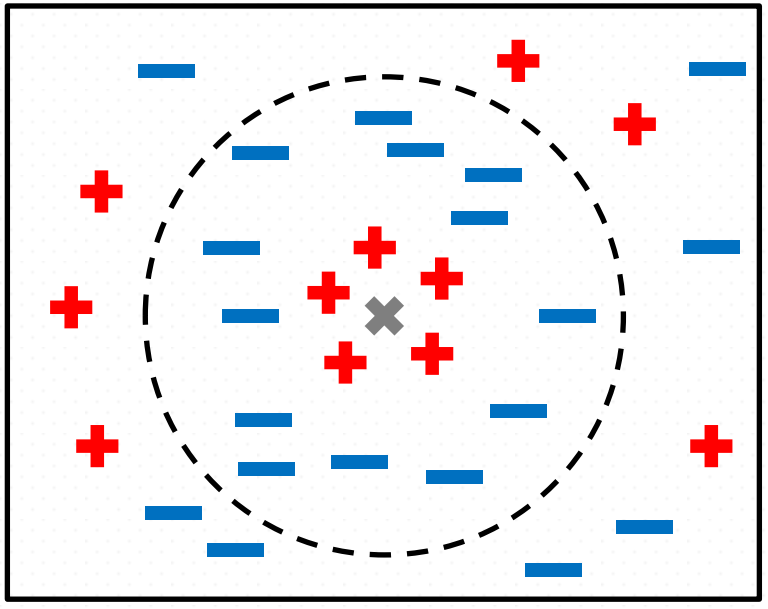
Choosing the k for kNN

- If k is too small, then the nearest-neighbor classifier may be susceptible to overfitting because of noise in the training data.
- If k is too large, the nearest-neighbor classifier may misclassify the test instance because its list of nearest neighbors may include data points that are located far away from its neighborhood.

Choosing the k for kNN



k-nearest neighbor classification with small k.



k-nearest neighbor classification with large k.

*k*NN Prediction: Classification

Classification: Take the majority vote of class labels among the k-nearest neighbors:

$$\text{Majority voting:} \quad y = \underset{v}{\text{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$$

where v is a class label, y_i is the class label for one of the nearest neighbors, and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

*k*NN Prediction: Regression

Regression: Predict the average value of the class value of the k-nearest neighbors.

Average value: $$y = \frac{1}{(\mathbf{x}_i, y_i) \in D_z} \sum_{(\mathbf{x}_i, y_i) \in D_z} y_i$$

where v is a class label and y_i is the class label for one of the nearest neighbors.

kNN Prediction: Distance-Weighted

- In the majority voting approach, every neighbor has the same impact on the classification.
- Instead, the influence of each nearest neighbor x_i can be weighted according to distance: $w_i = 1/d(\mathbf{x}', \mathbf{x}_i)^2$.
- Then, the class label can be determined as follows:

$$y = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i) .$$

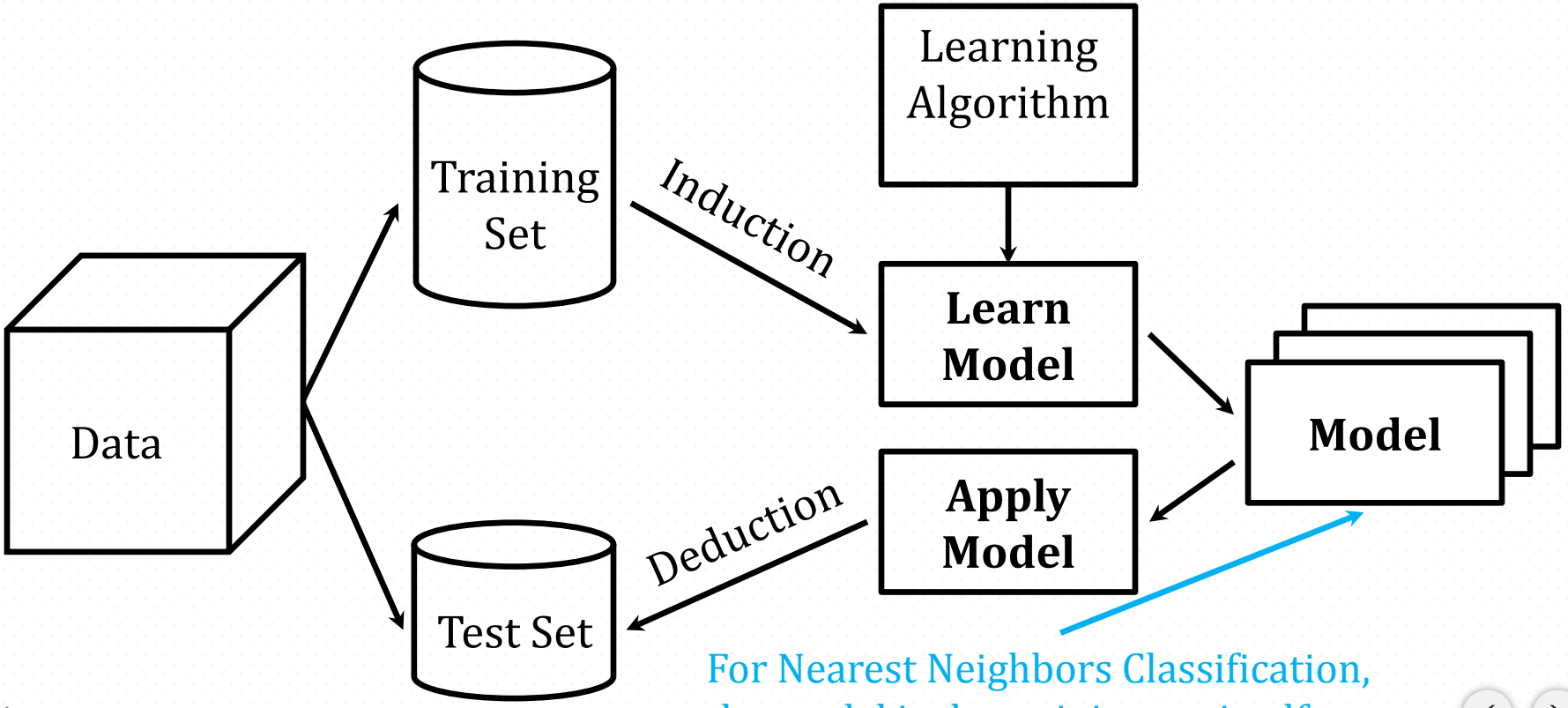
*k*NN Efficiency

- Very efficient in model induction (training)
  - Only store the training data.
- Not particularly efficient in testing
  - Computation of distance measure to every training instance
- Note that 1NN and *k*NN are equally efficient
  - Retrieving the *k* nearest neighbors is (almost) no more expensive than retrieving a single nearest neighbor
  - *k* nearest neighbors can be maintained in a queue.

*k*NN Summary

- Nearest-neighbor classification is part of a more general technique known as instance-based learning.
- Lazy learners such as nearest-neighbor classifiers do not require model building.
- Generate their predictions based on local information.
- Can produce arbitrarily shaped decision boundaries.
- Can easily produce wrong predictions without appropriate data preprocessing.

Back to Building a Classification Model



For Nearest Neighbors Classification, the model is the training set itself.

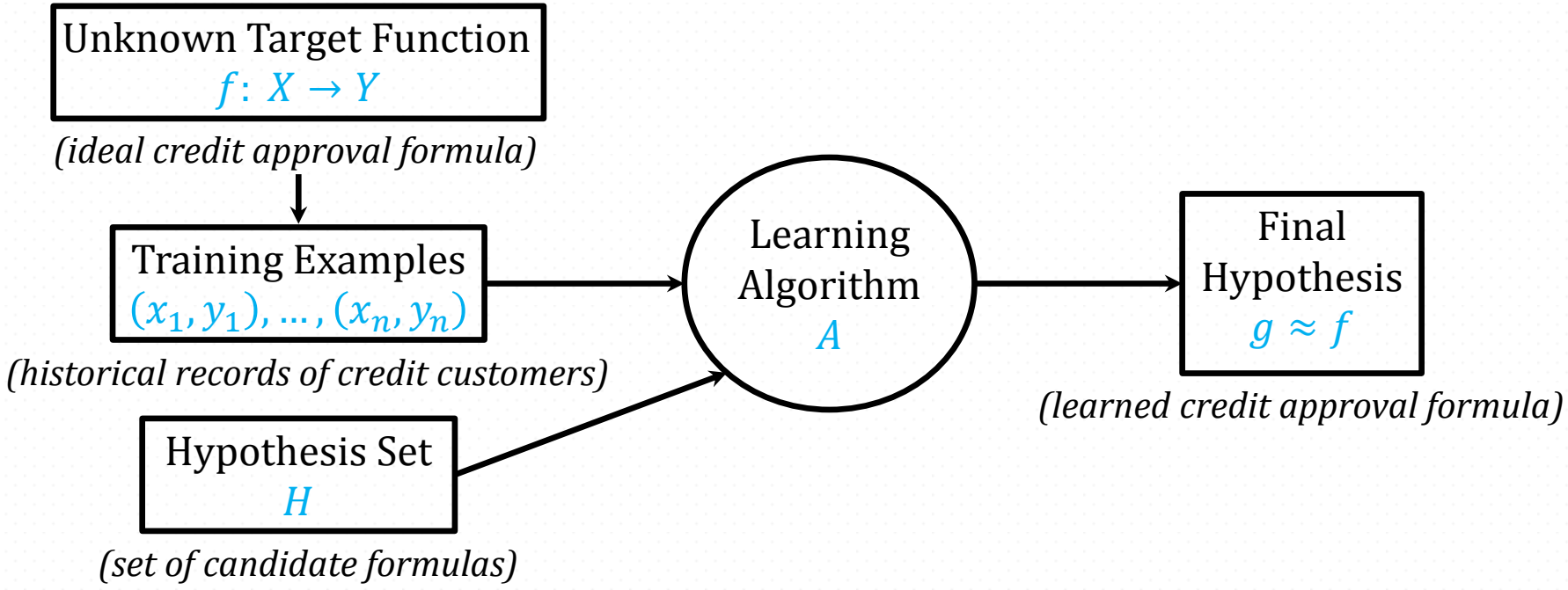
*Let's see how a model that **actually** learns works.*

Components of Learning

- Suppose that a bank wants to automate the process of evaluating credit card applications.
  - Input x (customer information that is used to make a credit application).
  - Target function $f: X \to Y$ (ideal formula for credit approval), where X and Y are the input and output space, respectively.
  - Dataset D of input-output examples $(x_1, y_1), \ldots, (x_n, y_n)$.
  - Hypothesis (skill) with hopefully good performance:

$$g: X \to Y \text{ (“learned” formula to be used)}$$

Components of Learning



Use data to compute hypothesis g that approximates target f

Simple Learning Model: The Perceptron

For $\mathbf{x} = x_1, \ldots, x_d$ ("features of the customer"), compute a weighted score and:

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

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

Simple Learning Model: The Perceptron

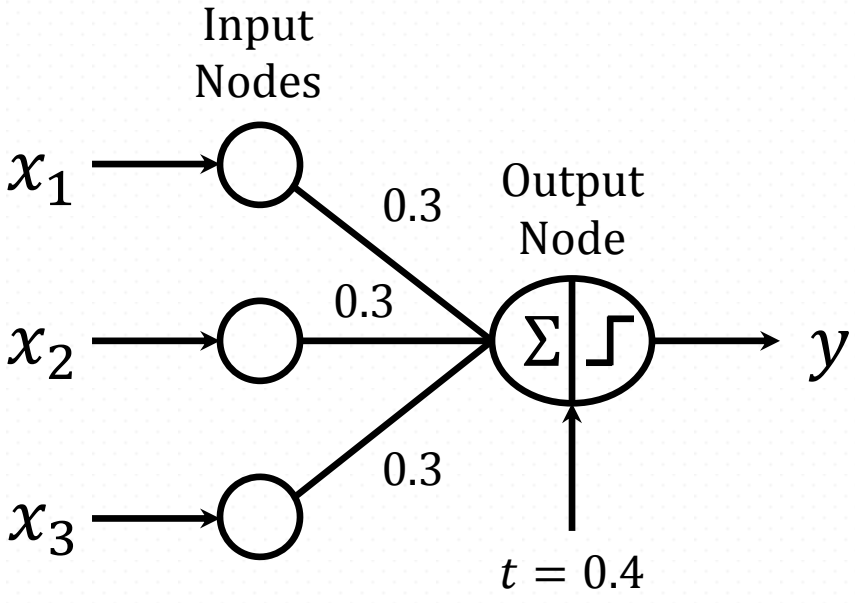
This formula can be written more compactly as

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

where $h(\mathbf{x}) = +1$ means 'approve credit' and $h(\mathbf{x}) = -1$ means 'deny credit'; $\text{sign}(s) = +1$ if $s > 0$ and $\text{sign}(s) = -1$ if $s < 0$. This model is called a *perceptron*.

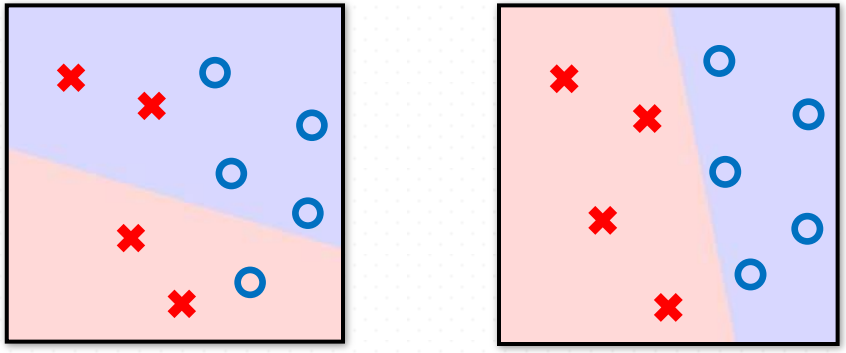
Simple Learning Model: The Perceptron

x_1	x_2	x_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Perceptrons (in $\mathbb{R}^2$)

$$h(\mathbf{x}) = sign(w_0 + w_1 x_1 + w_2 x_2)$$



The *perceptron* is a linear (binary) classifier:

- Customer features **x**: points on the plane
- Labels y: (+1), (-1)
- Hypothesis h: line (divide positive and negative)

And now...

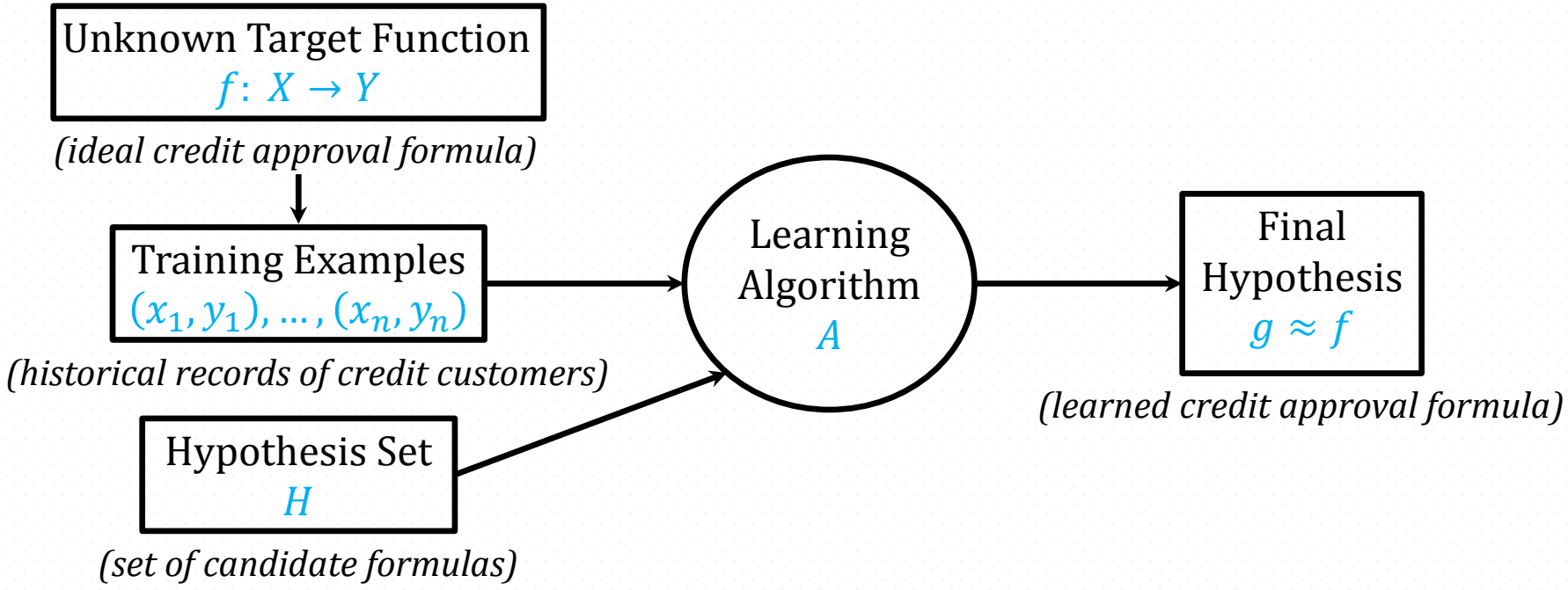
Let's see some classifying!

Components of Learning

- Suppose that a bank wants to automate the process of evaluating credit card applications.
  - Input x (customer information that is used to make a credit application).
  - Target function $f: X \to Y$ (ideal formula for credit approval), where X and Y are the input and output space, respectively.
  - Dataset D of input-output examples $(x_1, y_1), \ldots, (x_n, y_n)$.
  - Hypothesis (skill) with hopefully good performance:

$$g: X \to Y \text{ (“learned” formula to be used)}$$

Components of Learning



Use data to compute hypothesis g that approximates target f

Simple Learning Model: The Perceptron

For $\mathbf{x} = x_1, \ldots, x_d$ ("features of the customer"), compute a weighted score and:

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

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

Perceptron: A Mathematical Description

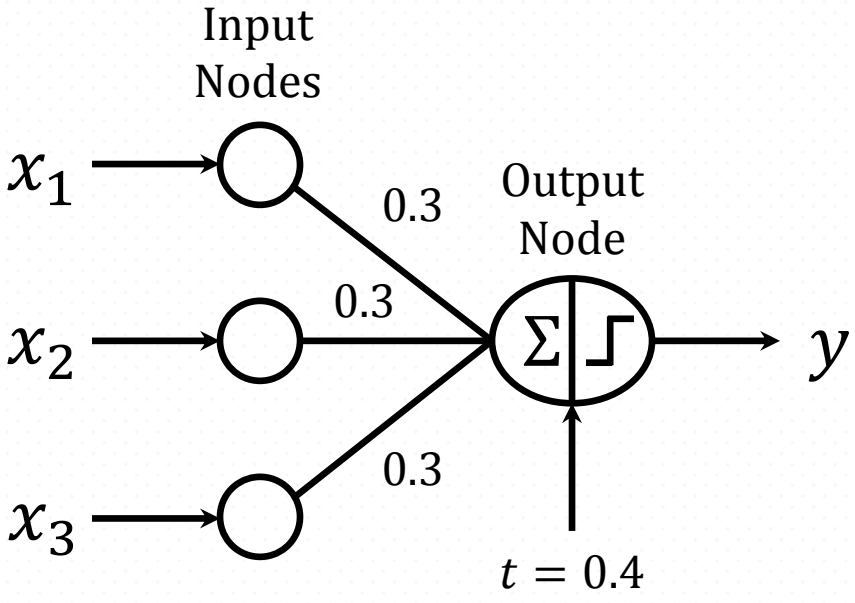
This formula can be written more compactly as

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

where $h(\mathbf{x}) = +1$ means 'approve credit' and $h(\mathbf{x}) = -1$ means 'deny credit'; $\text{sign}(s) = +1$ if $s > 0$ and $\text{sign}(s) = -1$ if $s < 0$. This model is called a *perceptron*.

Perceptron: A Visual Description

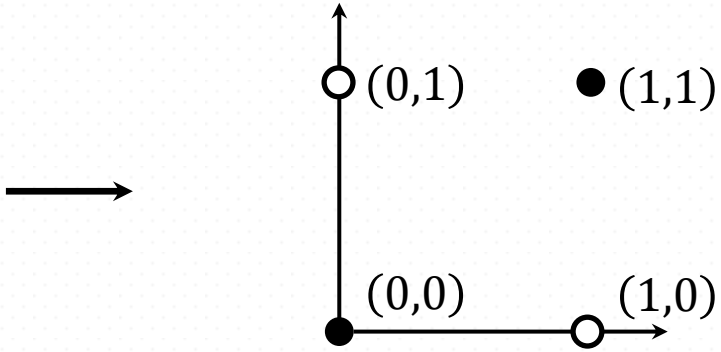
x_1	x_2	x_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



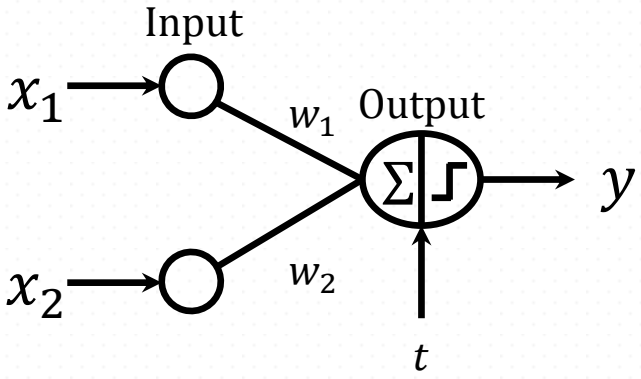
Perceptron Limitations

Data

x_1	x_2	$y = x_1$ XOR x_2
1	1	0
1	0	1
0	1	1
0	0	0



Model

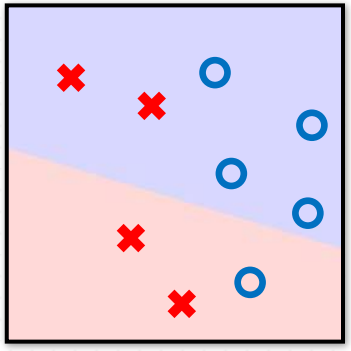


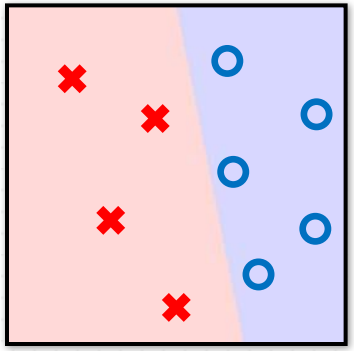
The following cannot all be true:

$$w_1 \times 1 + w_2 \times 1 < t$$
$$w_1 \times 1 + w_2 \times 0 > t$$
$$w_1 \times 0 + w_2 \times 1 > t$$
$$w_1 \times 0 + w_2 \times 0 < t$$

Perceptrons (in $\mathbb{R}^2$)

$$h(\mathbf{x}) = sign(w_0 + w_1 x_1 + w_2 x_2)$$





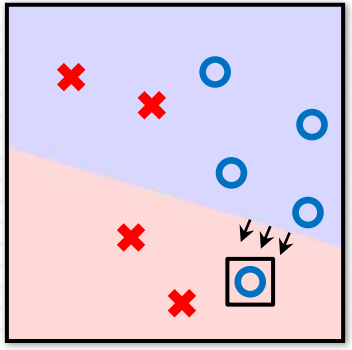
The *perceptron* is a linear (binary) classifier:

- Customer features **x**: points on the plane
- Labels y: (+1), (-1)
- Hypothesis h: line (divide positive and negative)

Perceptron Learning (in $\mathbb{R}^2$)

The algorithm will find an initial $\mathbf{w} = \{w_0, w_1, w_2\}$ from the training data. At each iteration t, $\mathbf{w}$ has a current value, $\mathbf{w}(t)$. The algorithm picks a misclassified instance, say $\big(\mathbf{x}(t), y(t)\big)$, and uses it to update $\mathbf{w}(t)$:

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



Summary of Perceptrons

- Simple and intuitive mathematical formulation
- Fast.
- A linear classifier or discriminator.
  - Only capable of learning linearly separable patterns.
- Models a binary output variable.

Bayesian Learning

- In many applications, the relationship between the feature set and the class variable is non-deterministic.

