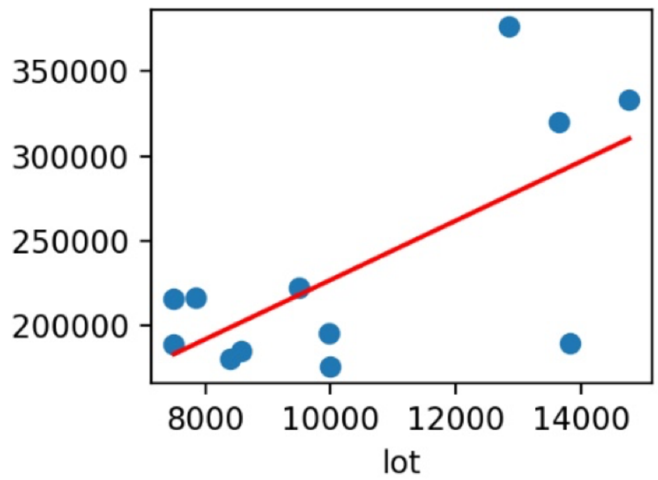
CMSC 478
Intro. to Machine Learning
Spring 2024

KMA Solaiman

ksolaima@umbc.edu

Visual version of linear regression: Learning



Let $h_\theta(x) = \sum_{j=0}^{d} \theta_j x_j$ want to choose θ so that $h_\theta(x) \approx y$. One popular idea called **least squares**

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2.$$

Choose

$$\theta = \underset{\theta}{\operatorname{argmin}}\, J(\theta).$$

Solving the least squares optimization problem.

Gradient Descent

	size	bedrooms	lot size
$x^{(1)}$	2104	4	45k
$x^{(2)}$	2500	3	30k

	Price
$y^{(1)}$	400
$y^{(2)}$	900

What's a prediction here?

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3.$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2.$$

$$\theta^{(0)} = 0$$

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \qquad \text{for } j = 0, \ldots, d.$$

Gradient Descent Computation

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \text{ for } j = 0, \ldots, d.$$

Note that α is called the **learning rate** or **step size**.

Let's compute the derivatives...

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta^{(t)}) &= \sum_{i=1}^{n} \frac{1}{2} \frac{\partial}{\partial \theta_j} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\
&= \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)})
\end{aligned}$$

Gradient Descent Computation

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \text{ for } j = 0, \ldots, d.$$

Note that α is called the **learning rate** or **step size**.

Let's compute the derivatives...

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) &= \sum_{i=1}^{n} \frac{1}{2} \frac{\partial}{\partial \theta_j} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\ &= \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)}) \end{aligned}$$

For our *particular* h_θ we have:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_d x_d \text{ so } \frac{\partial}{\partial \theta_j} h_\theta(x) = x_j$$

Gradient Descent Computation

Thus, our update rule for component j can be written:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}.$$

Gradient Descent Computation

Thus, our update rule for component j can be written:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}.$$

We write this in *vector notation* for $j = 0, \ldots, d$ as:

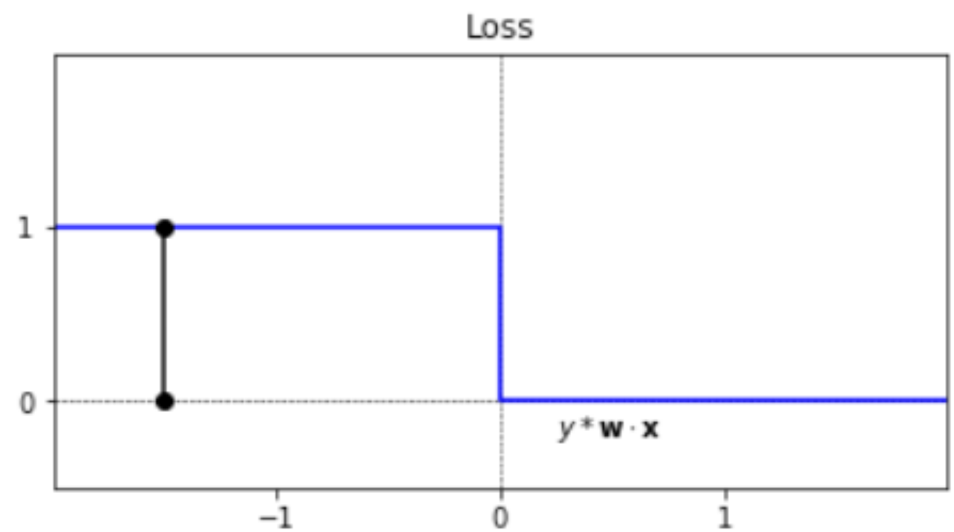
$$\theta^{(t+1)} = \theta^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}.$$

Saves us a lot of writing! And easier to understand ... eventually.

Loss Function for Classification: 0-1 Loss

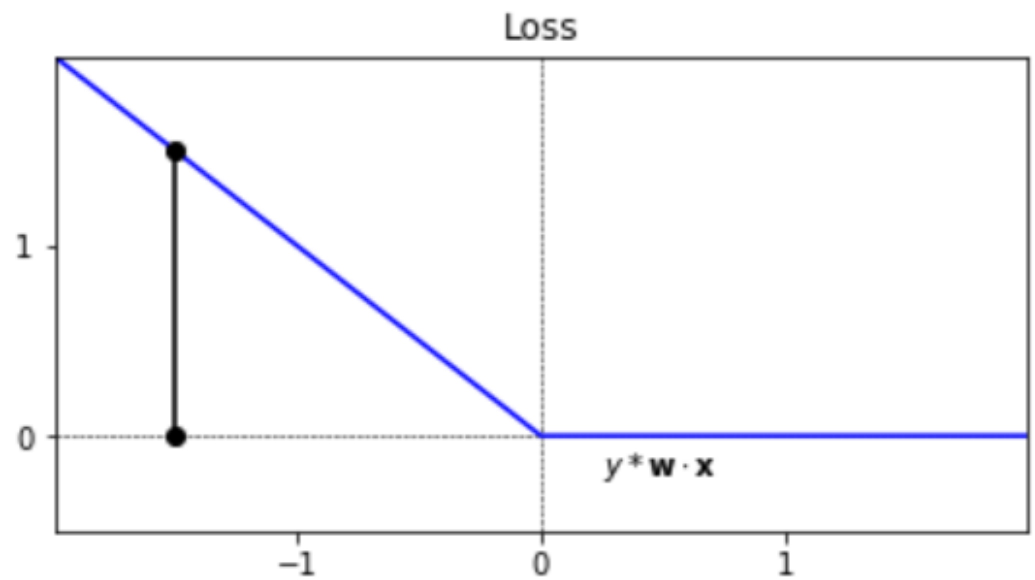
L_{0-1}	$\hat{y} = -1$	$\hat{y} = 1$
$y = -1$	0	1
$y = 1$	1	0

$$L_{0-1}(y, \mathbf{w} \cdot \mathbf{x}) = \begin{cases} 0 & \text{if } y * \mathbf{w} \cdot \mathbf{x} > 0 \\ 1 & \text{otherwise} \end{cases}$$



Perceptron Loss

$$L_P(y, \mathbf{w} \cdot \mathbf{x}) = \begin{cases} 0 & \text{if } y * \mathbf{w} \cdot \mathbf{x} > 0 \\ -y * \mathbf{w} \cdot \mathbf{x} & \text{otherwise} \end{cases}$$



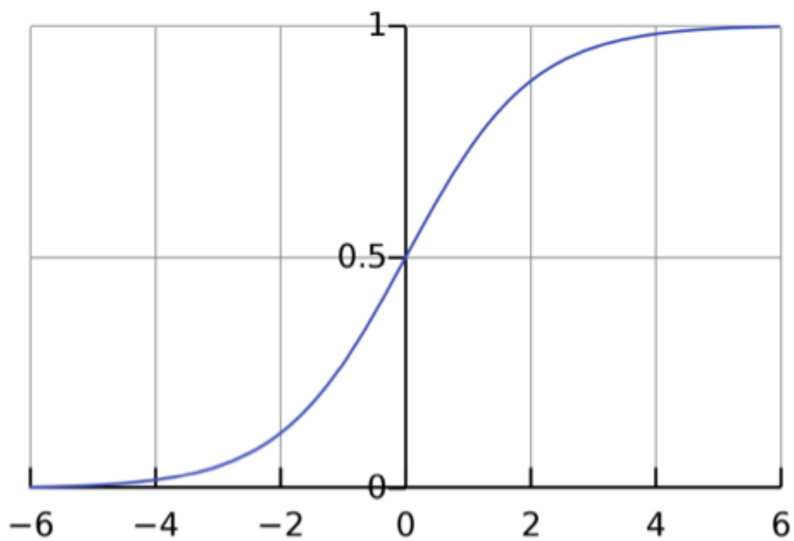
Logistic Regression: Link Functions

Given a training set $\{(x^{(i)}, y^{(i)}) \text{ for } i = 1, \ldots, n\}$ let $y^{(i)} \in \{0, 1\}$.
Want $h_\theta(x) \in [0, 1]$. Let's pick a smooth function:

$$h_\theta(x) = g(\theta^T x)$$

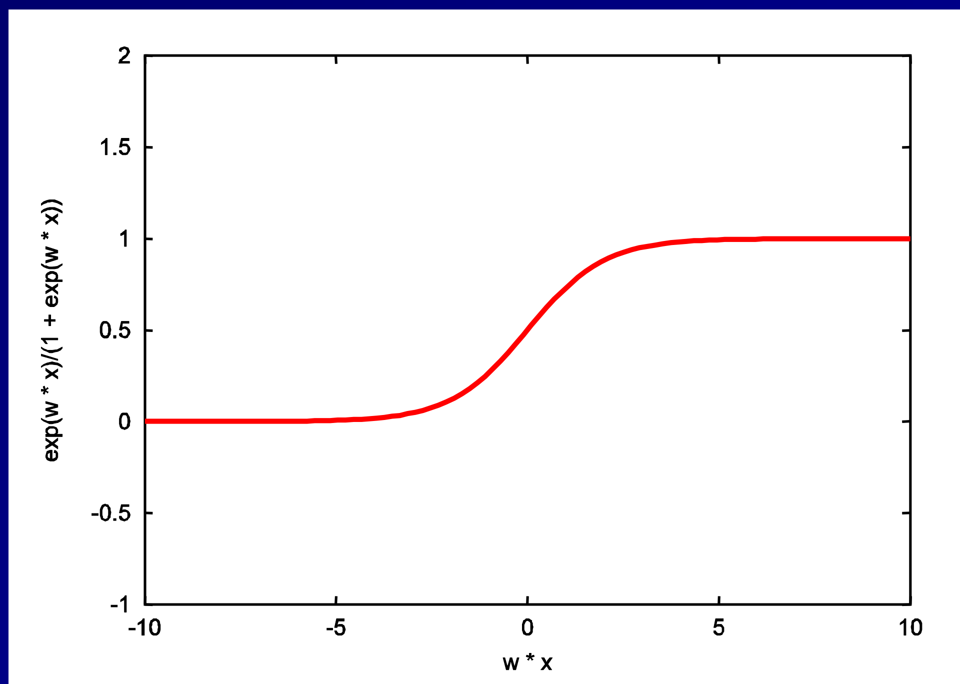
Here, g is a link function. There are *many*... but we'll pick one!

