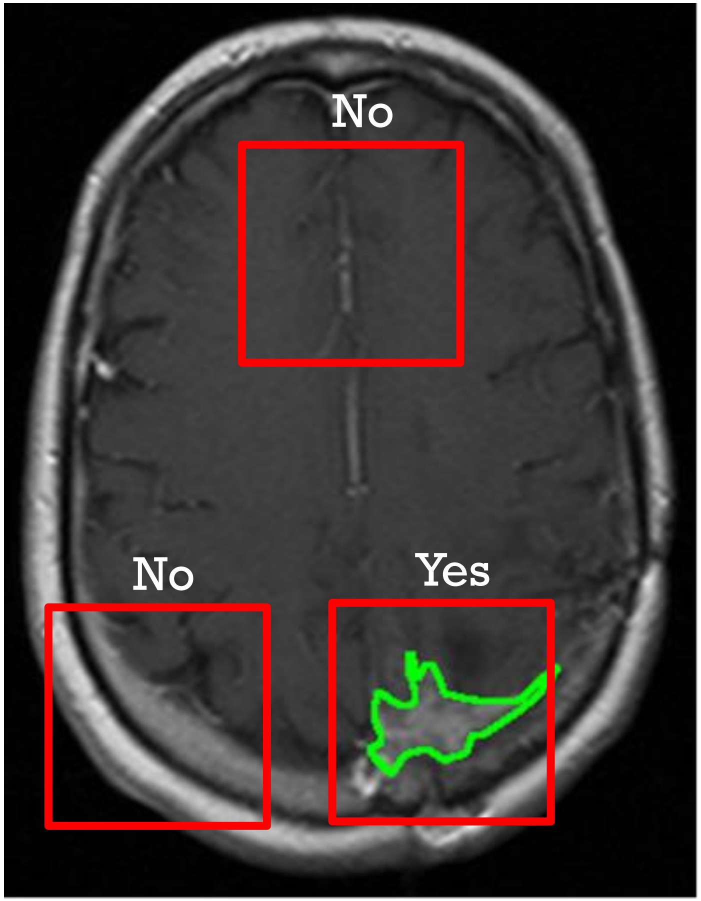
CLASSIFICATION

TUMOR CLASSIFICATION

Tumor?

Yes ~ +1
No ~ −1



SPAM FILTERS

Spam?

Yes ~ +1
No ~ −1

CLASSIFICATION

Machine Learning
> Supervised Learning
> Classification

- **Task.** Find $h: \mathbb{R}^d \to \{-1, +1\}$ such that $y \approx h(x; \theta)$
- **Experience.** Training data $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$
- **Performance.** Prediction error on test data