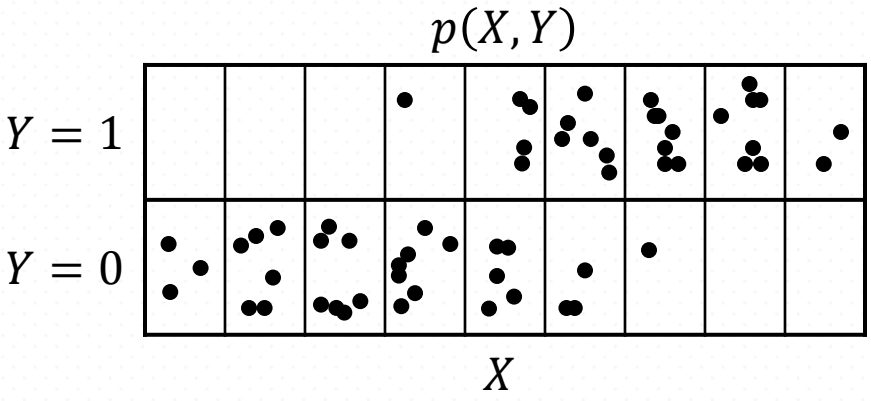
Conditional Probability

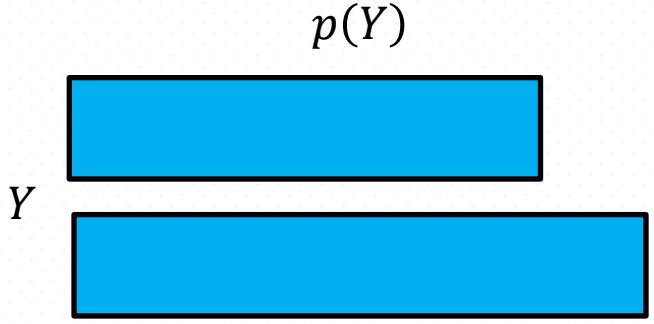
Conditional Probability is useful for understanding the dependencies among random variables. The conditional probability for the random variable Y given the random variable X, denoted as $P(Y|X)$, is defined as:

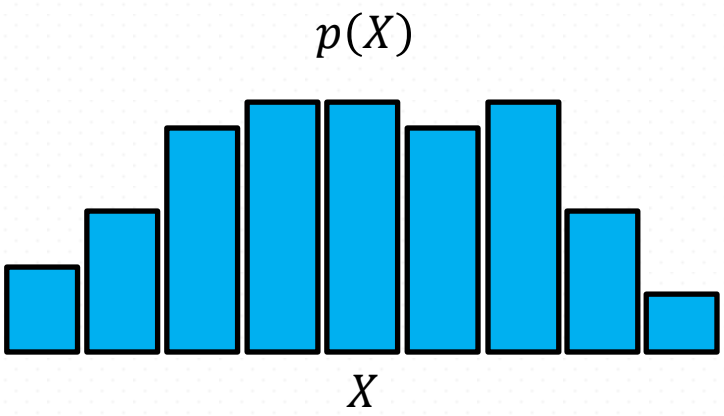
$$P(Y|X) = \frac{P(X,Y)}{P(X)}$$

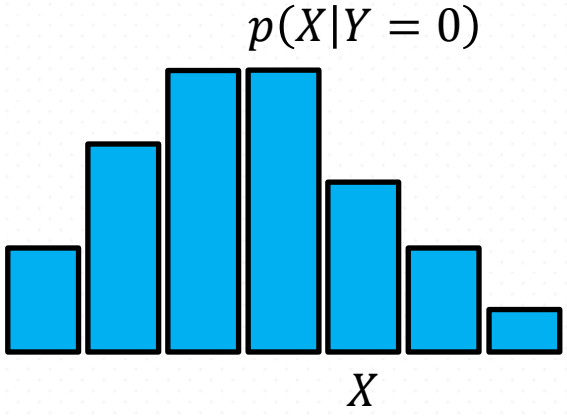
If X and Y are independent (the value of one variable has no impact on the other), then $P(Y|X) = P(Y)$.

Conditional Probability









Bayes' Theorem

The conditional probabilities $P(Y|X)$ and $P(X|Y)$, where X and Y are random variables, can be expressed in terms of one another using a formula known as **Bayes' theorem**:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}.$$

Law of Total Probability

If $\{X_1, X_2, \ldots, X_k\}$ is the set of mutually exclusive and exhaustive outcomes of a random variable X, then the denominator of Bayes' theorem can be expressed as:

$$P(X) = \sum_{i=1}^{k} P(X, Y_i) = \sum_{i=1}^{k} P(X|Y_i)P(Y_i)\,.$$

This is called the **law of total probability.**

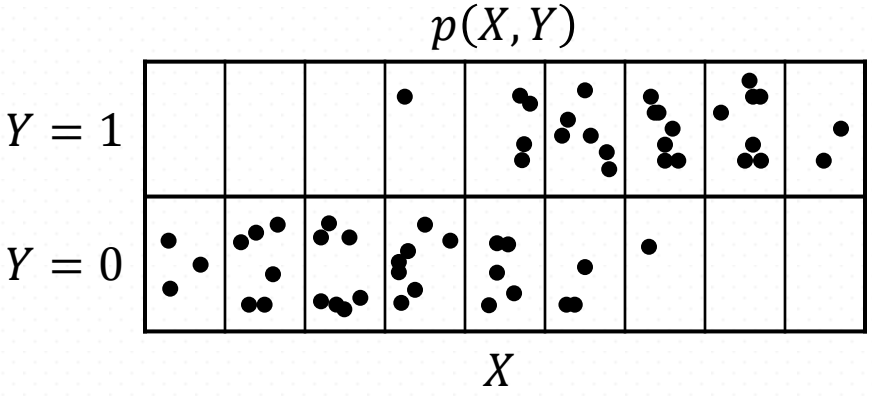
Bayes' Theorem for Classification

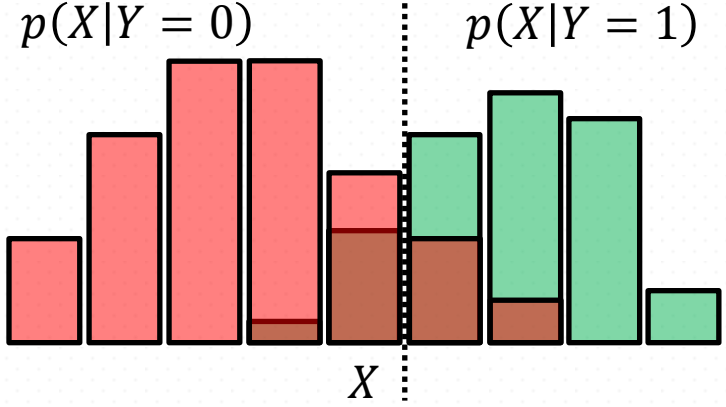
Using Bayes' theorem for classification problems, we get

$$P(Class|Feature\ Set) = \frac{P(Feature\ Set|Class)P(Class)}{P(Feature\ Set)}$$

Posterior Probability $\propto$ Class Conditional Probability $\times$ Prior

Bayesian Decisions





- Notice that feature values for class 1 ($Y = 1$) tend to be larger than those for class 0 ($Y = 0$). The dashed line denotes the decision boundary between classes 0 and 1 based on feature X.

Example Application of Bayes' Theorem

Consider a football game between two rival teams: Team 0 and Team 1. Support Team 0 wins 65% of the time and Team 1 wins the remaining matches. Among the games won by Team 0, only 30% of them come from playing on Team 1's football field. On the other hand, 75% of the victories for Team 1 are obtained while playing at home. If Team 1 is to host the next match between the two teams, which team will most likely emerge as the winner?

Example Application of Bayes' Theorem

Let X be the team hosting the match and Y be the winner of the match. Both X and Y can take on values from the set $\{0,1\}$. Then:

Probability Team 0 wins is $P(Y = 0) = 0.65$.

Probability Team 1 wins is $P(Y = 1) = 1 - 0.65 = 0.35$.

Probability Team 1 hosted the match it won is

$$P(X = 1|Y = 1) = 0.75.$$

Probability Team 1 hosted the match won by Team 0 is

$$P(X = 1|Y = 0) = 0.3.$$

Example Application of Bayes' Theorem

$$P(Y = 1|X = 1) = \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1)}$$

$$= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1, Y = 1) + P(X = 1, Y = 0)}$$

$$= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1|Y = 1)P(Y = 1) + (X = 1|Y = 0)P(Y = 0)}$$

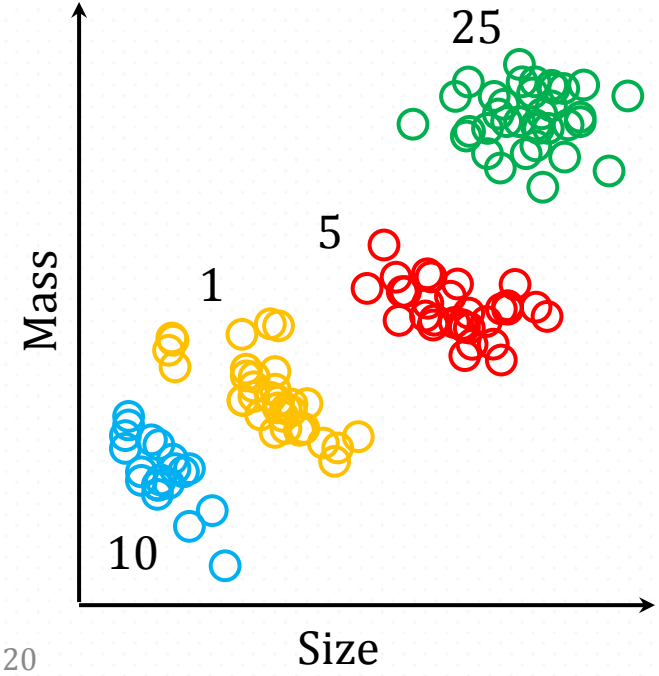
$$= \frac{0.75 \times 0.35}{0.75 \times 0.35 + 0.3 \times 0.65}$$

$$= 0.5738$$

Classification Paradigms

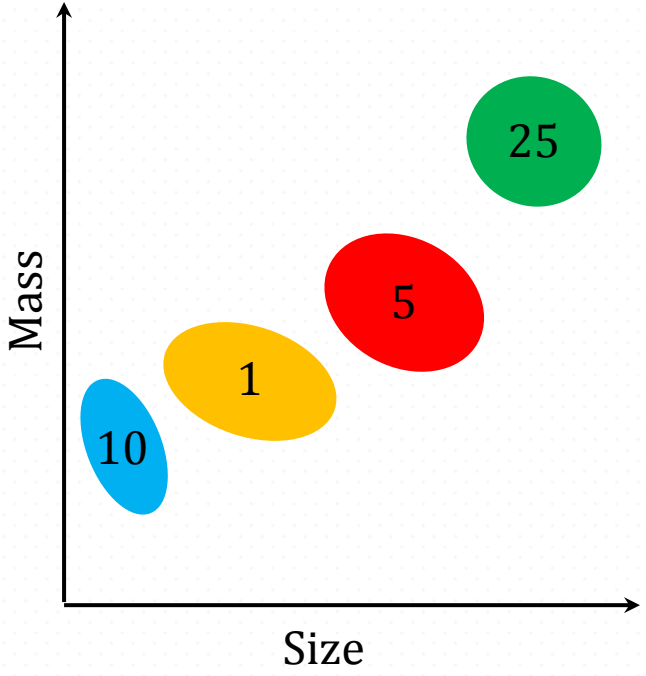
Discriminative

Determine $P(Class_k|\mathbf{x})$ Directly



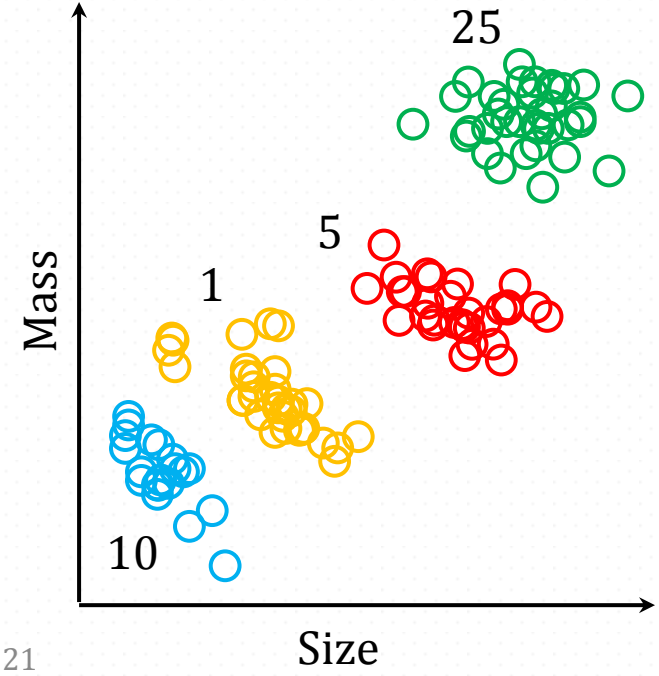
Generative

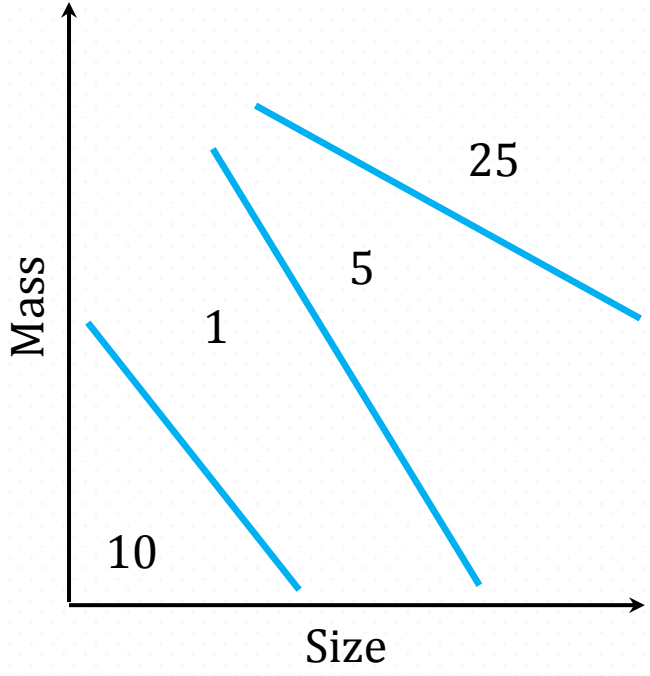
Model $P(\mathbf{x}|Class_k)$ and $P(Class_k)$ Separately and Use Bayes' Theorem to Find $P(Class_k|\mathbf{x})$



Classification Paradigms: Discriminative

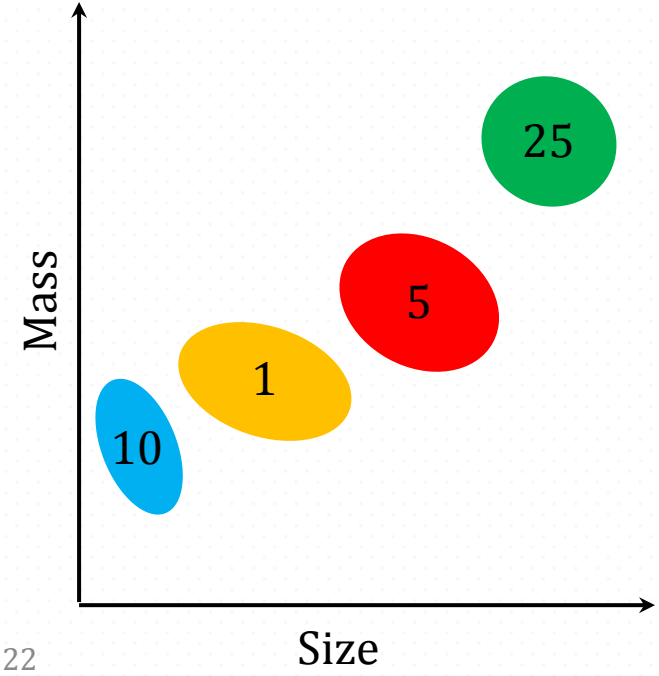
- Consider learning coin classification

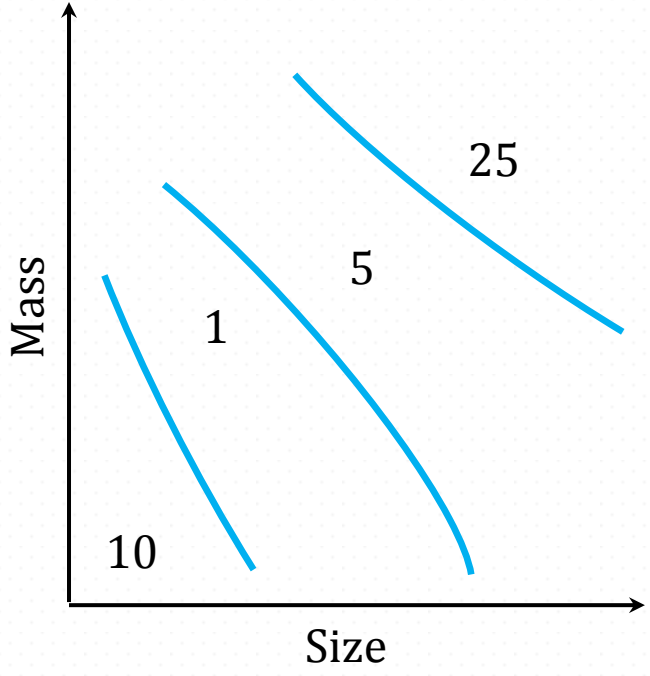




Classification Paradigms: Generative

- Consider learning coin classification





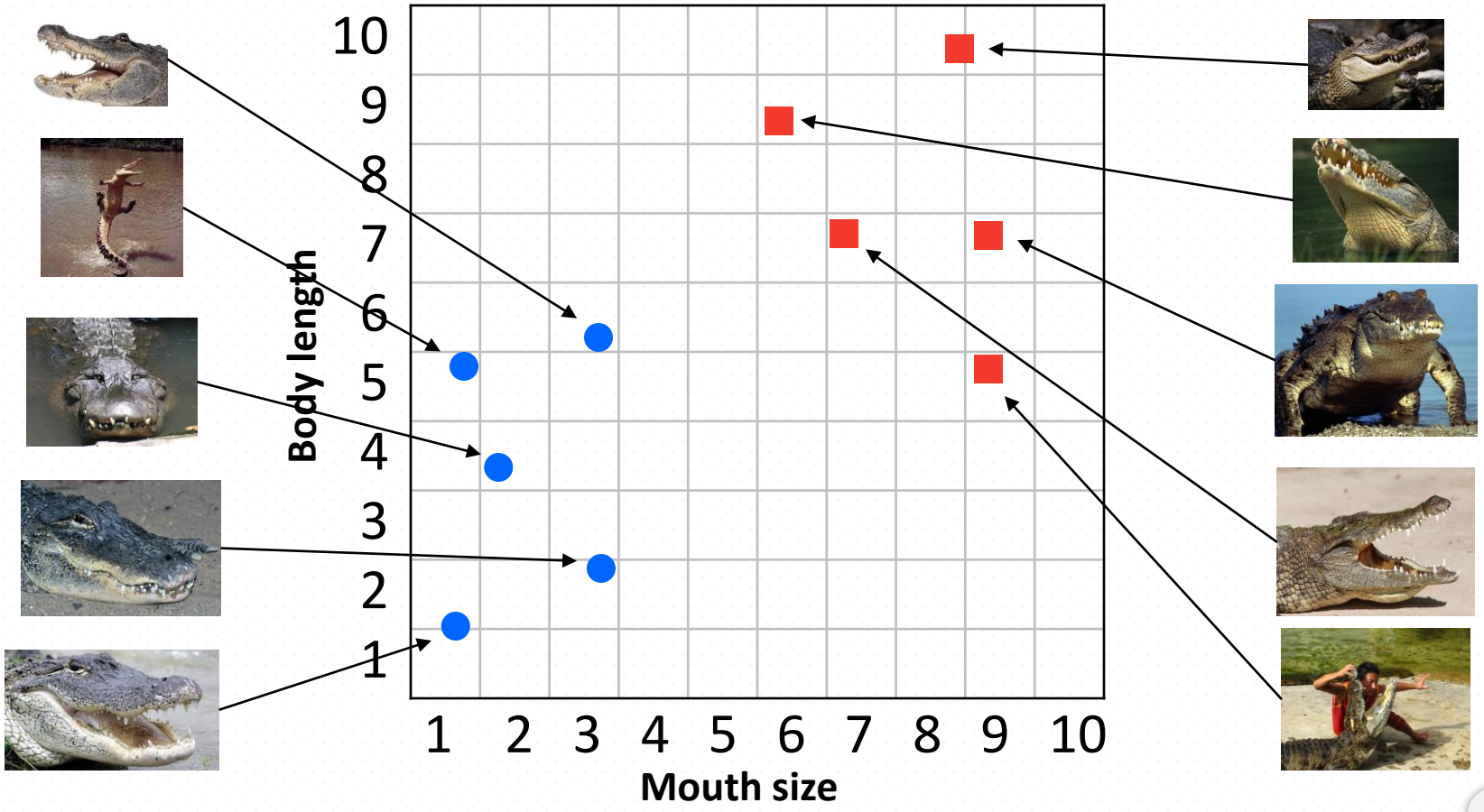
Naïve Bayes Classifier

(pages 231–238 on text book)

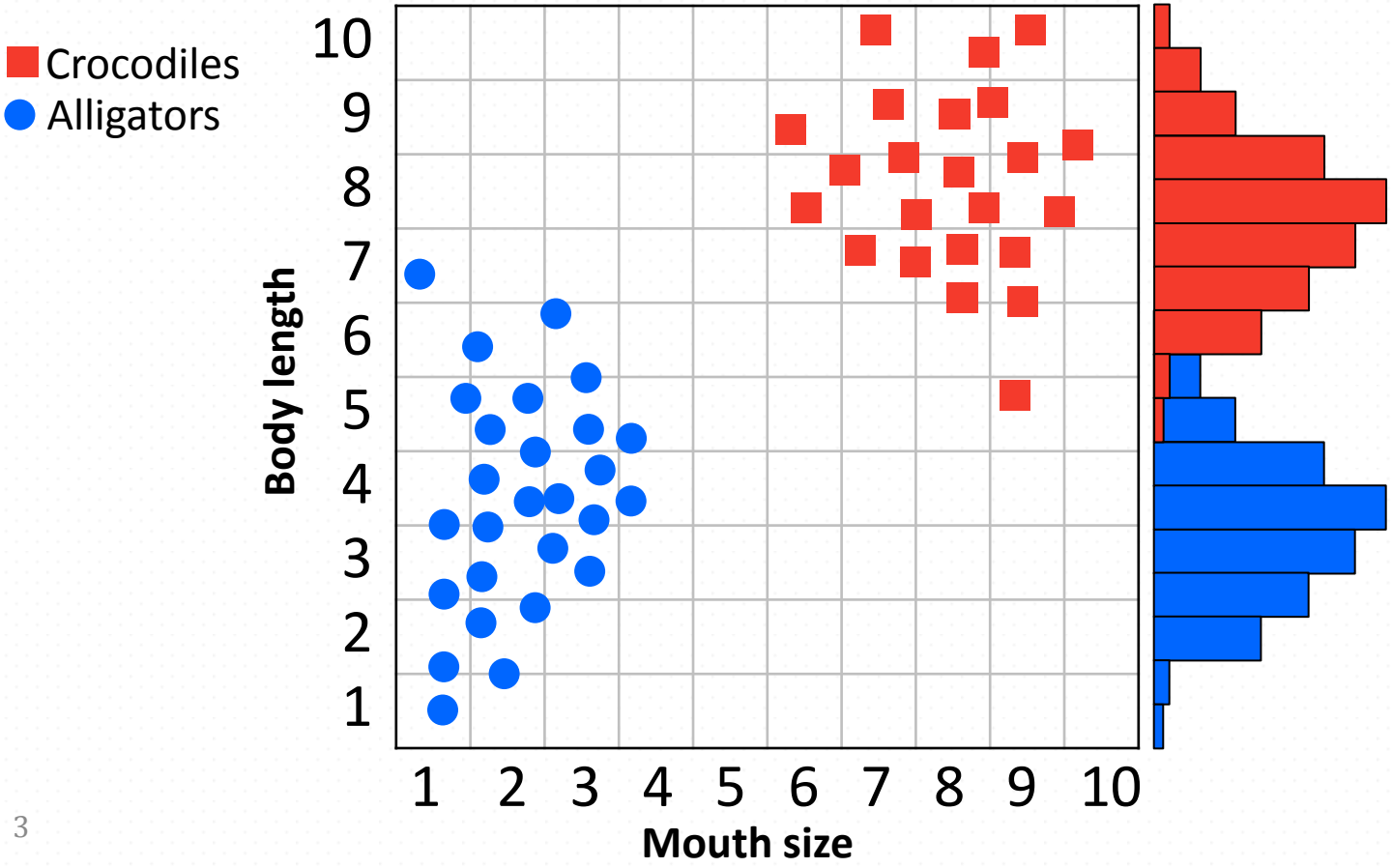
- Lets start off with a visual intuition
  - Adapted from Dr. Eamonn Keogh's lecture – UCR

