Lecture 3: Classification

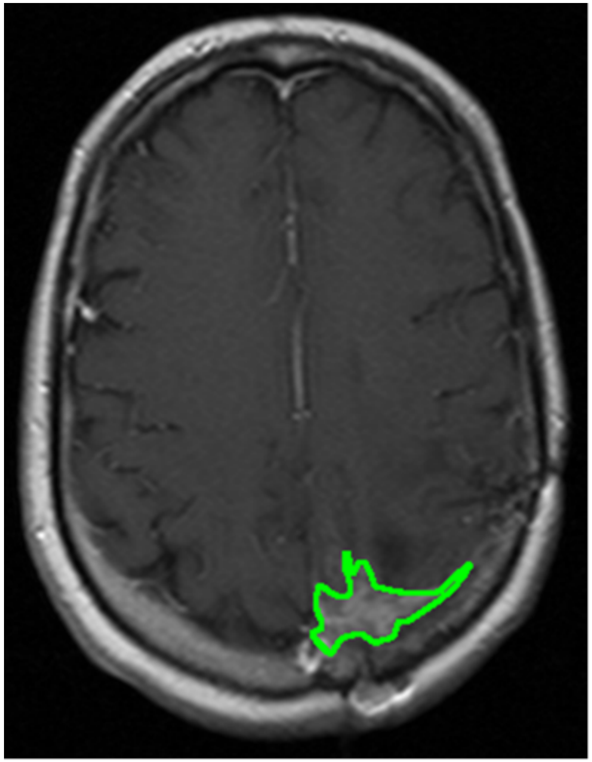
Liang Zheng

TUMOR CLASSIFICATION

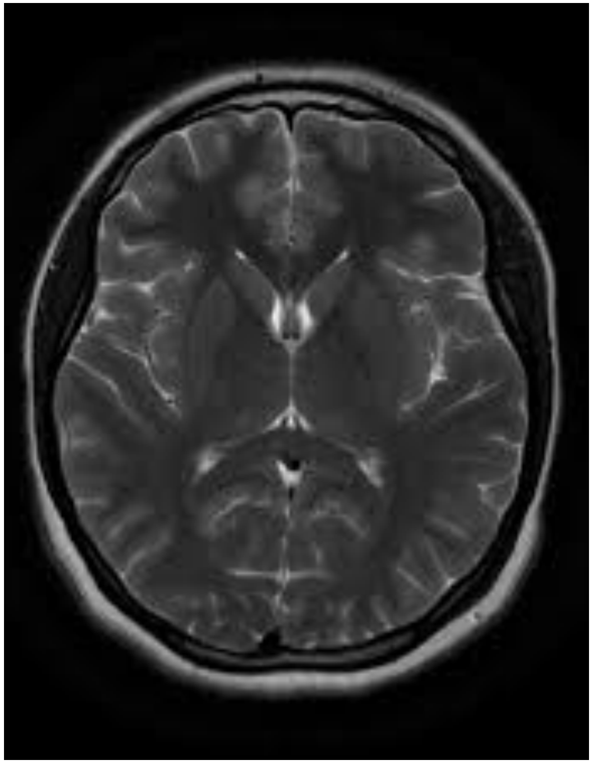
Tumor?

Yes: $+1$

No: -1



Yes



No

Spam Filters

Spam?

Yes: $+1$
No: -1

Regression - Revision

Machine Learning
> Supervised Learning
> Regression

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

Classification

Machine Learning
> Supervised Learning
>> Classification

- **Task.** Find $\boxed{h\colon \mathbb{R}^d \to \{-1, +1\}}$ 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

Regression

Training data

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

- Features/Inputs $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)^\top \in \mathbb{R}^d$
- Response/Output $\boxed{y^{(i)} \in \mathbb{R}}$

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 $\boxed{y^{(i)} \in \{-1, +1\}}$

Model

Set of *linear* classifiers $h: \mathbb{R}^d \to \{-1, +1\}$

$$h(x; \theta, \theta_0) = \operatorname{sign}(\theta_d x_d + \cdots + \theta_1 x_1 + \theta_0)$$
$$= \operatorname{sign}(\theta^\top x + \theta_0)$$

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

Some folks define $\operatorname{sign}(0) = 0$ but we will not adopt that here.