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



Why the exp function?

- One reason: A linear function has a range from $[-\infty, \infty]$ and we need to force it to be positive and sum to 1 in order to be a probability:



Logistic Regression: Link Functions

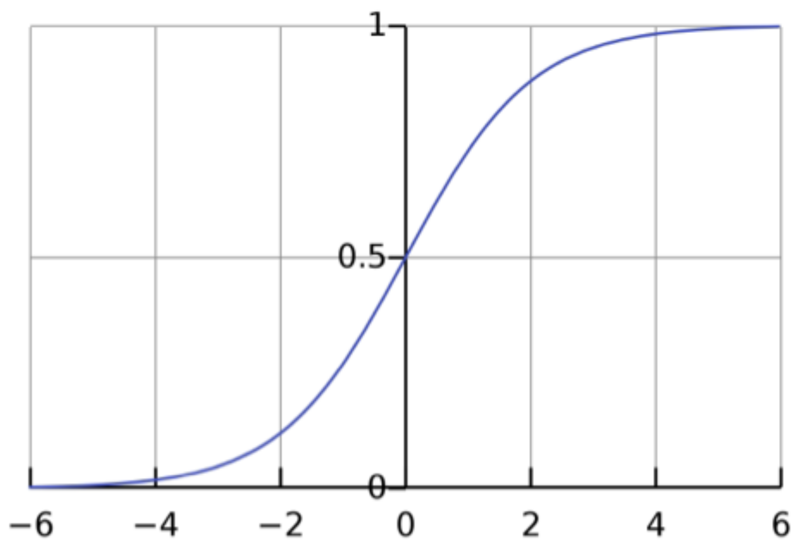
Given a training set $\{(x^{(i)}, y^{(i)})$ for $i = 1, \ldots, n\}$ let $y^{(i)} \in \{0, 1\}$.
Want $h_\theta(x) \in [0, 1]$. Let's pick a smooth function:

$$h_\theta(x) = g(\theta^T x)$$

Here, g is a link function. There are *many*... but we'll pick one!

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

SIGMOID



How do we interpret $h_\theta(x)$?

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Then,

$$L(\theta) = P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$$

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Then,

$$L(\theta) = P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$$

Conditional Distribution P(y | X)

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Then,

$$L(\theta) = P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$$

How do we go to something similar to a cost function from P (y l X; θ) ?

- Maximum Likelihood Estimation (MLE)

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

$$
\begin{aligned}
P(y = 1 \mid x; \theta) =& h_\theta(x) \\
P(y = 0 \mid x; \theta) =& 1 - h_\theta(x)
\end{aligned}
$$

Then,

$$
\begin{aligned}
L(\theta) =& P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\
=& \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \quad \text{exponents encode ``if-then''}
\end{aligned}
$$

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

$$
\begin{aligned}
P(y = 1 \mid x; \theta) =& h_\theta(x) \\
P(y = 0 \mid x; \theta) =& 1 - h_\theta(x)
\end{aligned}
$$

Then,

$$
\begin{aligned}
L(\theta) =& P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\
=& \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \quad \text{exponents encode "if-then"}
\end{aligned}
$$

Taking logs to compute the log likelihood $\ell(\theta)$ we have:

$$
\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))
$$

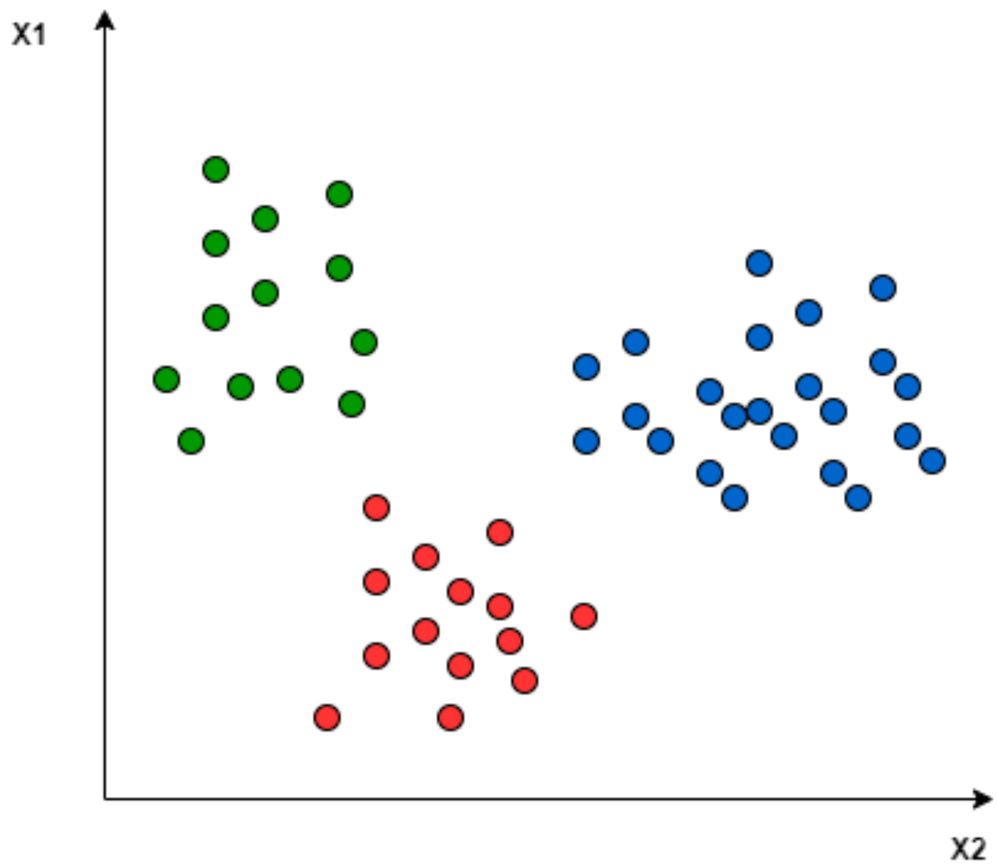
Now to solve it...

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

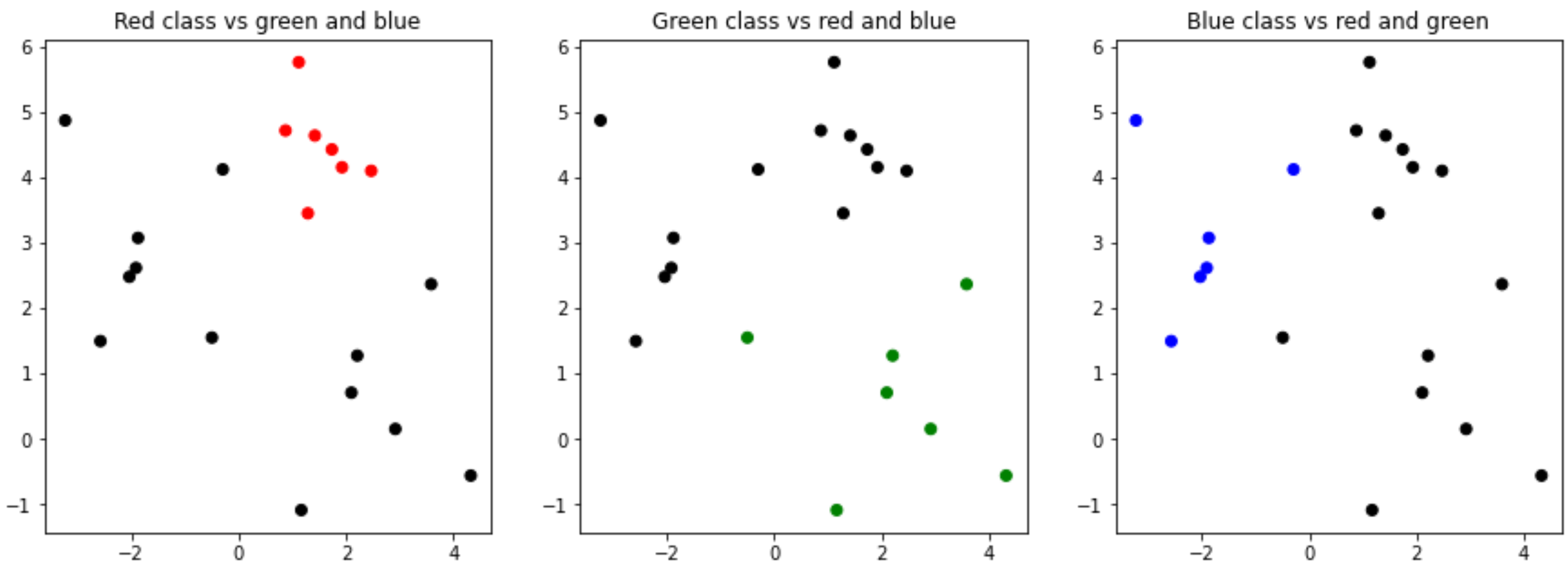
We maximize for θ but we already saw how to do this! Just compute derivative, run (S)GD and you're done with it!

Takeaway: This is *another* example of the max likelihood method: we setup the likelihood, take logs, and compute derivatives.

Extending LR to K>2 classes



1 vs All



Red class vs green and blue
Green class vs red and blue
Blue class vs red and green
6
5
4
3
2
1
0
−1
−2
0
2
4

A Quick and Dirty Intro to Multiclass Classification.
This technique is *the daily workhorse of modern AI/ML*

Multiclass

Suppose we want to choose among k discrete values, e.g., $\{\text{'Cat'}, \text{'Dog'}, \text{'Car'}, \text{'Bus'}\}$ so $k = 4$.