Alligators

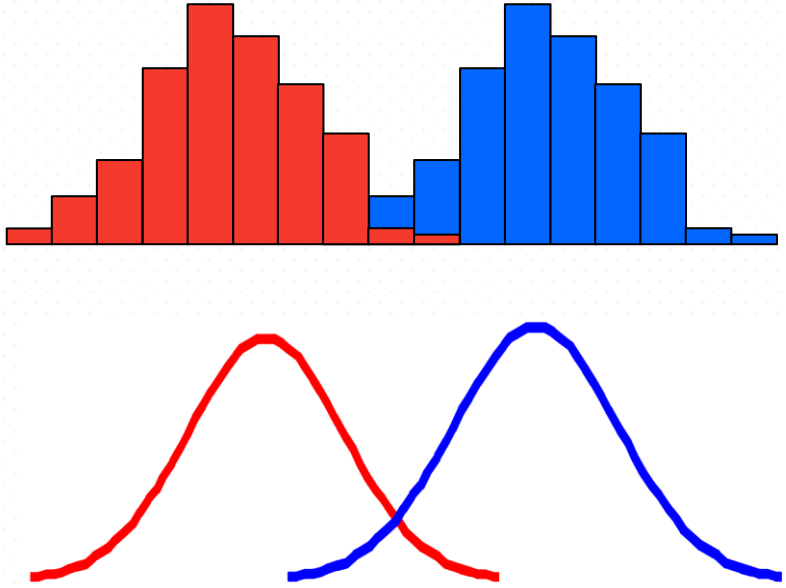
Crocodiles



Suppose we had a lot of data. We could use that data to build a histogram. Below is one built for the *body length* feature:

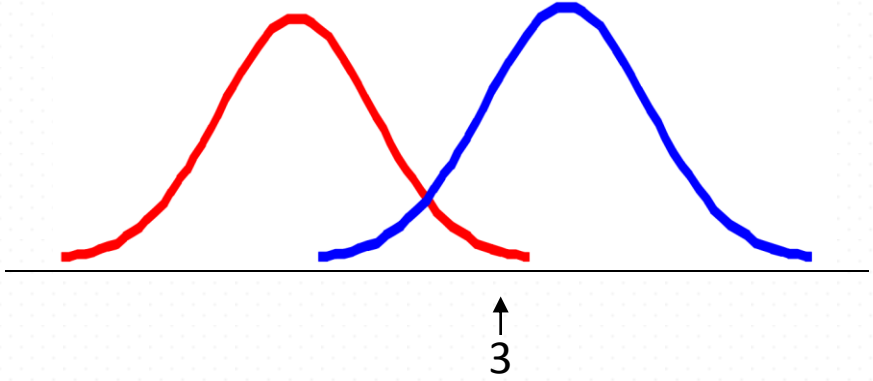


We can summarize these histograms as two normal distributions.



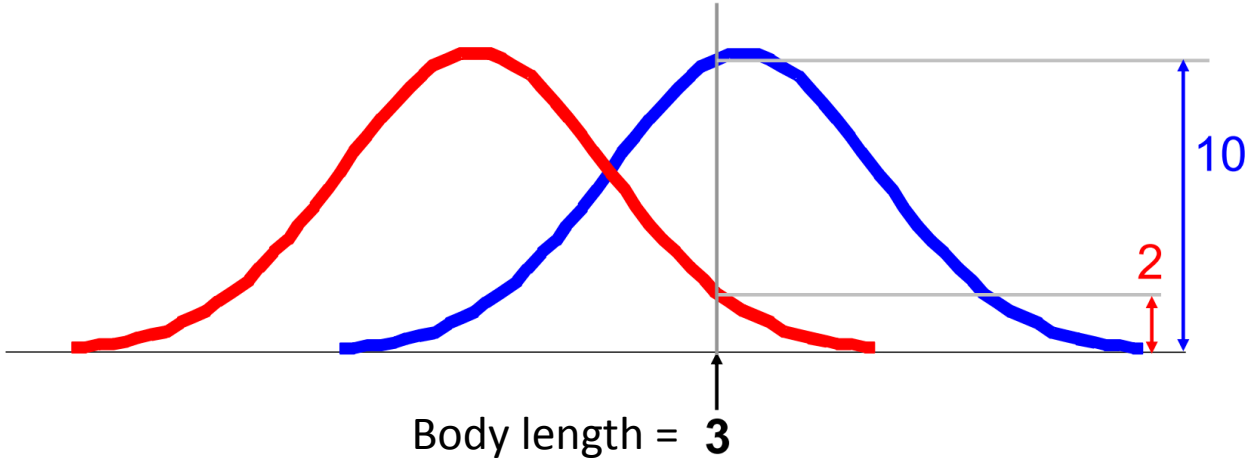
- Suppose we wish to classify a new animal that we just found. Its body length is 3 units. How can we classify it?
- One way to do this is, given the distributions of that feature, we can analyze which class is more *probable:* Crocodile or Alligator.
- More formally:

$$p(c_j|d) = probability\ of\ class\ \boldsymbol{c_j}, given\ that\ we\ observed\ \boldsymbol{d}$$



$$p(c_j|d) = probability\ of\ class\ \boldsymbol{c_j}, given\ that\ we\ observed\ \boldsymbol{d}$$

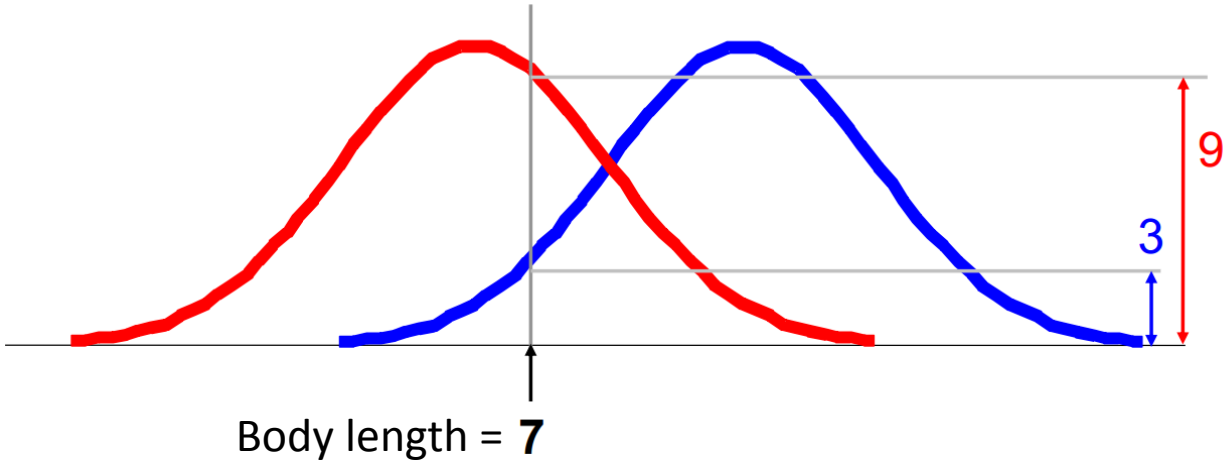
$$p(Alligator|body\ length = 3) = 10/(10+2) = \mathbf{0.833}$$
$$p(Crocodile|body\ length = 3) = 2/(10+2) = \mathbf{0.166}$$



$$p(c_j|d) = probability\ of\ class\ \boldsymbol{c_j}, given\ that\ we\ observed\ \boldsymbol{d}$$

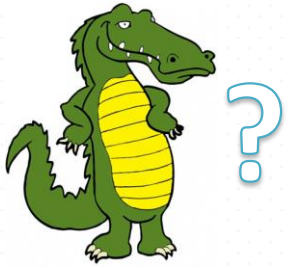
$$p(Alligator|body\ length = 7) = 3/(3+9) = \mathbf{0.25}$$
$$p(Crocodile|body\ length = 7) = 9/(3+9) = \mathbf{0.75}$$

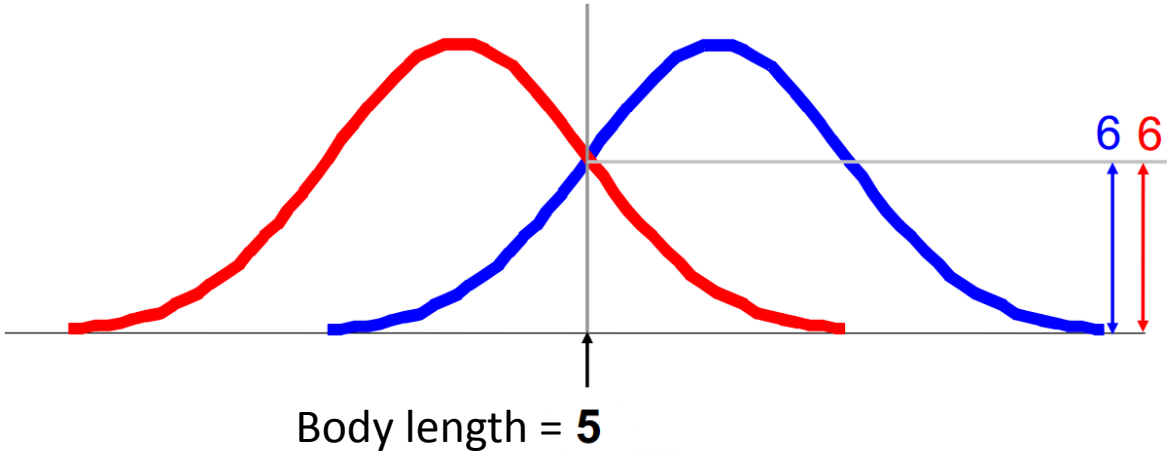
?



$$p(c_j|d) = probability\ of\ class\ \boldsymbol{c_j}, given\ that\ we\ observed\ \boldsymbol{d}$$

$$p(Alligator|body\ length = 5) = 6/(6+6) = \mathbf{0.5}$$
$$p(Crocodile|body\ length = 5) = 6/(6+6) = \mathbf{0.5}$$





Naïve Bayes Classifier

- This visual intuition describes a simple Bayes classifier commonly known as:
  - Naïve Bayes
  - Simple Bayes
  - Idiot Bayes
- While going through the math, keep in mind the basic idea:

Given a new unseen instance, we (1) find its probability of it belonging to each class, and (2) pick the most probable.

Bayes Theorem

Posterior probability

Likelihood

Class prior probability

$$P(c_j|d) = \frac{P(d|c_j)P(c_j)}{P(d)}$$

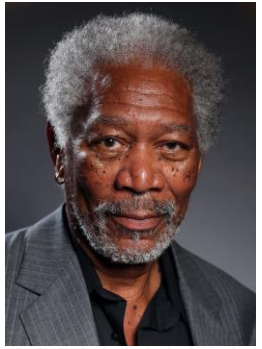
Predictor prior probability

- $\boldsymbol{P(c_j|d)}$ = probability of instance d being in class c_j
- $\boldsymbol{P(d|c_j)}$ = probability of generating instance d given class c_j
- $\boldsymbol{P(c_j)}$ = probability of occurrence of class c_j
- $\boldsymbol{P(d)}$ = probability of instance d occurring

Suppose we have another binary classification problem with the following two classes: $c_1 = male$, and $c_2 = female$

We now have a person called *Morgan*. How do we classify them as male or female?

Morgan Fairchild



Morgan Freeman

What is the probability of being called Morgan given that you are a male?

What is the probability of being a male?

$$P(male|morgan) = \frac{P(morgan|male)P(male)}{P(morgan)}$$

What is the probability of being called Morgan

Suppose this individual on your left (Morgan) was arrested for money laundering. Is Morgan male or female?

Assume we are given the following database of names. We can then apply Bayes rule.

Name	Sex
Morgan	Male
Reid	Female
Morgan	Male
Morgan	Female
Everaldo	Male
Francis	Male
Jennifer	Female

Name	Sex
Morgan	Male
Reid	Female
Morgan	Male
Morgan	Female
Everaldo	Male
Francis	Male
Jennifer	Female

$$P(c_j|d) = \frac{P(d|c_j)P(c_j)}{P(d)}$$

$$P(female|morgan) = \frac{1/3 * 3/8}{3/8} = \frac{0.125}{3/8}$$

$$P(male|morgan) = \frac{2/5 * 5/8}{3/8} = \frac{0.25}{3/8}$$

Money launderer Morgan is more likely to be male.

Naïve Bayes Classifier

- Both examples that we looked at considered only a single feature (i.e., *body length* and *name*)
- What if we have several features?

Name	Over 6ft	Eye color	Hair style	Sex
Morgan	Yes	Blue	Long	Female
Bob	No	Brown	None	Male
Vincent	Yes	Brown	Short	Male
Amanda	No	Brown	Short	Female
Reid	No	Blue	Short	Male
Lauren	No	Purple	Long	Female
Elisa	Yes	Brown	Long	Female

Naïve Bayes Classifier

- Naïve Bayes assumes that all features are independent (i.e., they have independent distributions).
- The probability of class c_j generating instance d can then be estimated as:

$$P(d|c_j) = P(d_1|c_j) \times P(d_2|c_j) \times \ \ldots \ \times \ P(d_n|c_j)$$

Probability of class c_j generating the observed value for feature 1

Probability of class c_j generating the observed value for feature 2

…

Naïve Bayes Classifier

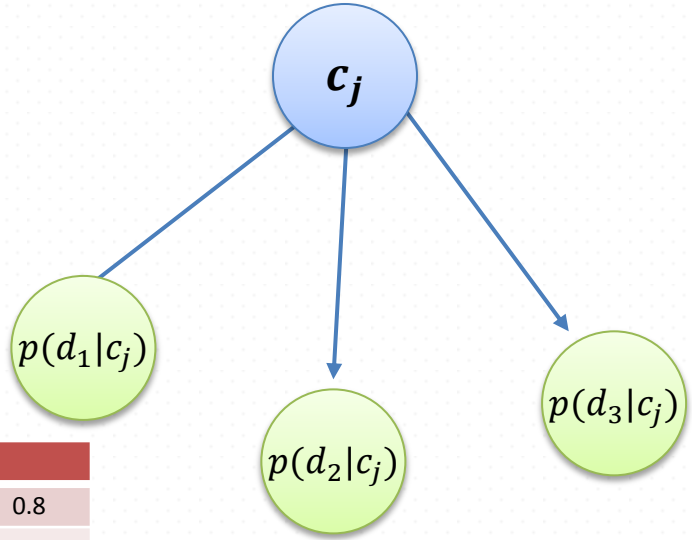
- Suppose we have Amanda's data:

Name	Over 6ft	Eye color	Hair style	Sex
Amanda	No	Brown	Short	?

$$\boldsymbol{P(Amanda|c_j)} = P(over6ft = No|c_j) \times P(eyecolor = Brown|c_j) \times P(hair = Short|c_j)$$

Naïve Bayes Classifier

- A visual representation:
  - Each class implies certain features with certain probabilities



Sex	Over 6ft	
Male	Yes	0.8
	No	0.2
Female	Yes	0.01
	No	0.99

Sex	Eye color	
Male	Brown	0.67
	Blue	0.33
Female	Brown	0.45
	Blue	0.55

Sex	Hair	
Male	None	0.1
	Short	0.8
	Long	0.1
Female	None	0.05
	Short	0.15
	Long	0.8

All tables can be computed with a single scan of our dataset: **Very efficient!**

Naïve Bayes Classifier

- Naïve Bayes is not sensitive to irrelevant features.
  - For instance, if we were trying to classify a person's sex based on a set of features that included *eye color* (which is irrelevant to their gender):

$$P(Reid|c_j) = P(eye = blue|c_j) \times P(wears\ makeup = Yes|c_j) \times \ \ldots$$

$$P(Reid|Female) = 9{,}000/10{,}000 \times 9{,}500/10{,}000 \times \ldots$$

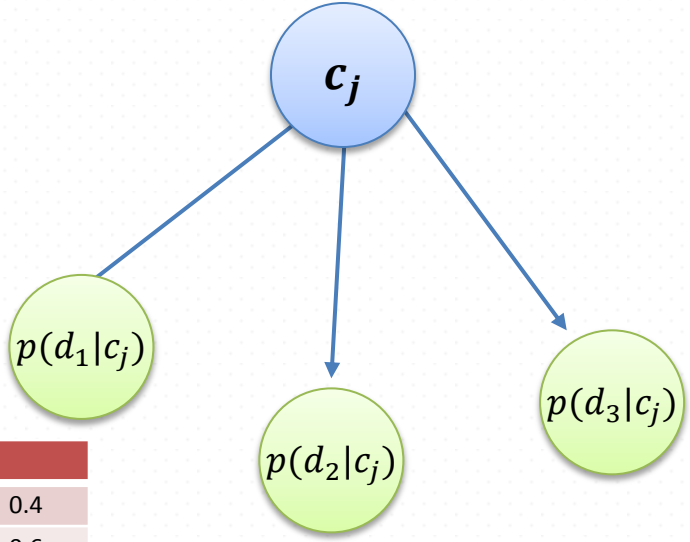
$$P(Reid|Male) = 9{,}007/10{,}000 \times 1/10{,}000 \times \ldots$$

Nearly identical!

  - The more data, the better!

Naïve Bayes Classifier

- Naïve Bayes works with any number of classes or features:

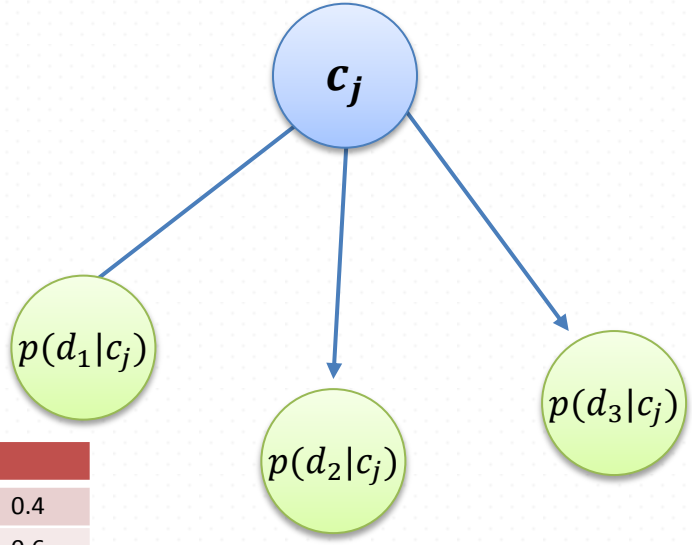


Sex	Friendly	
Crocodile	Yes	0.4
	No	0.6
Alligator	Yes	0.3
	No	0.7
T-Rex	Yes	0.05
	No	0.95

Sex	Color	
Crocodile	Black	0.1
	Brown	0.2
	Green	0.7
Alligator	Black	0.2
	Brown	0.3
	Green	0.5
T-Rex	Black	0.2
	Brown	0.2
	Green	0.6

Naïve Bayes Classifier - Problems

- Naïve Bayes assumes all features are independent (often **bad!**)



Sex	Friendly	
Crocodile	Yes	0.4
	No	0.6
Alligator	Yes	0.3
	No	0.7
T-Rex	Yes	0.05
	No	0.95

...

Sex	Color	
Crocodile	Black	0.1
	Brown	0.2
	Green	0.7
Alligator	Black	0.2
	Brown	0.3
	Green	0.5
T-Rex	Black	0.2
	Brown	0.2
	Green	0.6

Naïve Bayes Classifier - Problems

- What if one of the probabilities is zero? Ops.

$$P(d|c_j) = P(d_1|c_j) \times P(d_2|c_j) \times \ldots \times P(d_n|c_j)$$
$$= 0.15 \times \mathbf{0} \times \cdots \times 0.55$$

- This is not uncommon, especially when the number of instances is small
- Solution: **m-estimate**

m-estimate

- To avoid trouble when a probability $P(d_1|c_j) = 0$, we fix its prior probability and the number of samples to some non-zero value beforehand
  - Think of it as adding a bunch of fake instances before we start the whole process
- If we create $m > 0$ fake samples of feature X with value of x, and we assign a prior probability p to them, then posterior probabilities are obtained as:

$$P(X = x|c_j) = \frac{\#(X=x,c_j)+mp}{\#(c_j)+m}$$

Naïve Bayes Pros

- Can be used for both continuous and discrete features.
- Can be used for binary or multi-class problems.
- It is robust to isolated noise/outliers.
- Missing values are easily handled. They are simply ignored.
- It is robust to irrelevant attributes.

Naïve Bayes Cons

- Correlated features degrade performance. If we have the same feature multiple times in the dataset, we can change classification results.
- When m-estimate is used, the choice of *m's* and *p's* is very arbitrary.

Naïve Bayes – Class exercise

- Predict if Bob will default his loan

Bob

Home owner: *No*

Marital status: *Married*

Job experience: *3*

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

Naïve Bayes – Class exercise (1)

Bob

Home owner: *No*

Marital status: *Married*

Job experience: *3*

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

- $P(Y = No) = 7/10$
- $P(Home\ owner = No | Y = No) = 4/7$
- $P(Marital\ status = Married | Y = No) = 4/7$
- $P(Job\ experience = 3 | Y = No) = 2/7$

$$P(Bob\ will\ NOT\ default) = \frac{7}{10} \times \frac{4}{7} \times \frac{4}{7} \times \frac{2}{7} = \mathbf{0.065}$$

Naïve Bayes – Class exercise (2)

Bob

Home owner: *No*

Marital status: *Married*

Job experience: *3*

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

- $P(Y = Yes) = 3/10$
- $P(Home\ owner = No | Y = Yes) = 1/3$
- $P(Marital\ status = Married | Y = Yes) = 1/3$
- $P(Job\ experience = 3 | Y = Yes) = 1/3$

$$P(Bob\ will\ default) = \frac{3}{10} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} = \mathbf{0.011}$$

Naïve Bayes – Class exercise (3)

Bob

Home owner: *No*

Marital status: *Married*

Job experience: *3*

Home owner	Marital Status	Job experience (1-5)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	2	Yes

- $P(Bob\ will\ NOT\ default) = \mathbf{0.065}$
- $P(Bob\ will\ default) = \mathbf{0.011}$

Predict: BOB WILL NOT DEFAULT

Decision Trees

- Example of inductive learning
  - The process of learning by example – where a system tries to induce a general rule from a set of observed instances.
- Directed structure comprised of nodes
  - Each node specifies a test on an attribute
  - Each branch corresponds to an attribute value or condition
  - Leaves represent a class (or decision)
- Very wide application range

Constructing Decision Trees

- Top-down, recursive, divide and conquer
  1. Select best feature for root node. Construct a branch for every possible value of that feature.
  2. Split data into mutually exclusive subsets for each branch
  3. Repeat this process recursively using only the portion of data arriving at each node
  4. Stop when training examples can be perfectly classified → create a leaf node with the class decision

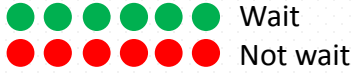
How to choose the splitting attribute?

