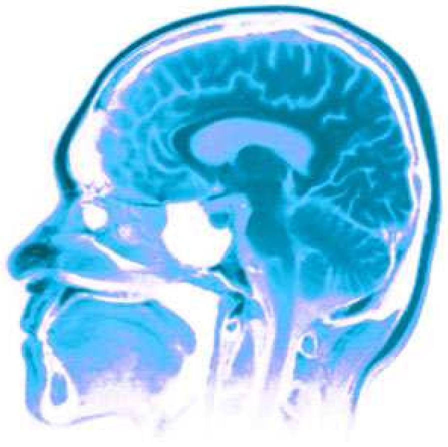
CPSC540



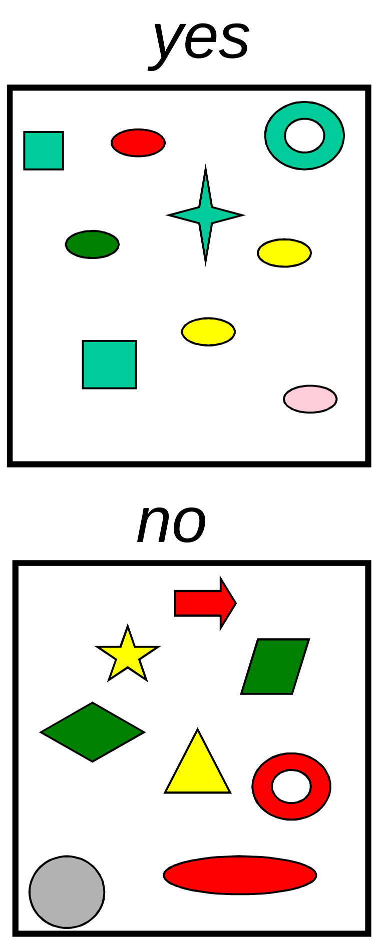
Classification

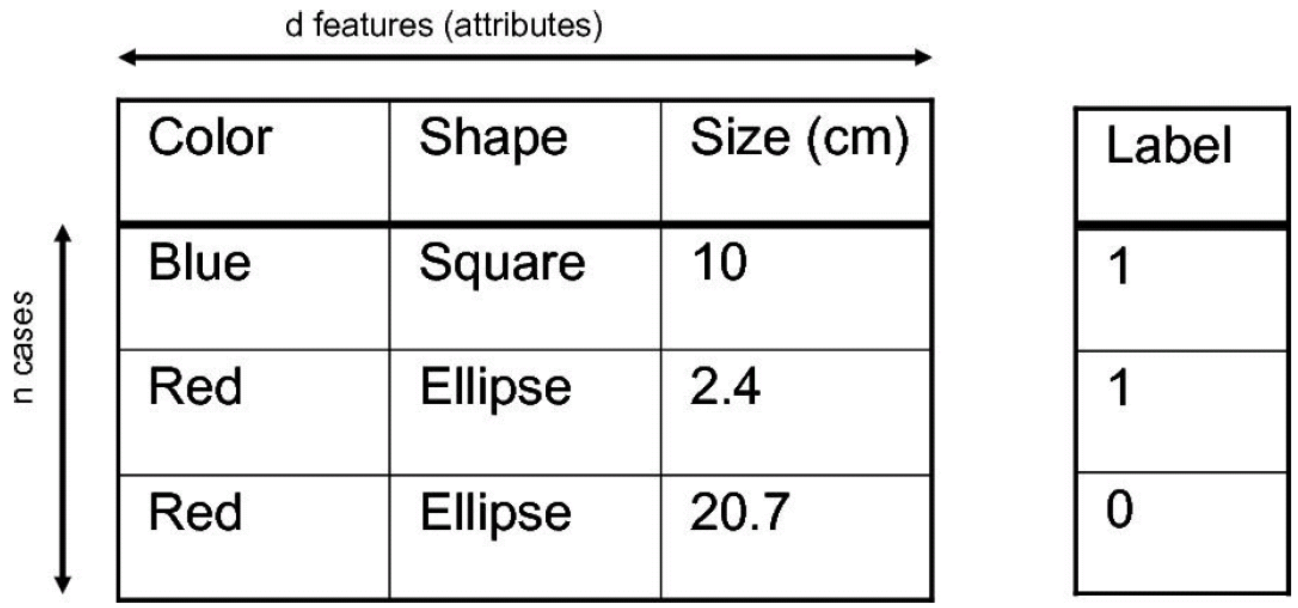
Nando de Freitas
2011
KPM Book Sections: *1.1, 1.2, 2.3.1 and 2.3.2*

Supervised Learning

- In the **predictive** or **supervised learning** approach, the goal is to learn a mapping from inputs $\mathbf{x}$ to outputs y, given a labeled set of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$.
- Here $\mathcal{D}$ is called the **training set**, and N is the number of training examples.
- The form of the inputs can in principle be anything, but most methods assume that $\mathbf{x}_i$ is a fixed-length vector of **features** (also called **attributes** or **covariates**), such as the height and weight of a person.
- Similarly the form of the output or **response variable** can in principle be anything, but most methods assume that y_i is a **categorical** or **nominal** variable from some finite set, $y_i \in \{1, \ldots, C\}$ (such as male or female), or that y_i is a real-valued scalar (such as income level).
- When y_i is categorical, the problem is known as **classification** and when y_i is real-valued, the problem is known as **regression**.

Classification hypothesis: decision tree

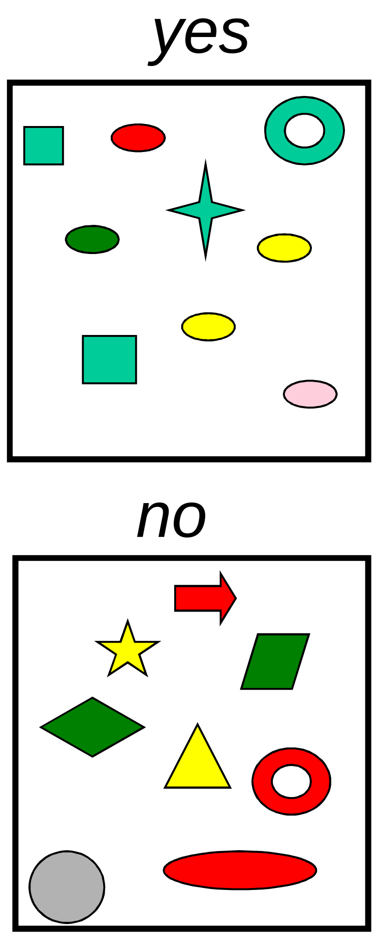


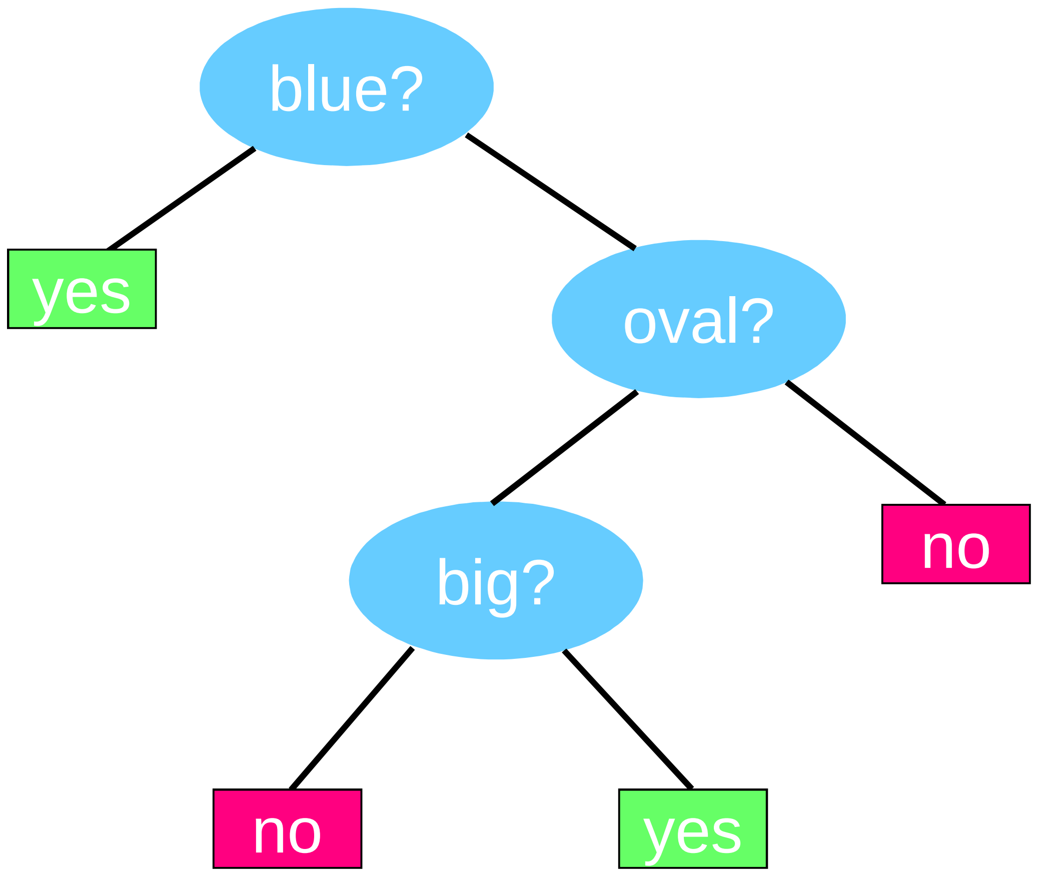

d features (attributes)

n cases

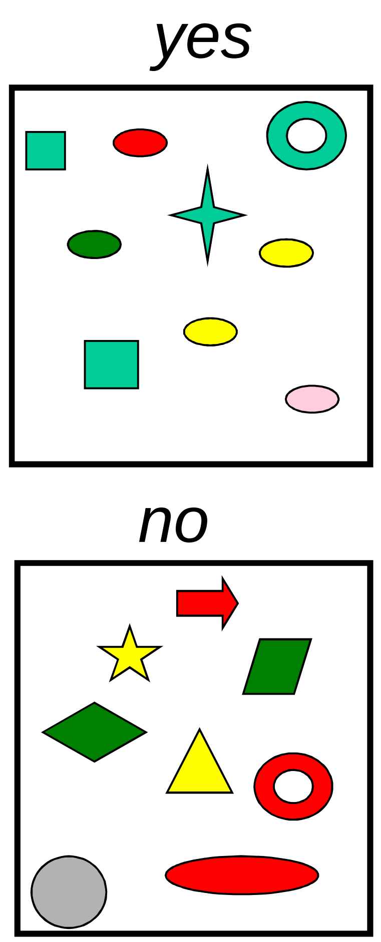
Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