We encode with **one-hot** vectors i.e. $y \in \{0,1\}^k$ and $\sum_{j=1}^{k} y_j = 1$.

$$\underset{\text{`Cat'}}{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}} \quad \underset{\text{`Dog'}}{\begin{pmatrix}0\\1\\0\\0\end{pmatrix}} \quad \underset{\text{`Car'}}{\begin{pmatrix}0\\0\\1\\0\end{pmatrix}} \quad \underset{\text{`Bus'}}{\begin{pmatrix}0\\0\\0\\1\end{pmatrix}}$$

A prediction here is actually a *distribution* over the k classes. This leads to the SoftMax function described below (derivation in the notes!). That is our hypothesis is a vector of k values:

$$P(y = j | x; \bar{\theta}) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^{k} \exp(\theta_i^T x)}.$$

Here each θ_j has the *same dimension* as x, i.e., $x, \theta_j \in R^{d+1}$ for $j = 1, \ldots, k$.

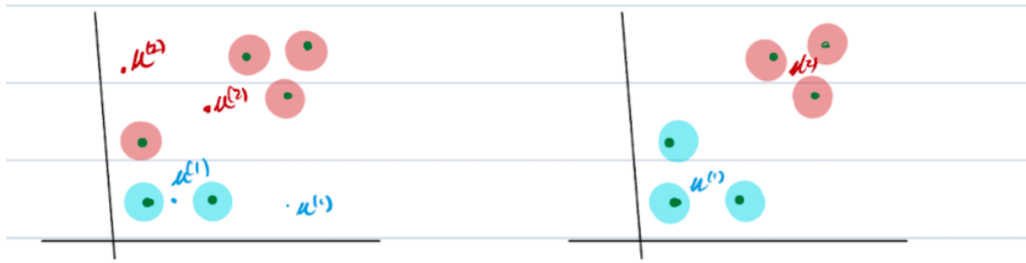
Extending Logistic Regression to K > 2 classes

- Choose class K to be the "reference class" and represent each of the other classes as a logistic function of the odds of class *k* versus class K:

$$
\begin{aligned}
\log \frac{P(y=1|\mathbf{x})}{P(y=K|\mathbf{x})} &= \mathbf{w}_1 \cdot \mathbf{x} \\
\log \frac{P(y=2|\mathbf{x})}{P(y=K|\mathbf{x})} &= \mathbf{w}_2 \cdot \mathbf{x} \\
&\vdots \\
\log \frac{P(y=K-1|\mathbf{x})}{P(y=K|\mathbf{x})} &= \mathbf{w}_{K-1} \cdot \mathbf{x}
\end{aligned}
$$

- Gradient ascent can be applied to simultaneously train all of these weight vectors $\mathbf{w}_k$

How do we find these clusters? (Iterative Approach)



- (Randomly) Initialize Centers $\mu^{(1)}$ and $\mu^{(2)}$.
- Assign each point, $x^{(i)}$, to closest cluster

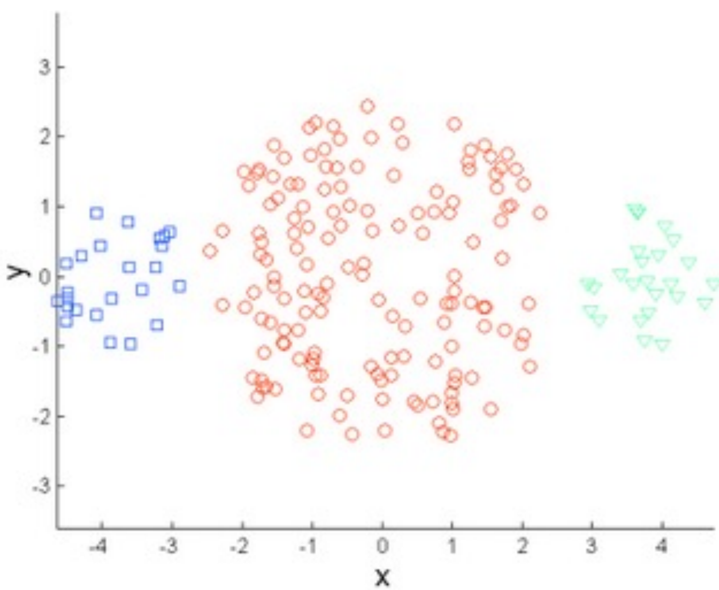
$$C^{(i)} = \operatorname*{argmin}_{j=1,\dots,k} \|\mu^{(j)} - x^{(i)}\|^2 \text{ for } i = 1, \dots, n$$

- Compute new center of each cluster:

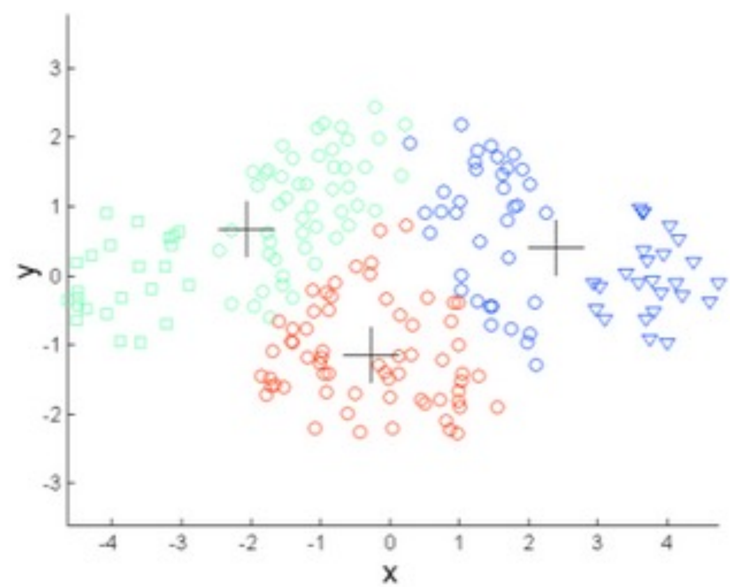
$$\mu^{(j)} = \frac{1}{|\Omega_j|} \sum_{i \in \Omega_j} x^{(i)} \text{ where } \Omega_j = \{i : C^{(i)} = j\}$$

Repeat until clusters stay the same!

Different number of clusters

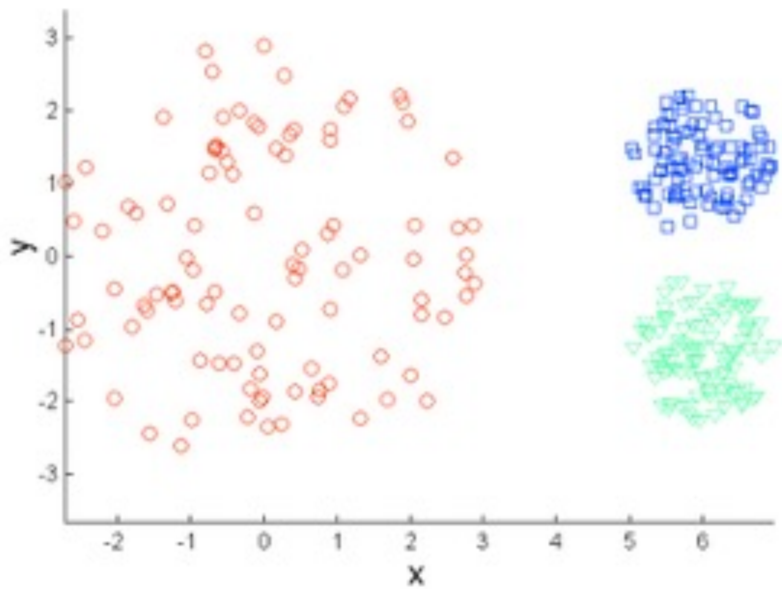


Original Points

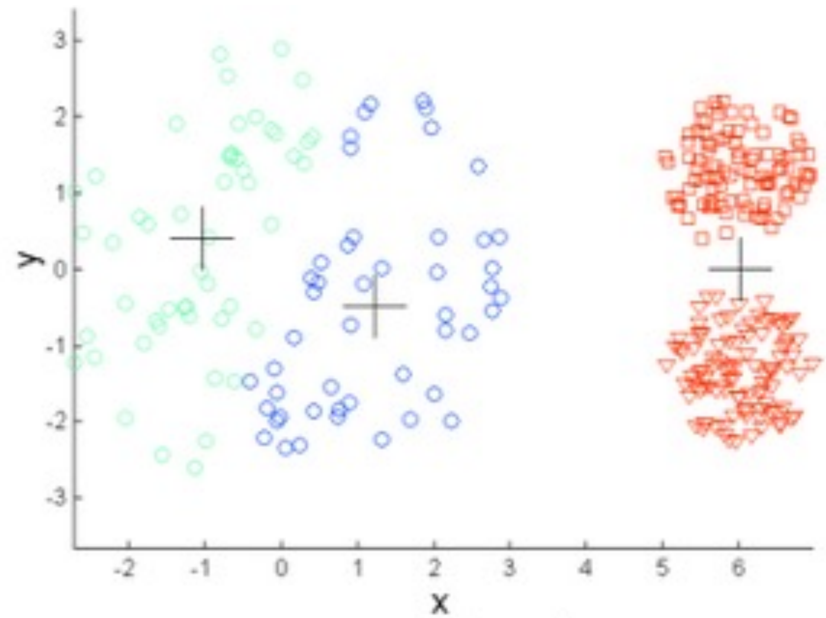


K-means (k = 3)