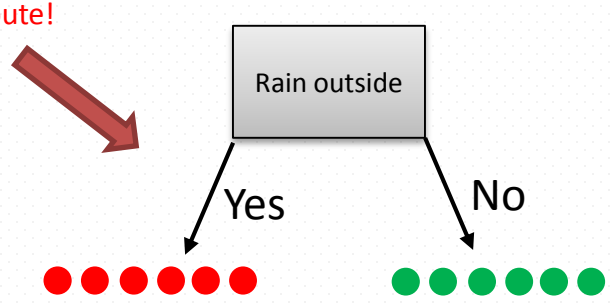
- Information Gain (used in ID3, C4.5)
- Gain Ratio (used in C4.5)
- Gini Measure (used in CART)

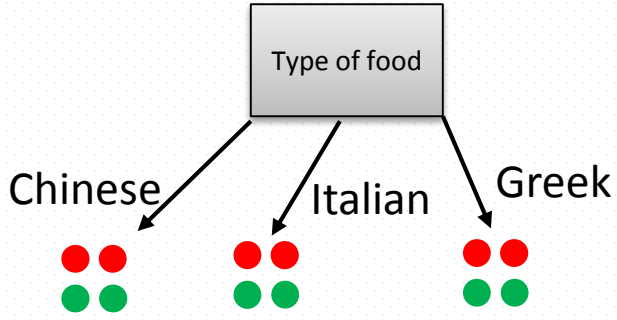
Determining the best split

- Greedy approach:
  - Choose nodes with **homogeneous** class distributions
  - Suppose we are trying to analyze a dataset to figure out if people will wait outside a restaurant for food

Homogeneous
Low degree of impurity
Lower ***entropy***
Better attribute!



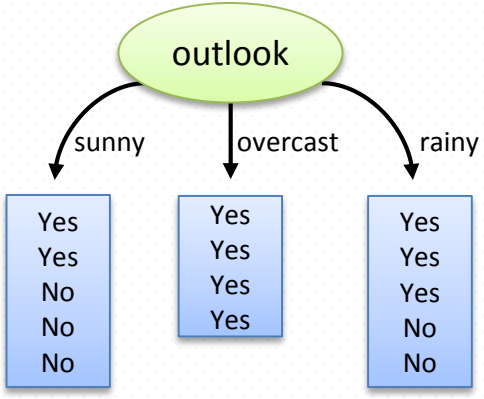


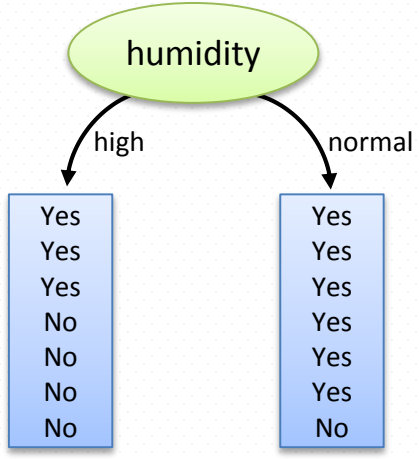


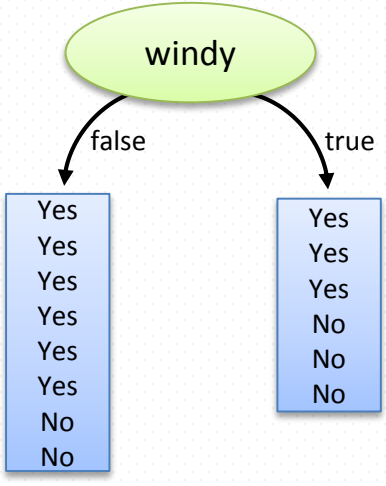
Weather Data

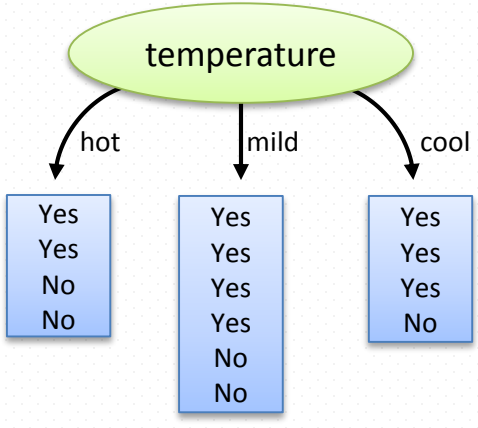
Outlook	Temp	Humidity	Windy	Play?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Which attribute to select?









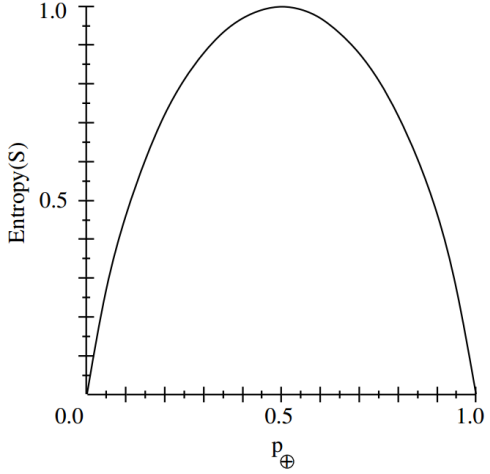
Information Gain

- Information gain (IG) measures how much "information" an attribute gives us about the class.
  - attributes that perfectly partition should give maximal information
  - unrelated attributes should give no information
- It measures the reduction in **entropy**
  - Entropy: (im)purity in an arbitrary collection of examples

Aside on Entropy

- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$\boldsymbol{Entropy(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} log_2 p_{\ominus}}$$



Aside on Entropy

- $Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)
- Why?
  - Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p
  - So, expected number of bits to encode $\oplus$ or $\ominus$ of a random member of S:

$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropy(S) = H(S) = p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

Aside on Entropy

- Minimum number of bits needed for *c* different classes (general case):

$$H(Y) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 \ldots - p_c \log_2 p_c$$

$$H(Y) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

- Properties of entropy
  1. High entropy: uniform distribution
  2. Low entropy: varied distribution (more desirable)

Conditional Entropy

- For an example at random, the conditional entropy of Y (class-label) conditioned on the m feature values taken for a feature x_k is given by:

$$H(Y|x_k) = \sum_{j=1}^{m} P(x_k = v_j) H(Y|x_k = v_j)$$

$$\boldsymbol{InfGain(Y|x_k) = H(Y) - H(Y|x_k)}$$

Example

School	Likes football?
ND	Yes
MSU	No
ND	No
ND	Yes
ND	No
USC	Yes
MSU	No
USC	Yes

Compute $H(Y|X)$

v_j	$\mathbf{P}(x = v_j)$	$H(Y|x)$
ND	0. 5	1
MSU	0.25	0
USC	0.25	0

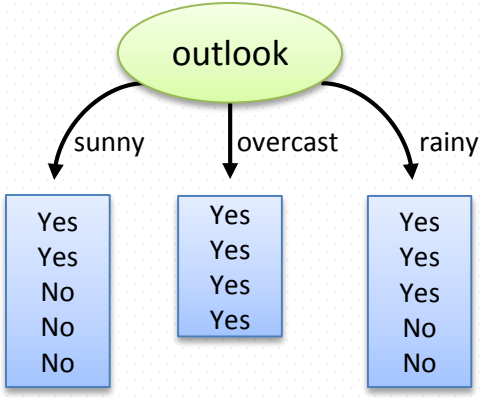
$$H(Y|X) = 0.5 * 1 + 0.25 * 0 + 0.25 * 0$$
$$H(Y|X) = 0.5$$

$$H(Y) - \frac{4}{8} log_2\left(\frac{4}{8}\right) - \frac{4}{8} log_2\left(\frac{4}{8}\right)$$
$$H(Y) = 1$$

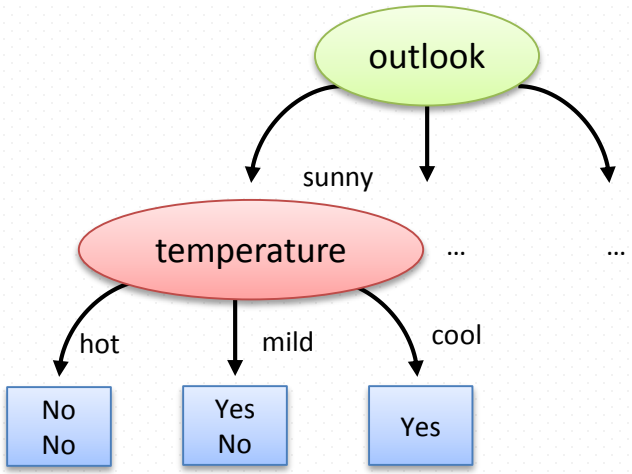
$$InfGain(Y|x_k) = H(Y) - H(Y|x_k)$$
$$InfGain(Y|x_k) = 1 - 0.5 = 0.5$$

Back to the Decision Tree

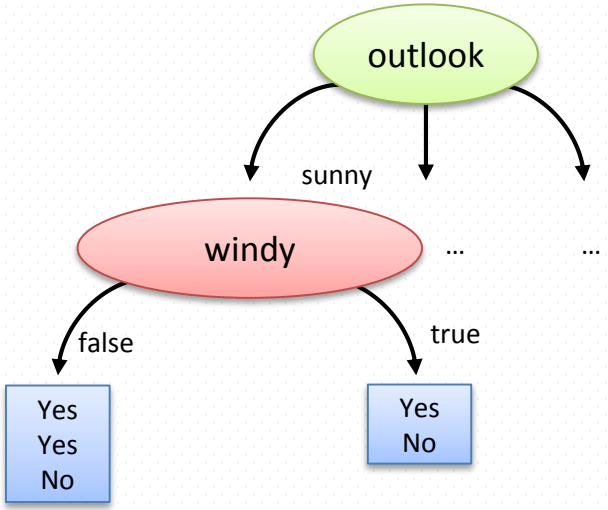
- Information gain
  - (entropy before split) – (entropy after split)
- Information gain for *outlook*:
  $InfGain(\text{outlook}) = IG([9,5]) - IG([2,3],[4,0],[3,2])$
  $InfGain(\text{outlook}) = 0.94 - 0.693 = \mathbf{0.247}$
- For other features:
  $InfGain(\text{“}temperature\text{”}) = \mathbf{0.029}$
  $InfGain(\text{“}humidity\text{”}) = \mathbf{0.152}$
  $InfGain(\text{“}windy\text{”}) = \mathbf{0.048}$



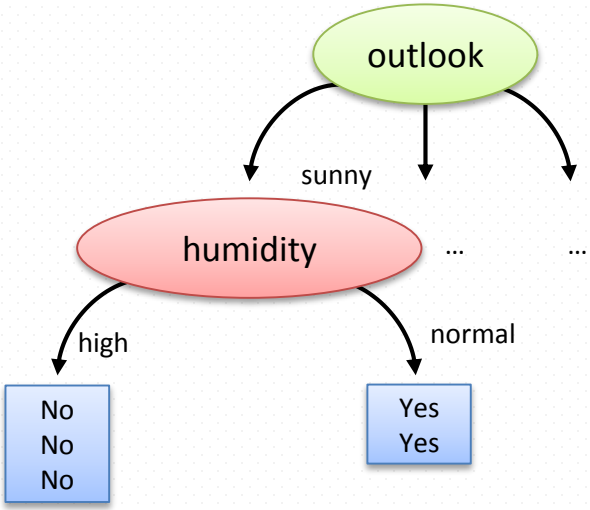
Continuing to split



$$Infogain(\text{“}temperature\text{”}) = 0.571$$

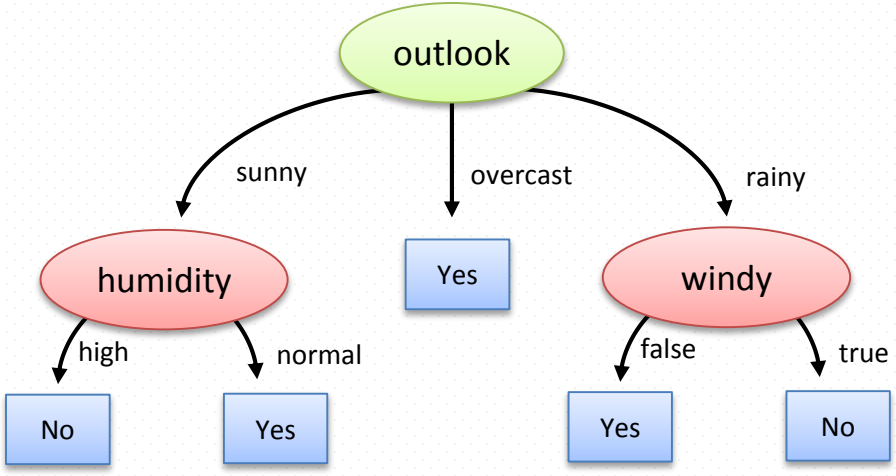


$$Infogain(\text{“}windy\text{”}) = 0.020$$



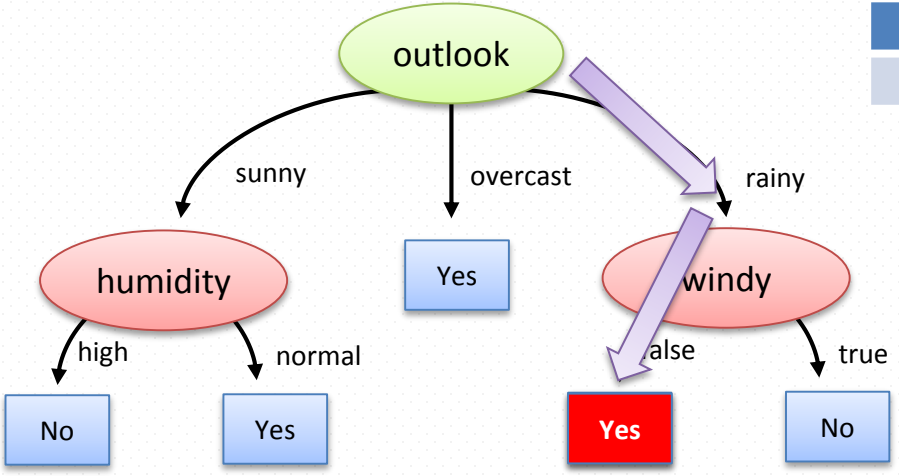
$$Infogain(\text{“}humidity\text{”}) = 0.971$$

Final Tree



- Note: Leaves need not be pure as there can often be similar instances with different classes.

Applying Model to test Data



Test data:

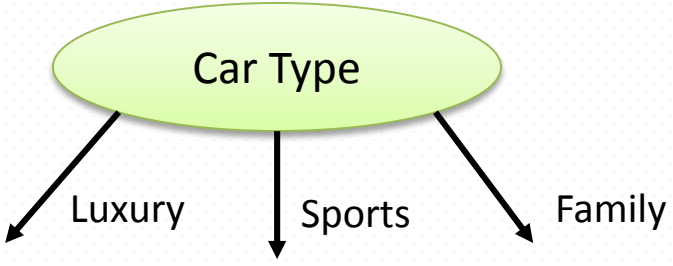
Outlook	Temp	Humidity	Windy	Play?
Rainy	Hot	High	False	?

How to Specify Test Condition

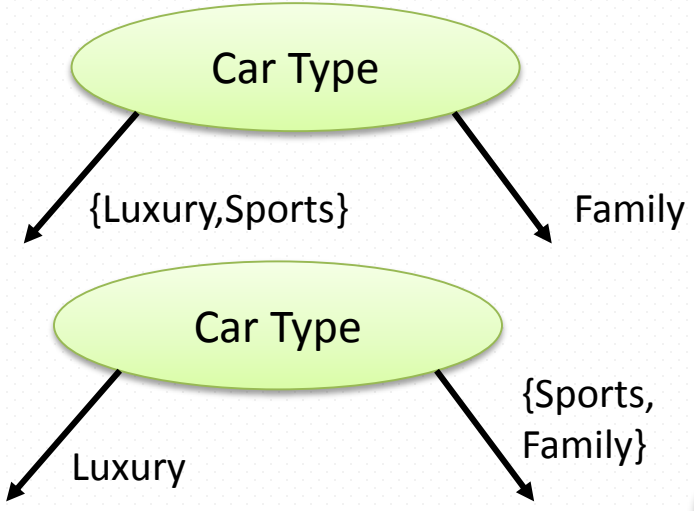
- Depends on:
  - Type of attributes/features:
    - Nominal
    - Continuous
  - Number of ways to split
    - 2-way split
    - Multi-way split

Splitting Based on Nominal Attributes

- Multi-way split:
  - Use as many partitions as there are values



- Binary split:
  - Divide values into two subsets



Splitting Based on Continuous Attributes

- **Discretization:**
  - Form an ordinal categorical feature
  - It can be done at the beginning (static – global), or at each level individually (dynamic – local)
- **Binary Decision:**
  - $(A < v)$ or $(A \geq v)$
  - Considers all splits and chooses the best
  - More computationally intensive

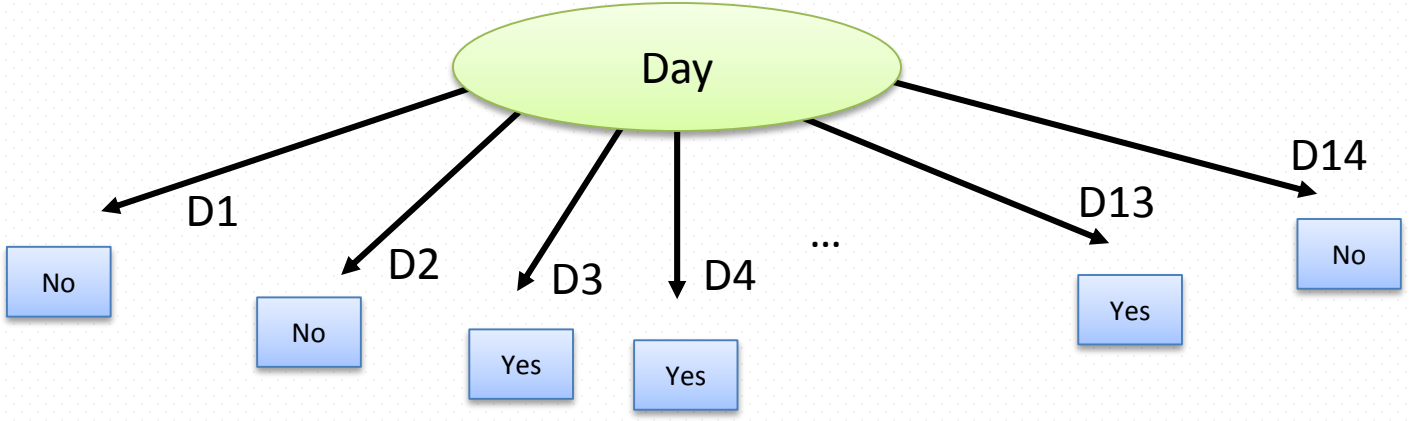
Highly branching attributes

- Problematic: attributes with a large number of values (extreme case: ID code)
- Subsets are more likely to be pure if there is a large number of values
  - Information gain is biased towards choosing attributes with a large number of values
  - This may result in **overfitting** (selection of an attribute that is non-optimal for prediction)

Highly branching attributes – Example

Day	Outlook	Temp	Humidity	Windy	Play?
D1	Sunny	Hot	High	False	No
D2	Sunny	Hot	High	True	No
D3	Overcast	Hot	High	False	Yes
D4	Rainy	Mild	High	False	Yes
D5	Rainy	Cool	Normal	False	Yes
D6	Rainy	Cool	Normal	True	No
D7	Overcast	Cool	Normal	True	Yes
D8	Sunny	Mild	High	False	No
D9	Sunny	Cool	Normal	False	Yes
D10	Rainy	Mild	Normal	False	Yes
D11	Sunny	Mild	Normal	True	Yes
D12	Overcast	Mild	High	True	Yes
D13	Overcast	Hot	Normal	False	Yes
D14	Rainy	Mild	High	True	No

Highly branching attributes – Example



- Entropy of split = 0 → each leaf is "pure"
- Information gain is maximum for this feature
- Is that good?

Information Gain

- Information gain (IG) measures how much "information" a feature gives us about the class.
  - Features that perfectly partition should give maximal information.
  - Unrelated features should give no information.
- It measures the reduction in **entropy**.
  - Entropy: (im)purity in an arbitrary collection of examples.

Criterion of a Split

Suppose we want to split on the first variable (x_1):

x_1	1	2	3	4	5	6	7	8
y	0	0	0	1	1	1	1	1

If we split at $x_1 < 3.5$, we get an optimal split.

If we split at $x_1 < 4.5$, we make a mistake (misclassification).

Idea: *A better split should make the samples "pure" (homogeneous).*

Measures for Selecting the Best Split

Impurity measures include:

$$\text{Entropy} = -\sum_{i=1}^{K} p_k \log_2 p_k$$

$$\text{Gini} = 1 - \sum_{i=1}^{K} {p_k}^2$$

$$\text{Classification error} = 1 - \max_i p_k$$

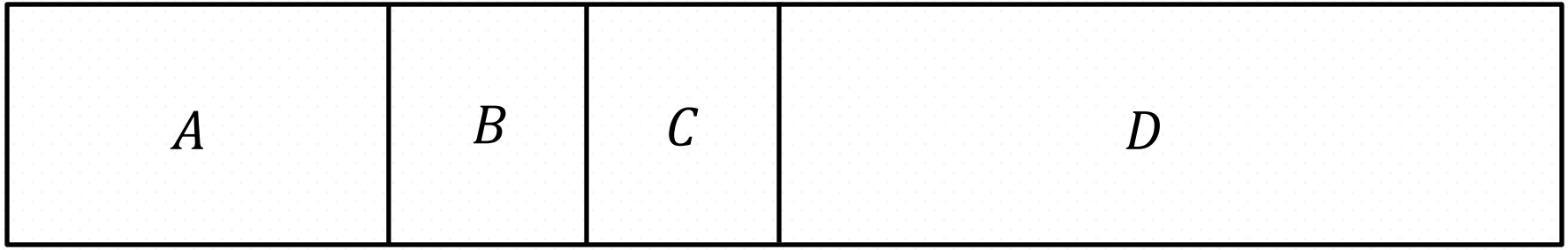
where p_k denotes the proportion of instances belonging to class k $(K = 1, \ldots, k)$, and $0 \log_2 0 = 0$.

What is Entropy?

And how do we compute it?

Measuring Information

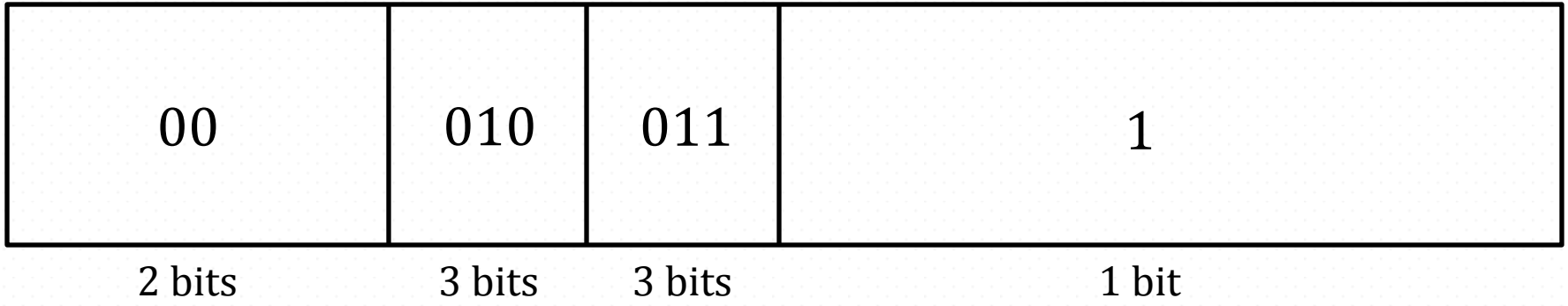
Consider the following probability space with a random variable that takes four values: A, B, C, and D.



If we select a random variable, it gives us information about its horizontal position.

Measuring Information

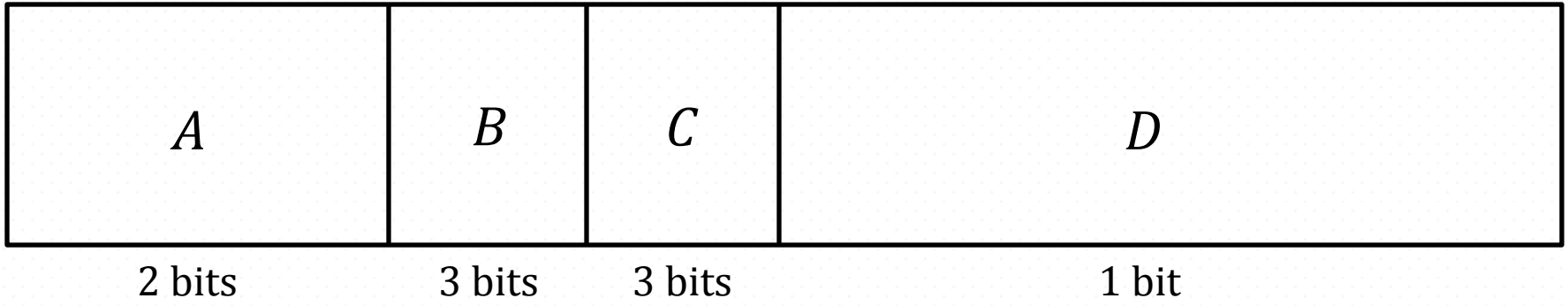
Let's say the horizontal position of the point is represented by a string of zeros and ones.