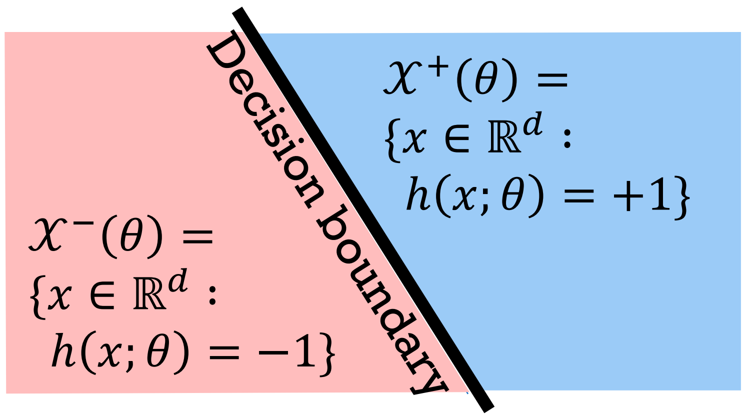
LINEAR CLASSIFICATION

Training data

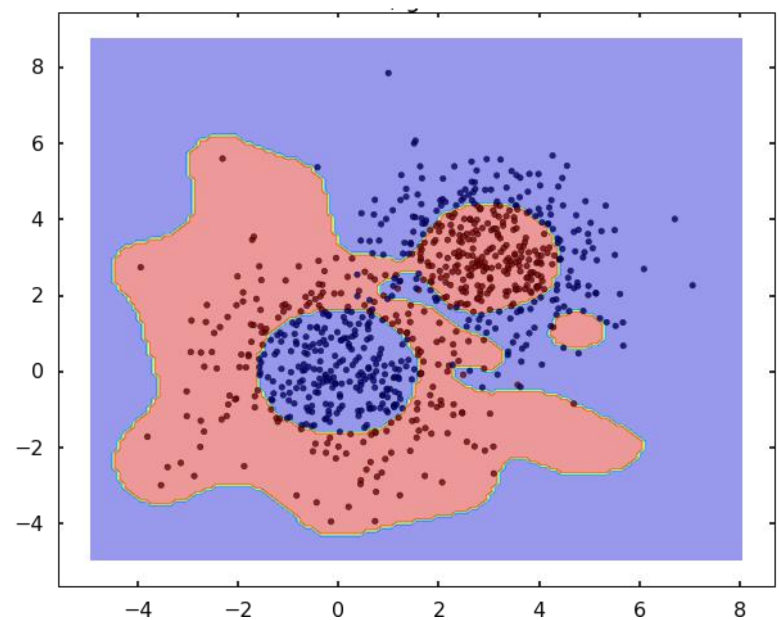
$$\mathcal{S}_n = \left\{ \left(x^{(i)}, y^{(i)}\right) \,\middle|\, i = 1, \dots, n \right\}$$

- Features/Inputs $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^\top \in \mathbb{R}^d$
- Labels/Output $y^{(i)} \in \{-1, +1\}$

DECISION REGIONS



linear classifier



non-linear classifier

A classifier h partitions the space into decision regions that are separated by decision boundaries. In each region, all the points map to the same label. Many regions could have the same label.

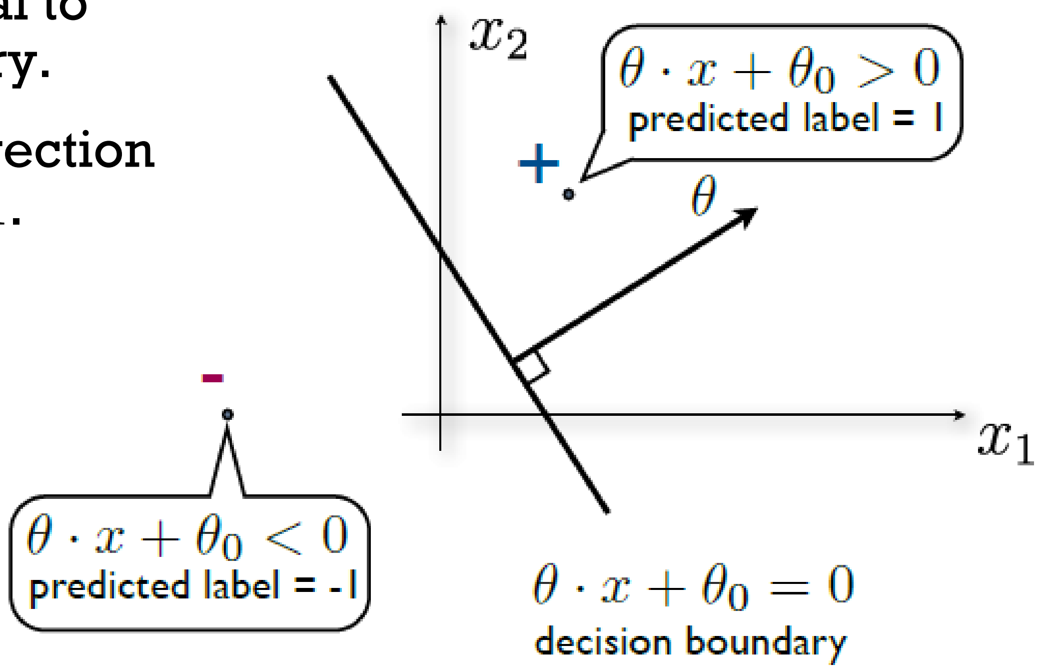
For linear classifiers, these regions are half spaces.

DECISION BOUNDARIES

For linear classifiers, the decision boundary is a hyperplane of dimension $d - 1$.

Vector θ is orthogonal to the decision boundary.