Different Densities



Original Points



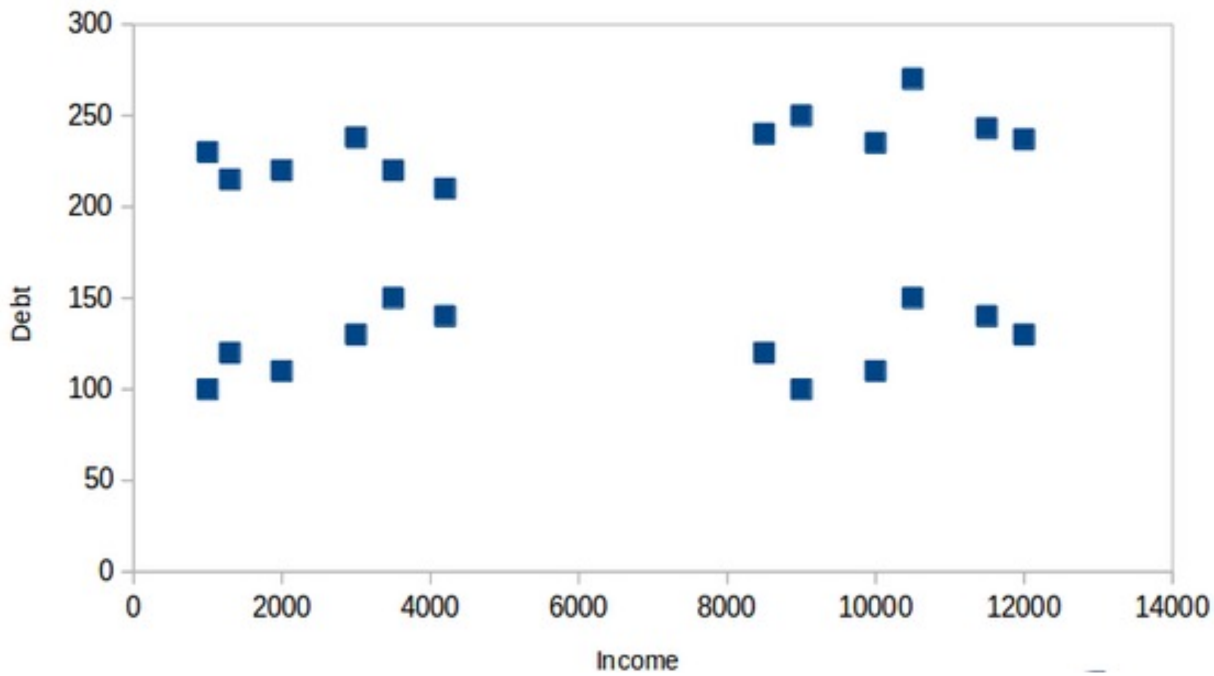
K-means (k = 3)

K-means++

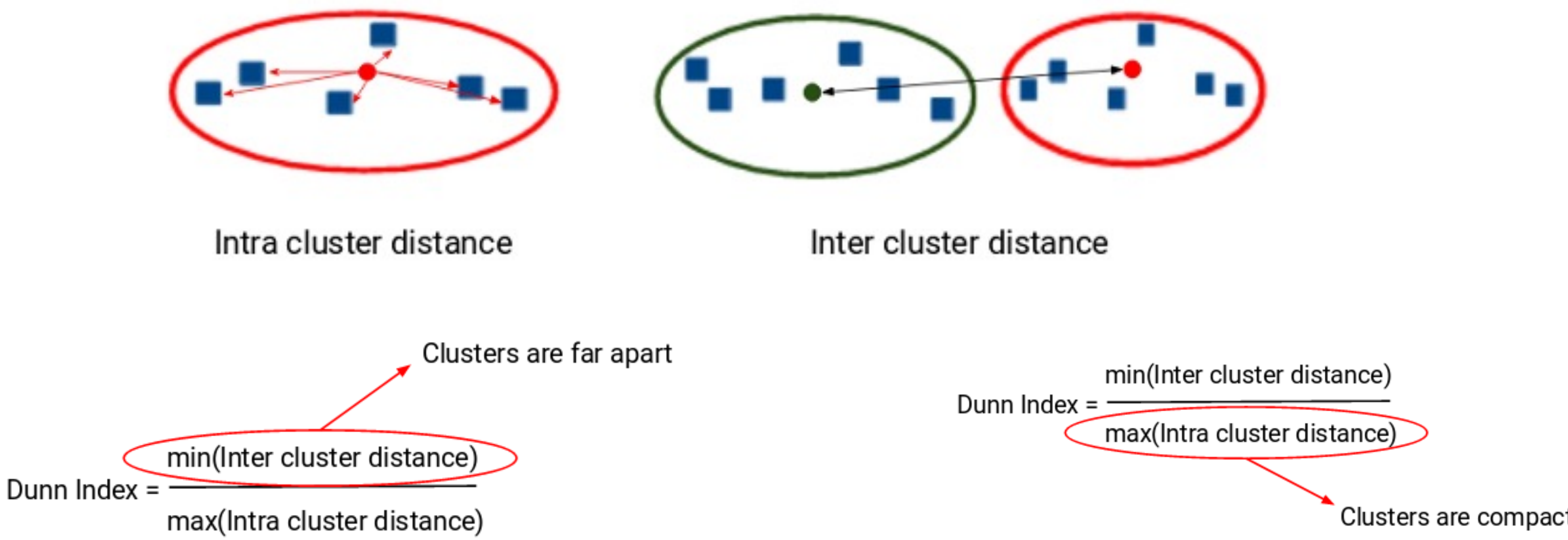
- Steps to Initialize the Centroids Using K-Means++

1. The first cluster is chosen uniformly at random from the data points we want to cluster. This is similar to what we do in K-Means, but instead of randomly picking all the centroids, we just pick one centroid here
2. Next, we compute the distance (D(x)) of each data point (x) from the cluster center that has already been chosen
3. Then, choose the new cluster center from the data points with the probability of x being proportional to $(D(x))^2$
4. We then repeat steps 2 and 3 until *k* clusters have been chosen

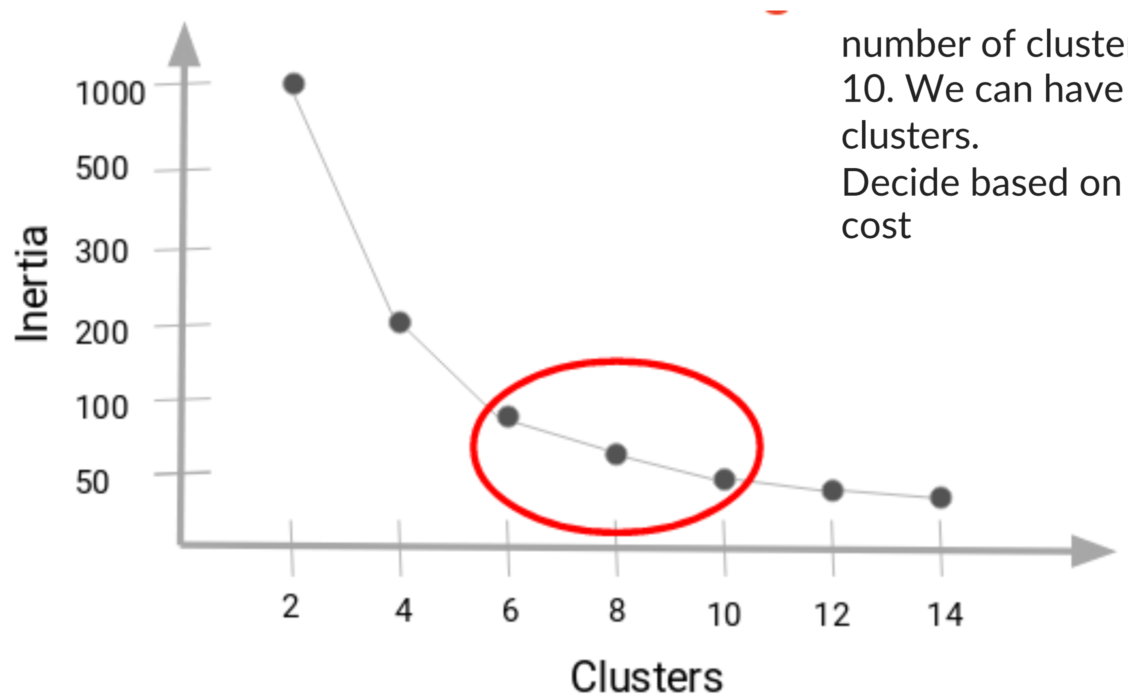
How to Choose the Right Number of Clusters?



- Dunn index


Intra cluster distance
Inter cluster distance
Clusters are far apart
Dunn Index = min(Inter cluster distance) / max(Intra cluster distance)
Dunn Index = min(Inter cluster distance) / max(Intra cluster distance)
Clusters are compact

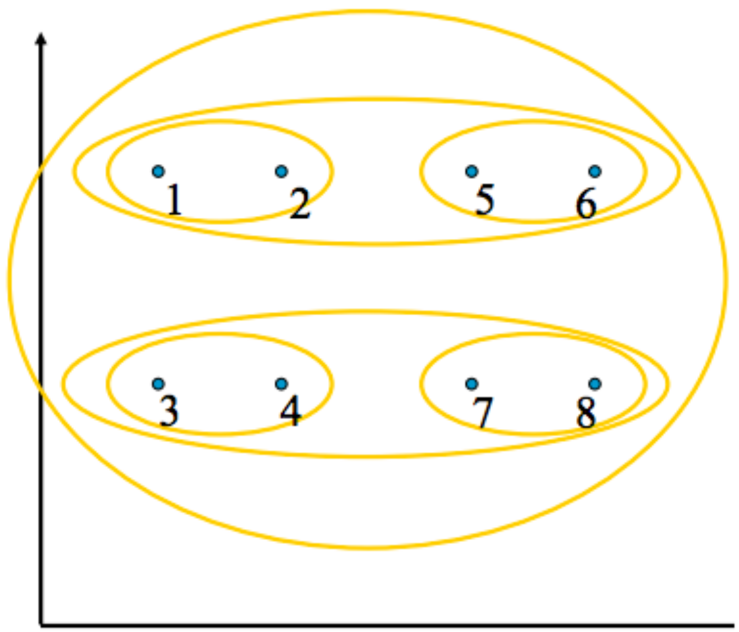
Empirical Choice of K



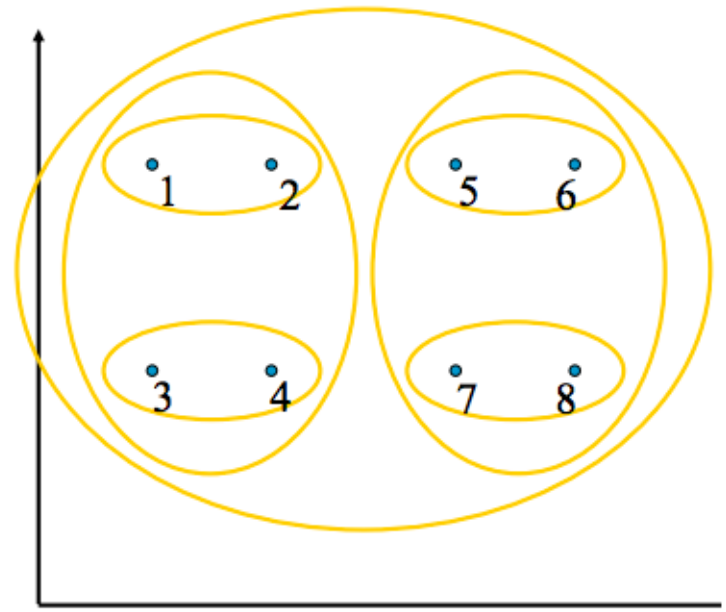
number of clusters between 6 and 10. We can have 7, 8, or even 9 clusters.
Decide based on computational cost