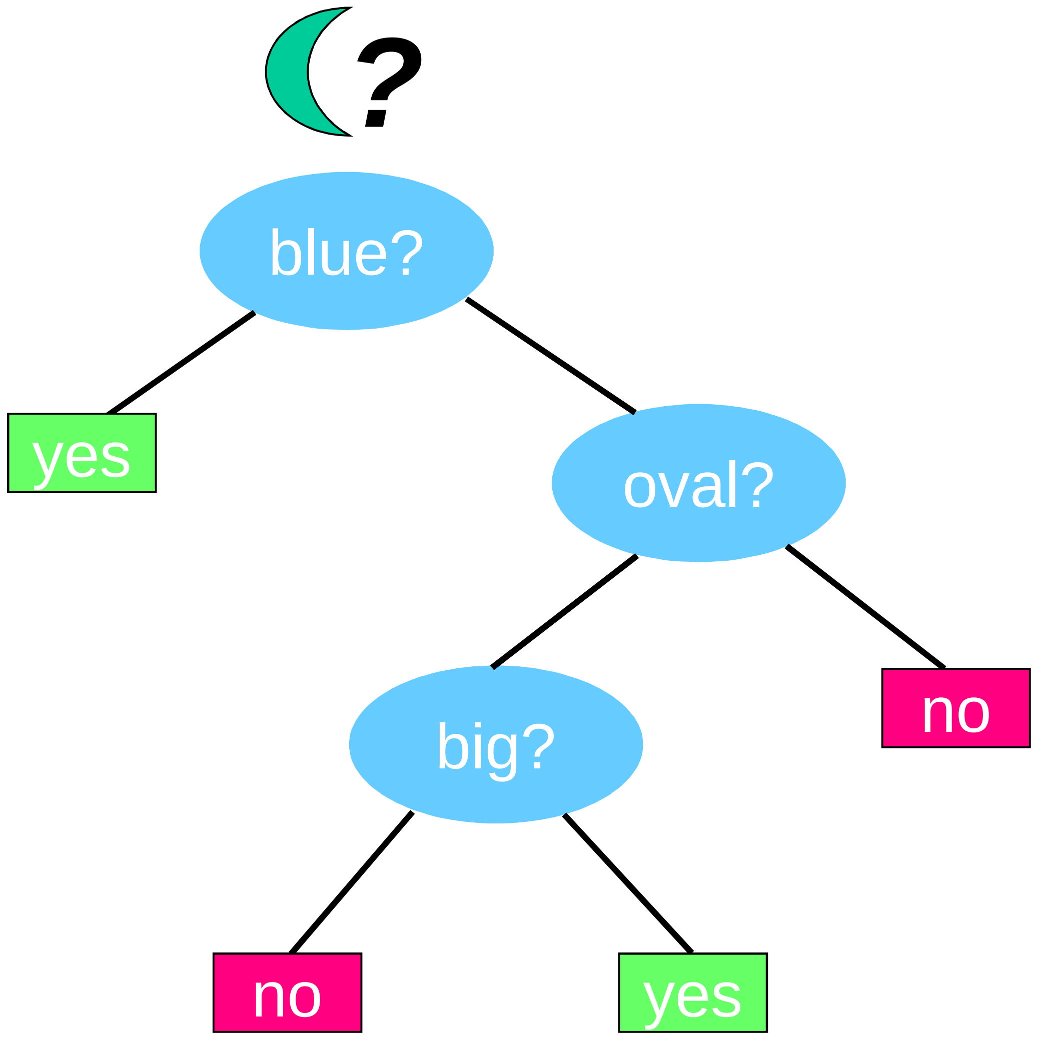
Classification hypothesis: decision tree



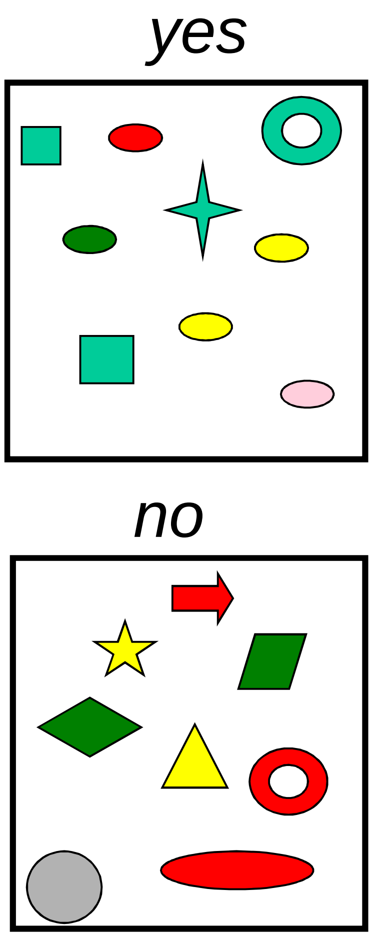


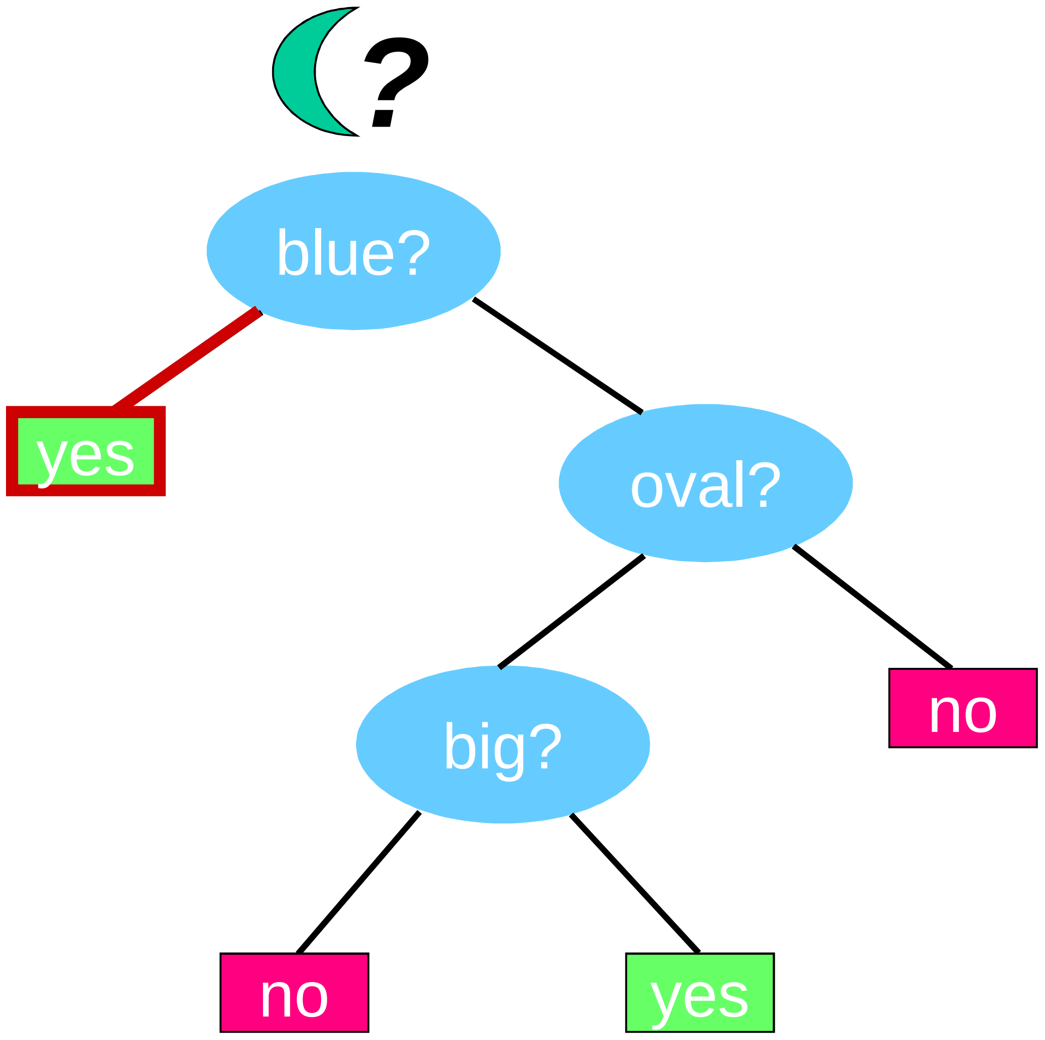
Classification hypothesis: decision tree