Larger regions are encoded with fewer bits; smaller regions are encoded with more bits.

Expected Information

The *expected value* is the sum over all values of the product of the probability of the value and the value.



$$\text{Expected Value} = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3$$

Expected Information

Each time a region of color got smaller by one half:

- The number of bits of information we got when it did happen went up by one.
- The change that it did happen went down by a factor of ½.

That is, the information of an event x is the logarithm of one over its probability:

$$Information(x) = \log_2 \left(\frac{1}{P(R = x)} \right)$$

Expected Information

So in general, the expected information or “entropy” of a random variable is the same as the expected value with the Value filled in with the Information:

$$\text{Entropy of } R = \sum_x P(R = x) \cdot Information(x)$$

$$= \sum_x P(R = x) \cdot \log_2 \left(\frac{1}{P(R = x)} \right)$$

$$= -\sum_x P(R = x) \cdot \log_2 P(R = x)$$

Properties of Entropy

Maximized when elements are heterogeneous (impure):

$$\text{If } p_k = \frac{1}{k}, \text{then}$$

$$\text{Entropy} = H = -K \cdot \frac{1}{k} \log_2 \frac{1}{k} = \log_2 K$$

Minimized when elements are homogenous (pure):

$$\text{If } p_i = 1 \text{ or } p_i = 0, \text{then}$$

$$\text{Entropy} = H = 0$$

Information Gain

With entropy defined as:

$$\mathrm{H} = -\sum_{i=1}^{K} p_k \log_2 p_k$$

Then the change in entropy, or *Information Gain*, is defined as:

$$\Delta H = H - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

where m is the total number of instances, with m_k instances belonging to class k, where $K = 1, \ldots, k$.

Information Gain: Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Information Gain: Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

$$\mathrm{H}(Y) = -\sum_{i=1}^{K} p_k \log_2 p_k$$

$$= -\frac{5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14}$$

$$= 0.94$$

Information Gain: Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	**High**	False	***No***
Sunny	Hot	**High**	True	***No***
Overcast	Hot	**High**	False	***Yes***
Rainy	Mild	**High**	False	***Yes***
Rainy	Cool	**Normal**	False	***Yes***
Rainy	Cool	**Normal**	True	***No***
Overcast	Cool	**Normal**	True	***Yes***
Sunny	Mild	**High**	False	***No***
Sunny	Cool	**Normal**	False	***Yes***
Rainy	Mild	**Normal**	False	***Yes***
Sunny	Mild	**Normal**	True	***Yes***
Overcast	Mild	**High**	True	***Yes***
Overcast	Hot	**Normal**	False	***Yes***
Rainy	Mild	**High**	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$= 0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

Information Gain: Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	**Normal**	False	***Yes***
Rainy	Cool	**Normal**	True	***No***
Overcast	Cool	**Normal**	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	**Normal**	False	***Yes***
Rainy	Mild	**Normal**	False	***Yes***
Sunny	Mild	**Normal**	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	**Normal**	False	***Yes***
Rainy	Mild	High	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

$$H_L = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}$$

Information Gain: Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	**High**	False	***No***
Sunny	Hot	**High**	True	***No***
Overcast	Hot	**High**	False	***Yes***
Rainy	Mild	**High**	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	**High**	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	**High**	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	**High**	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

$$H_L = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.592$$

$$H_R = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$

Information Gain: Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} 0.592 - \frac{7}{14} 0.985$$

$$= 0.94 - 0.296 - 0.4925$$

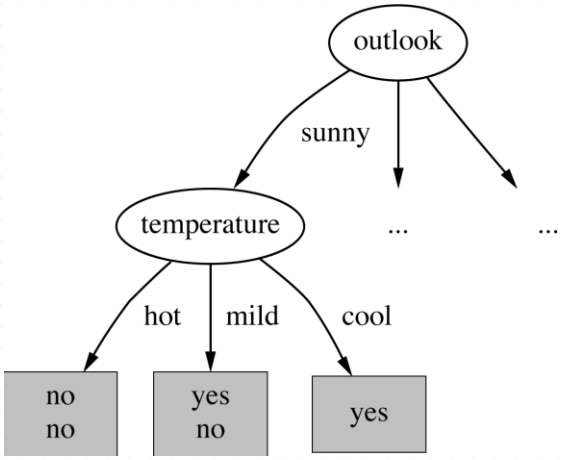
$$= 0.1515$$

Information Gain: Example

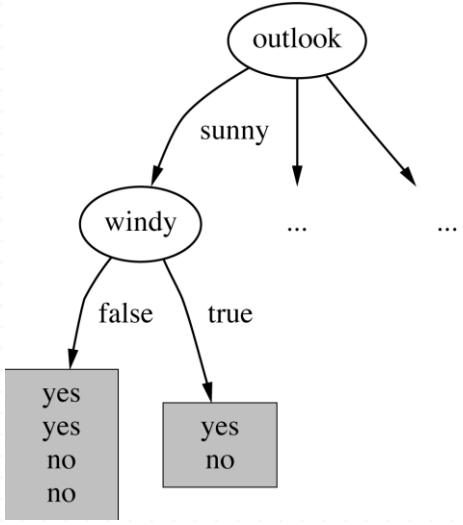
- Information gain for each feature:
  - Outlook = 0.247
  - Temperature = 0.029
  - Humidity = 0.152
  - Windy = 0.048
- Initial split is on outlook, because it is the feature with the highest information gain.

Information Gain: Example

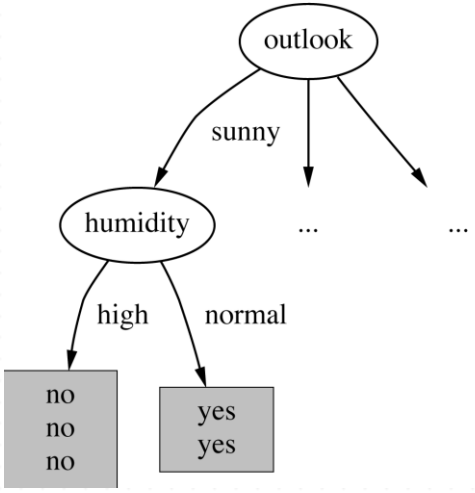
- Now we search for the best split at the next level:



Temperature = 0.571



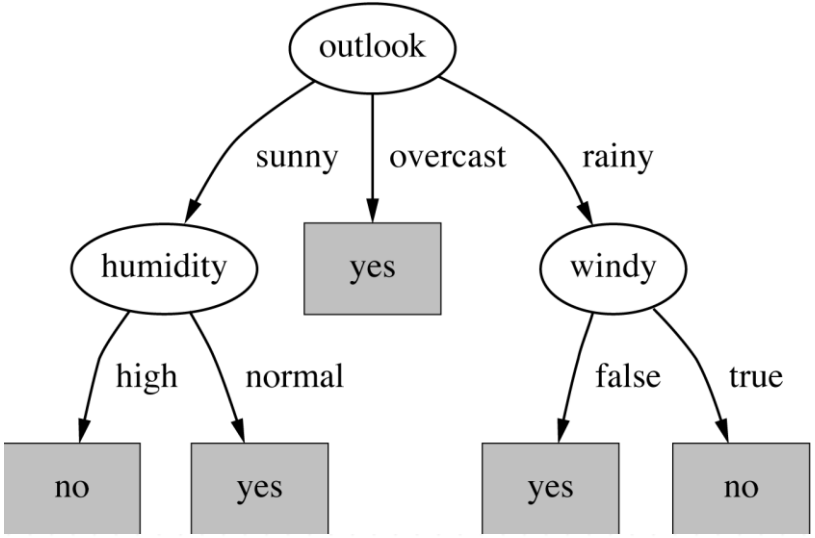
Windy = 0.020



Humidity = 0.971

Information Gain: Example

- The final decision tree:



Note that not all leaves need to be pure; sometimes similar (even identical) instances have different classes. Splitting stops when data cannot be split any further.

Gini Index

The Gini index is defined as:

$$\text{Gini} = 1 - \sum_{i=1}^{K} {p_k}^2$$

where p_k denotes the proportion of instances belonging to class k $(K = 1, \ldots, k)$.

Gini Index Properties

Maximized when elements are heterogeneous (impure):

$$\text{If } p_k = \frac{1}{k}, \text{ then}$$

$$\text{Gini} = 1 - \sum_{k=1}^{K} \frac{1}{k^2} = 1 - \frac{1}{k}$$

Minimized when elements are homogenous (pure):

$$\text{If } p_i = 1 \text{ or } p_i = 0, \text{ then}$$

$$\text{Gini} = 1 - 1 - 0 = 0$$

Gini Index Example

Suppose we want to split on the first variable (x_1):

x_1	1	2	3	4	5	6	7	8
y	0	0	0	1	1	1	1	1

$$Gini = 1 - \left({}^{3}/_{8}\right)^2 - \left({}^{5}/_{8}\right)^2 = {}^{15}/_{32}$$

If we split at $x_1 < 3.5$: $\Delta Gini = \frac{15}{32} - \frac{3}{8} \cdot 0 - \frac{5}{8} \cdot 0 = {}^{15}/_{32}$

If we split at $x_1 < 4.5$: $\Delta Gini = \frac{15}{32} - \frac{4}{8} \cdot \frac{3}{8} - \frac{4}{8} \cdot 0 = {}^{9}/_{32}$

Classification Error

The classification error is defined as:

$$\text{Classification error} = 1 - \max_i p_k$$

where p_k denotes the proportion of instances belonging to class k $(K = 1, \ldots, k)$.

Classification Error Properties

Tends to create impure nodes:

x_1	1	2	3	4	5	6	7	8
y	0	0	0	1	1	0	0	0

a b

Splitting at b has lower classification error than a, but results in both nodes being impure.

Splitting Based on Nominal Features

- **Multi-way split:** Use as many partitions as distinct values.

- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.

Splitting Based on Continuous Features

- **Discretization**: Form an ordinal categorical feature
  - Static: discretize once at the beginning (global)
  - Dynamic: discretize ranges at different levels (local)

- **Binary decision:** Consider all possible splits and finds the best cut.

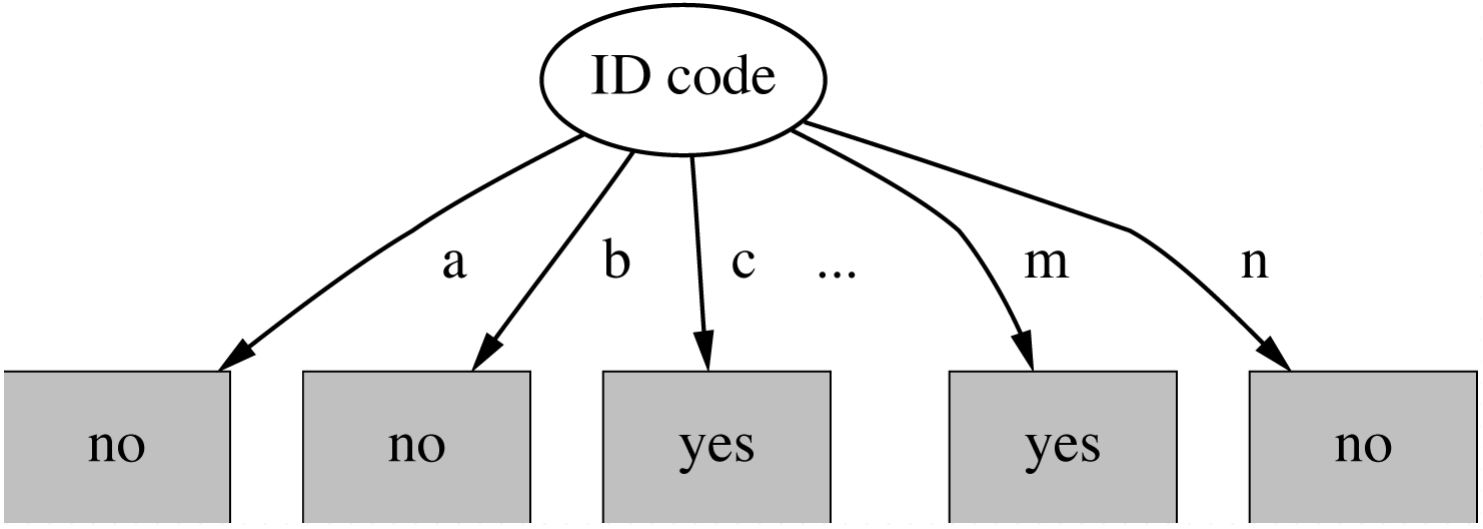
Highly Branching Features

- Features with a large number of values can be problematic
  - e.g.: ID code
- Subsets are more likely to be pure if there are a large number of values
  - Information gain is biased toward choosing features with a large number of values
  - The selection of a feature that is non-optimal for predication can result in *overfitting*.

Dataset with Highly Branching Features

ID	Outlook	Temperature	Humidity	Windy	Play
A	Sunny	Hot	High	False	***No***
B	Sunny	Hot	High	True	***No***
C	Overcast	Hot	High	False	***Yes***
D	Rainy	Mild	High	False	***Yes***
E	Rainy	Cool	Normal	False	***Yes***
F	Rainy	Cool	Normal	True	***No***
G	Overcast	Cool	Normal	True	***Yes***
H	Sunny	Mild	High	False	***No***
I	Sunny	Cool	Normal	False	***Yes***
J	Rainy	Mild	Normal	False	***Yes***
K	Sunny	Mild	Normal	True	***Yes***
L	Overcast	Mild	High	True	***Yes***
M	Overcast	Hot	Normal	False	***Yes***
N	Rainy	Mild	High	True	***No***

IG with Highly Branching Features



The entropy of the split is 0, since each leaf node is “pure”, having only one case.

Gain Ratio

- A modification of information gain that reduces its bias on highly branching features.
- It takes into account the number and size of branches when choosing a feature.
- It does this by normalizing information gain by the "*intrinsic information*" of a split, which is defined as the information need to determine the branch to which an instance belongs.

Intrinsic Information

- The intrinsic information represents the potential information generated by splitting the dataset into v partitions:

$$IntrinsicInfo(D) = -\sum_{j=1}^{v} \frac{|D_j|}{D} \cdot \log_2\left(\frac{|D_j|}{D}\right)$$

- High intrinsic info: partitions have more or less the same size
- Low intrinsic info: few partitions hold most of the tuples.

Gain Ratio Defined

- The gain ratio is defined as:

$$GainRatio(F) = \frac{Gain(F)}{IntinsicInfo(F)}$$

- The feature with the maximum gain ratio is selected as the splitting feature.

Comparing Feature Selection Measures

- Information Gain
  - Biased toward multivalued features.
- Gain Ratio
  - Tends to prefer unbalanced splits in which one partition is much smaller than the other.
- Gini Index
  - Has difficulties when the number of classes is large.
  - Favors tests that result in equal-sized partitions with purity.
- Classification Error
  - No. Just, no.

Overfitting in Decision Trees

- If a decision tree is fully grown, it may lose some generalization capability.
- This is a phenomenon known as *overfitting*.

Definition of Overfitting

Consider the error of hypothesis h. We let error on the training data be $\text{error}_{train}(h)$ and error over the entire distribution D of data be $\text{error}_D(h)$.

Then a hypothesis h "overfits" the training data if there is an alternative hypothesis, h', such that:

$$\text{error}_{train}(h) < \text{error}_{train}(h')$$
$$\text{error}_D(h) < \text{error}_D(h')$$

Model Overfitting

Errors committed by classification models are generally divided into two types:

1. **Training Errors**
   The number of misclassification errors committed on training records; also called resubstitution error.

2. **Generalization Errors**
   The expected error of the model on previously unseen records.

Causes of Overfitting

1. **Overfitting Due to Presence of Noise**
   Mislabeled instances may contradict the class labels of other similar records.

2. **Overfitting Due to Lack of Representative Instances**
   Lack of representative instances in the training data can prevent refinement of the learning algorithm.

3. **Overfitting and the Multiple Comparison Procedure**
   Failure to compensate for algorithms that explore a large number of alternatives can result in spurious fitting.

Overfitting Due to Noise: An Example

An example training set for classifying mammals. Asterisks denote mislabelings.

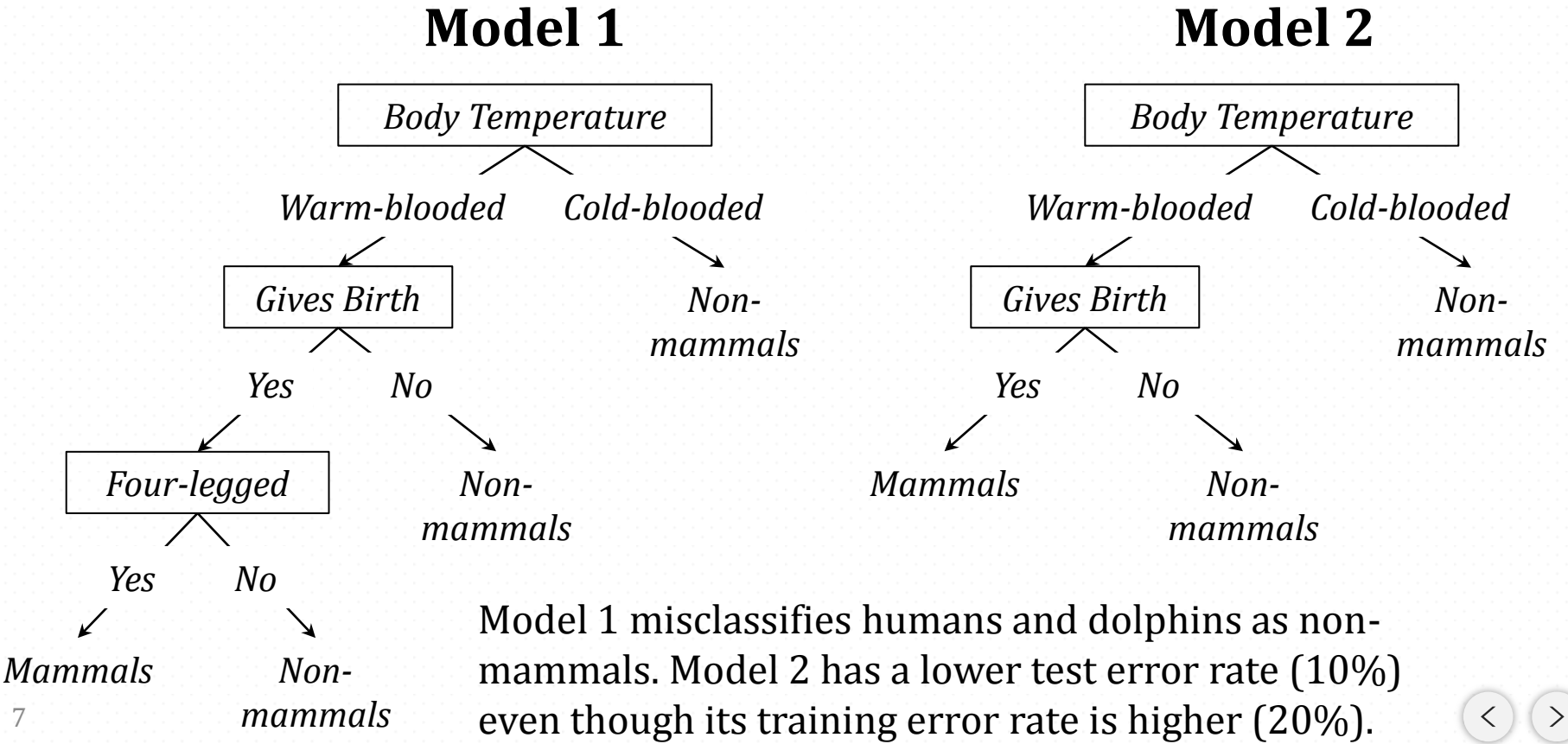
Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Porcupine	Warm-blooded	Yes	Yes	Yes	***Yes***
Cat	Warm-blooded	Yes	Yes	No	***Yes***
Bat	Warm-blooded	Yes	No	Yes	***No****
Whale	Warm-blooded	Yes	No	No	***No****
Salamander	Cold-blooded	No	Yes	Yes	***No***
Komodo dragon	Cold-blooded	No	Yes	No	***No***
Python	Cold-blooded	No	No	Yes	***No***
Salmon	Cold-blooded	No	No	No	***No***
Eagle	Warm-blooded	No	No	No	***No***
Guppy	Cold-blooded	Yes	No	No	***No***

Overfitting Due to Noise

An example testing set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Human	Warm-blooded	Yes	No	No	***Yes***
Pigeon	Warm-blooded	No	No	No	***No***
Elephant	Warm-blooded	Yes	Yes	No	***Yes***
Leopard shark	Cold-blooded	Yes	No	No	***No***
Turtle	Cold-blooded	No	Yes	No	***No***
Penguin	Cold-blooded	No	No	No	***No***
Eel	Cold-blooded	No	No	No	***No***
Dolphin	Warm-blooded	Yes	No	No	***Yes***
Spiny anteater	Warm-blooded	No	Yes	Yes	***Yes***
Gila monster	Cold-blooded	No	Yes	Yes	***No***

Overfitting Due to Noise



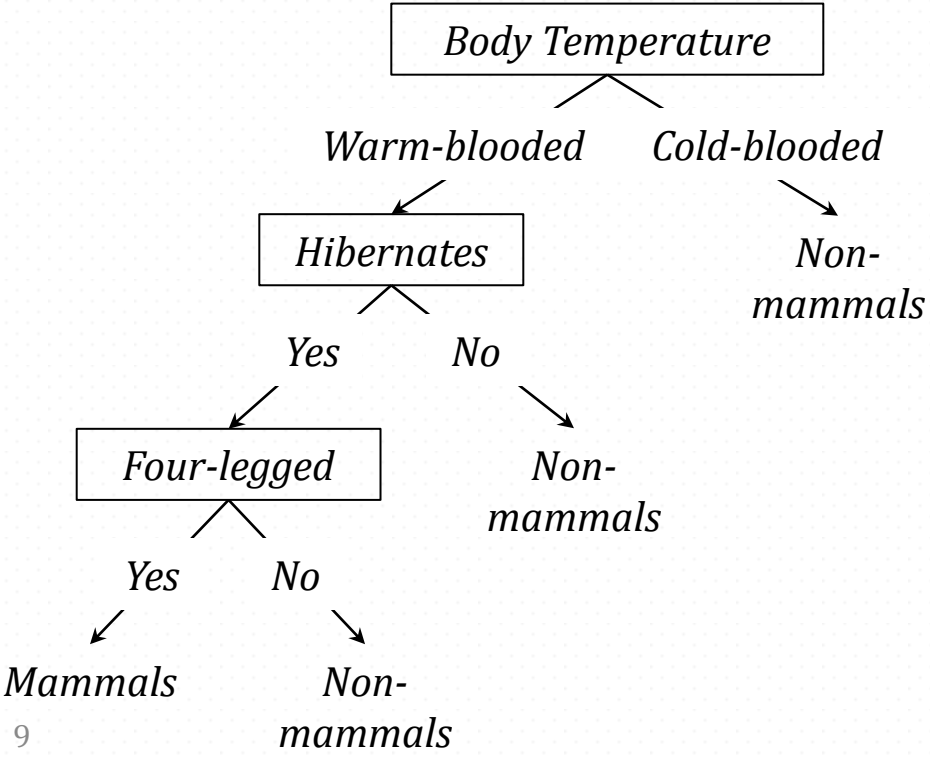
Model 1 misclassifies humans and dolphins as non-mammals. Model 2 has a lower test error rate (10%) even though its training error rate is higher (20%).