Agglomerative clustering

Closest pair
(single-link clustering)



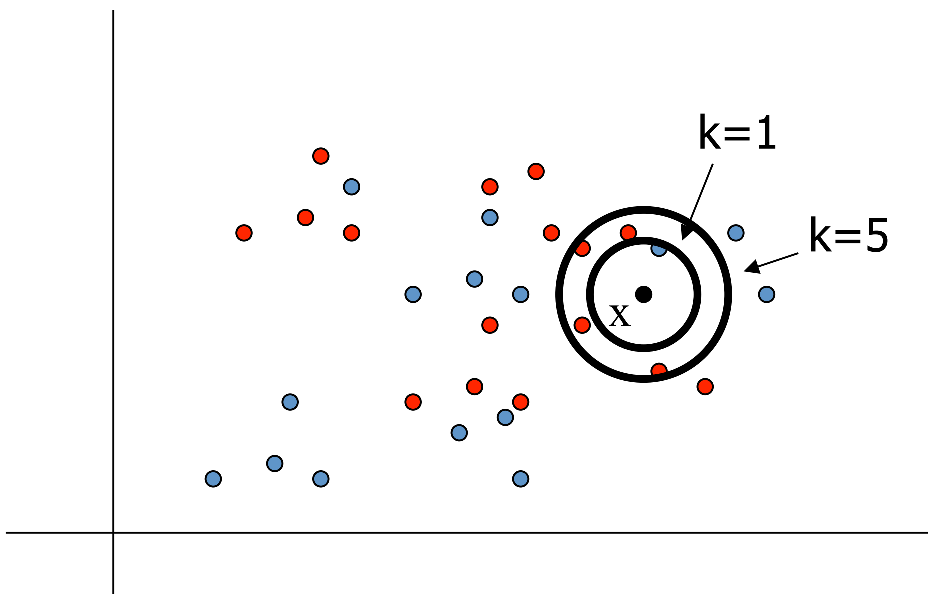
Farthest pair
(complete-link clustering)



[Pictures from Thorsten Joachims]

K-Nearest Neighbor Methods

- To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class

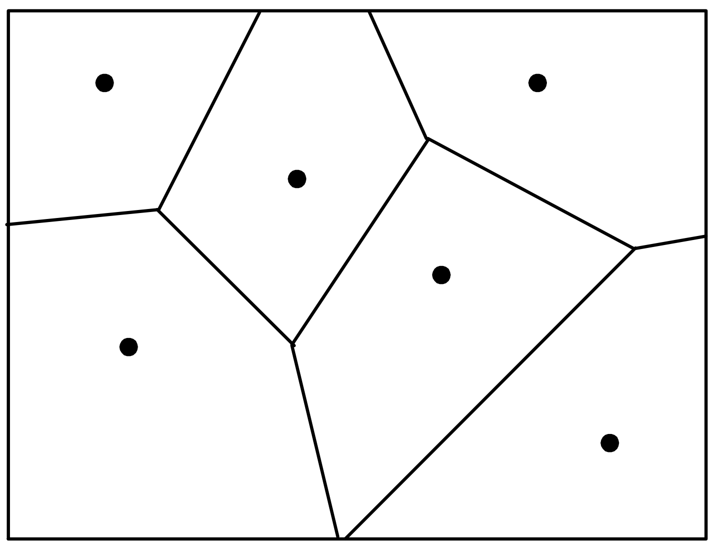


common values for k: 3, 5

Decision Boundaries

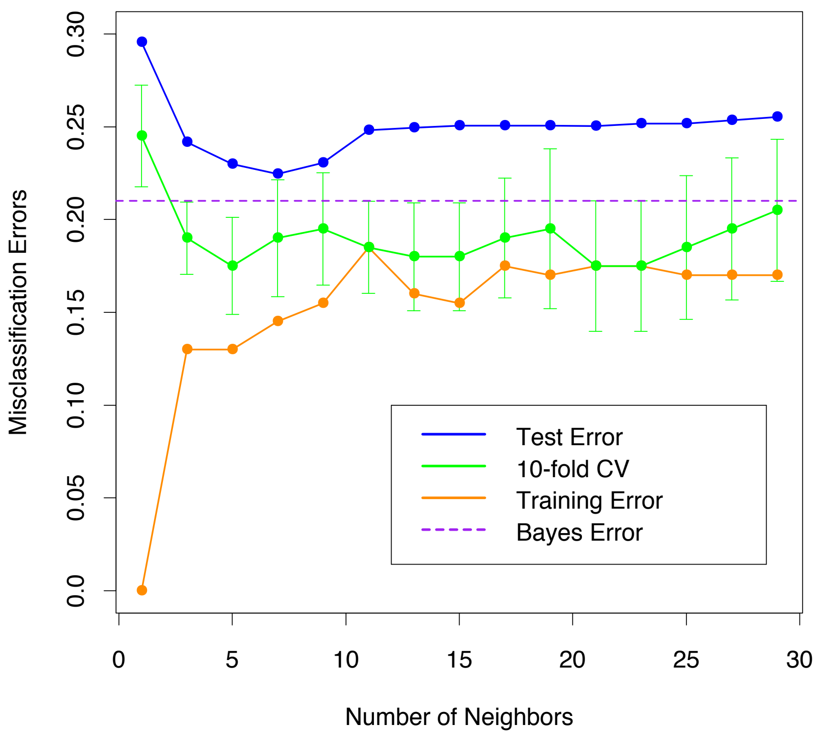
- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

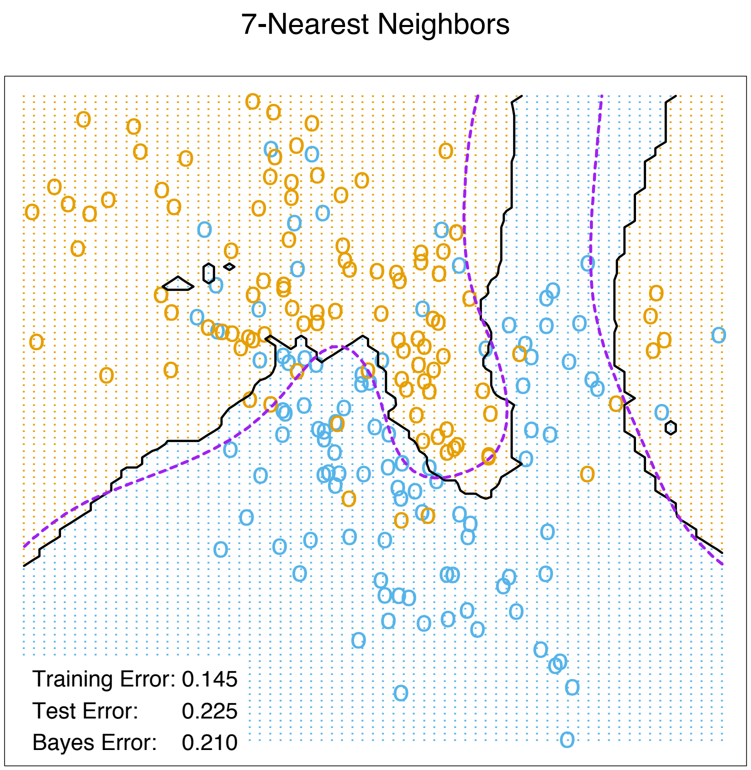
1-NN Decision Surf ace



- The more examples that are stored, the more complex the decision boundaries can become

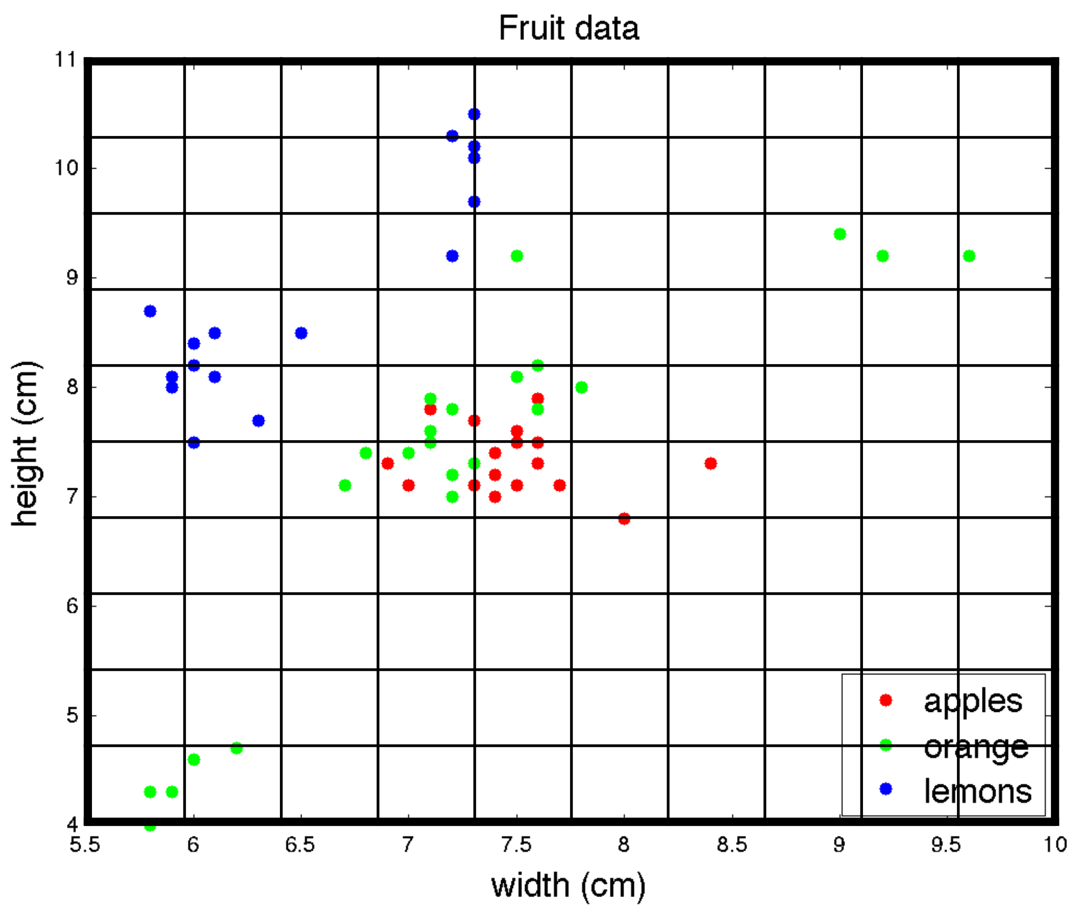
Example results for k-NN





[Figures from Hastie and Tibshirani, Chapter 13]

Practical issue when using kNN: Curse of dimensionality



#bins = 10x10

d = 2

#bins = 10^d

d = 1000

Atoms in the universe: $\sim 10^{80}$

How many neighborhoods are there?