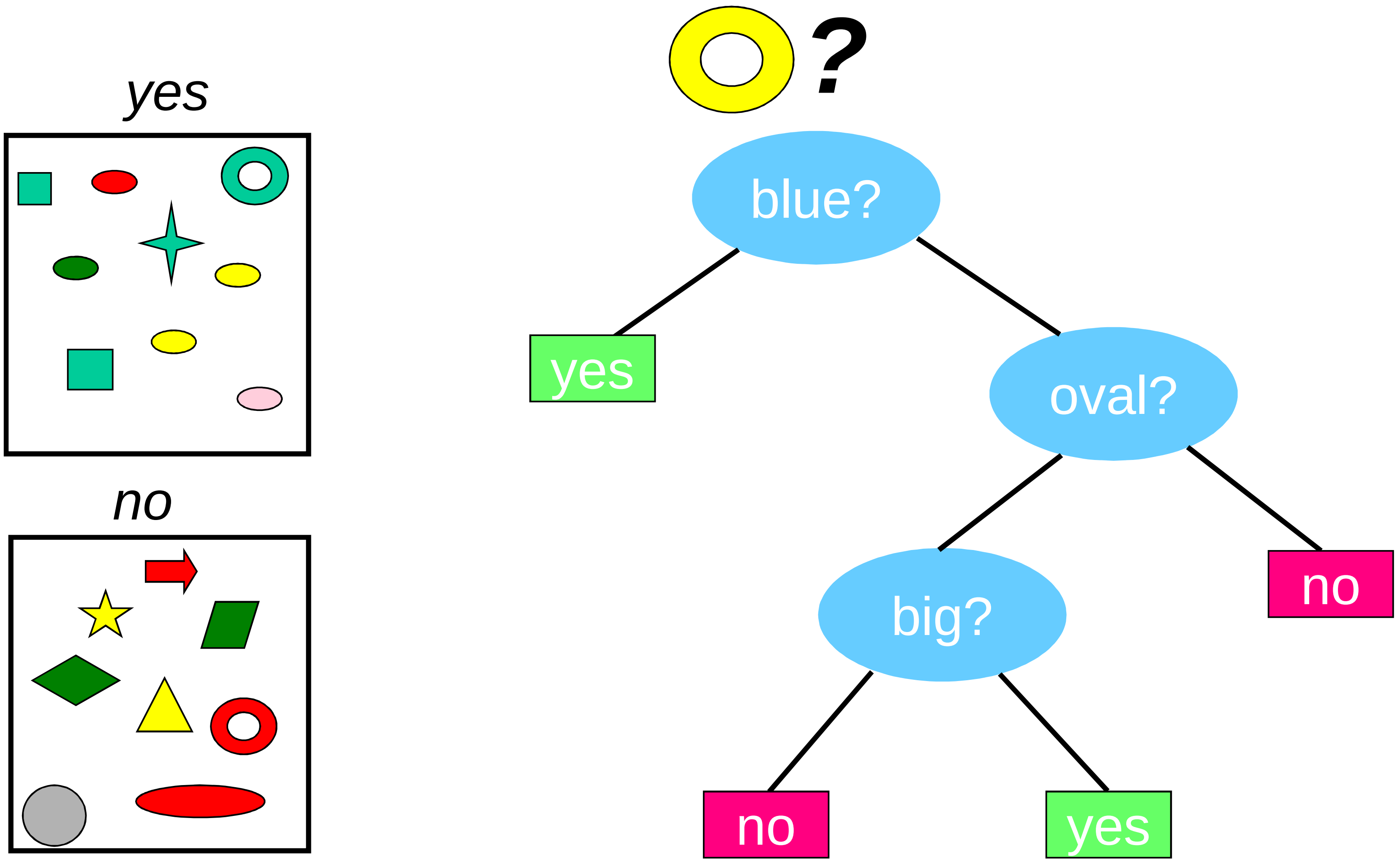


Classification hypothesis: decision tree

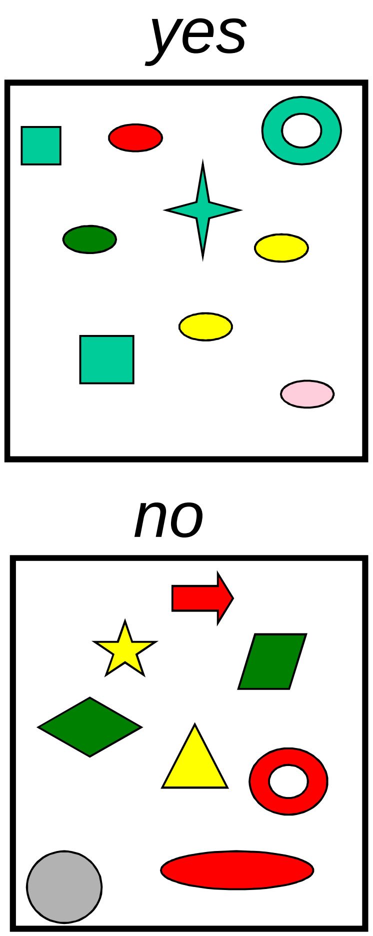


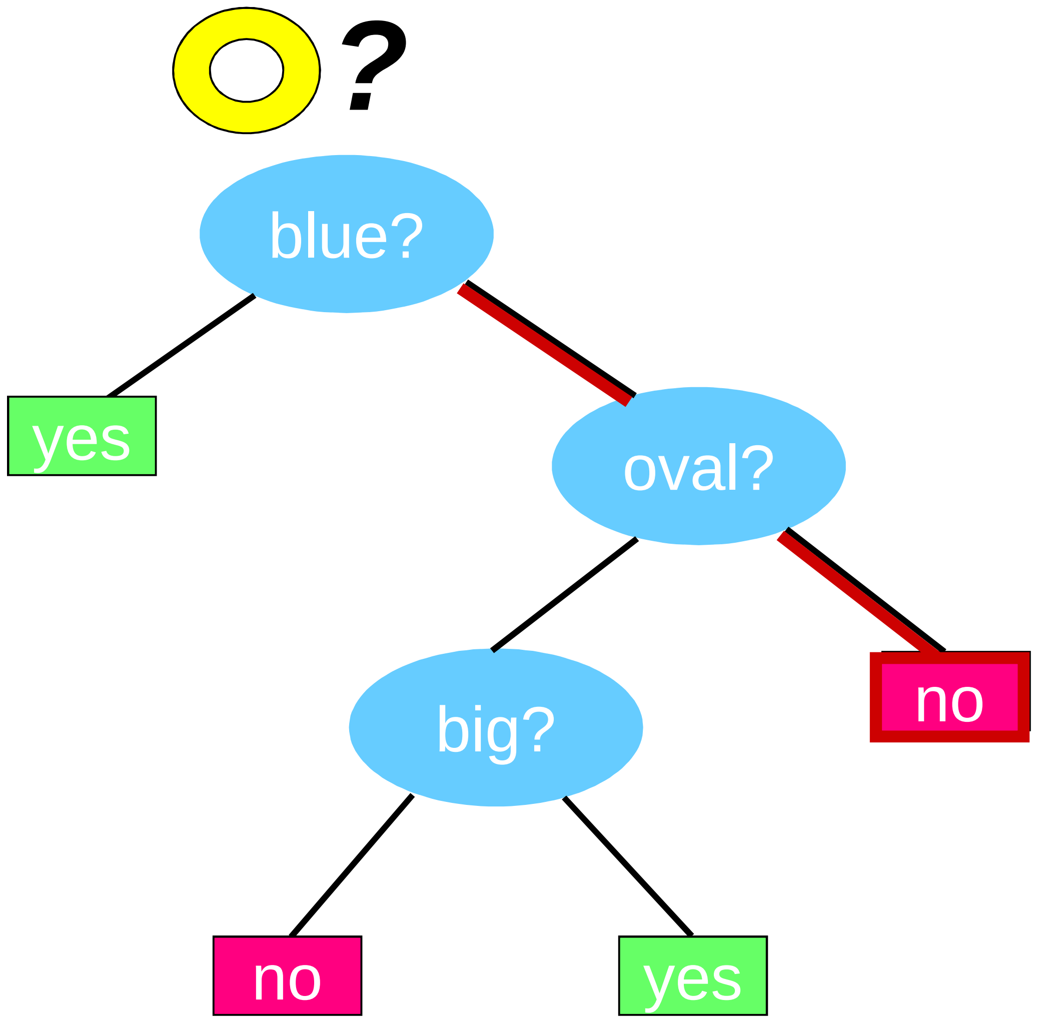


Classification hypothesis: decision tree



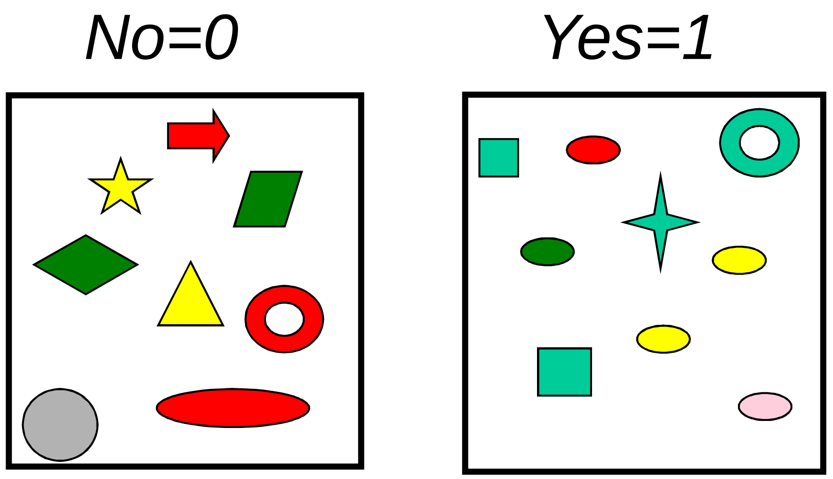
Classification hypothesis: decision tree

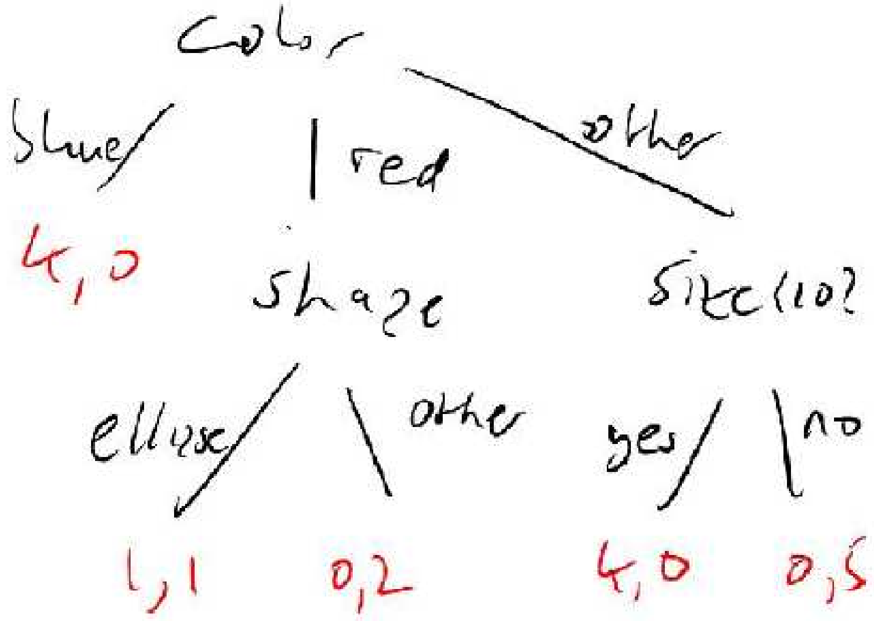




Probabilistic classification

- We first check the color of the object.
- If it is blue, we predict $p(y = 1|\mathbf{x}) = 4/4$;
- if it is red, we then check the shape: if it is an ellipse, we predict $p(y = 1|\mathbf{x}) = 1/2$, otherwise we predict $p(y = 1|\mathbf{x}) = 0/2$;
- if it is some other colour, we check the size: if less than 10, we predict $p(y = 1|\mathbf{x}) = 4/4$, otherwise $p(y = 1|\mathbf{x}) = 0/5$.





Probabilistic Classification

- With $p(y|\mathbf{x}, \mathcal{D})$, we explicitly denote that our probabilities are conditional on the test input $\mathbf{x}$, and the training set $\mathcal{D}$, by putting these terms on the right hand side of the conditioning bar $|$.

- We are also implicitly conditioning on the form of model that we use to make predictions.

- When choosing between different models, we will make this assumption explicit by writing $p(y|\mathbf{x}, \mathcal{D}, M)$, where M denotes the model.

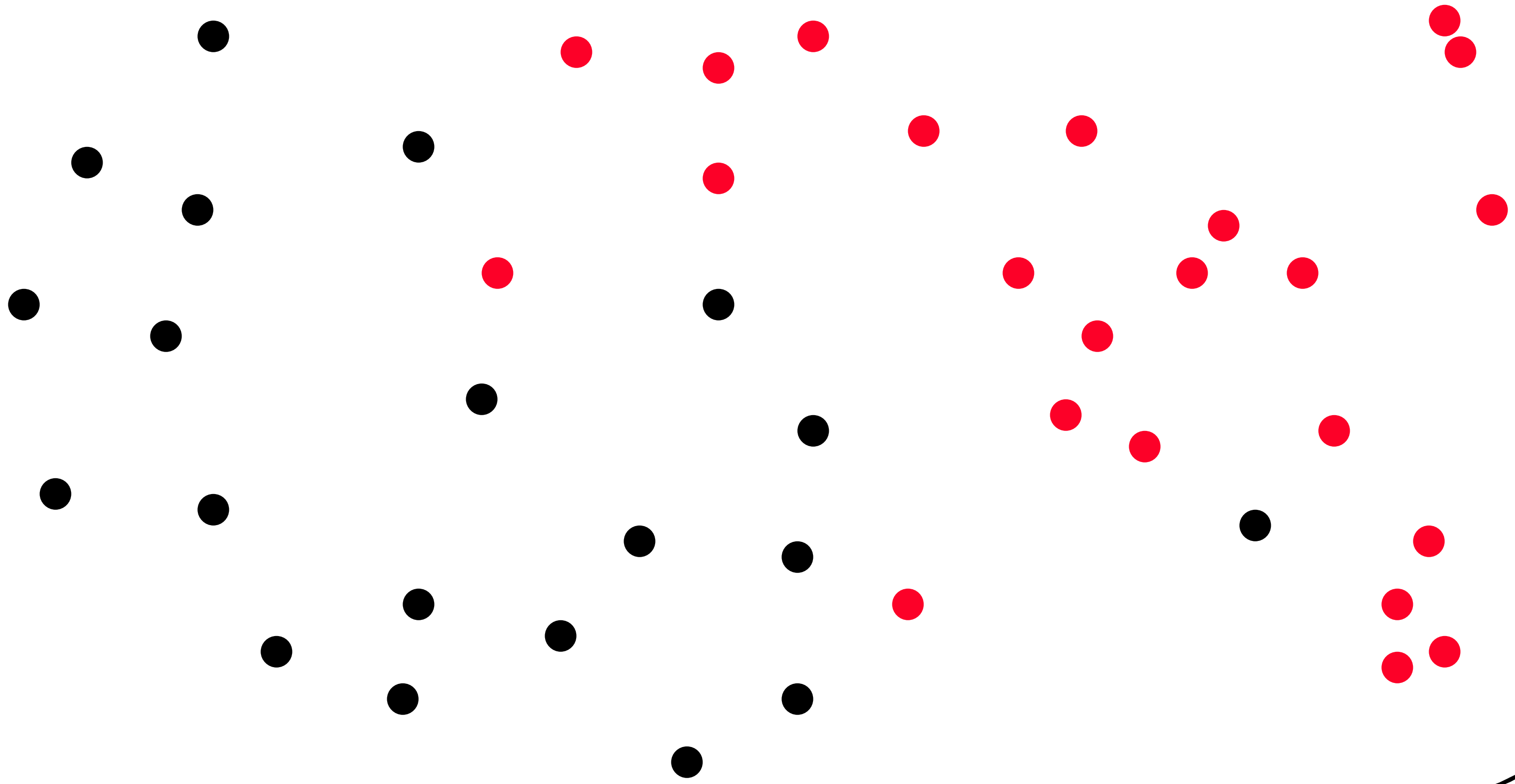
- Given a probabilistic output, we can always compute our "best guess" as to the "true label" using

$$\hat{y} = \hat{f}(\mathbf{x}) = \arg\max_{c=1}^{C} p(y = c|\mathbf{x}, \mathcal{D}) \tag{1}$$