Model Parameters

$$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

Also called the *offset*

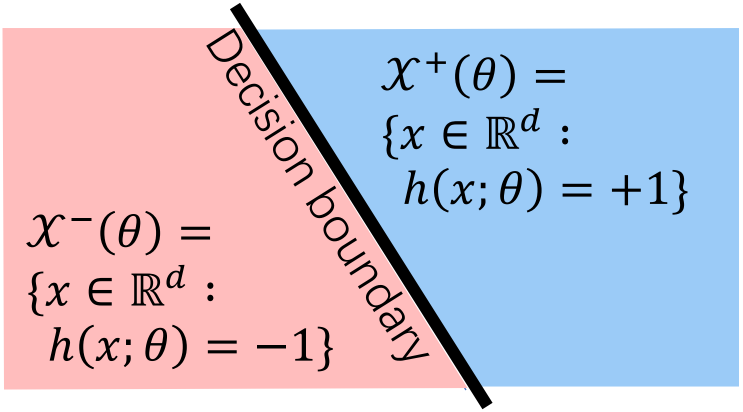
Test Loss

$[\![\cdot]\!]$ is the *indicator* function that returns a 1 if its argument is true, and 0 otherwise.

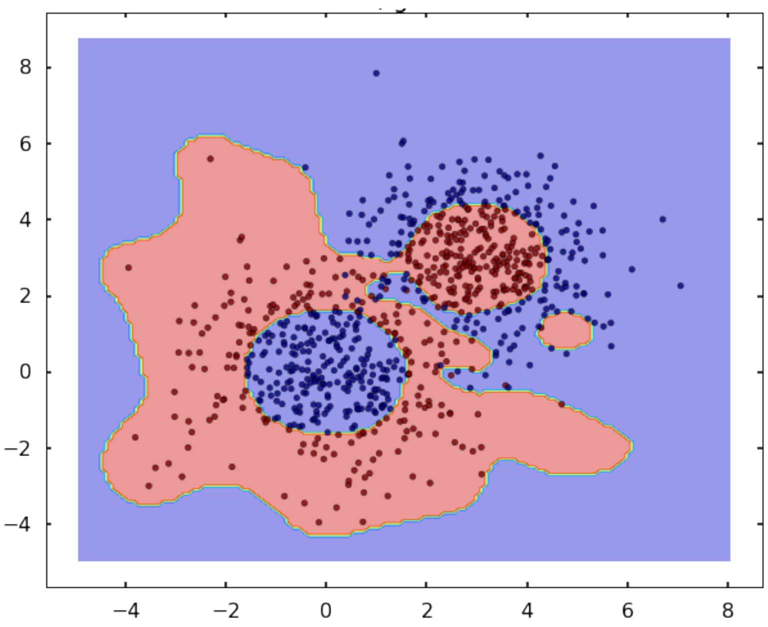
$$\mathcal{R}_1(\theta, \theta_0; x, y) = [\![y \neq h(x; \theta, \theta_0)]\!]$$

$$\mathcal{R}(\theta, \theta_0; \mathcal{S}_*) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_*} \mathcal{R}_1(\theta, \theta_0; x, y)$$

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.