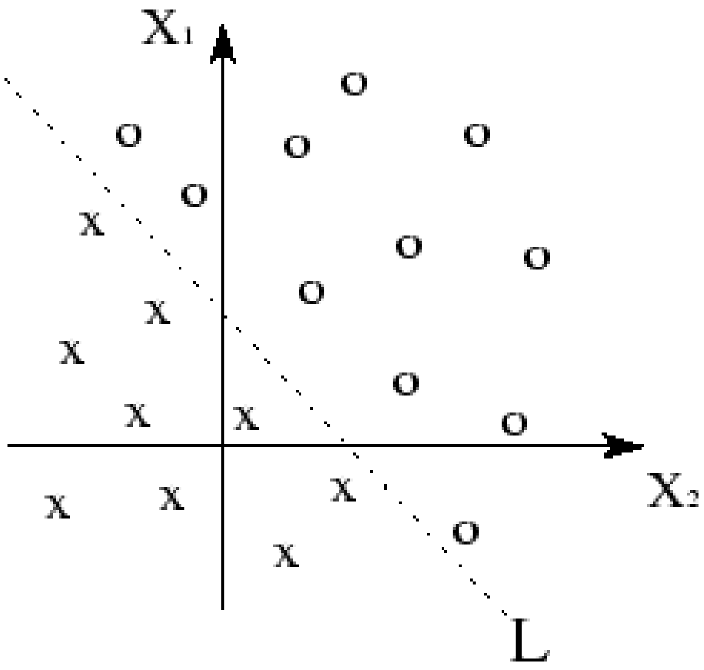
Vector θ points in direction of region labelled $+1$.



LINEARLY SEPARABLE

The training data $\mathcal{S}_n$ is
linearly separable
if there exists a parameters θ and θ_0 such that for all $(x, y) \in \mathcal{S}_n$,

$$y(\theta^\top x + \theta_0) > 0.$$

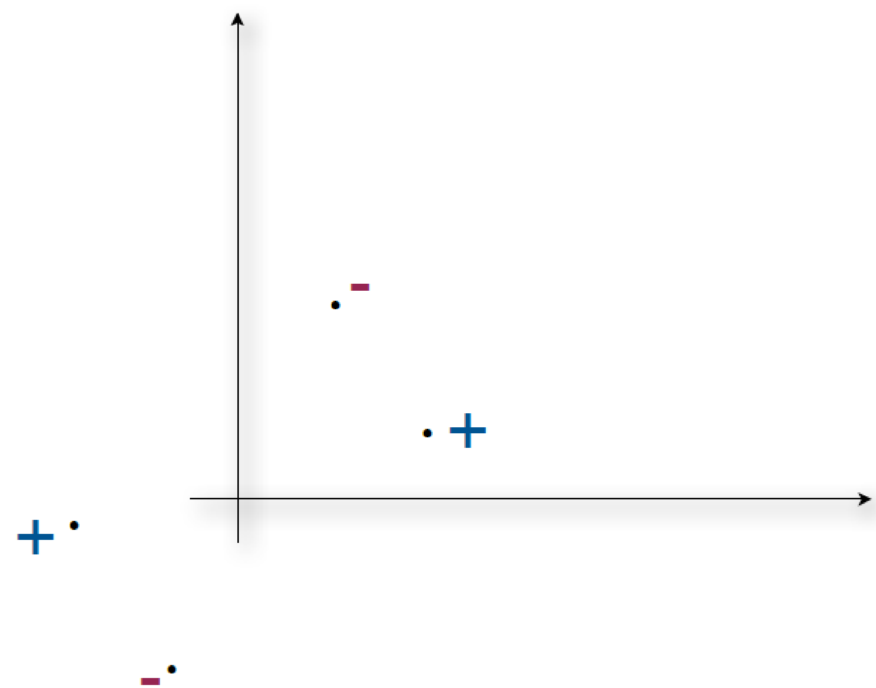


* not in syllabus

NOT LINEARLY SEPARABLE

Challenge
How do you prove the training data on the right is not linearly separable?

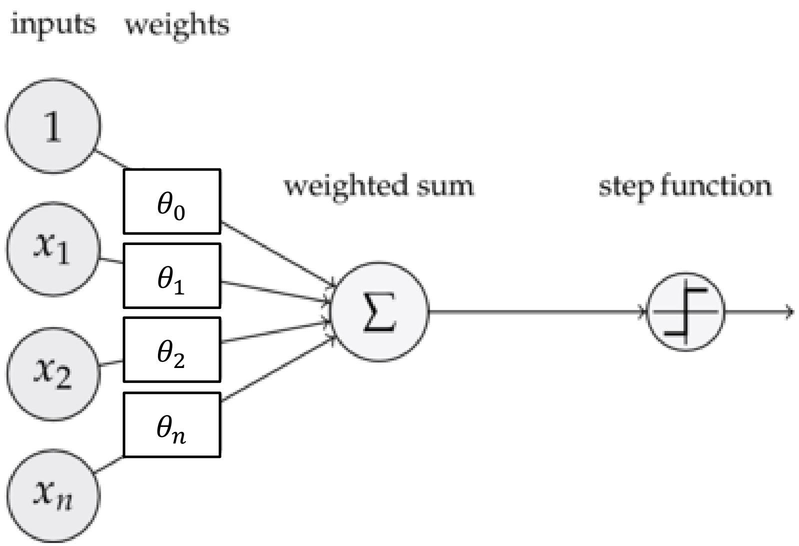
Hint
Draw a line between the points labelled $+1$, and a line between the points labelled -1.