Overfitting Due to Lack of Samples

An example training set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Salamander	Cold-blooded	No	Yes	Yes	***No***
Guppy	Cold-blooded	Yes	No	No	***No***
Eagle	Warm-blooded	No	No	No	***No***
Poorwill	Warm-blooded	No	No	Yes	***No***
Platypus	Warm-blooded	No	Yes	Yes	***Yes***

Overfitting Due to Lack of Samples



Although the model's training error is zero, its error rate on the test set if 30%.

Humans, elephants, and dolphins are misclassified because the decision tree classifies all warm-blooded vertebrates that do not hibernate as non-mammals. The tree arrives at this classification decision because there is only one training records, which is an eagle, with such characteristics.

Model Overfitting

A good model must not only fit the training data well but also accurately classify records it has never seen.

In other words, a good model must have *low training error* **and** *low generalization error*.

Model Overfitting

A good model must not only fit the training data well but also accurately classify records it has never seen.

In other words, a good model must have *low training error* **and** *low generalization error*.

Occam's Razor

"Everything should be made as simple as possible, but no simpler."

All other things being equal, simple theories are preferable to complex ones.

Occam's Razor

But *why* prefer a short hypothesis?

1. There are fewer short hypotheses than long hypotheses.
2. A short hypothesis that fits the data is unlikely to be a coincidence.
3. A long hypothesis that fits the data might be a coincidence.

Minimum Description Length Principle

- A formalization of Occam's razor.
- The Idea:

The best hypothesis for a given set of data is the one that leads to the best compression of the data.

How do we measure "compression"?

MDL: Inuitive Explanation

Occam's razor: prefer the shortest hypothesis.

MDL: prefer the hypothesis h that minimizes the space required to describe a theory plus the space required to describe the theory's mistakes.

MDL: Formal Explanation

Occam's razor: prefer the shortest hypothesis.

MDL: prefer the hypothesis h that minimizes

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} L_{C_1}(h) + L_{C_2}(D|h)$$

where L_{C_x} is the description length of x under encoding C.

MDL Example

Let H be a set of decision trees (hypotheses) and D be a set of training data labels. Then,

$L_{C_1}(h)$ is the number of bits to describe tree h.

$L_{C_2}(D|h)$ is the number of bits to describe D given h.

- Note that $L_{C_2}(D|h) = 0$ if all training instances are classified perfectly by h. It need only describe exceptions.

Hence h_{MDL} trades-off tree size for training errors.

MDL for Classification Models

- The hypothesis is the classification model and the description length is the combined description of the model and its errors on the training data.
- Using the MDL principle, we seek a classifier with *shortest* description.
- Used this way, the MDL principle is a *model selection criterion*—a way to select between potential models or hypotheses.

Model Selection Criteria

Model selection criteria attempt to find a good compromise between:

a) The model's complexity

b) The model's prediction accuracy on unseen data

- Reasoning: a good model is a simple model that achieves high accuracy on the given data
- Also known as Occam's Razor: the best theory is the smallest one that describes all the facts

Elegance vs. Errors

Consider the following two theories of some data:

Theory 1: very simple, elegant theory that explains the data almost perfectly

Theory 2: significantly more complex theory that reproduces the data without mistakes

Theory 1 is probably preferable.

Elegance vs. Errors Example

Canonical example: Kepler's laws of planetary motion.

- Actually *less* accurate the Copernicus's latest refinement of the Ptolemaic theory of epicycles.
- But far *simpler*.

"I have cleared the Augean stables of astronomy of cycles and spirals, and left behind me a single cartload of dung."
–Johannes Kepler

From Theory to Practice

Let's look at how to turn these ideas of model selection criteria into practice.

Avoiding Overfitting in Decision Trees

- Stop growing the tree when the data split is not statistically significant
- Grow the full tree, then prune
  - Do we really needs all the “small” leaves with perfect coverage?

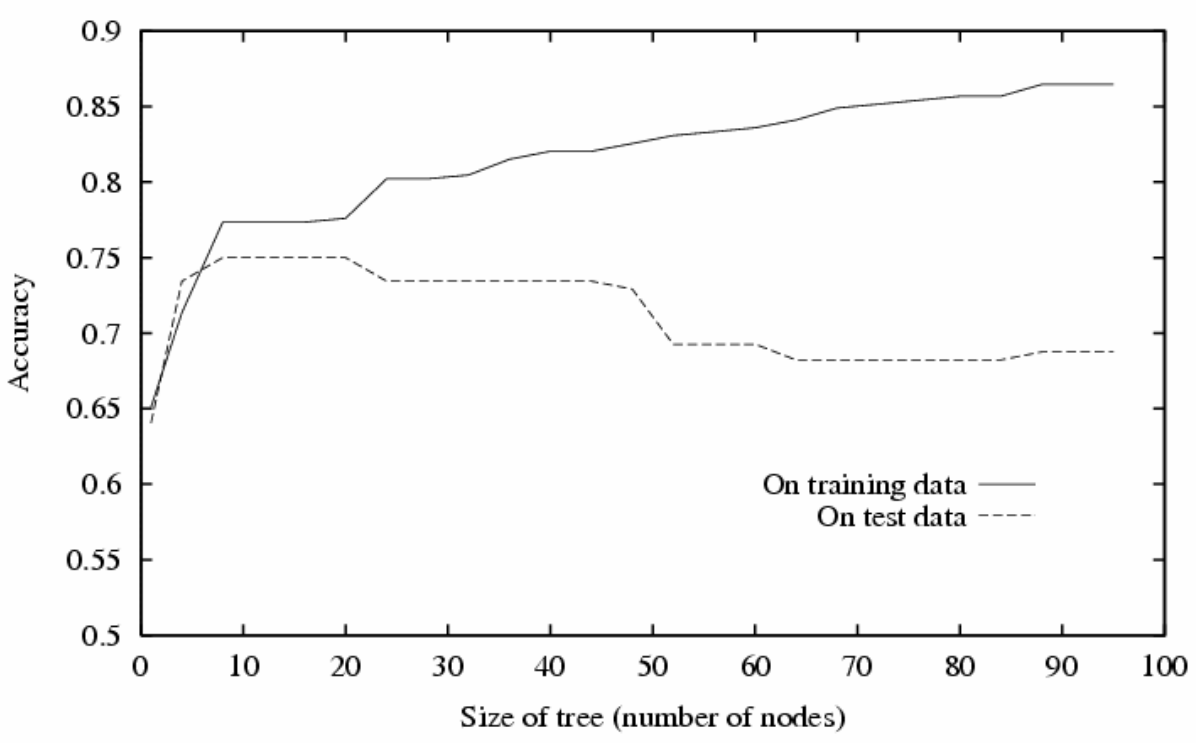
Avoiding Overfitting in Decision Trees

- How to select
  - Measure performance over training data (and include some estimates for generalization)
  - Measure performance over separate validation data
  - Use Minimum Description Length Principle (MDL)
    - Minimize, $size(tree) + size\big(misclassiciation(tree)\big)$

Decision Tree Pruning Methodologies

- Pre-pruning (top-down)
  - Stopping criteria while growing the tree
- Post-pruning (bottom-up)
  - Grow the tree, then prune
  - More popular

Decision Tree Overfitting



Decision Tree Pre-Pruning

- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node
  - Stop if all instances belong to the same class
  - Stop if all the feature values are the same

Decision Tree Pre-Pruning

- More restrictive conditions
  - Stop if the number of instances is less than some use-specified threshold
  - Stop if the class distribution of instances are independent of the available features
    - Stop if expanding the current node does not improve impurity.

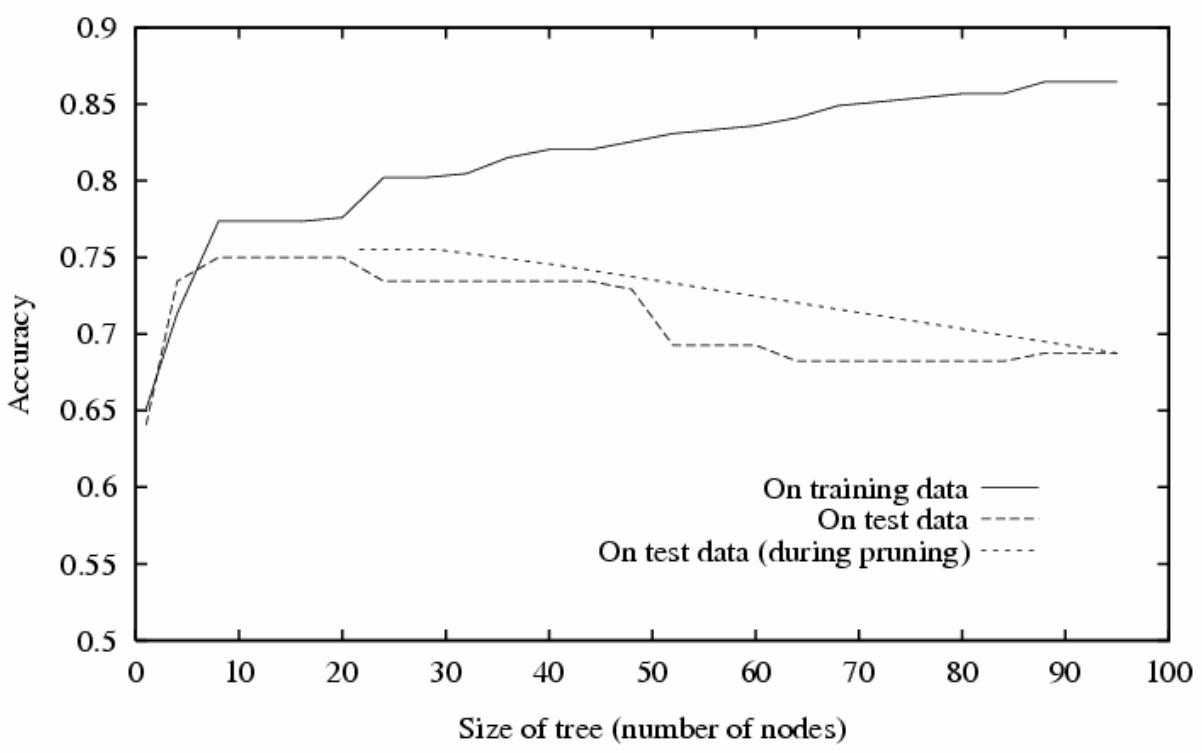
Decision Tree Post-Pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming, replace sub-tree by a leaf node
  - Class label of leaf node is determined from majority class of instances in the sub-tree
- Can use MDL for post-prunning

Decision Tree Post-Pruning

- Reduced Error Pruning
  - Split data into training and validation set
  - Remove one node at a time and evaluate the performance on the validation data
  - Remove the one that decreases the error
  - Usually produces the smallest version of a tree
  - But always requires a validation set

Decision Tree Pruning



Decision Trees: Pros and Cons

- Pros:
  - Fast in implementation
  - Works with all types of features
  - Insensitive to outliers
  - Few tuning parameters
  - Efficient in handling missing values
  - Interpretable model representation

Decision Trees: Pros and Cons

- Cons:
  - Not effective at approximating linear or smooth functions or boundaries
  - Trees always include high-order interactions
  - Large variance
    - Each split is conditional on all of its ancestor splits.

Evaluation & Credibility Issues

- What measure should we use?
  - Classification accuracy might not be enough.
- How reliable are the predicted results?
- How much should we believe in what was learned?
  - Error on the training data is not a good indicator of performance on future data.
  - The classifier was computed from the very same training data, any estimate based on that data will be optimistic.

Evaluation Questions

- How to evaluate the performance of a model?
- How to obtain reliable estimates of performance?
- How to compare the relative performance among competing models?
- Given two equally performing models, which one should we prefer?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model.
- Confusion matrix:

		Predicted Class	
		+	-
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	-	f_{-+} (FP)	f_{--} (TN)

Accuracy

		Predicted Class	
		+	-
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	-	f_{-+} (FP)	f_{--} (TN)

The most widely-used metric is accuracy:

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Misleading Accuracy

- Consider a two-class problem:
  - Number of class 0 instances = 9990
  - Number of class 1 instances = 10
- Suppose a model predicts everything to be class 0.
  - It's accuracy is 9990/10000=99.9%.
  - It's accuracy is misleading, because the model does not predict any class 1 instance.

Cost Matrix

		Predicted Class	
		+	-
Actual Class	+	$C(+\|+)$	$C(-\|+)$
	-	$C(+\|-)$	$C(-\|-)$

$C(i|j)$ is the cost of misclassifying a class j instance as class i

Computing the Cost of Classification

		Predicted Class	
		+	-
Actual Class	+	−1	100
	-	1	0

		Predicted Class	
		+	-
Actual Class	+	150	40
	-	60	250

$Accuracy = 80\%$
$Cost = 3910$

		Predicted Class	
		+	-
Actual Class	+	250	45
	-	5	200

$Accuracy = 90\%$
$Cost = 4255$

Accuracy

		Predicted Class	
		+	-
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	-	f_{-+} (FP)	f_{--} (TN)

True positive (TP) or f_{++}: positive instances correctly predicted.
False negative (FN) or f_{+-}: positive instances wrongly predicted.
False positive (FP) or f_{-+}: negative instances wrongly predicted.
True negative(TN) or f_{--}: negative instances correctly predicted.

Confusion Matrix

		Predicted Class	
		+	-
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	-	f_{-+} (FP)	f_{--} (TN)

True positive rate (TPR): fraction of positive instances correctly predicted.
False positive rate (FPR): fraction of positive instances wrongly predicted.
False negative rate (FNR): fraction of negative instances wrongly predicted.
True negative rate (TNR): fraction of negative instances correctly predicted.

Confusion Matrix

		Predicted Class	
		+	-
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	-	f_{-+} (FP)	f_{--} (TN)

True positive rate (TPR): $TPR = TP/(TP + FN)$.
False positive rate (FPR): $FPR = FP/(TN + FP)$.
False negative rate (FNR): $FNR = FN/(TP + FN)$.
True negative rate (TNR): $TNR = TN/(TN + FP)$.

Cost-Sensitive Measures

$$\text{Precision}(p) = \frac{TP}{TP + FP}$$

$$\text{Recall}(r) = \frac{TP}{TP + FN}$$

As precision ↑, false positives (TN) ↓.

As recall ↑, false negatives (FN) ↓.

F_1 Measure

$$F_1 \text{ measure } = \frac{2rp}{r+p} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

$$F_1 \text{ measure} = \frac{2}{\frac{1}{r} + \frac{1}{p}}$$

As F_1 measure ↑, false positives (FP) and false negatives (FN) ↓.

F_β Measure

$$F_\beta \text{ measure} = \frac{(\beta^2 + 1)rp}{r + \beta^2 p}$$

Both precision and recall are special cases of F_β where $\beta = 0$ and $\beta = \infty$, respectively. Low values of β make F_β closer to precision; high values make it closer to recall.

Precision and Recall

$$\text{Precision, } p = \frac{TP}{TP+FP}$$

$$\text{Recall, } r = \frac{TP}{TP+FN}$$

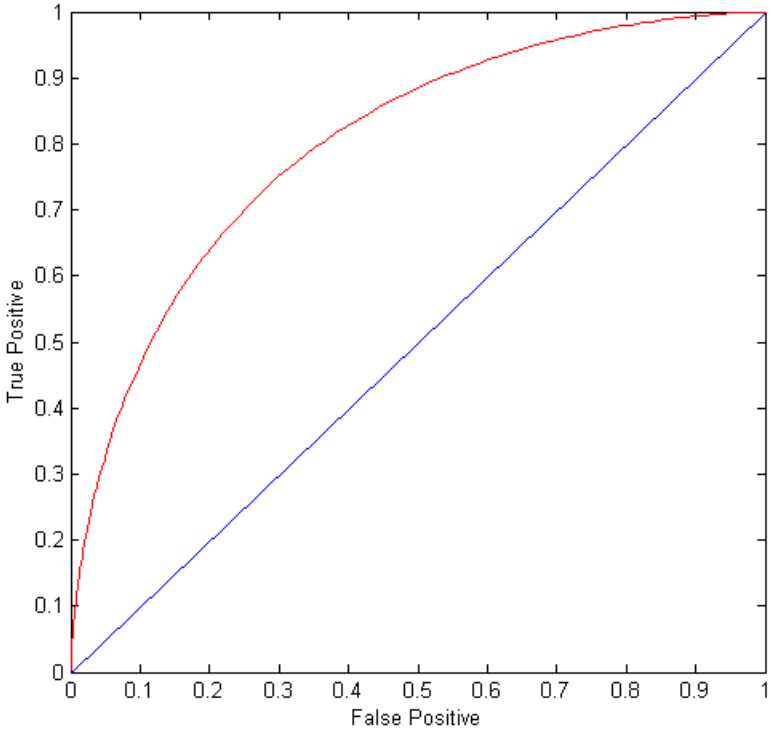
$$F_1\text{measure} = \frac{2rp}{r+p} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

$$F_1\text{measure} = \frac{2}{\frac{1}{r} + \frac{1}{p}}$$

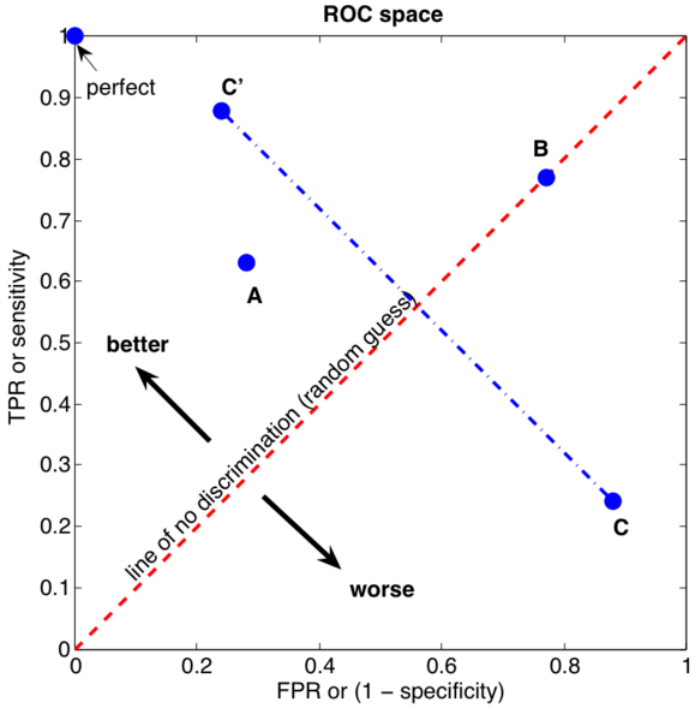
Receiver Operating Characteristic (ROC)

- Developed in the 1950s for signal detection theory to analyze noisy signals.
- ROC curve plots TP (on the y-axis) against FP (on the x-axis).
  - Performance of each classifier represented as a point on the ROC curve.
  - Changing the threshold of algorithm, sample distribution, or cost matrix changes the location of the point.

ROC Curve



ROC Curve



A

TP=63	FP=28	91
FN=37	TN=72	109
100	100	200

TPR = 0.63
FPR = 0.28
ACC = 0.68

B

TP=77	FP=77	154
FN=23	TN=23	46
100	100	200

TPR = 0.77
FPR = 0.77
ACC = 0.50

C

TP=24	FP=88	112
FN=76	TN=12	88
100	100	200

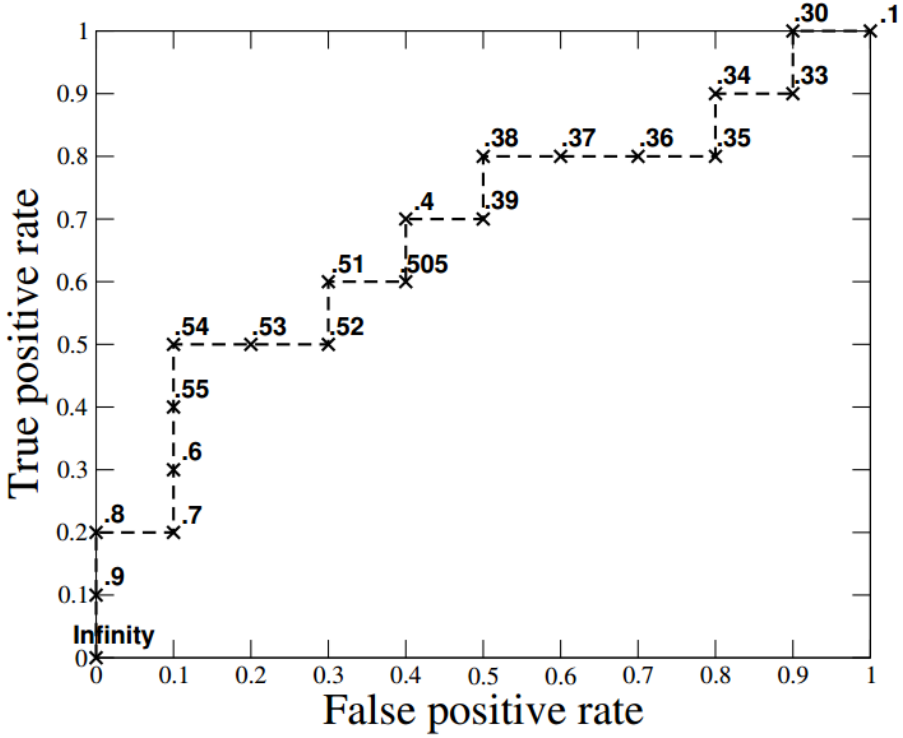
TPR = 0.24
FPR = 0.88
ACC = 0.18

C'

TP=88	FP=24	112
FN=12	TN=76	88
100	100	200

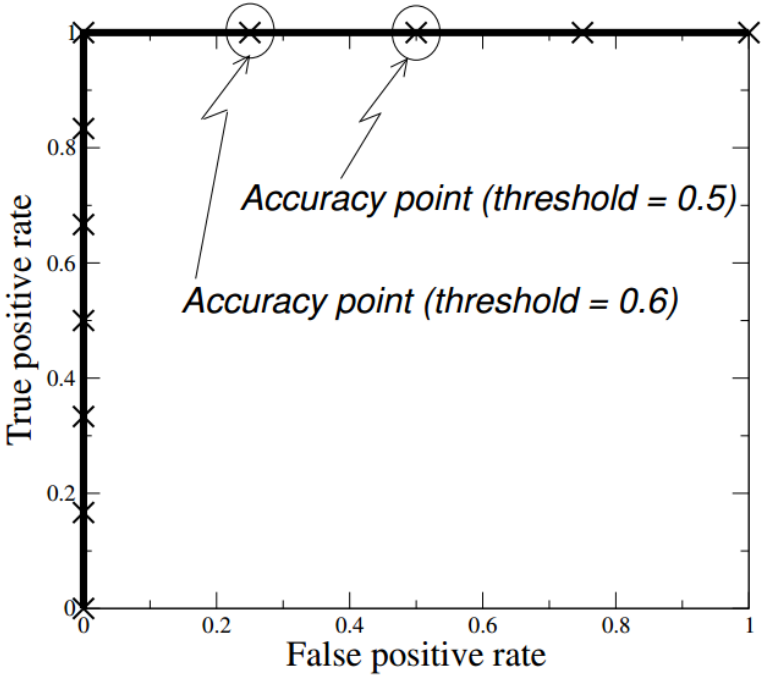
TPR = 0.88
FPR = 0.24
ACC = 0.82

Generating ROC Curves



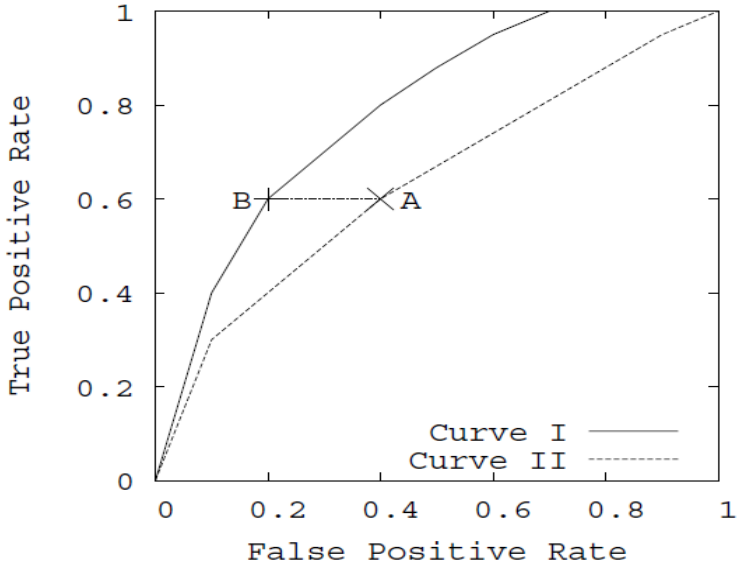
Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Generating ROC Curves