- This corresponds to the most probable class label, and is called the **mode** of the distribution $p(y|\mathbf{x}, \mathcal{D})$; it is also known as a **MAP estimate** (MAP stands for **maximum a posteriori**).

$$p(y=0|\mathbf{x},\mathcal{D},K=3)=\frac{1}{3}\sum_{i\in N_3(\mathbf{x},\mathcal{D})}\mathbb{I}(y_i=0)$$

$$p(y=1|\mathbf{x},\mathcal{D},K=3)=\frac{1}{3}\sum_{i\in N_3(\mathbf{x},\mathcal{D})}\mathbb{I}(y_i=1)$$



K-nearest neighbor classifier

- Another example of a simple classifier, which is arguably better-suited to real-valued inputs than a decision tree, is the K **nearest neighbor** (**KNN**) classifier. This simply "looks at" the K points in the training set that are nearest to the test input $\mathbf{x}$, counts how many members of each class are in this set, and returns that empirical fraction as the estimate. More formally,

$$p(y = c|\mathbf{x}, \mathcal{D}, K) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x}, \mathcal{D})} \mathbb{I}(y_i = c)$$

where $N_K(\mathbf{x}, \mathcal{D})$ are the (indices of the) K nearest points to $\mathbf{x}$ in $\mathcal{D}$ and $\mathbb{I}(e)$ is the **indicator function** defined as follows:

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

This method is an example of **memory-based learning** or **instance-based learning**.

K-nearest neighbor classifier

- If the features are real-valued, a standard way to measure distance between two feature vectors is **Euclidean distance**,

$$d(\mathbf{x}, \tilde{\mathbf{x}}) = ||\mathbf{x} - \tilde{\mathbf{x}}||_2 = \sqrt{\sum_{j=1}^{D} (x_j - \tilde{x}_j)^2}$$

- When using Euclidean distance, we are assuming all the features have the same scale. Consequently it is common to first **standardize** the data, which means ensuring it has zero mean and unit variance. We can do this by computing

$$z_{ij} = \frac{x_{ij} - \overline{x}_j}{\sigma_j}$$

where $\overline{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$ is the empirical mean of the j'th feature, and $\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \overline{x}_j)^2$ is the empirical variance.

K-nearest neighbor classifier

- If the features are discrete, a standard way to measure distance is **Hamming distance**, defined as

$$d(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{j=1}^{D} \mathbb{I}(x_j \neq \tilde{x}_j)$$

  which counts the number of features that differ between the two examples.

- If the features are both discrete and continuous, we can define a mixed distance measure, such as

$$d(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbb{I}(x_1 \neq \tilde{x}_1) + \mathbb{I}(x_2 \neq \tilde{x}_2) + \sqrt{(x_3 - \tilde{x}_3)^2}$$

  where x_1 represents color, x_2 represents shape, and x_3 represents size.

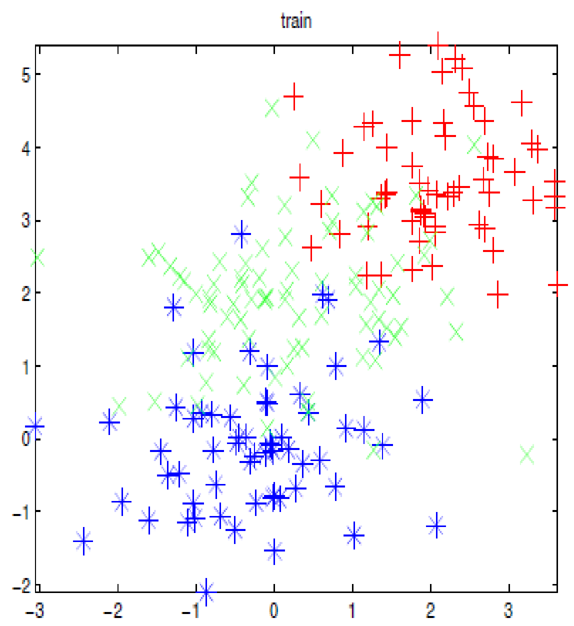
K-nearest neighbor classifier

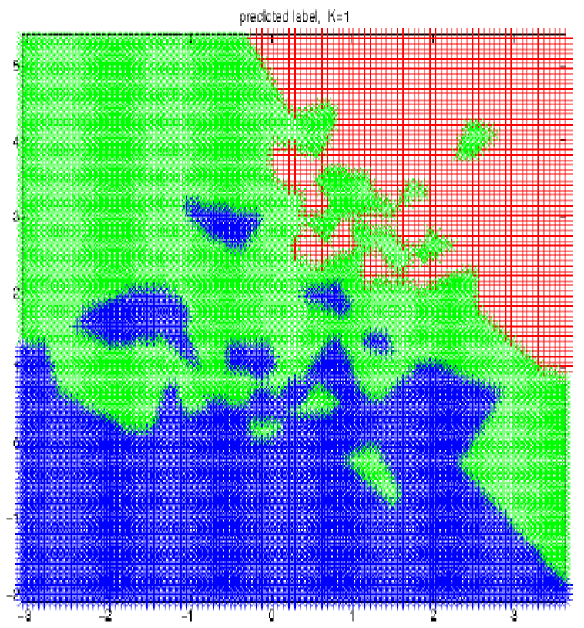
- We are free to associate weights with each dimension, if some features are more important than others. We can do this by using the following weighted distance metric:

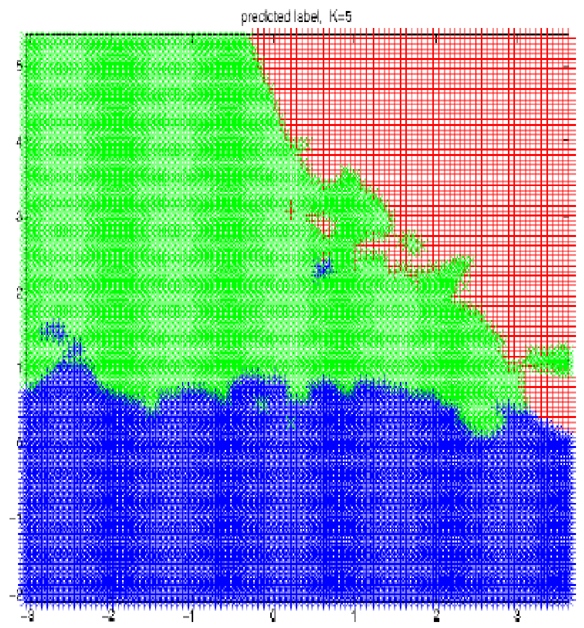
$$d(\mathbf{x}, \tilde{\mathbf{x}} | \mathbf{w}) = \sum_{j=1}^{D} w_j d_j(x_j, \tilde{x}_j)$$

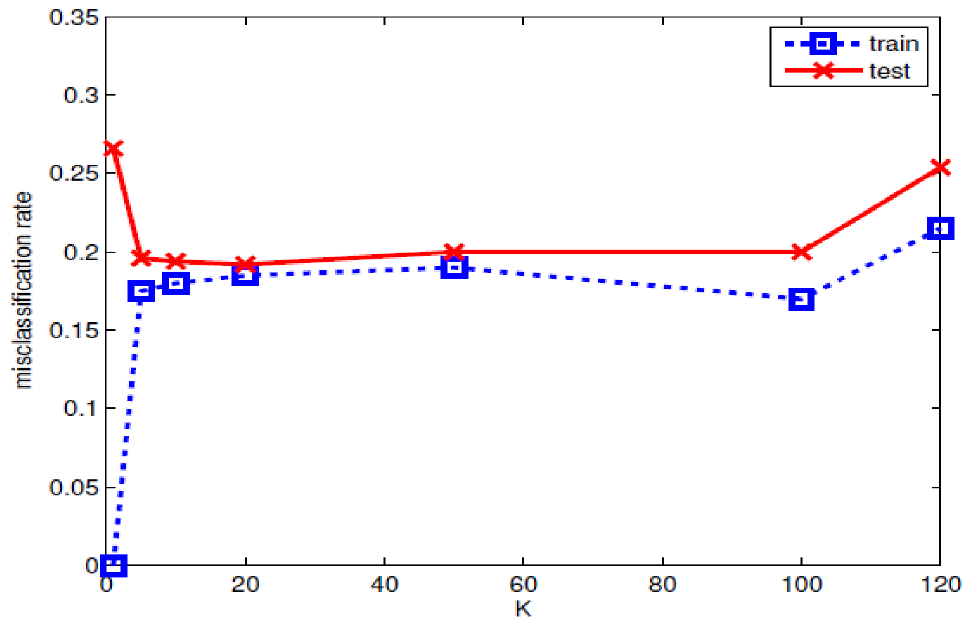
  where d_j is a distance measure appropriate for the attribute j. By setting some weights to zero, we can ignore certain attributes when comparing objects; this is known as **feature selection**.