Nearest Neighbor

When to Consider

- Instance map to points in R^n
- Less than 20 attributes per instance
- Lots of training data

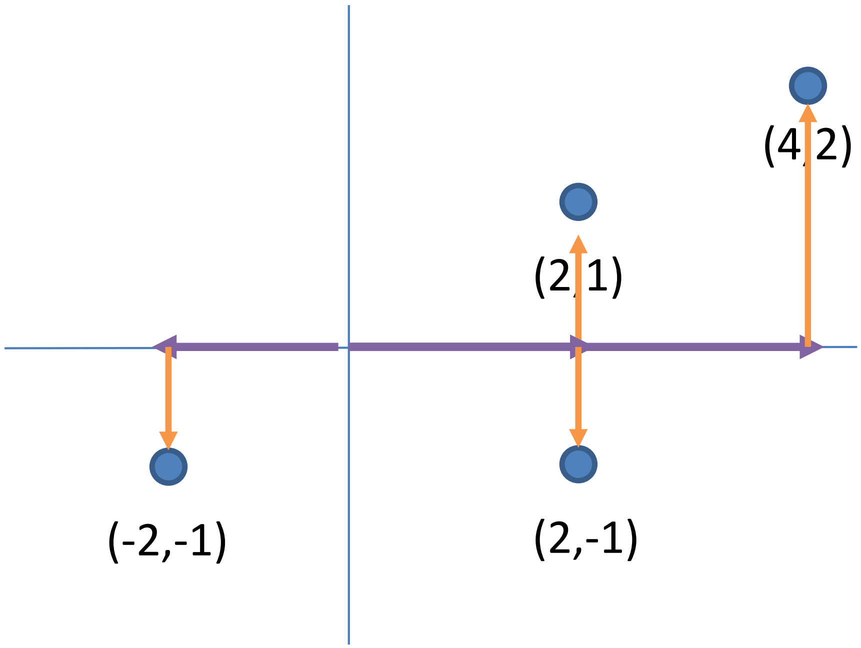
Advantages

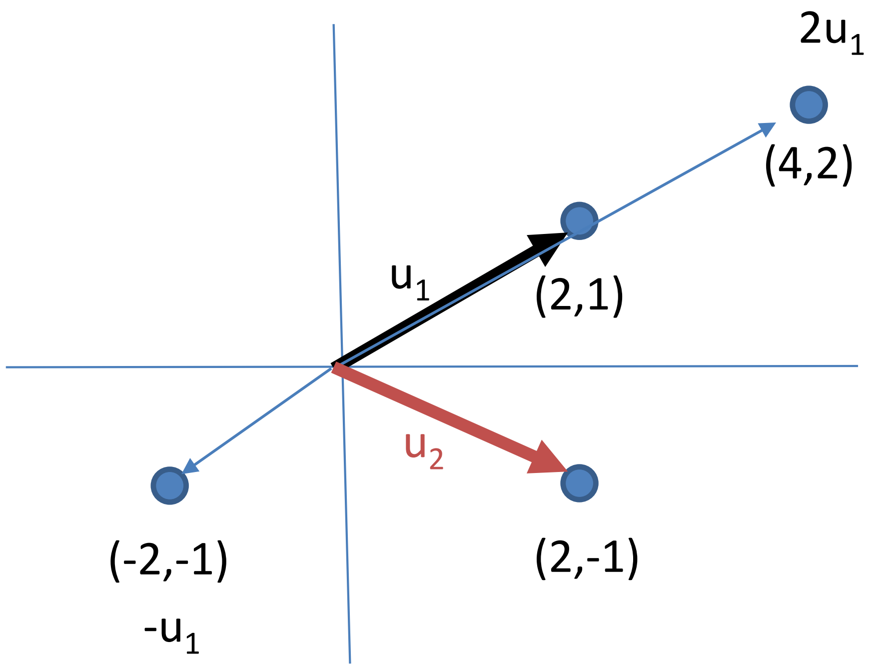
- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

Summarizing Redundant Information





(2,1) = 1*(2,1) + 0*(2,-1)
(4,2) = 2*(2,1) + 0*(2,-1)

(Is it the most general? These vectors aren't orthogonal)

Algorithm 37 PCA($\mathbf{D}$, K)

```
μ ← MEAN(X)                                            // compute data mean for centering
D ← (X − μ1^⊤)^⊤ (X − μ1^⊤)                            // compute covariance, 1 is a vector of ones
{λ_k, u_k} ← top K eigenvalues/eigenvectors of D
return (X − μ1) U                                      // project data using U
```

Finding PCA

There are two ways you can find PCA:

- Maximize the projected subspace of the data. (we see more)

$$\max_{u \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (u \cdot x^{(i)})^2.$$

- Minimize the residual

$$\min_{u \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - u \cdot x^{(i)})^2.$$

We need to recall some more linear algebra to solve this.

More PCA

- **Multiple Dimensions** What if we want multiple dimensions? We keep the top-k.

$$\max_{U \in \mathbb{R}^{k \times d} : UU^T = I_k} \frac{1}{n} \sum_{u=1}^{n} \| U x^{(i)} \|^2.$$

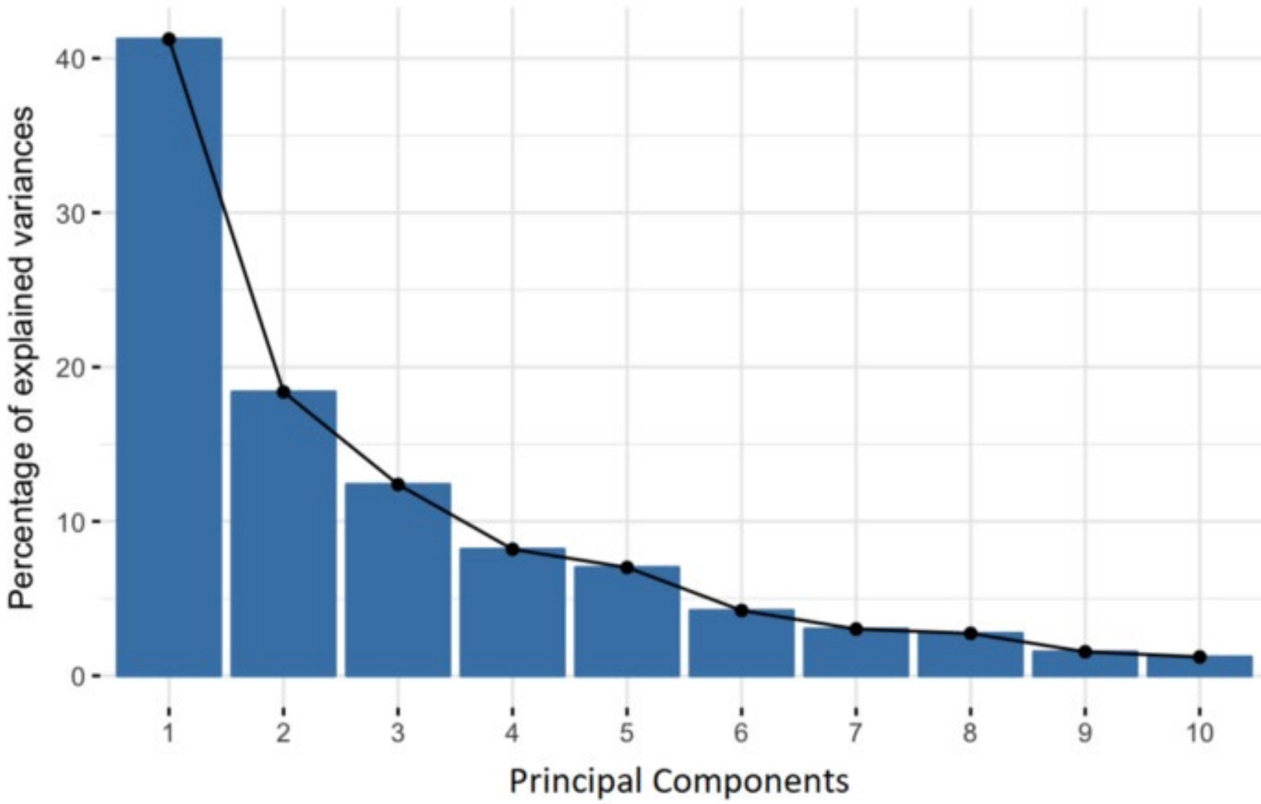
- **Reduce dimensionality**. How do we represent data with just those $k < d$ scalars α_j for $j = 1, \ldots, k$

$$x = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_d u_d \text{ keep only } (\alpha_1, \ldots, \alpha_k)$$

  - Lurking instability: what if $\lambda_j = \lambda_{j+1}$?
- **Choose k?** One approach is "amount of explained variance"