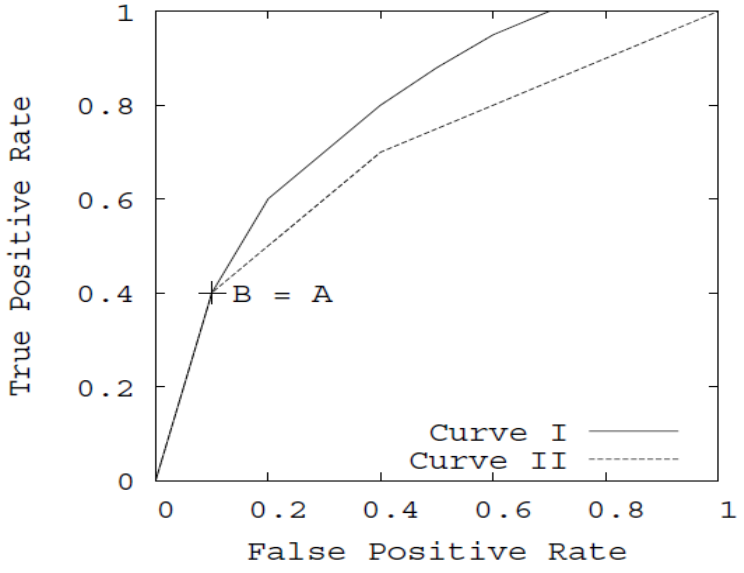


Inst no.	Class True	Class Hyp	Score
1	p	Y	0.99999
2	p	Y	0.99999
3	p	Y	0.99993
4	p	Y	0.99986
5	p	Y	0.99964
6	p	Y	0.99955
7	n	Y	0.68139
8	n	Y	0.50961
9	n	N	0.48880
10	n	N	0.44951

Dominating Classifiers in ROC Space

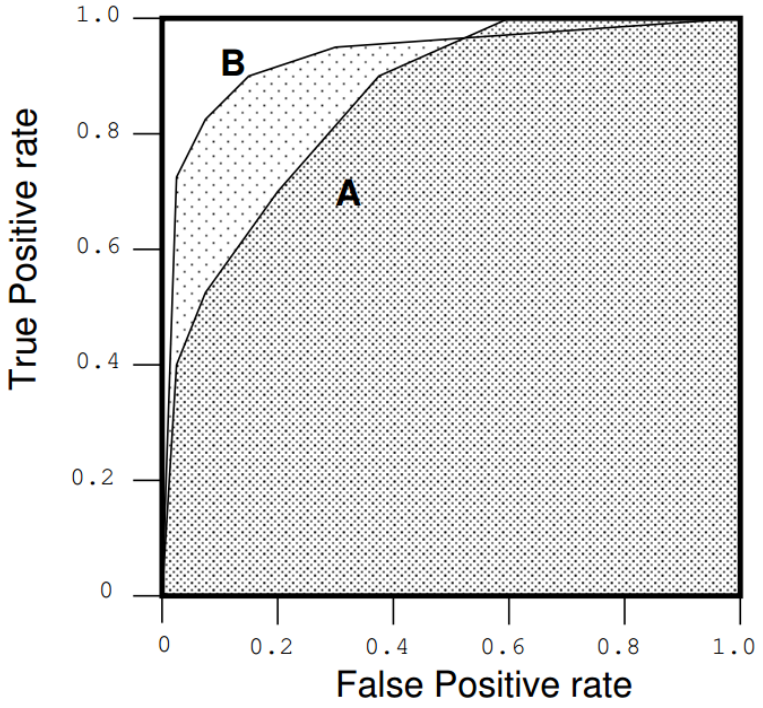


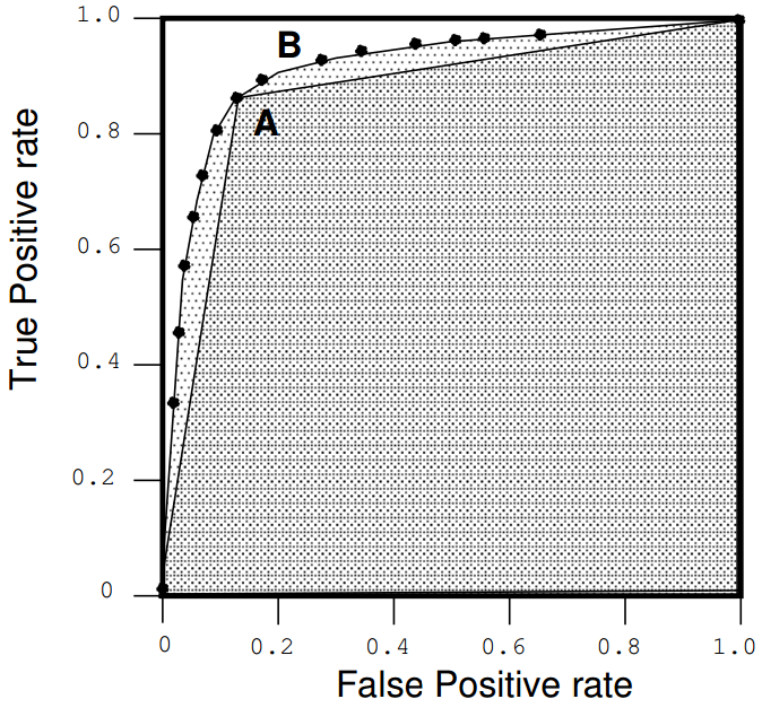
(a) Case 1: $FPR(A) > FPR(B)$



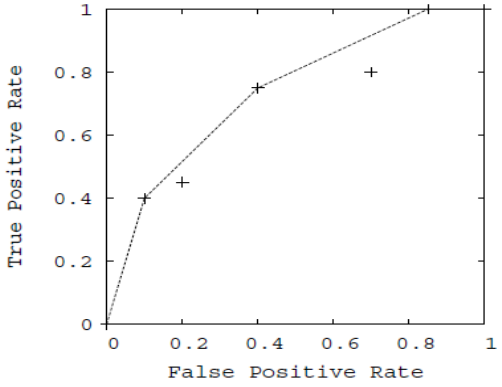
(b) Case 2: $FPR(A) = FPR(B)$

Area Under the ROC Curve

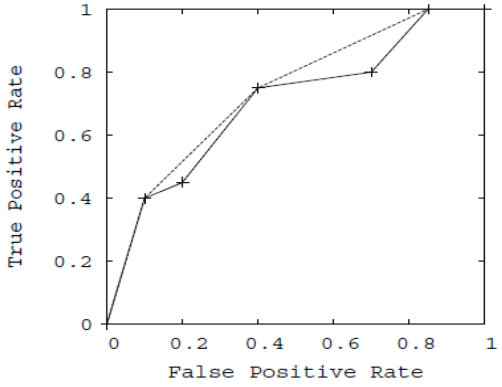




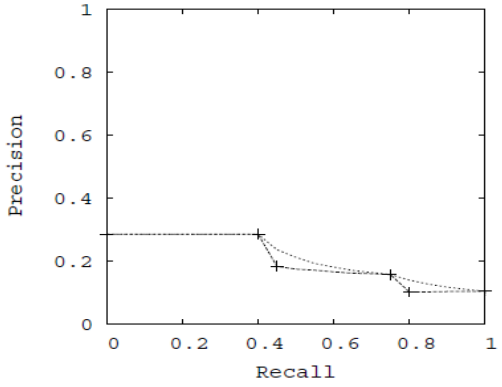
Precision-Recall Curves



(a) Convex hull in ROC space

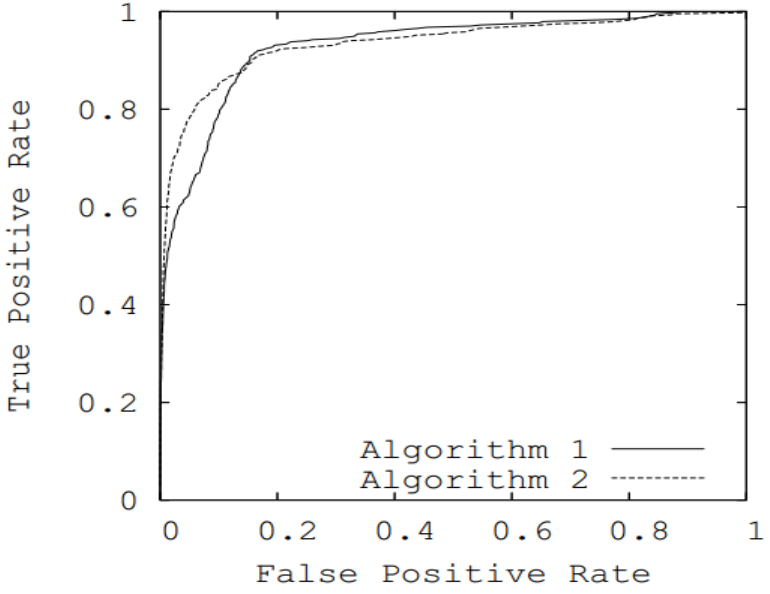


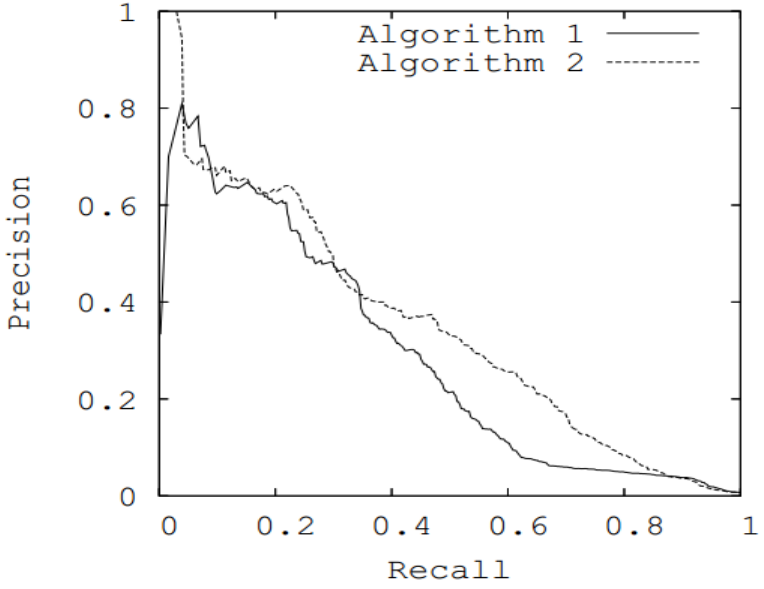
(b) Curves in ROC space



(c) Equivalent curves in PR space

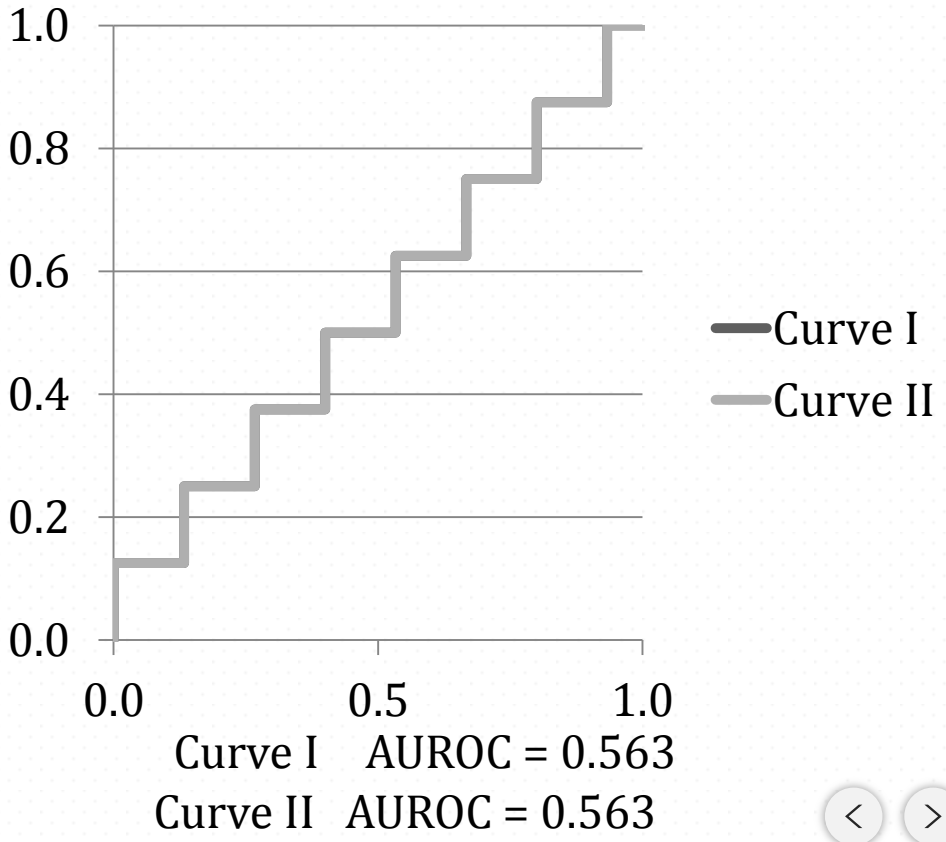
ROC and PR Curves





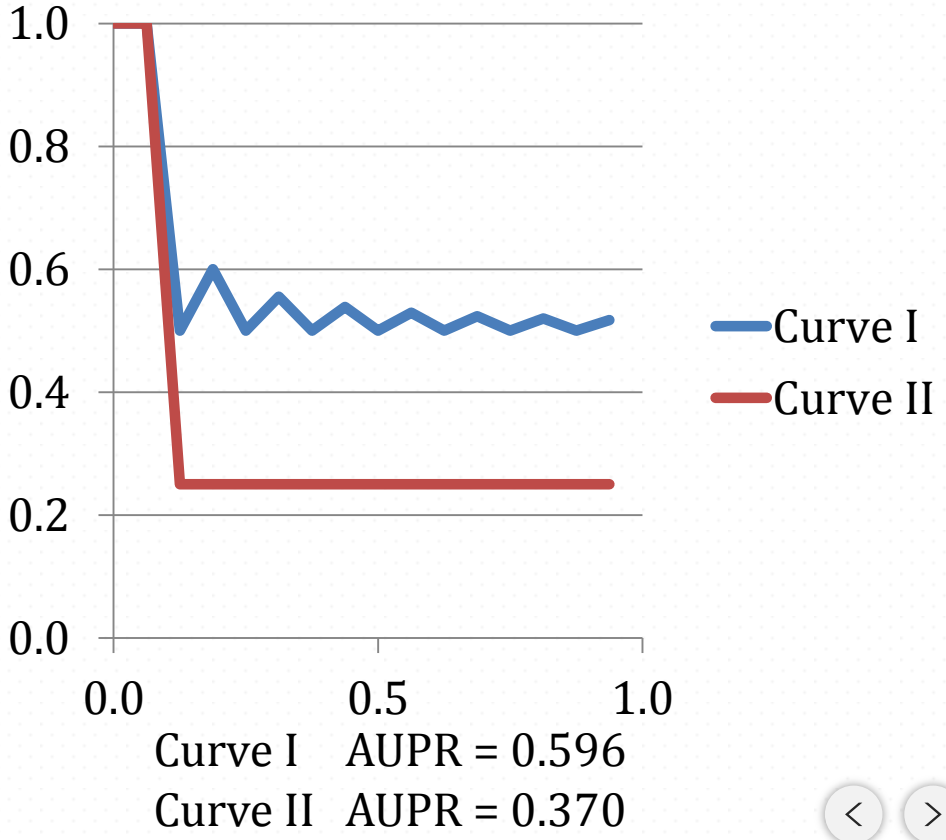
ROC is Skew Insensitive

Prediction	Class I	Class II
$\mu_0 = max$	1	1
$\mu_1 = \mu_0 - \epsilon_1$	1	0
$\mu_2 = \mu_1 - \epsilon_2$	0	0
$\mu_3 = \mu_2 - \epsilon_3$	0	0
$\mu_4 = \mu_3 - \epsilon_4$	1	1
$\mu_5 = \mu_4 - \epsilon_5$	1	0
$\mu_6 = \mu_5 - \epsilon_6$	0	0
$\mu_7 = \mu_6 - \epsilon_7$	0	0
.	.	.
.	.	.
$\mu_n = \mu_{n-1} - \epsilon_n$	.	.

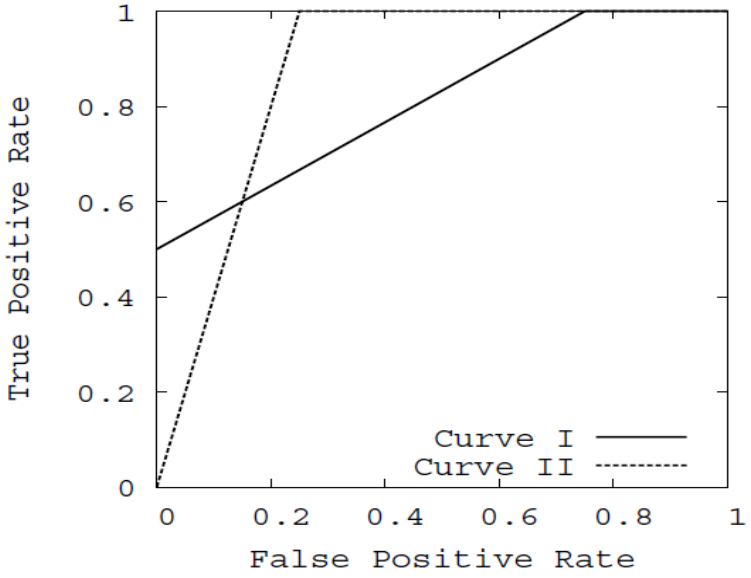


Precision-Recall is Skew Sensitive

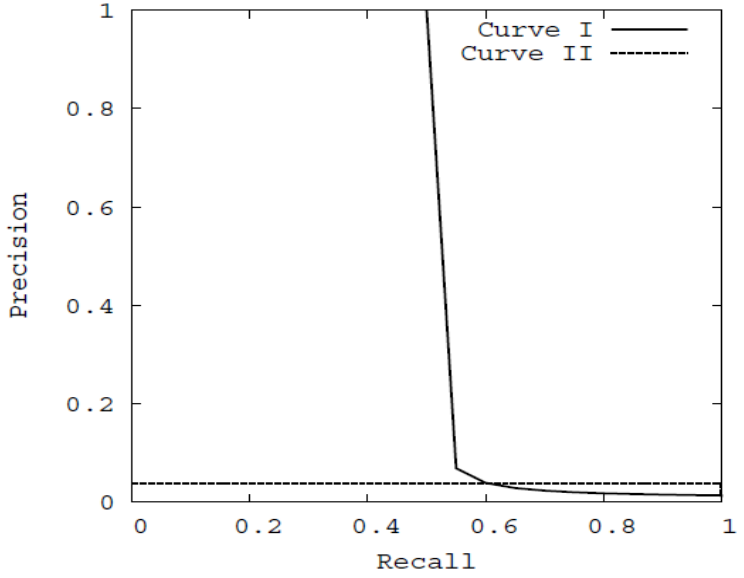
Prediction	Class I	Class II
$\mu_0 = max$	1	1
$\mu_1 = \mu_0 - \epsilon_1$	1	0
$\mu_2 = \mu_1 - \epsilon_2$	0	0
$\mu_3 = \mu_2 - \epsilon_3$	0	0
$\mu_4 = \mu_3 - \epsilon_4$	1	1
$\mu_5 = \mu_4 - \epsilon_5$	1	0
$\mu_6 = \mu_5 - \epsilon_6$	0	0
$\mu_7 = \mu_6 - \epsilon_7$	0	0
.	.	.
.	.	.
$\mu_n = \mu_{n-1} - \epsilon_n$	.	.



Optimizing the AUROC vs. AUPR



(a) Comparing AUC-ROC for two algorithms



(b) Comparing AUC-PR for two algorithms

Curve I

AUROC: 0.813

AUPR: **0.514**

Curve II

AUROC: **0.875**

AUPR: 0.038

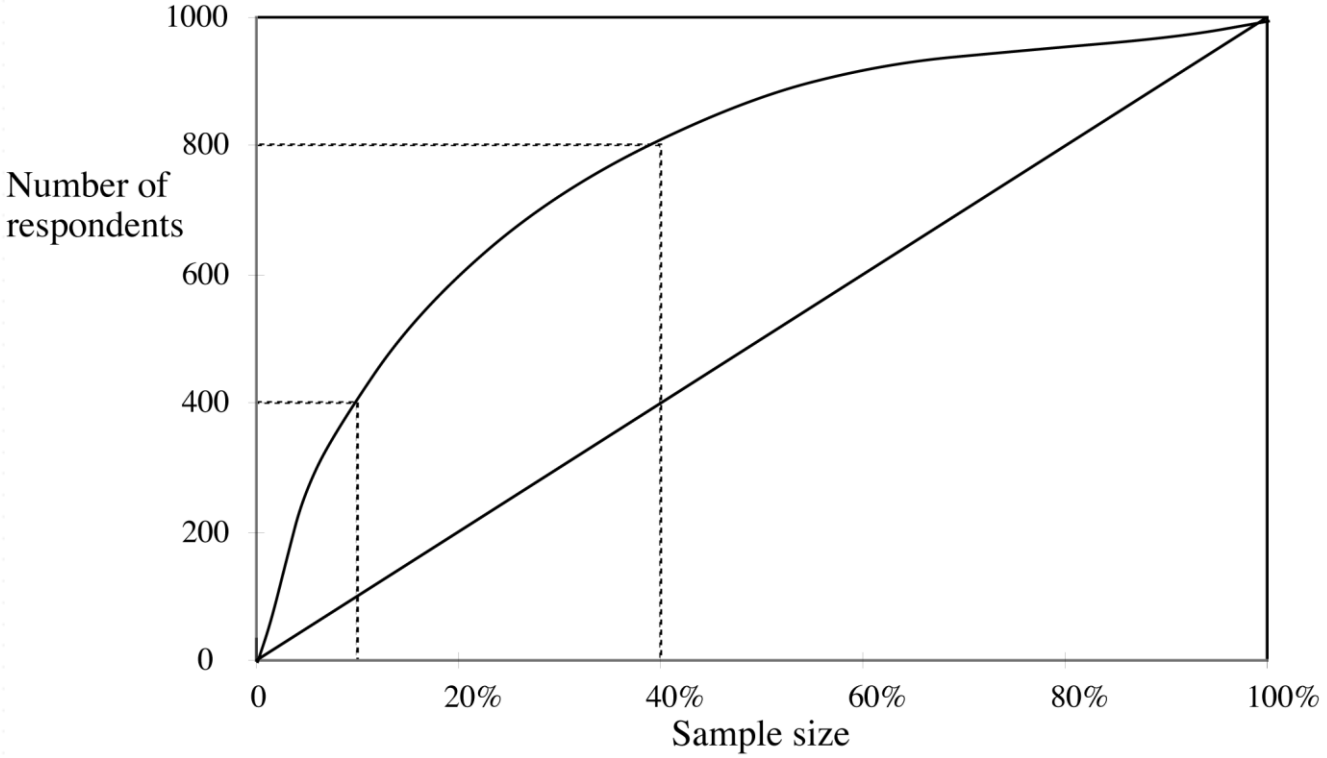
Lift Charts

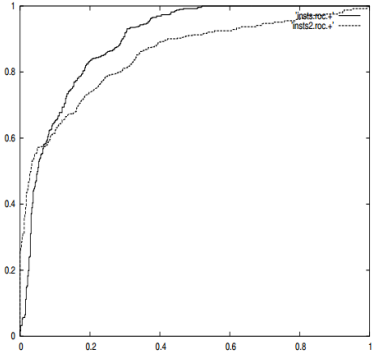
- *Lift* is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model.
- The greater the area between the lift curve and the baseline, the better the model.