K-nearest neighbor classifier







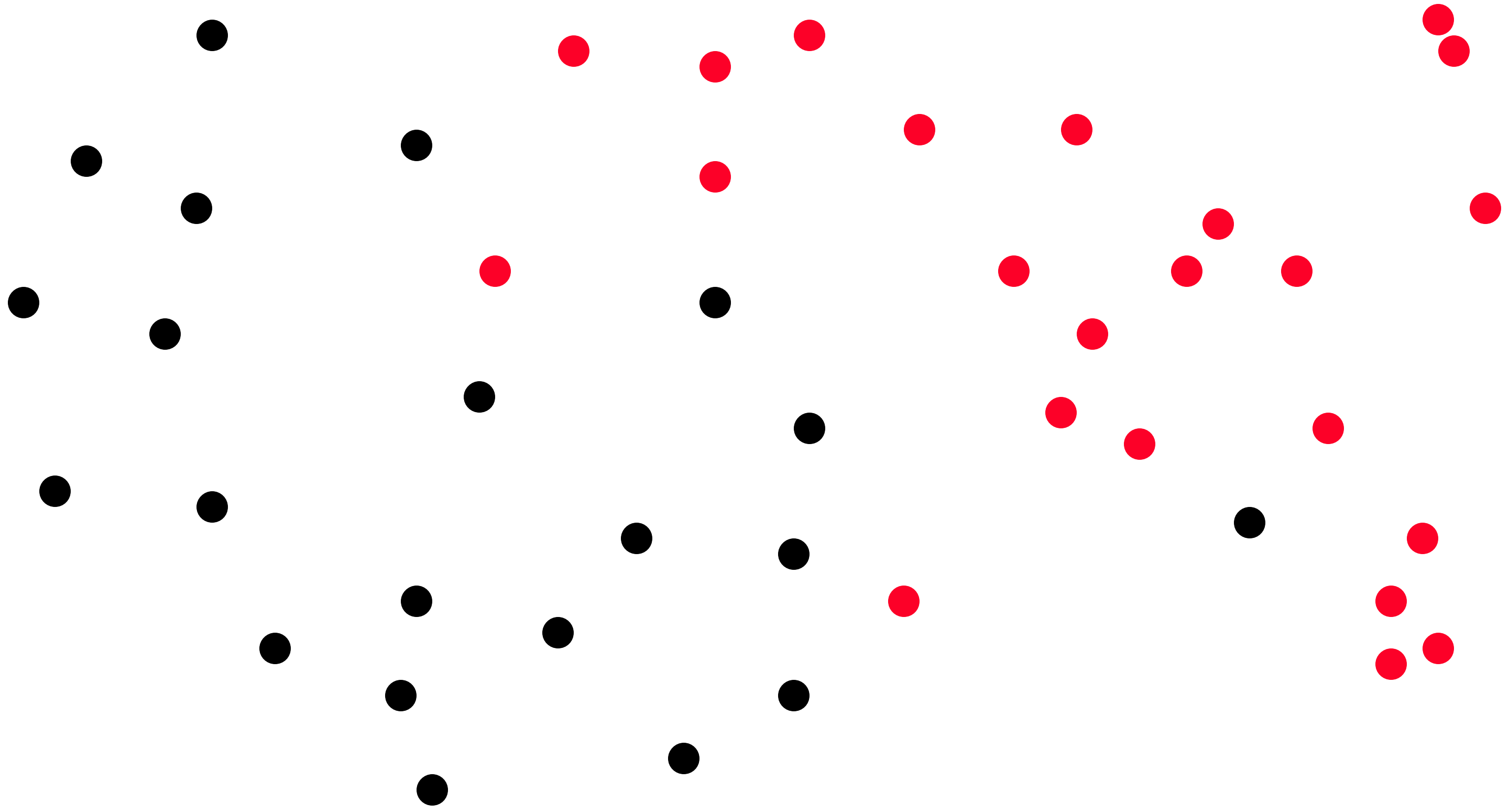


Parametric vs non-parametric

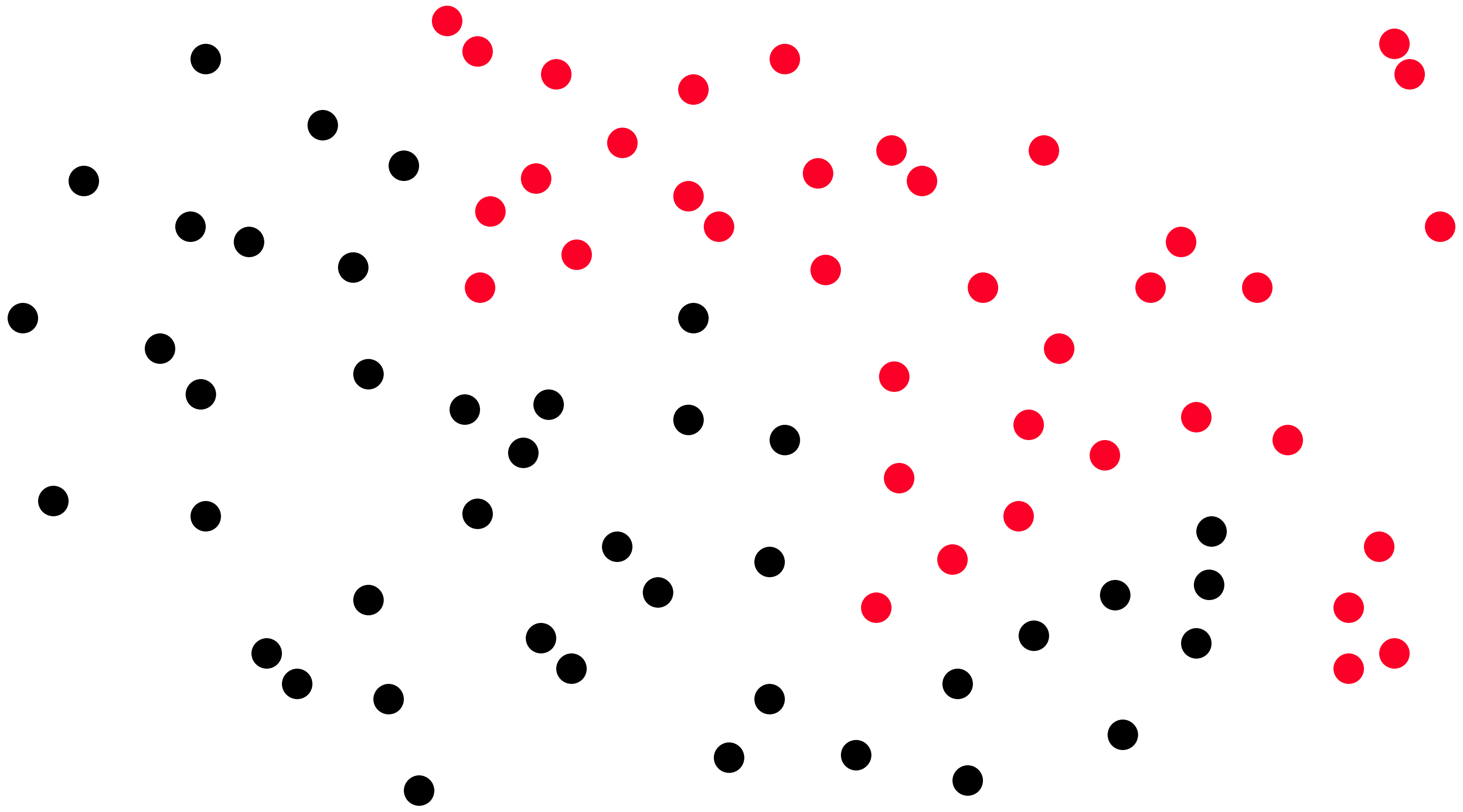
- A KNN classifier is an example of a **non-parametric model**, This does not mean the model has "no parameters", since it clearly does (namely K, the parameters inside the distance metric, and all the training data). Rather, "non-parametric" means (roughly speaking) that the number of parameters can grow with the amount of training data.

- By contrast, many popular methods for classification are based on **parametric models**, that have a number of parameters that is fixed ahead of time. The data is then used to estimate these parameters; this is called "learning" or "model fitting".

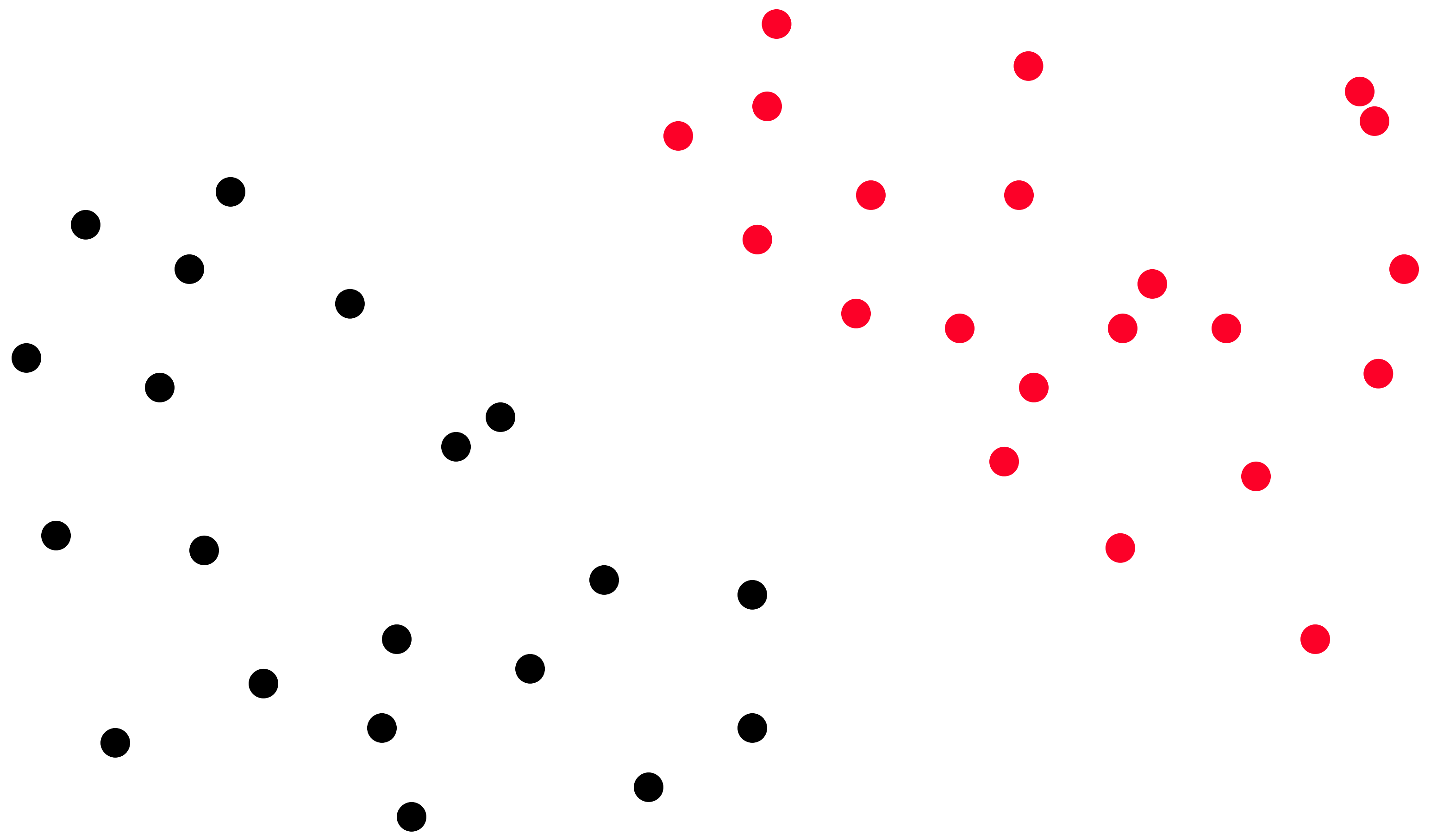
What discriminant curve best separates these two classes?



Next day, we get more data and a surprise!



If the discriminant is a line, what is the best line?



Binomial and Bernoulli r.v.s

- Suppose we toss a coin n times. Let $X \in \{0, \ldots, n\}$ be the number of heads. If the probability of heads is θ, then we say X has a **Binomial** distribution, written as $X \sim \mathrm{Bin}(n, \theta)$. The pmf is given by

$$\mathrm{Bin}(x|n,\theta) := \binom{n}{x} \theta^x (1-\theta)^{n-x}, \quad \text{where} \quad \binom{n}{x} := \frac{n!}{(n-x)!x!}$$

- Since this is a pmf, we have $0 \leq p(x|n,\theta) \leq 1$ and $\sum_{x=0}^{n} p(x|n,\theta) = 1$. One can easily show that the mean of this distribution is $\mathbb{E}\left[X\right] = n\theta$, and the variance is $\mathrm{var}\left[X\right] = \theta(1-\theta)$.

- Let $X \in \{0, 1\}$ be a binary random variable, with probability of "success" or "heads" of θ. We say that X has **Bernoulli** distribution. This is written as $X \sim \mathrm{Ber}(\theta)$, where the pmf is defined as

$$\mathrm{Ber}(x|\theta) = \theta^{\mathbb{I}(x=1)}(1-\theta)^{\mathbb{I}(x=0)}$$

  In other words,

$$\mathrm{Ber}(x|\theta) = \begin{cases} \theta & \text{if } x = 1 \\ 1-\theta & \text{if } x = 0 \end{cases}$$

Logistic regression

- Logistic regression is a model that specifies the probability of the output given the input as follows:

$$p(y|\mathbf{x}, \mathbf{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^T\mathbf{x}))$$

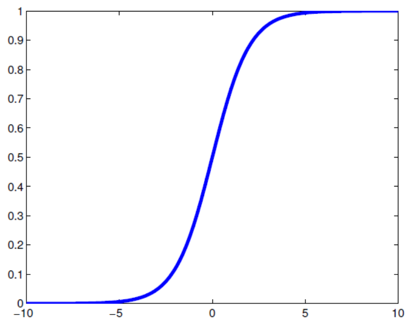
- The notation $\mathbf{w}^T\mathbf{x}$ refers to the scalar (inner) product

$$\mathbf{w}^T\mathbf{x} = w_0 + \sum_{j=1}^{D} w_j x_j$$

  $\mathbf{x} = (1, x_1, \ldots, x_D)$. w_0 is called an **offset** or **bias** term, and encodes the baseline probability that y is on even if there are no other features.