PERCEPTRON ALGORITHM

PERCEPTRON

Linear classifiers are often also called perceptrons.



Perceptrons (1957) were designed to resemble neurons.

The hypothesis function is given by

$$h_\theta\left(x^{(i)}\right) = \operatorname{sgn}\left(\left\langle \theta, x^{(i)} \right\rangle\right),$$

where

$$\operatorname{sgn}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0. \end{cases}$$

Perceptron criterion

- Using the count of the number of misclassified points as a loss function is not good as this is a piecewise constant, which means that its gradient is zero almost everywhere and gradient descent methods will not work.
- Instead we define the perceptron criterion:

$$\mathcal{L}_P(\theta) = -\sum_{i \in \mathcal{M}} y^{(i)} \left\langle \theta, x^{(i)} \right\rangle,$$

  where $\mathcal{M}$ is the set of misclassified points.
- Note that this is a piecewise-linear function.

Perceptron algorithm

- Step 1: pick a point $x^{(i)}$ and check if $y^{(i)} \left\langle \theta(t), x^{(i)} \right\rangle \geq 0$.
- Step 2: if yes, do nothing; if no perform the following update rule:

$$
\begin{aligned}
\theta(t+1) &= \theta(t) - \alpha \nabla \mathcal{L}_P(\theta(t)) \\
&= \theta(t) + \alpha x^{(i)} y^{(i)}
\end{aligned}
$$

- Cycle through the rest of the data with steps 1 and 2.

- The update rule reduces the error with respect to the selected point because

$$
\begin{aligned}
-y^{(i)}\left\langle\theta(t+1), x^{(i)}\right\rangle &= -y^{(i)}\left\langle\theta(t),\, x^{(i)}\right\rangle - y^{(i)}\left\langle\alpha x^{(i)} y^{(i)},\, x^{(i)}\right\rangle \\
&< -y^{(i)}\left\langle\theta(t),\, x^{(i)}\right\rangle
\end{aligned}
$$

  since $\alpha\left\|y^{(i)}x^{(i)}\right\|^2 > 0$.
- However, this does not guarantee that the total error function is reduced at each stage as:
  - the contribution to the error from other misclassified points may have increased;
  - previously correctly classified points may have become misclassified.

* proof not in syllabus

PERCEPTRON CONVERGENCE

Theorem. If the training data is linearly separable, then the perceptron algorithm terminates after a finite number of steps.

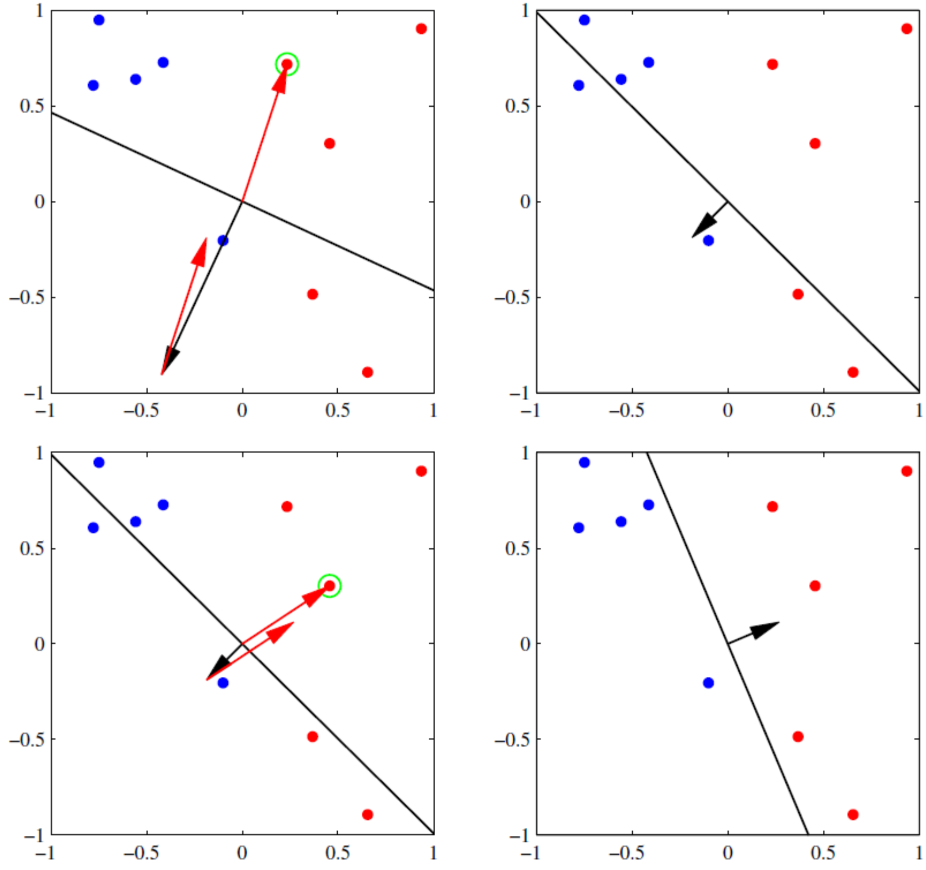
Proof. Not in the scope of this class, but it is not difficult. Basic idea is that with every mistake, lower bound for $||\theta||$ grows quickly but upper bound grows slowly. Eventually it must stop.