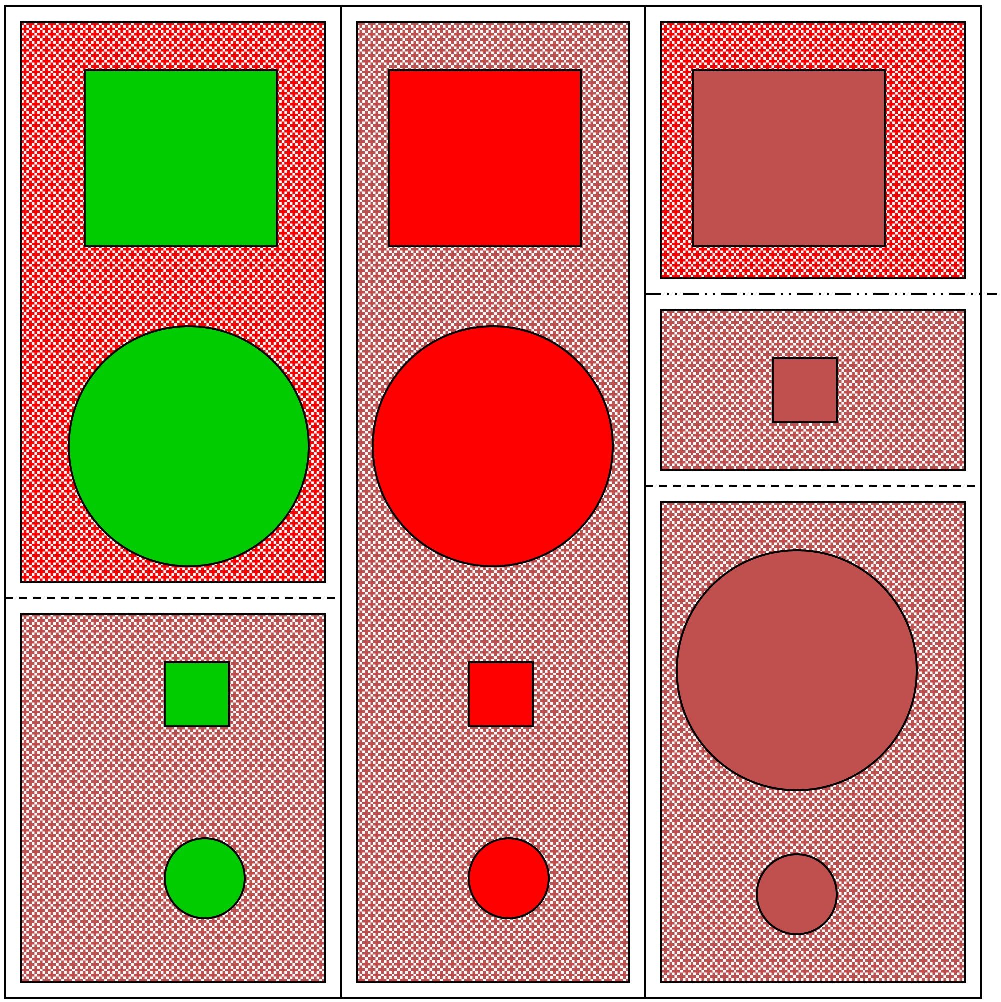
$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{n} \lambda_i} \geq 0.9 \text{ note } \operatorname{tr}(C) = \sum_{i=1}^{n} C_{i,i} = \sum_{i=1}^{n} \lambda_i$$

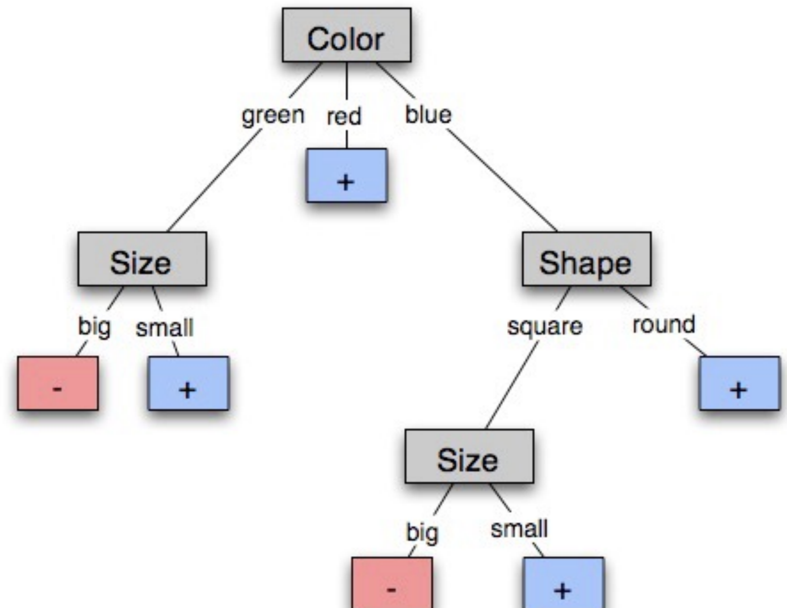
Recall $\lambda_j \geq 0$ since C is a covariance matrix.


Percentage of explained variances
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
Principal Components

A decision tree-induced partition

The red groups are negative examples, blue positive





Negative things are big, green shapes and big, blue squares

Choosing best attribute

- **Key problem:** choose attribute to split given set of examples
- Possibilities for choosing attribute:
  - **Random:** Select one at random
  - **Least-values:** one with smallest # of possible values
  - **Most-values:** one with largest # of possible values
  - **Max-gain:** one with largest expected ***information gain***
  - **Gini impurity:** one with smallest gini impurity value
- The last two measure the **homogeneity** of the target variable within the subsets
- The ID3 and C4.5 algorithms uses **max-gain**

A Simple Example

For this data, is it better to start the tree by asking about the restaurant **type** or its current **number of patrons**?

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

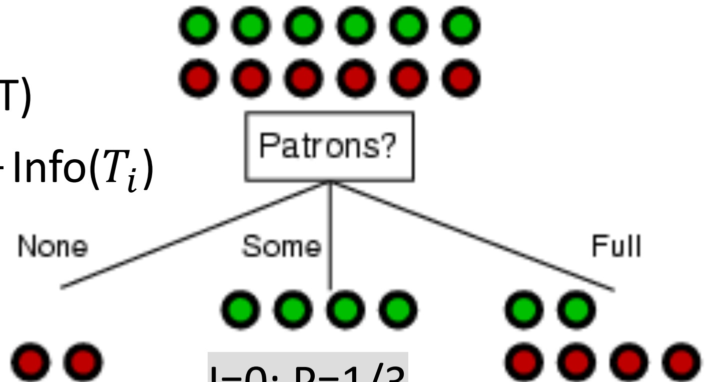
Information Gain

stay
leave

I = Info(T)
$= -\sum_c \widehat{p_c} \log_2 \widehat{p_c}$

I = -(.5*log$_2$(.5) + .5*log$_2$(.5)) = 0.5+0.5 => **1.0**

Info(X, T)
$= \sum_i \frac{|T_i|}{|T|} \text{Info}(T_i)$

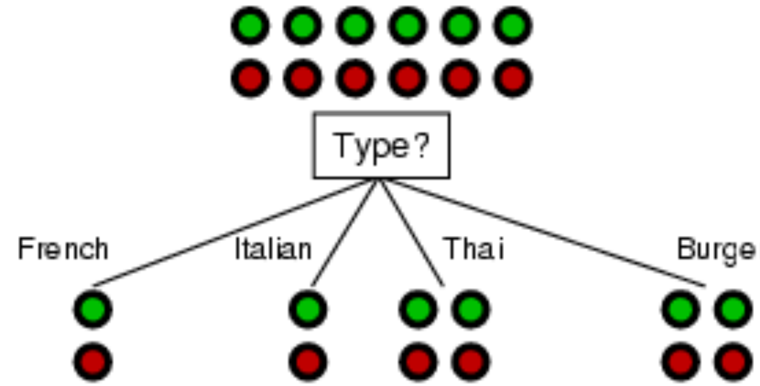


I=0; P=1/6

I=0; P=1/3

I=-(1/3*log$_2$(1/3)+2/3*log$_2$(2/3)),
P=6/12=1/2 => 0.91/2 = 0.46

Information gain = 1 - 0.46 => **0.54**



I=1;P=1/6 I=1; P=1/6 I=1; P=2/6 I=1; P=2/6

I = 6/6*1 => **1.0**

Information gain = 1 - 1 => **0.0**

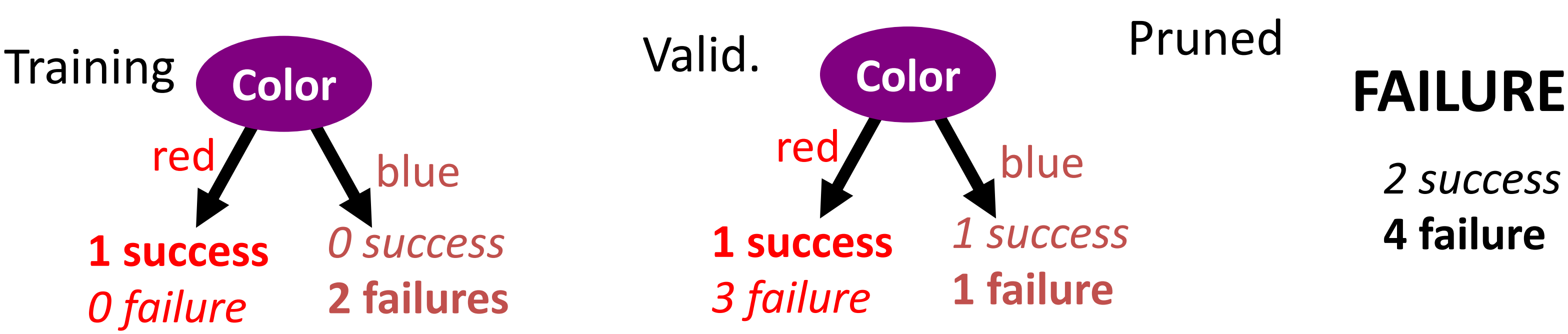
- **Information gain** for asking **Patrons = 0.54**, for asking **Type = 0**
- Note: If only one of the N categories has any instances, the information entropy is always 0

Avoiding Overfitting

- Remove obviously **irrelevant features**
  - E.g., remove 'year observed', 'month observed', 'day observed', 'observer name' from the attributes used
- Get **more training data**
- **Pruning** lower nodes in a decision tree
  - E.g., if info. gain of best attribute at a node is below a threshold, stop and make this node a leaf rather than generating children nodes

Pruning decision trees

- Pruning a decision tree is done by replacing a whole subtree by a leaf node
- Replacement takes place if the expected error rate in the subtree is greater than in the single leaf, e.g.,
  - **Training data:** 1 training red success and 2 training blue failures
  - **Validation data**: 3 red failures and one blue success
  - Consider replacing subtree by a single node indicating failure
- After replacement, only 2 errors instead of 4



Principles for Estimating Probabilities

Principle 1 (maximum likelihood):

- choose parameters θ that maximize **P(data | θ)**
- e.g.,

$$\hat{\theta}^{MLE} = \frac{\alpha_1}{\alpha_1 + \alpha_0}$$

$$\frac{P(data|\theta)\,P(\theta)}{P(data)} =$$

Principle 2 (maximum a posteriori prob.):