- $\mathrm{sigm}(\eta)$ refers to the **sigmoid** function, also known as the **logistic** or **logit** function, defined as

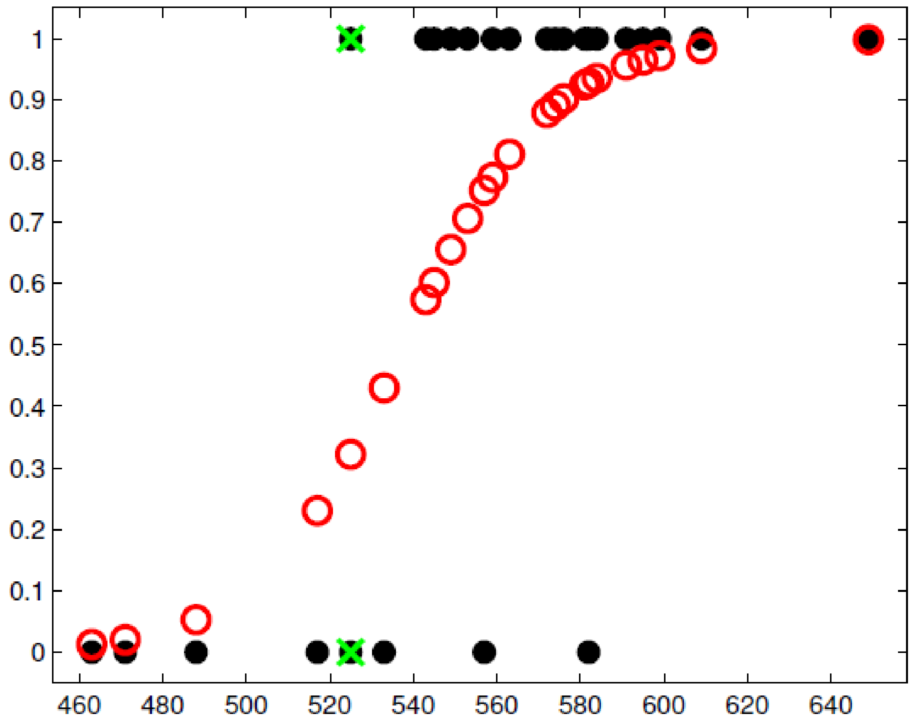
$$\mathrm{sigm}(\eta) := \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}$$



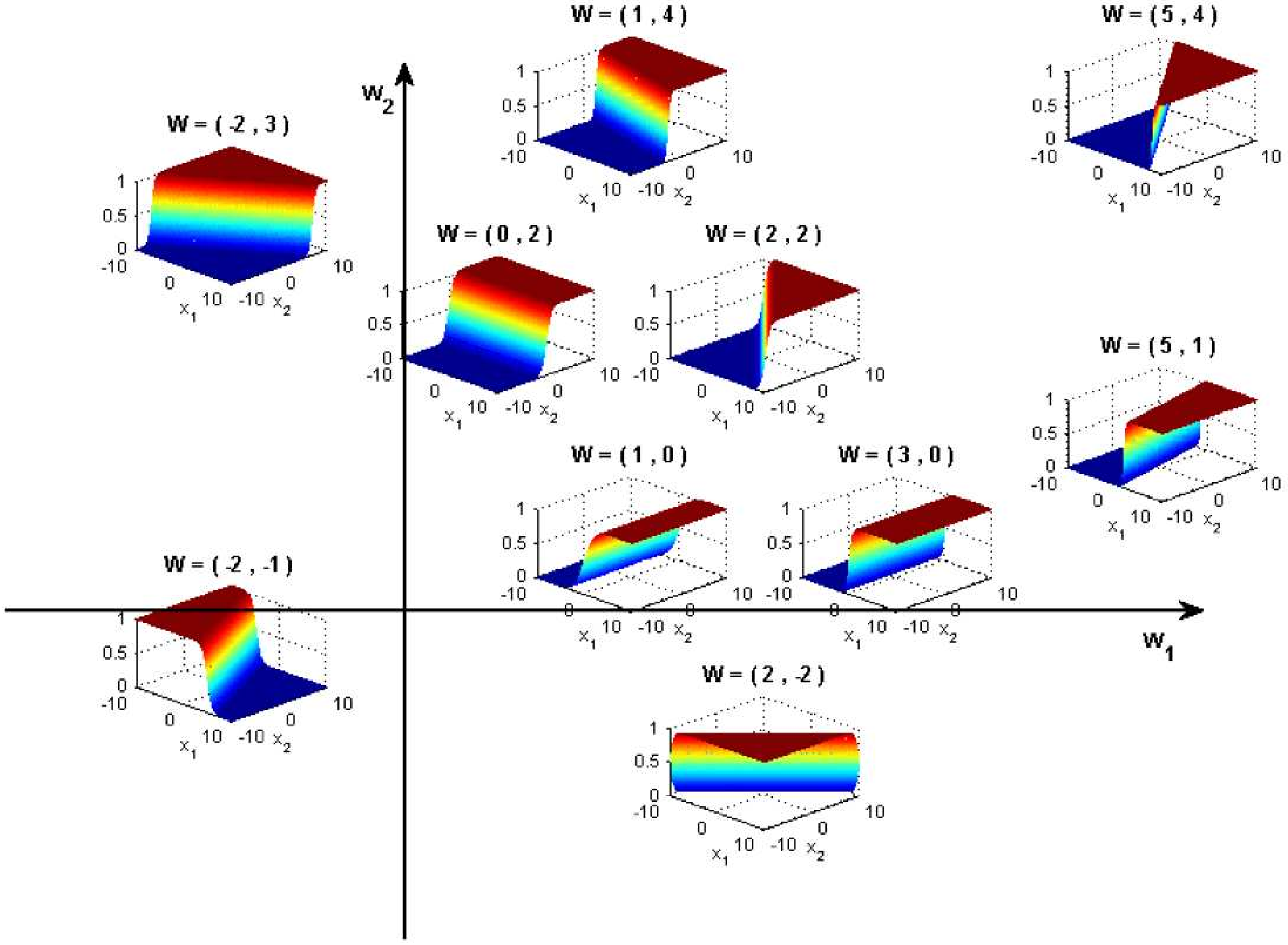
Logistic regression

- $$p(y_i = 1|x_i, \mathbf{w}) = \text{sigm}(w_0 + w_1 x_i)$$

x_i is the "Scholastic Aptitude Test" (SAT) score of student i and y_i is whether they passed or failed a class. The solid black dots show the training data, and the red circles plot $p(y = 1|\mathbf{x}_i, \hat{\mathbf{w}})$, where $\hat{\mathbf{w}}$ are the parameters estimated from the training data. See `logregSATdemo.m`



Logistic regression



Logistic regression - neuroscience

- One motivation comes from neuroscience. In the 1950s, McCulloch and Pitts made a simple model of how a **neuron** works. They proposed that a neuron forms a weighted sum of its inputs (coming in along its **dendrites**) and then "fires" an output pulse (along its **axon**) if the weighed sum of inputs exceed a threshold. That is, "fire" if $p(y=1|\mathbf{x}) > p(y=0|\mathbf{x})$.

- To see this, consider the **log odds ratio**, defined as follows:

$$\mathrm{LOR}(\mathbf{x}) := \log \frac{p(y=1|\mathbf{x},\mathbf{w})}{p(y=0|\mathbf{x},\mathbf{w})}$$

For a logistic regression model, the log odds ratio has the following form:

$$\mathrm{LOR}(\mathbf{x}) = \log \left[\frac{e^\eta}{1+e^\eta} \frac{1+e^\eta}{1} \right] = \log e^\eta = \eta = \mathbf{w}^T\mathbf{x}$$

So the neuron fires iff $\mathrm{LOR}(\mathbf{x}) = w_0 + \sum_{j=1}^D w_j x_j > 0$

Logistic regression – categorical inputs

- The standard approach to handling categorical inputs is to re-code such features using a 1-of-K vector, where K is the number of categories. This is sometimes called a **dummy variable**, a **factor**, a **1-of-K encoding**, or a **one-hot encoding** (since only one “wire” is “hot” or “on” at a time).

- For example, $x \in \{r, g, b\}$ can be encoded as a bit vector of length 3:

$$\boldsymbol{\phi}(x) = (\mathbb{I}(x = r), \mathbb{I}(x = g), \mathbb{I}(x = b))$$

  so $\boldsymbol{\phi}(g) = [0, 1, 0]$, etc. This maps $\{r, g, b\}$ to $\{0, 1\}^3$. We now use $\boldsymbol{\phi}(x_{ij})$ instead of x_{ij}.

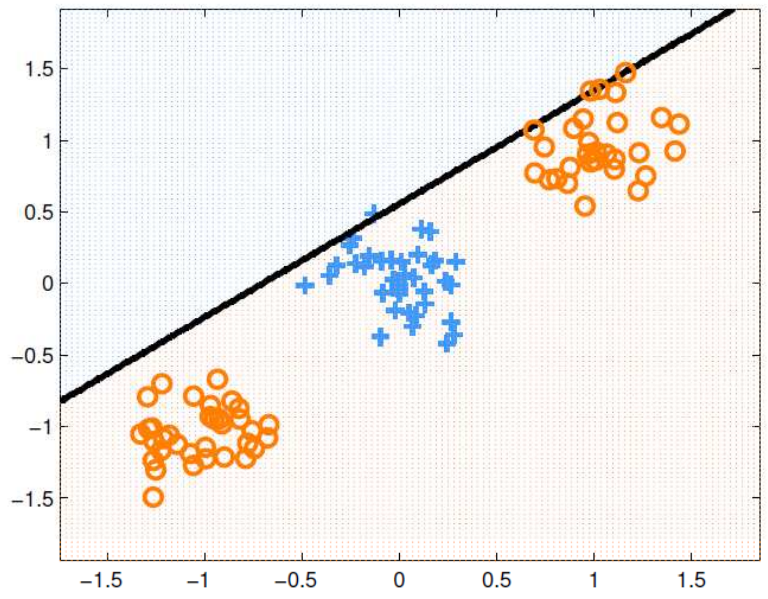
Logistic regression – decision boundary

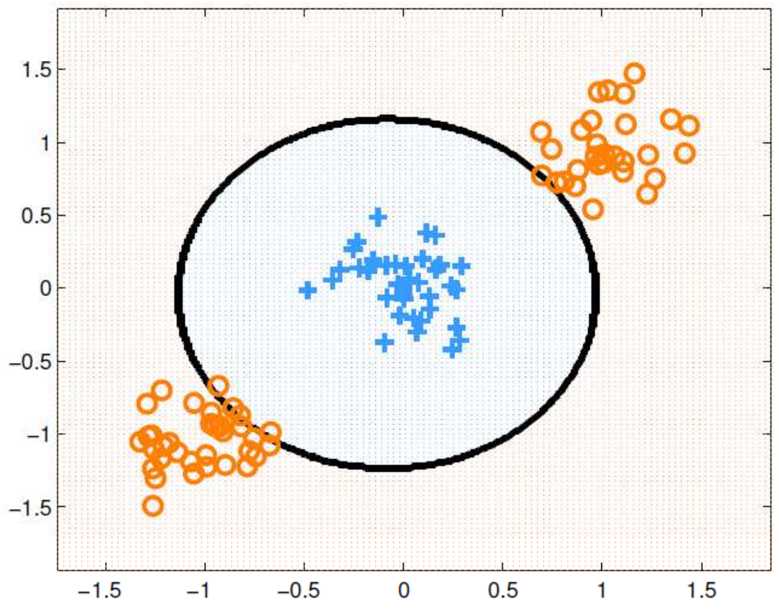
- Logistic regression essentially partitions the input space into two regions: those for which $\mathrm{LOR}(\mathbf{x}) < 0$ and those for which $\mathrm{LOR}(\mathbf{x}) > 0$.