Non linearly-separable. In this case, the perceptron algorithm will never terminate, because there will always be a mistake for all values of θ. Other learning algorithms are needed.

http://www.cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf

Example

Red circles: +1, Blue circles: -1 (see pg 195 in Bishop)

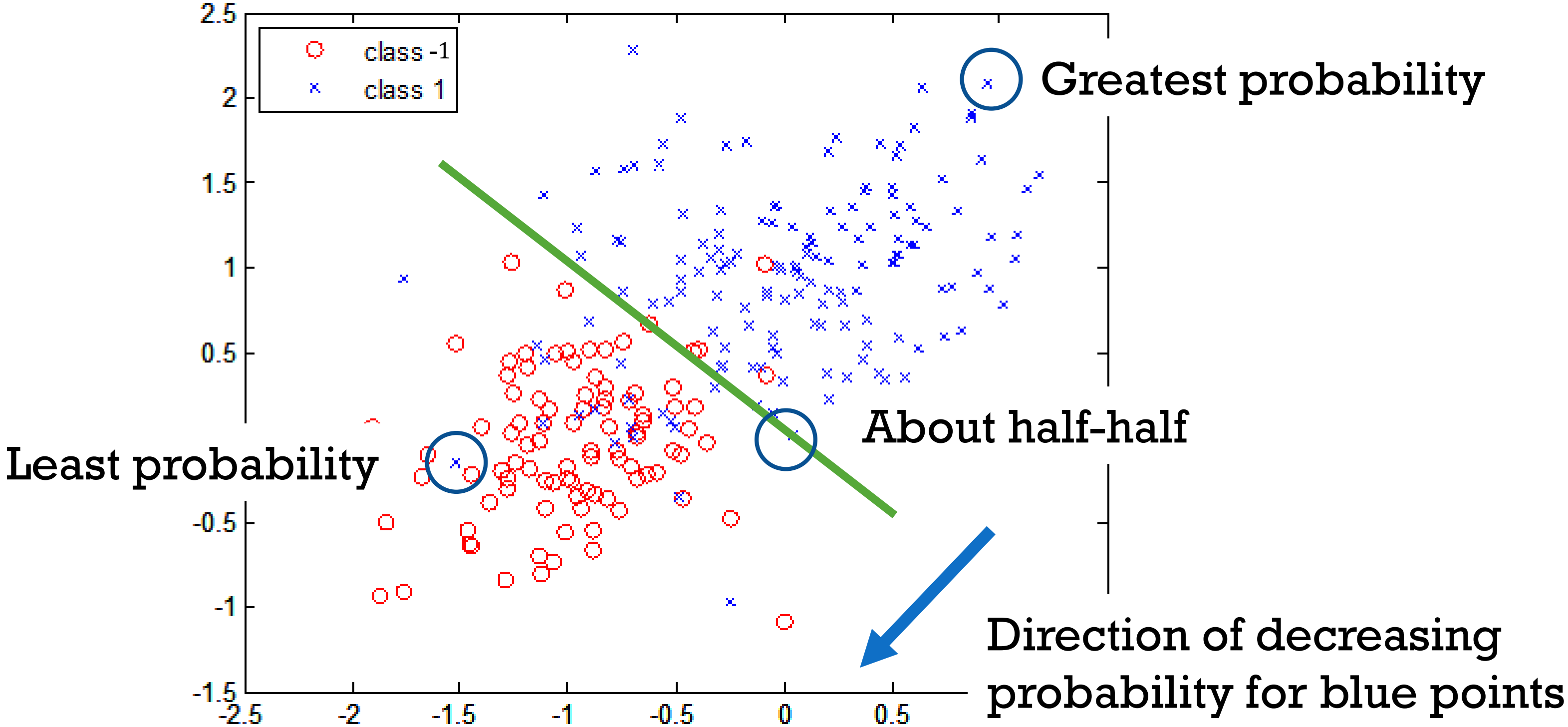


Disadvantages of the perceptron

- Learning algorithm difficulties:
  - Not easy to differentiate slow convergence from cases where there will be no convergence due to not having linear separability.