Example: Direct Marketing

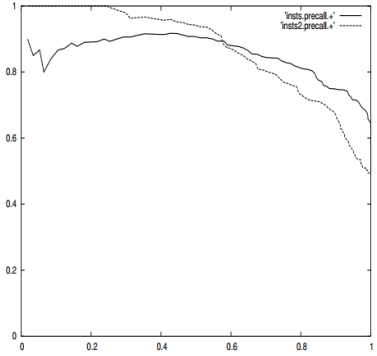
- Mass mailout of promotional offers (1,000,000).
- The proportion who normally respond is 0.1% (1,000).
- A data mining tool can identify a subset of a 100,000 for which the response rate is 0.4% (400).
- In marketing terminology, the increase of response rate is known as the *lift factor* yielded by the model.
- The same data mining tool may be able to identify 400,000 households for which the response rate is 0.2% (800).
- The overall goal is to find subsets of test instances that have a high proportion of true positives.

Example: Direct Marketing

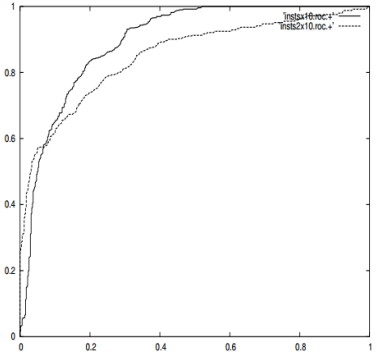


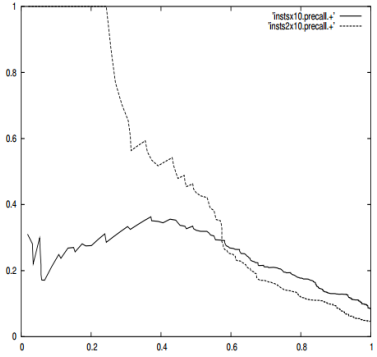


(a) ROC curves, 1:1



(b) Precision-recall curves, 1:1





How do we obtain reliable estimates of performance measures?

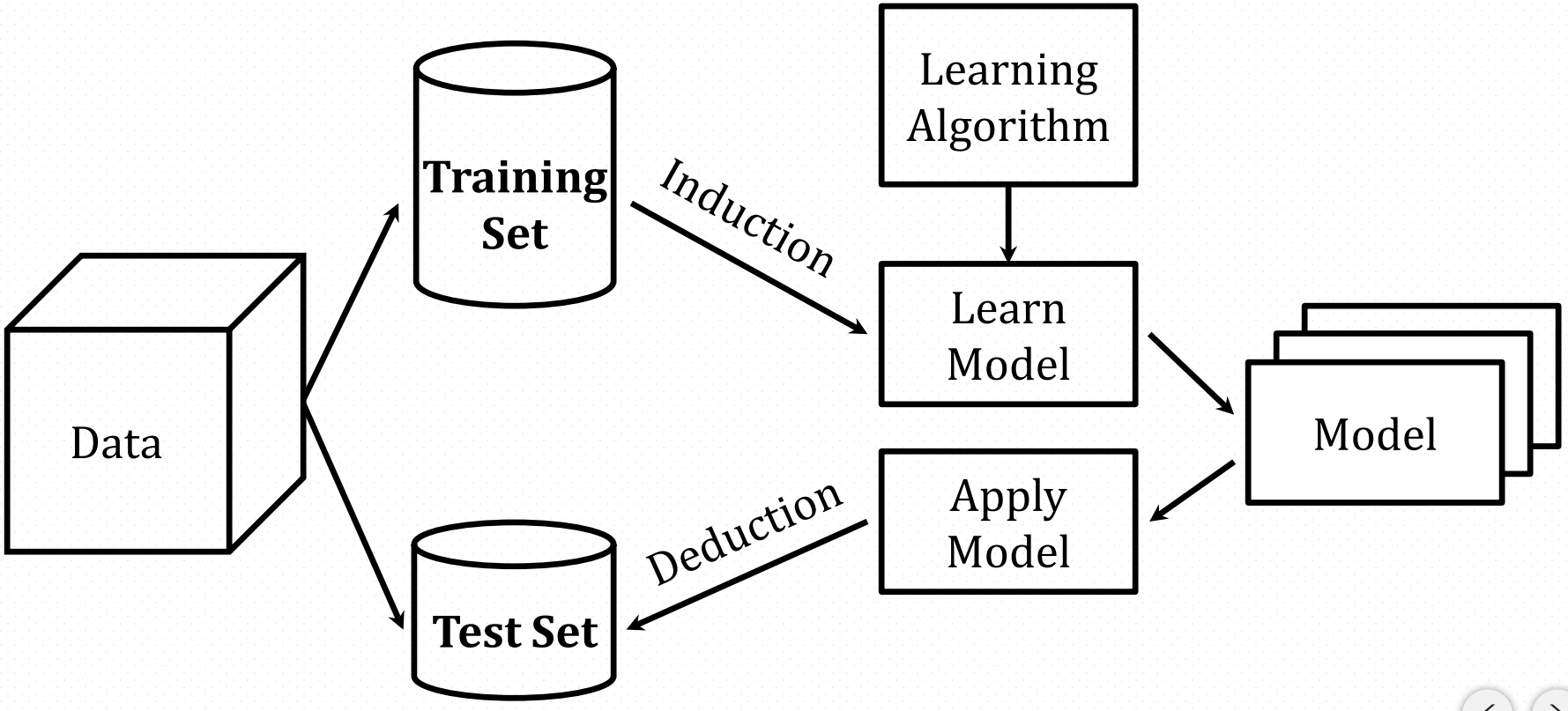
Estimating Model Performance

- How do we estimate performance measures?

- Error on training data?
  - Also called resubstitution error.
  - Not a good indicator of the performance on future data.

- Simple solution
  - Spit the available data into training and testing sets.

Training and Testing Sets



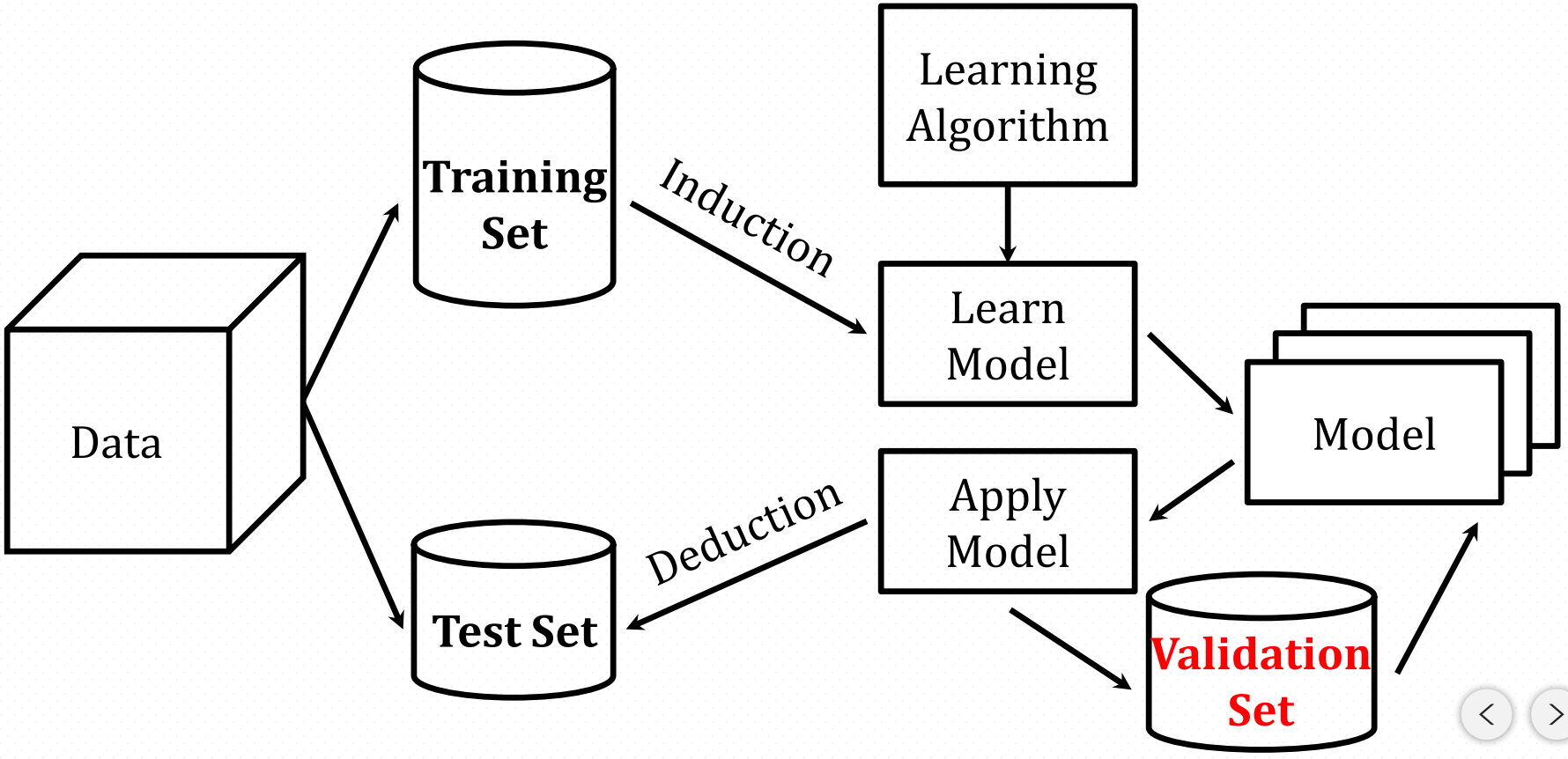
Avoiding Data Snooping

- It is important that the test data is not used in any way to create the classifier.
- Some learning schemes operating in two stages
  - Stage 1: builds the basic structure
  - Stage 2: optimizes parameter settings
- The test data cannot be used for parameter tuning.
- Proper procedure uses three sets: training data, validation data, and test data.

Validation Data

- A validation dataset is a subset of the data used to tune parameters.
- Typically used when an appropriate model needs to be chosen from several rivaling approaches.

Validation Data



Methods of Estimating Performance

- Holdout
  - Reserve ½ for training and ½ for testing.
  - Reserve 2/3 for training and 1/3 for testing.
- Random subsampling
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on $k-1$ partitions, test on the remaining one
  - Leave-one-out: $k = n$

Methods of Estimating Performance

- Holdout
  - Single holdout
  - Repeated holdout
- Cross validation
  - k-fold validation
  - Leave-one-out validation
- Stratified sampling
- Bootstrap

Holdout Estimation

Key Idea:

Reserve a certain amount of data for testing and use the remainder for training.

Problems:

- For small or “unbalanced” datasets, instances might not be representative.
- The data used for training and testing may vary significantly.

Stratified Holdout

Generate holdout using *stratified sampling*.

- Generates new subsets of instances with an approximately equal proportions of classes.

Ensures that the classes are equally represented in the samples.

Repeated Holdout

- Repeated holdout, or "random subsampling," improves the reliability of the holdout estimate by repeating the process with different subsamples.
  - In each iteration, a certain proportion of data is randomly selected for training.
  - The error rates on different iterations are averaged to yield an overall error rate.
- Problem: overlapping test sets.

Cross-Validation

- Cross-validation ensures non-overlapping test sets.
- In k-fold cross-validation:
  - The data is split into k stratified subsets of equal size.
  - Each of the k subsets is used for testing and the combination of the rest for training.
- The error estimates are averaged across each of the k folds.

Example of Cross-Validation

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

1. Divide a dataset into k folds.

Example of Cross-Validation

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

1. Divide a dataset into k folds.
2. Use one subset for testing and the remainder for training.

Example of Cross-Validation

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

1. Divide a dataset into k folds.
2. Use one subset for testing and the remainder for training.
3. Iterate.

Example of Cross-Validation

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

1. Divide a dataset into k folds.
2. Use one subset for testing and the remainder for training.
3. Iterate.
4. Average the error rates over all k folds.

Properties of Cross-Validation

- Cross-validation uses sampling without replacement.
  - The same instance, once selected, cannot be selected again for a particular training/testing set.
- Computationally expensive.
- Variance tends to be high.

The Bootstrap

- The bootstrap uses sampling with replacement to form the training set.
  - Sample a dataset of n instances n times with replacement to form a new dataset of n instances.
  - Use this data as the training set.
  - Use the instances from the original dataset that don't occur in the new training set for testing.

The Bootstrap

- An instance has a probability of $1 - 1/n$ of not being picked for training.
- Thus, its probability of ending up in the test data is:

$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

- This means the training data will contain approximately 63.2% of the instances.

Estimating Error using the Bootstrap

- The error estimate on the test data will be very pessimistic, since training was on just ~63% of the instances.
- Therefore, combine it with the resubstitution error:

$$err = 0.632 \times e_{\text{test instances}} + 0.368 \times e_{\text{training instances}}$$

- The resubstitution error gets less weight than the error on the test data.

Properties of the Bootstrap

- For small sample size n, bootstrap will have much smaller variability than the cross-validation estimate.
- Bootstrap and CV estimates will generally be close for large sample sizes.
  - Their ratio will approach unity as the sample size approaches infinity.

How do we compare the relative performance among competing models?

Comparing Data Mining Methods

- Frequent problem: we want to know which of the two learning techniques is better
  - How to reliably say Model A is better or worse than Model B?
- We can:
  - Compare on different test sets
  - Compare 10-fold CV estimates
- Both require significance testing.

Significance Tests

- Significance tests tell us how (statistically) confident we can be that there is truly a difference.
- For example:
  - Null hypothesis: there is no "real" difference
  - Alternative hypothesis: there is a difference
- A significance test measures how much evidence there is in favor of rejecting the null hypothesis

Methods for Comparing Classifiers

- Two models:
  - Model M1: accuracy = 85%, tested on 30 instances
  - Model M2: accuracy = 75%, tested on 5,000 instances
- Can we say M1 is better than M2?
- How much confidence can have in the accuracy of both models?
- Can the difference in performance measure be explained as a result of random fluctuations in the test set?

Confidence Intervals

- We can say: error lies within a certain specified interval within a certain specified confidence
- Example: $S = 750$ successes in $n = 1000$ test examples
- Estimated error rate: 25%
- How close is this to the true error rate?
- With 95% confidence [22.32,27.68]

Confidence Interval for Accuracy

- Prediction can be regarded as a Bernoulli trial with two possible outcomes, correct or incorrect.
- A collection of Bernoulli trials has a Binomial distribution.
- Given the number of correct test predictions x and the number of test instances N, accuracy $acc = x/N$.
- Can we predict the true accuracy of the model from acc?

Confidence Interval for Accuracy

- For large test sets ($N > 30$), the accuracy acc has a normal distribution with mean p and variance $p(1-p)/N$.
- Confidence interval for p is:

$$P\left(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}\right) = 1 - \alpha$$

Confidence Interval for Accuracy

- Consider a testing set containing 1,000 examples ($N = 1000$).
- 750 examples have been correctly classified ($x = 750$, $acc = 75\%$).
- If we want an 80% confidence level, then the true performance p is between 73.2% and 76.7%.
- If we only have 100 training examples and 75 correctly classified examples, the true performance p is between 69.1% and 80.1%.

Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - $N = 100, acc = 0.8$
  - Let $1 - \alpha = 0.95$ (95% confidence)
  - From probability table, $Z_{\alpha/2} = 1.96$

N	50	100	500	1000	5000
p (lower)	0.670	0.711	0.763	0.774	0.789
p (upper)	0.888	0.866	0.833	0.824	0.811

$1 - \alpha$	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

Establishing Confidence Intervals

- If S contains n examples drawn independently and $n \geq 30$, then

- With approximately 95% probability (or confidence), $\text{error}_D(h)$ lies in the interval

$$\text{error}_S(h) \pm 1.96 \sqrt{\frac{\text{error}_D(h)\big(1 - \text{error}_D(h)\big)}{n}}$$

Comparing Performance of Two Models

- Two models, say M1 and M2, which is better?
- M1 is tested on D1 (size $= n_1$), found error rate $= e_1$.
- M2 is tested on D2 (size $= n_2$), found error rate $= e_2$.
- Assume D1 and D2 are independent.
- If n_1 and n_2 are sufficiently large, then:

$$e_1 \sim N(\mu_1, \sigma_1) \qquad e_2 \sim N(\mu_2, \sigma_2)$$

- Approximate:

$$\hat{\sigma}_i = \frac{e_i(1 - e_i)}{n_i}$$

Comparing Performance of Two Models

- To test if the difference between the performance of M1 and M2 is statistically significant, we consider $d = e1 - e2$.
- $d \sim N(d_t, \sigma_t)$, where d_t is the true difference.
- Since D1 and D2 are independent:

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_1^2$$
$$= \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}$$

- At $(1-\alpha)$ confidence level: $d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$.

Example of Comparing Two Models

- Given M1 with $n_1 = 30$ and $e_1 = 0.15$ and M2 with $n_2 = 5000$ and $e_2 = 0.25$, $d = 0.1$ (2-sided test). Thus,

$$\hat{\sigma}_d = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

- At 95% confidence level, $Z_{\alpha/2} = 1.96$:

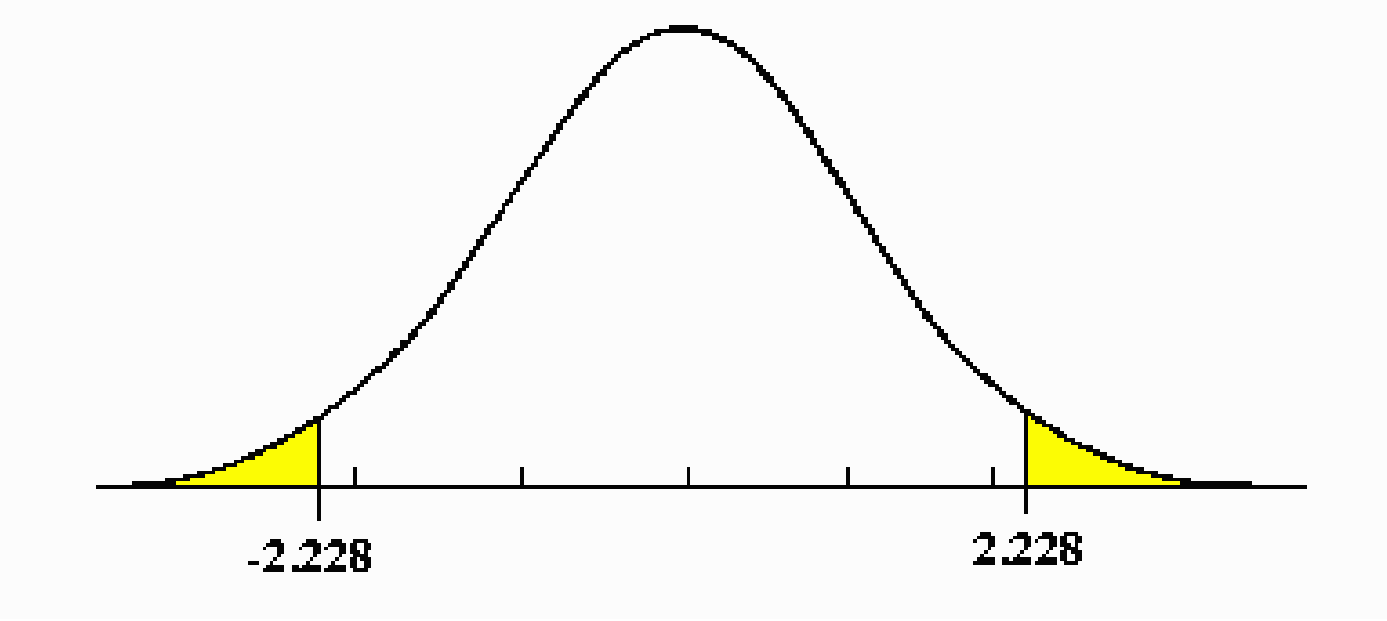
$$d_t = 0.100 \pm 1.96 \cdot \sqrt{0.0043} = 0.100 \pm 0.128.$$

- The interval contains 0, therefore the difference may not be statistically significant.

Student's t-Test

- Student's t-test tells us whether the means of two samples are significantly different
- Take individual samples from the sets of all possible cross-validation estimates
- Use a paired t-test because the individual samples are paired
  - The same CV is applied twice

Two-tailed t-Test



Comparing Two Classifiers

- Suppose we want to compare the performance of two classifiers using the k-fold cross-validation approach.
  - Assume we did 10-fold CV for two classifiers
- We want to know if there is a statistically significant difference between the two means.

Comparing Algorithms A and B

- Partition data D into k stratified disjoint subsets $T_1, T_2, \ldots, T_k$ of equal size.
- For $i = 1$ to k do

Use T_i as the testing set, and the remaining data for training set S_i

$$S_i \leftarrow \{D - T_i\}$$
$$h_A \leftarrow L_A(S_i)$$
$$h_B \leftarrow L_B(S_i)$$
$$\delta_i \leftarrow \text{error}_i(h_A) - \text{error}_i(h_B)$$

Return $\bar{\delta}$, where

$$\bar{\delta} \equiv \frac{1}{k}\sum_{i=1}^{k} \delta_i$$

Comparing Classifiers A and B

- The difference of the means also has a Student's distribution with $k-1$ degrees of freedom
- $N\%$ confidence interval for δ: $\bar{\delta} \pm t_{N,k-1}\ s_{\bar{\delta}}$

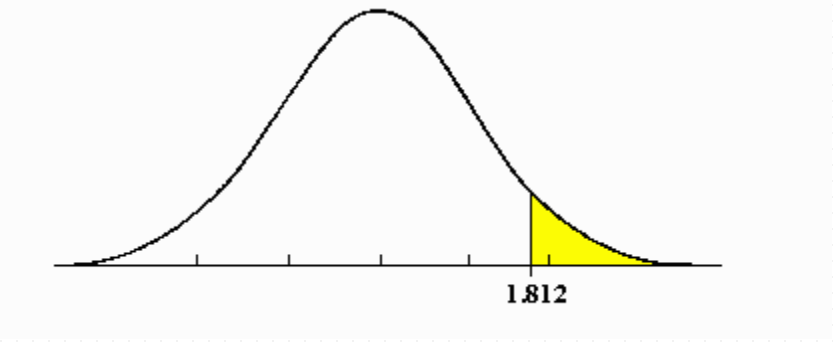
$$s_{\bar{\delta}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} \left(\delta_i - \bar{\delta}\right)^2}$$

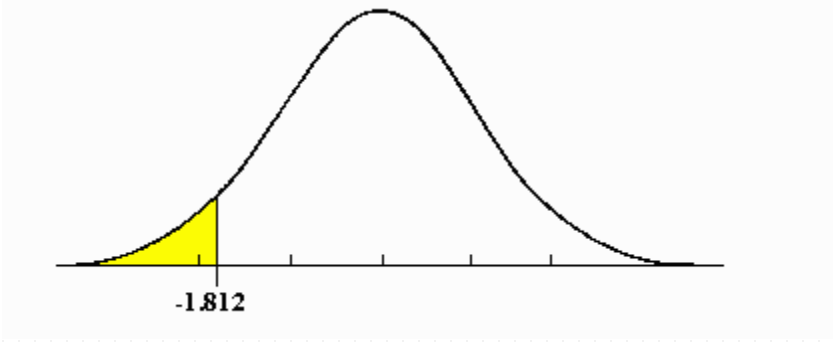
$$t_{N,K-1} = \frac{\bar{\delta}}{s_{\bar{\delta}}}$$

Performing the t-Test

1. Fix a significance level α
   - If a difference is significant at the α% level, there is a $(100 - \alpha)$% chance that there really is a difference
2. Divide the significance level by two, because the test is two-tailed
   - i.e., the true difference can be positive or negative
3. If $t_{N,k-1} < -t$ or $t \geq t_{N,k-1}$ then the difference is significant
   - i.e., the null hypothesis can be rejected

One-tailed t-Test





Unpaired Observations

- If the CV estimates are from different randomizations, they are no longer paired
- Then we have to use an unpaired t-test with $\min(k, j) - 1$ degrees of freedom
- The t-statistic becomes:

$$t = \frac{m_d}{\frac{{\sigma_d}^2}{k}} \rightarrow t = \frac{m_x - m_y}{\sqrt{\frac{{\sigma_x}^2}{k} + \frac{{\sigma_y}^2}{j}}}$$

Evaluation Measures Summary

- What you should know?
  - Confidence intervals
  - Evaluation schemes—hold-out, 10-fold CV, bootstrap, etc.
  - Significance tests
  - Different evaluation measures for classification
    - Error/accuracy, ROC, f-measure, lift curves, cost-sensitive classification