- The point that separates these two regions is called the **decision boundary**, i.e., the set $\{\mathbf{x} : \mathrm{LOR}(\mathbf{x}) = 0\}$.

- In 1d, the decision boundary is a single point, where $x^* = -\frac{w_0}{w_1}$. The value of w_0/w_1 determines the location of the threshold, and the magnitude of w_1 determines the "steepness" of the sigmoid function, that is, the sensitivity of the response to changes in x.

Logistic regression





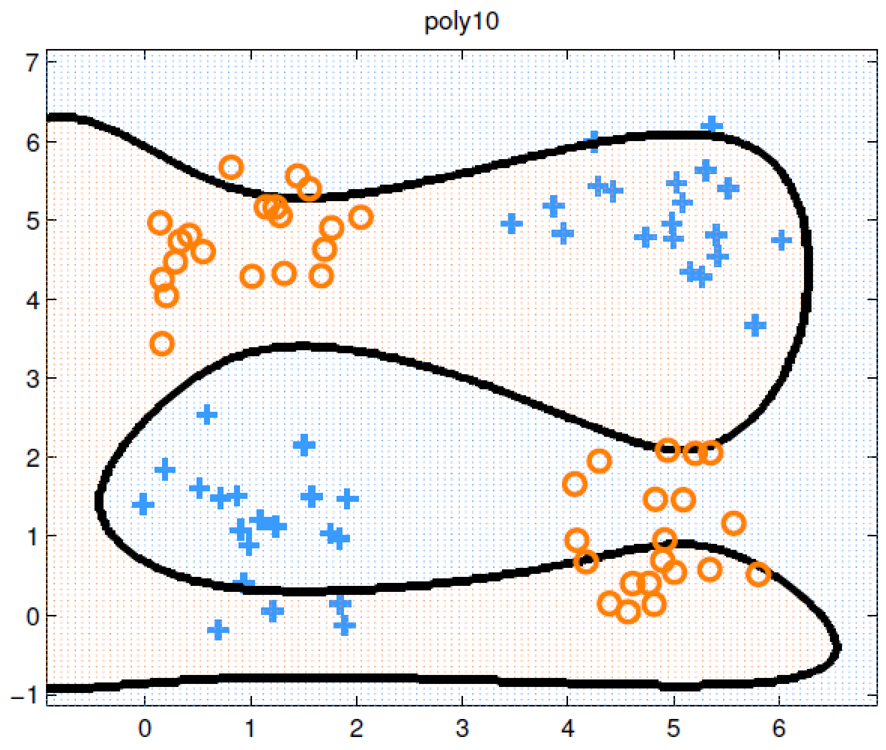
- There is an easy way to make linear models represent non-linear functions, called **basis function expansion**. The idea is that we replace the original features $\mathbf{x}$ by some (fixed) non-linear function $\boldsymbol{\phi}(\mathbf{x})$, and then use $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$ instead of $\mathbf{w}^T\mathbf{x}$.

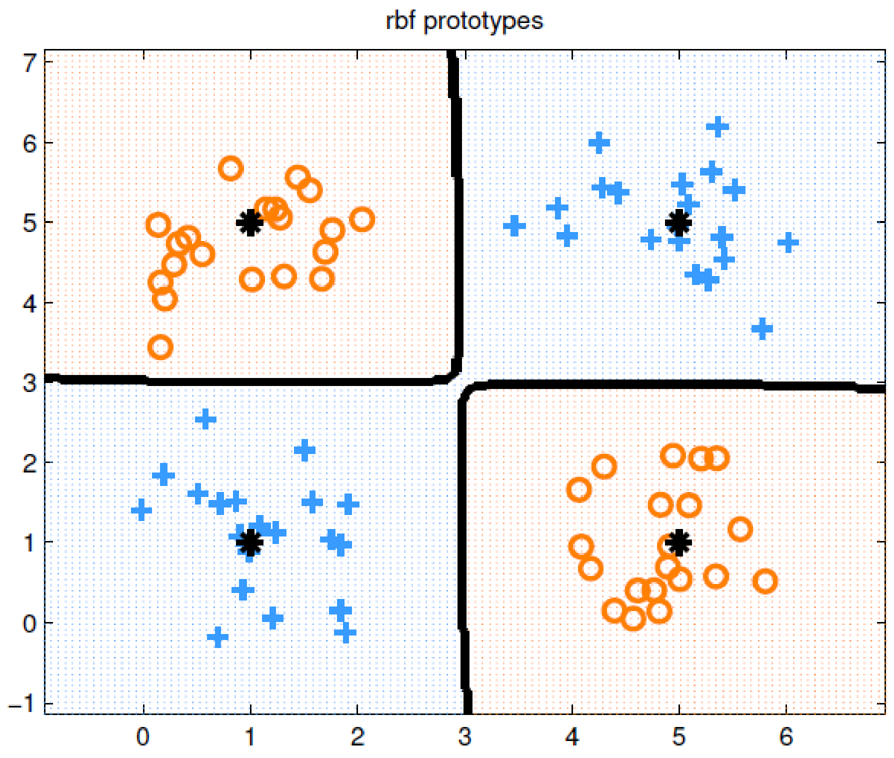
- A simple example of basis function expansion is to use a polynomial of degree d:

$$\boldsymbol{\phi}(x) = [1, x^1, x^2, \ldots, x^d]$$

See `logregBasisFnDemo.m`

Logistic regression





- See `logregXorDemo.m`

Logistic regression

- Another kind of basis function expansion is based on **radial basis functions** (**RBF**), which have the form

$$\boldsymbol{\phi}(\mathbf{x}) = [\kappa(\mathbf{x}, \boldsymbol{\mu}_1), \ldots, \kappa(\mathbf{x}, \boldsymbol{\mu}_{D'})]$$

  where

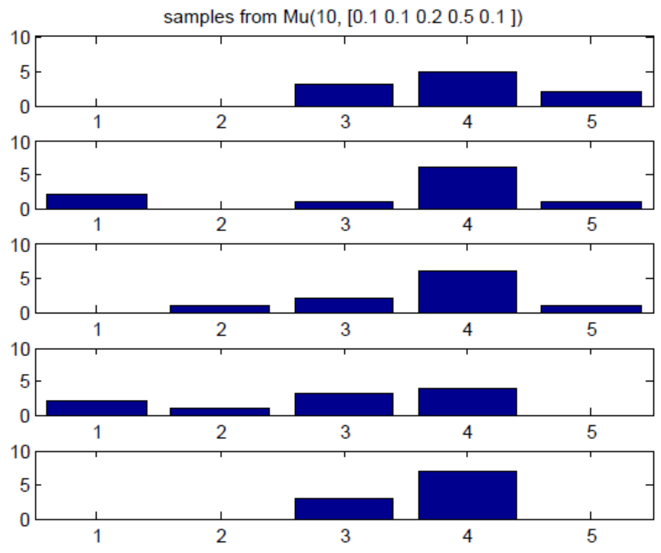
$$\kappa(\mathbf{x}, \boldsymbol{\mu}_k) = \exp(-\frac{1}{2\sigma^2}||\boldsymbol{\mu}_k - \mathbf{x}||^2))$$

  The $\boldsymbol{\mu}_k$ are prototypes or exemplars, and σ^2 is known as the **bandwidth**.

- The quantity $\kappa(\mathbf{x}, \boldsymbol{\mu}_k) \geq 0$ is called a **kernel function**; it measures the similarity between $\mathbf{x}$ and $\boldsymbol{\mu}_k$, where similar objects are defined to be ones that are close in Euclidean distance in the original feature space.

- Later, we discuss more general kinds of kernel functions, which allow us to measure the similarity between structured objects such as strings (sequences of characters), trees, molecular structures, etc.

Multinomial distribution



- The multivariate version of a binomial is called a **multinomial** distribution. As an example, suppose we have a dice with K sides/ faces. Let the probability that we roll face j be θ_j. Suppose we roll the dice n times in total. Let $\mathbf{x} = (x_1, \ldots, x_K)$ be a random vector, where x_j is the number of times face j occurs. Then $\mathbf{x}$ has the following pmf:

$$\mathrm{Mu}(\mathbf{x}|n, \boldsymbol{\theta}) := \binom{n}{x_1 \ldots x_K} \prod_{j=1}^{K} \theta_j^{x_j}, \quad \text{where} \quad \binom{n}{x_1 \ldots x_K} := \frac{n!}{x_1! x_2! \cdots x_K!}$$

and $n = \sum_{k=1}^{K} x_k$.

Multinomial distribution

- Now suppose $n = 1$. This is like rolling a K-sided dice once, so $\mathbf{x}$ will be a vector of 0s and 1s (a bit vector), in which only one bit can be turned on. In this case, we can think of x as being a scalar categorical random variable with K states (values), and $\mathbf{x}$ is its dummy encoding. For example, if $K = 3$, we encode the states 1, 2 and 3 as $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. In this case, the pmf becomes

$$\text{Mu}(\mathbf{x}|1, \boldsymbol{\theta}) = \prod_{j=1}^{K} \theta_j^{\mathbb{I}(x_j=1)}$$

- This very common special case is known as a **categorical** or **discrete** distribution. We will use the following notation for this case:

$$\text{Cat}(x|\boldsymbol{\theta}) := \text{Mu}(\mathbf{x}|1, \boldsymbol{\theta})$$

  In otherwords, if $x \sim \text{Cat}(\boldsymbol{\theta})$, then $p(x = j|\boldsymbol{\theta}) = \theta_j$.

Multinomial Logistic regression

- In multiclass regression, we replace the logistic function $\eta = \text{sigm}(\mathbf{w}^T\mathbf{x})$ with $\boldsymbol{\eta} = \mathcal{S}(\mathbf{W}^T\mathbf{x})$, where $\mathcal{S}$ is the **softmax** function, defined as follows:

$$\mathcal{S}(\boldsymbol{\eta})_c = \frac{e^{\eta_c}}{\sum_{c'=1}^{C} e^{\eta_{c'}}}$$

  We use the non-standard notation $\mathcal{S}(\boldsymbol{\eta})$ to denote the $C \times 1$ vector of probabilities, and $\mathcal{S}(\boldsymbol{\eta})_c$ to denote the c'th component.

- The overall model becomes

$$p(y|\mathbf{x}, \mathbf{W}) = \text{Cat}(y|\mathcal{S}(\mathbf{W}^T\mathbf{x}))$$

  where $\mathbf{W}$ is a $D \times C$ weight *matrix*, with one column per class.

- This is called **multinomial logistic regression**, or the **multinomial logit model**.