  - Different initialization of the parameters and presentation of the data lead to different solutions.
- Does not provide probabilistic outputs.
- Does not generalize well to more than two classes.

5 min break

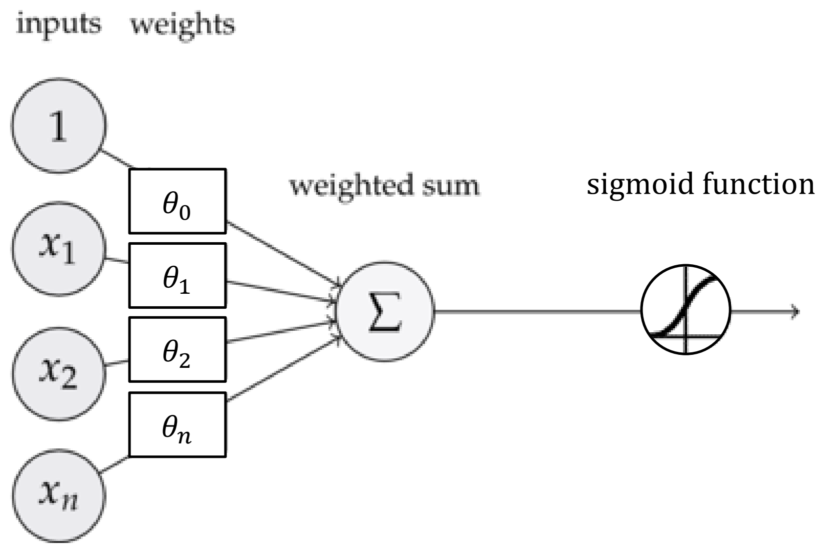
LOGISTIC REGRESSION

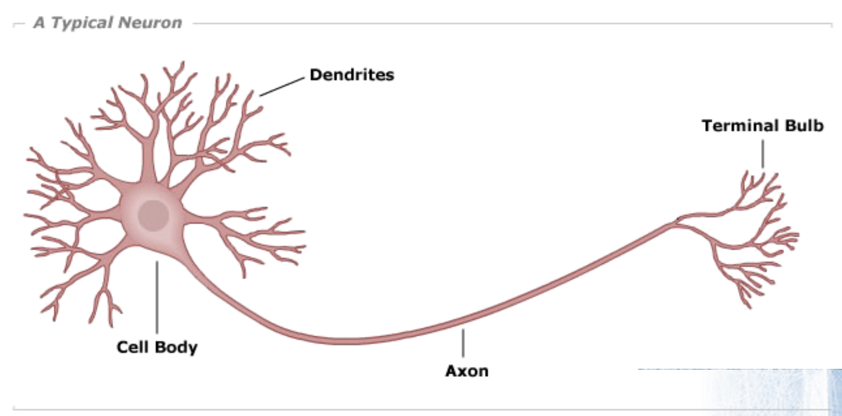

class -1
class 1
Greatest probability
About half-half
Least probability
Direction of decreasing probability for blue points
2.5
2
1.5
1
0.5
-0.5
-1
-1.5
-2.5
-2
-1.5
-1
-0.5
0
0.5

SIGMOID NEURONS

Model consists of
sigmoid neurons.

They were popular
in the early days of
deep learning (2006).