- choose parameters θ that maximize **P(θ | data)**
- e.g.

$$\hat{\theta}^{MAP} = \frac{\alpha_1 + \#\text{hallucinated\_1s}}{(\alpha_1 + \#\text{hallucinated\_1s}) + (\alpha_0 + \#\text{hallucinated\_0s})}$$

Maximum Likelihood Estimation

X=1 X=0

P(X=1) = θ P(X=0) = (1-θ)

Data D: = {1 0 0 1} 1

$P(D|\theta) = \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta = \theta^{\alpha_1} (1-\theta)^{\alpha_0}$

Flips produce data D with α_1 heads, α_0 tails

- flips are independent, identically distributed 1's and 0's (Bernoulli)
- α_1 and α_0 are counts that sum these outcomes (Binomial)

$$P(D|\theta) = P(\alpha_1, \alpha_0|\theta) = \theta^{\alpha_1}(1-\theta)^{\alpha_0}$$

Maximum Likelihood Estimate for Θ

$$\hat{\theta} = \arg\max_{\theta} \ln P(\mathcal{D} \mid \theta)$$
$$= \arg\max_{\theta} \ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

- Set derivative to zero: $\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = 0$

[C. Guestrin]

$$\hat{\theta} = \arg\max_{\theta} \ln P(D|\theta)$$

$$= \arg\max_{\theta} \ln \left[\theta^{\alpha_1}(1-\theta)^{\alpha_0}\right]$$

- Set derivative to zero: $\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = 0$

hint: $\frac{\partial \ln \theta}{\partial \theta} = \frac{1}{\theta}$

$$\frac{\partial}{\partial \theta} \alpha_1 \ln \theta + \alpha_0 \ln(1-\theta)$$

$$\alpha_1 \frac{1}{\theta} + \alpha_0 \frac{\partial \ln(1-\theta)}{\partial \theta}$$

$$\frac{\partial \ln(1-\theta)}{\partial(1-\theta)} \cdot \frac{\partial(1-\theta)}{\partial \theta}$$

$$\frac{1}{1-\theta} \qquad -1$$

$$0 = \alpha_1 \frac{1}{\theta} - \frac{\alpha_0}{1-\theta}$$

$$\theta = \frac{\alpha_1}{\alpha_1 + \alpha_0}$$

Summary: Maximum Likelihood Estimate

P(X=1) = θ
P(X=0) = 1-θ
(Bernoulli)

- Each flip yields boolean value for X

$$X \sim \text{Bernoulli: } P(X) = \theta^X (1-\theta)^{(1-X)}$$

- Data set D of independent, identically distributed (iid) flips produces α_1 ones, α_0 zeros (Binomial)

$$P(D|\theta) = P(\alpha_1, \alpha_0|\theta) = \theta^{\alpha_1}(1-\theta)^{\alpha_0}$$

$$\hat{\theta}^{MLE} = \text{argmax}_\theta \, P(D|\theta) = \frac{\alpha_1}{\alpha_1+\alpha_0}$$

Principles for Estimating Probabilities

Principle 1 (maximum likelihood):

- choose parameters θ that maximize $P(data \mid \theta)$

Principle 2 (maximum a posteriori prob.):

- choose parameters θ that maximize

$$P(\theta \mid data) = \frac{P(data \mid \theta)\, P(\theta)}{P(data)}$$

Beta prior distribution – P(θ)

- $$P(\theta) = \frac{\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}}{B(\beta_H,\beta_T)} \sim Beta(\beta_H,\beta_T)$$
- Likelihood function: $P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- Posterior: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$

Beta prior distribution – P(θ)

$$P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$

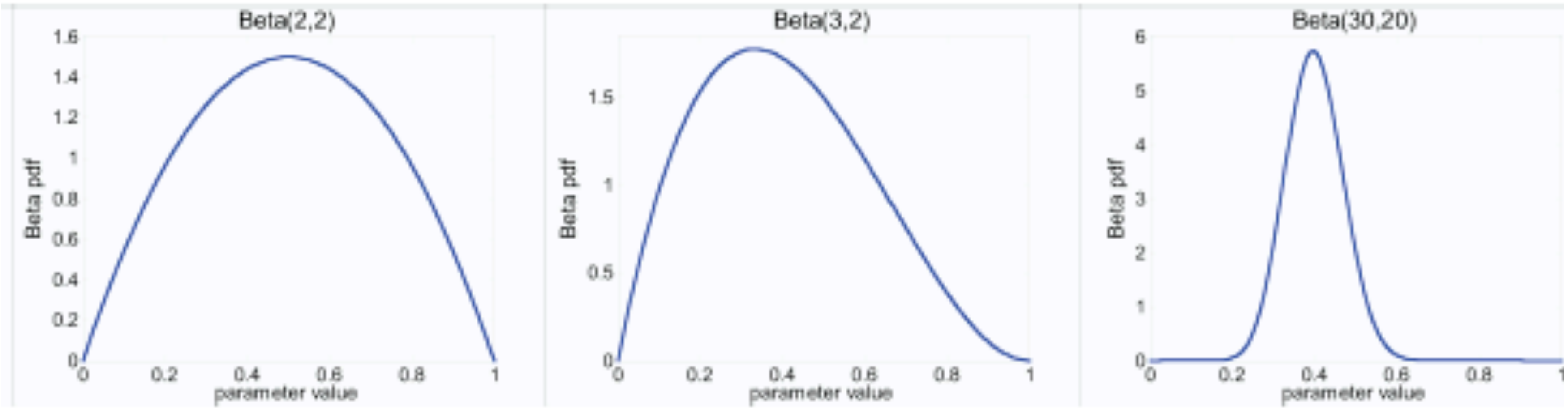
- Likelihood function: $P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- Posterior: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta) P(\theta)$

$$\propto \theta^{\alpha_H + \beta_H - 1}(1-\theta)^{\alpha_T + \beta_T - 1}$$

$$\hat{\theta}^{MAP} = \frac{(\alpha_H + \beta_H - 1)}{(\alpha_H + \beta_H - 1) + (\alpha_T + \beta_T - 1)}$$

Beta prior distribution – P(θ)

- $$P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$



[C. Guestrin]

Eg. 1 Coin flip problem

Likelihood is ~ Binomial

$$P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

If prior is Beta distribution,

$$P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$

Then posterior is Beta distribution

$$P(\theta|D) \sim Beta(\alpha_H + \beta_H, \alpha_H + \beta_H)$$

and MAP estimate is therefore

$$\hat{\theta}^{MAP} = \frac{\alpha_H + \beta_H - 1}{(\alpha_H + \beta_H - 1) + (\alpha_T + \beta_T - 1)}$$

Eg. 2 Dice roll problem (6 outcomes instead of 2)

Likelihood is ~ Multinomial($\theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$)

$$P(\mathcal{D} \mid \theta) = \theta_1^{\alpha_1}\theta_2^{\alpha_2}\ldots\theta_k^{\alpha_k}$$

If prior is Dirichlet distribution,

$$P(\theta) = \frac{\theta_1^{\beta_1-1}\ \theta_2^{\beta_2-1}\ \ldots\theta_k^{\beta_k-1}}{B(\beta_1,\ldots,\beta_k)} \sim \mathrm{Dirichlet}(\beta_1,\ldots,\beta_k)$$

Then posterior is Dirichlet distribution

$$P(\theta|D) \sim \mathsf{Dirichlet}(\beta_1+\alpha_1,\ldots,\beta_k+\alpha_k)$$

and MAP estimate is therefore

$$\hat{\theta}_i^{MAP} = \frac{\alpha_i+\beta_i-1}{\sum_{j=1}^{k}(\alpha_j+\beta_j-1)}$$

Can we reduce params using Bayes Rule?

Suppose X =<X_1,… X_n>
where X_i and Y are boolean RV's

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

How many parameters to define P(X_1,… X_n | Y)?

How many parameters to define P(Y)?

Can we reduce params using Bayes Rule?

Suppose X =<X_1,… X_n>
where X_i and Y are boolean RV's

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

How many parameters to define P(X_1,… X_n | Y)?

P(X|Y=1) ----- 2^n - 1
P(X|Y=0) ----- 2^n - 1

How many parameters to define P(Y)?

Can we reduce params using Bayes Rule?

Suppose X =<X_1,… X_n>
where X_i and Y are boolean RV's

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

how many params for $P(X_1 \ldots X_n|Y)$ $(2^n-1)\cdot 2$

how many for $P(Y) = 1$

Naïve Bayes

Naïve Bayes assumes

$$P(X_1 \ldots X_n | Y) = \prod_i P(X_i | Y)$$

i.e., that X_i and X_j are conditionally independent given Y, for all $i \neq j$

Naïve Bayes uses assumption that the X_i are conditionally independent, given Y

Given this assumption, then:

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$
$$= P(X_1|Y)P(X_2|Y)$$

Chain rule

Cond. Indep.

in general: $P(X_1 ... X_n|Y) = \prod_i P(X_i|Y)$

How many parameters to describe $P(X_1 ... X_n|Y)$? $P(Y)$?

- Without conditional indep assumption? $2(2^n-1)+1$
- With conditional indep assumption? $2n \quad +1$

Naïve Bayes: Subtlety #1

Often the X_i are not really conditionally independent

- We use Naïve Bayes in many cases anyway, and it often works pretty well
  - often the right classification, even when not the right probability (see [Domingos&Pazzani, 1996])
- What is effect on estimated P(Y|X)?
  - Extreme case: what if we add two copies: $X_i = X_k$

Extreme case: what if we add two copies: $X_i = X_k$

Extreme case: what if we add two copies: $X_i = X_k$

$$P(Y=y \mid X) \propto P(Y=y) \prod_i P(X_i = x \mid Y=y)$$

$$P(X_1, \ldots, X_n \mid Y=y)$$

Naïve Bayes: Subtlety #2

If unlucky, our MLE estimate for $P(X_i \mid Y)$ might be zero.
(for example, X_i = *birthdate*. X_i = *Jan_25_1992*)

- Why worry about just one parameter out of many?

- What can be done to address this?

Naïve Bayes: Subtlety #2

If unlucky, our MLE estimate for $P(X_i \mid Y)$ might be zero. (e.g., X_i = Birthday_Is_January_30_1992)

- Why worry about just one parameter out of many?

$$P(Y|X) \propto P(Y) \prod_i \underbrace{P(X_i = x^{New} | Y)}_{0}$$

- What can be done to address this?