Multinomial Logistic regression

- The softmax function is so-called since it acts a bit like the max function. To see this, let us divide each η_c by a constant T called the **temperature**. Then as $T \to 0$, we find

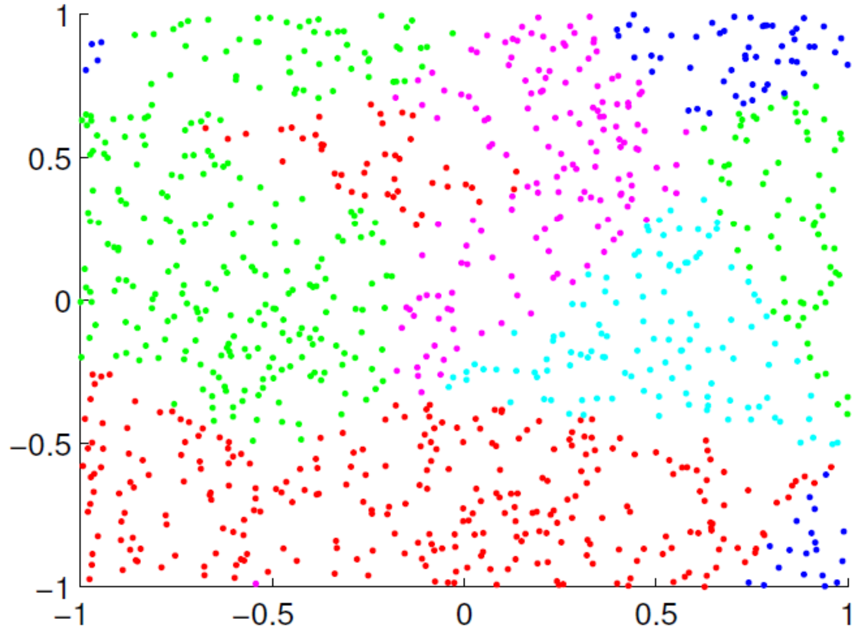
$$\mathcal{S}(\boldsymbol{\eta}/T)_c = \begin{cases} 1.0 & \text{if } c = \arg\max_{c'} \eta_{c'} \\ 0.0 & \text{otherwise} \end{cases}$$

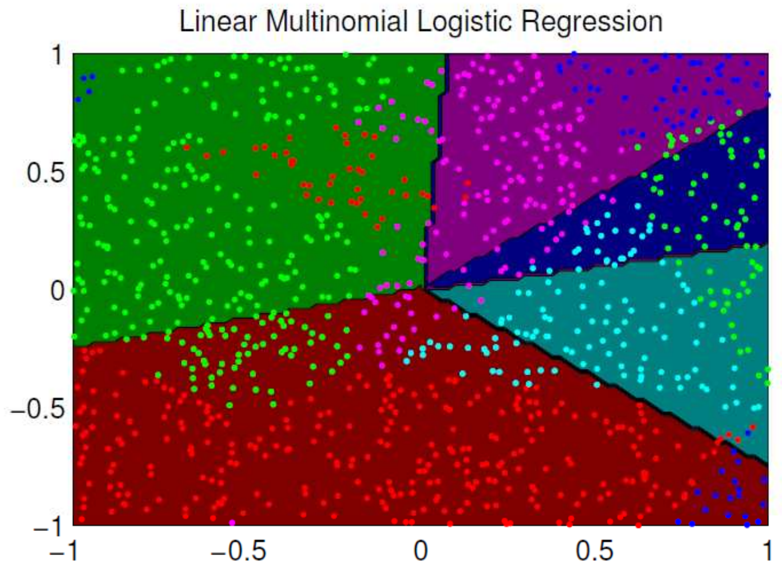
- We can arbitrarily define $\mathbf{w}_c = \mathbf{0}$ for one of the classes, say $c = C$, since $p(y = C|\mathbf{x}, \mathbf{w}) = 1 - \sum_{c=1}^{C-1} p(y = c|\mathbf{x}, \mathbf{w})$. In this case, the model has the form

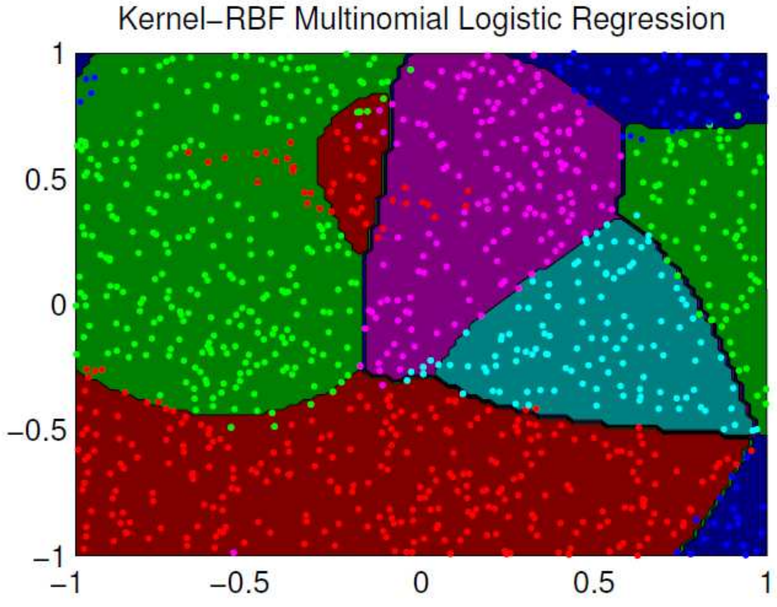
$$p(y = c|\mathbf{x}, \mathbf{W}) = \frac{\exp(w_{c0} + \mathbf{w}_c^T \mathbf{x})}{1 + \sum_{k=1}^{C-1} \exp(w_{k0} + \mathbf{w}_k^T \mathbf{x})}$$

  In general, if we have C classes, we only need to specify $C - 1$ vectors $\mathbf{w}_c$. If we don't "clamp" one of the vectors to some constant value, the parameters will be **unidentifiable**.

Multinomial Logistic regression







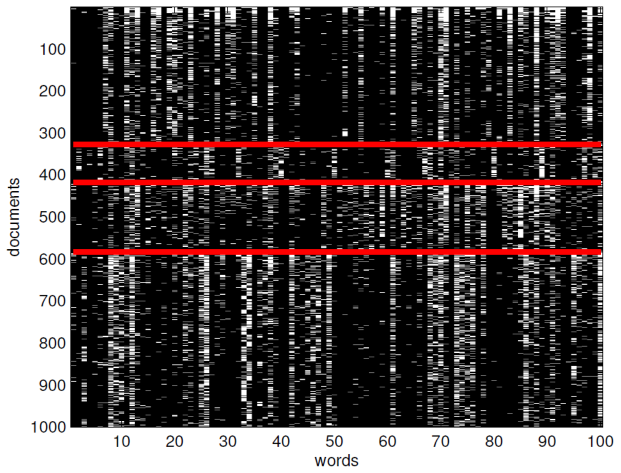
- See `logregMultinomKernelDemo.m`

Text classification

- In **document classification**, the goal is to classify a document, such as a web page or email message, into one of C classes, that is, to compute $p(y = c|\mathbf{x}, \mathcal{D})$, where $\mathbf{x}$ is some representation of the text. A special case of this is **email spam filtering**, where the classes are **spam** $y = 1$ or **ham** $y = 0$.

- A common way to represent variable-length documents in feature-vector format is to use a **bag of words** representation. For example, consider the following fragment of a famous nursery rhyme:

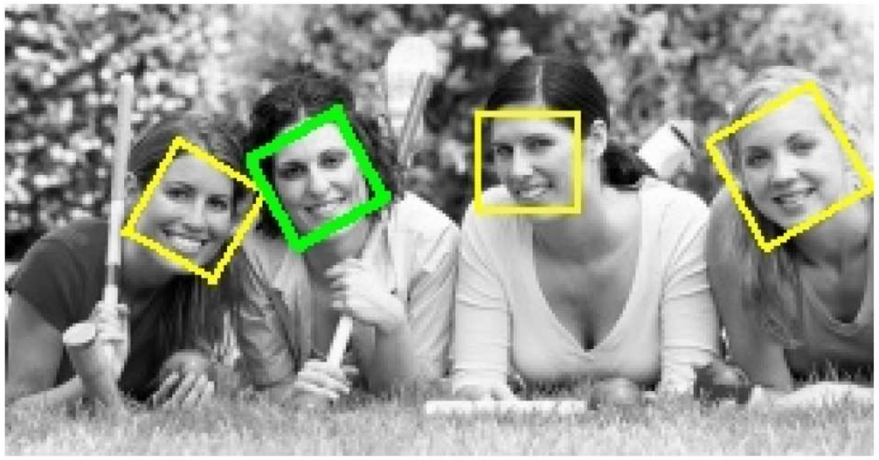
```
Mary had a little lamb,
little lamb, little lamb,
Mary had a little lamb,
whose fleece was white as snow.
```



Token	1	2	3	4	5	6	7	8	9	10
Word	mary	lamb	little	big	fleece	white	black	snow	rain	unk
Count	2	4	4	0	1	1	0	1	0	4
Binary	1	1	1	0	1	1	0	1	0	1

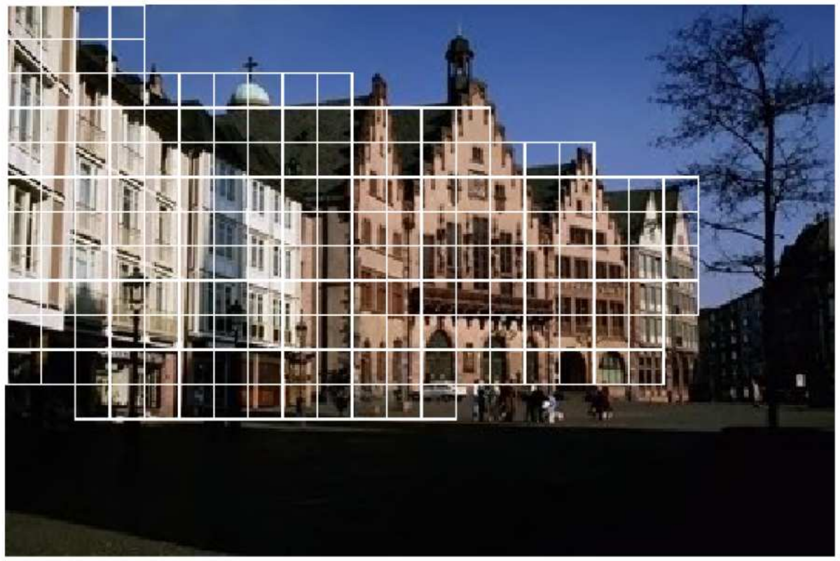
Face detection

- Divide the image into many small overlapping patches at different locations, scales and orientations, and classify each such patch based on whether it contains face-like texture or not. This is called a **sliding window detector**.



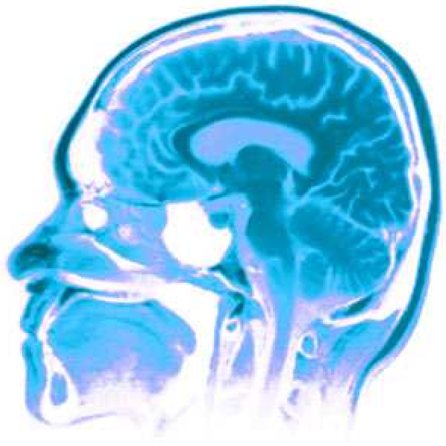
Structured classification

- In some problems, we need to predict multiple *related* response variables, i.e., we need to compute $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$, where $\mathbf{y} \in \mathcal{Y}$, and $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \cdots \mathcal{Y}_T$, where T is the output dimensionality.



- Another example concerns **part of speech tagging**. This is the task of determining if each word in a sentence is a noun, verb, adjective, etc. Locally this can be ambiguous, but again context can help. For example, the word "hit" could be a noun or a verb, but in the sentence "Bob hit the ball", it is obviously a verb, whereas in the sentence "The movie was a hit", it is obviously a noun.

Next class



Maximum Likelihood
and unconstrained optimization

Nando de Freitas
2011

KPM Book Sections: *3.2.1, 3.2.2, 3.4, 3.5, 3.6, 11.2, 11.3 and 30.4*