PROBABILISTIC MODEL

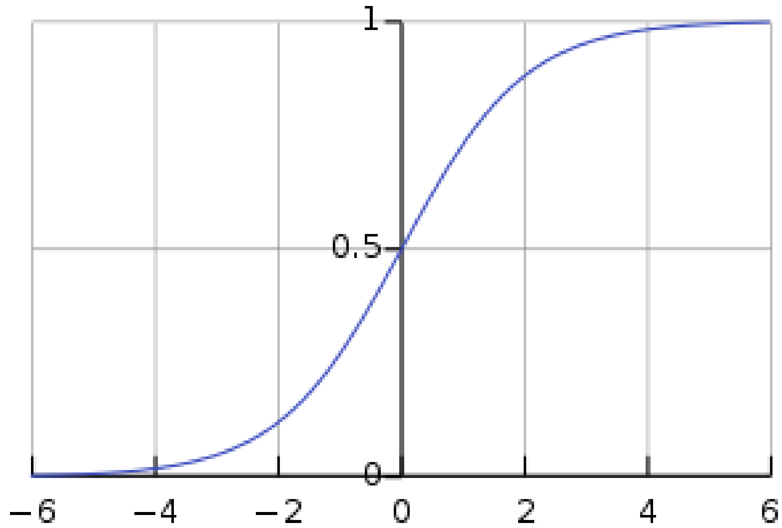
$[0, 1]$ denotes the interval $\{a \in \mathbb{R}: 0 \leq a \leq 1\}$

Model the probability that the label y is $+1$ given the feature is x.

$$h: \mathbb{R}^d \to [0, 1]$$

$$h(x; \theta) = \mathbb{P}(\, y = +1 \mid x \,) = \text{sigmoid}(\theta^\top x)$$

For small $\theta^\top x$, the label is very likely to be -1.



For large $\theta^\top x$, the label is very likely to be $+1$.

Logistic regression

- The formula for the sigmoid function $\sigma(z)$ is given by

$$\sigma\left(z\right) = \frac{1}{1 + e^{-z}},$$

and the hypothesis function is thus

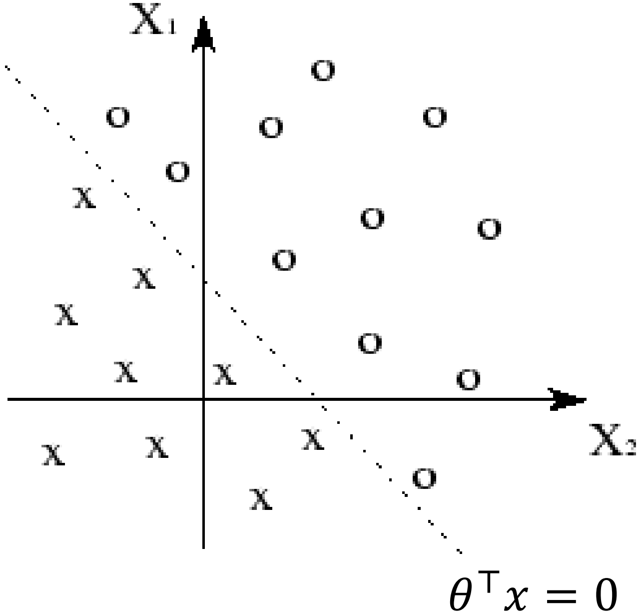
$$h_\theta\left(x^{(i)}\right) = \sigma\left(\left\langle \theta, x^{(i)} \right\rangle\right) = \frac{1}{1 + e^{-\left\langle \theta, x^{(i)} \right\rangle}}.$$

- The sigmoid function is also known as the logistic function.

DECISION BOUNDARY

$$h(x;\theta) \geq \frac{1}{2} \quad \Leftrightarrow \quad \text{sigmoid}(\theta^\top x) \geq \frac{1}{2} \quad \Leftrightarrow \quad \theta^\top x \geq 0$$

$$h(x;\theta) < \frac{1}{2} \quad \Leftrightarrow \quad \text{sigmoid}(\theta^\top x) < \frac{1}{2} \quad \Leftrightarrow \quad \theta^\top x < 0$$



The decision boundary is described by $\theta^\top x = 0$.

Formulas for the sigmoid function

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

(ii)
$$\begin{aligned}
\sigma'(z) &= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) \\
&= \left(\frac{1}{1+e^{-z}}\right)\left(1 - \frac{1}{1+e^{-z}}\right) \\
&= \sigma(z)\left(1-\sigma(z)\right) = \sigma(z)\sigma(-z)
\end{aligned}$$

- With $z^{(i)}$ denoting $\langle \theta, x^{(i)} \rangle$, we have

$$p_\theta\left(y = 1 \,\middle|\, x^{(i)}\right) = \sigma\left(z^{(i)}\right),$$

which means that

$$p_\theta\left(y = 0 \,\middle|\, x^{(i)}\right) = 1 - \sigma\left(z^{(i)}\right) = \sigma\left(-z^{(i)}\right).$$

- We can combine both expressions as

$$p_\theta\left(y = y^{(i)} \,\middle|\, x^{(i)}\right) = \sigma\left(z^{(i)}\right)^{y^{(i)}} \sigma\left(-z^{(i)}\right)^{1-y^{(i)}}.$$

Log-likelihood function

The log-likelihood function is thus

$$
\begin{aligned}
\ell(\theta) &= \log \prod_{i=1}^{m} p\left(y = y^{(i)} \,\middle|\, x^{(i)}\right) \\
&= \log \prod_{i=1}^{m} \sigma\left(z^{(i)}\right)^{y^{(i)}} \sigma\left(-z^{(i)}\right)^{1-y^{(i)}} \\
&= \sum_{i=1}^{m} y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(\sigma\left(-z^{(i)}\right)\right).
\end{aligned}
$$

The gradient of this loss function is

$$
\begin{aligned}
\frac{\partial \ell(\theta)}{\partial \theta_j} &= \sum_{i=1}^{m} \frac{y^{(i)} x_j^{(i)}}{\sigma\left(z^{(i)}\right)} \sigma\left(z^{(i)}\right) \sigma\left(-z^{(i)}\right) - \frac{\left(1 - y^{(i)}\right) x_j^{(i)}}{\sigma\left(-z^{(i)}\right)} \sigma\left(z^{(i)}\right) \sigma\left(-z^{(i)}\right) \\
&= \sum_{i=1}^{m} y^{(i)} x_j^{(i)} \sigma\left(-z^{(i)}\right) - \left(1 - y^{(i)}\right) x_j^{(i)} \sigma\left(z^{(i)}\right) \\
&= \sum_{i=1}^{m} x_j^{(i)} \left[y^{(i)} \left(1 - \sigma\left(z^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \sigma\left(z^{(i)}\right)\right] \\
&= \sum_{i=1}^{m} x_j^{(i)} \left(y^{(i)} - \sigma\left(z^{(i)}\right)\right).
\end{aligned}
$$

- We can then use the gradient to perform gradient ascent to maximize the likelihood:

$$\theta_j(t+1) = \theta_j(t) + \alpha \sum_{i=1}^{m} \left(y^{(i)} - \sigma\left(\left\langle \theta(t), x^{(i)} \right\rangle\right)\right) x_j^{(i)}, \quad j = 1, \ldots, n.$$

- Do you see the similarity with linear regression?

* not in syllabus

MULTICLASS CLASSIFICATION

Example. Predict color preference (e.g. yellow, green, blue).

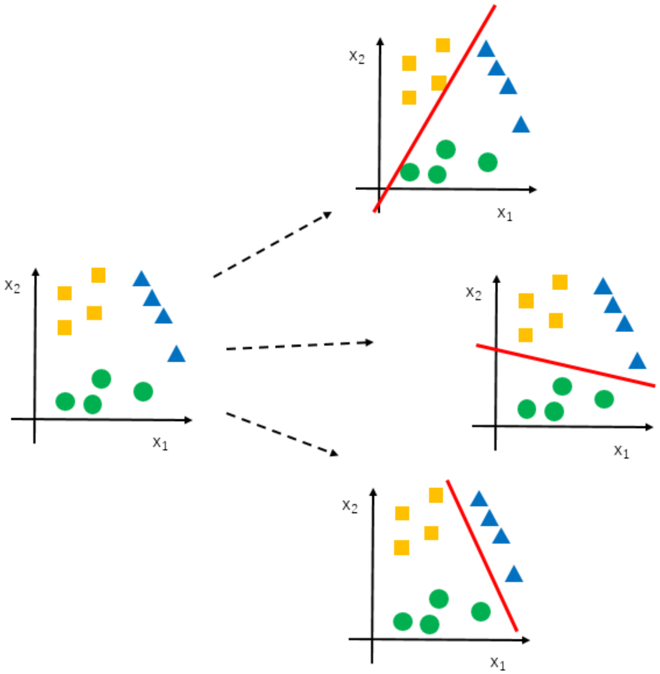
Solution.
Learn a 'one-vs-rest' function for each class.

$$h_{\text{yellow}}(x) = \text{sigmoid}(\alpha^\top x)$$

$$h_{\text{green}}(x) = \text{sigmoid}(\beta^\top x)$$

$$h_{\text{blue}}(x) = \text{sigmoid}(\gamma^\top x)$$

Rank the function values to predict the best class.



We will discuss a better method for multiclass classification next week called softmax classification.