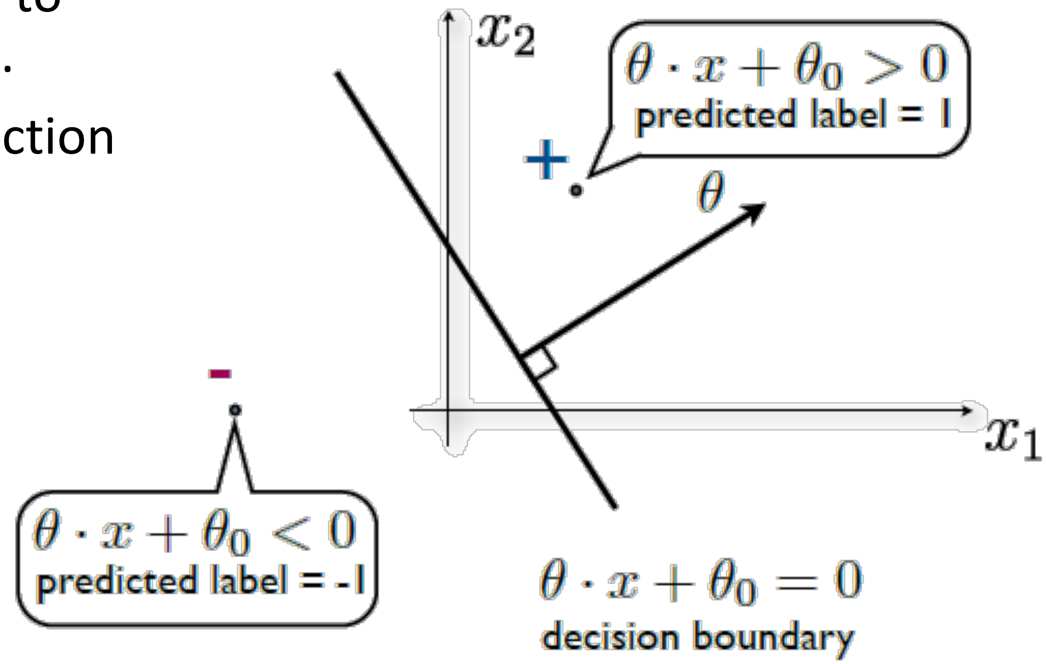
For linear classifiers, these regions are half spaces.

Decision Boundaries

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

Vector θ is orthogonal to the decision boundary.

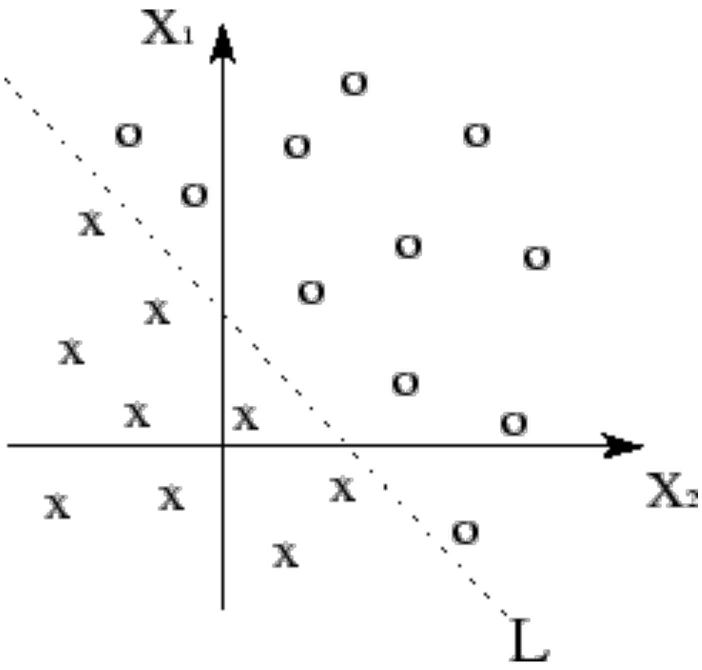
Vector θ points in direction of region labelled $+1$.



Linearly Separable

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

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



Constant Feature Trick

Define $x_0 = 1$ and set $\tilde{x} = (x_d, \ldots, x_1, x_0) \in \mathbb{R}^{d+1}$

Data

$$\left(\tilde{x}^{(1)}, y^{(1)}\right), \left(\tilde{x}^{(2)}, y^{(2)}\right), \ldots, \left(\tilde{x}^{(n)}, y^{(n)}\right),\ \ \tilde{x} \in \mathbb{R}^{d+1}, y \in \{-1, +1\}$$

Model

$$h(x; \theta, \theta_0) = \operatorname{sign}(\theta_1 x_1 + \cdots + \theta_d x_d + \theta_0 x_0) = \operatorname{sign}\left(\tilde{\theta}^\top \tilde{x}\right)$$

$$\tilde{\theta} = (\theta, \theta_0) \in \mathbb{R}^{d+1}$$

Test Loss

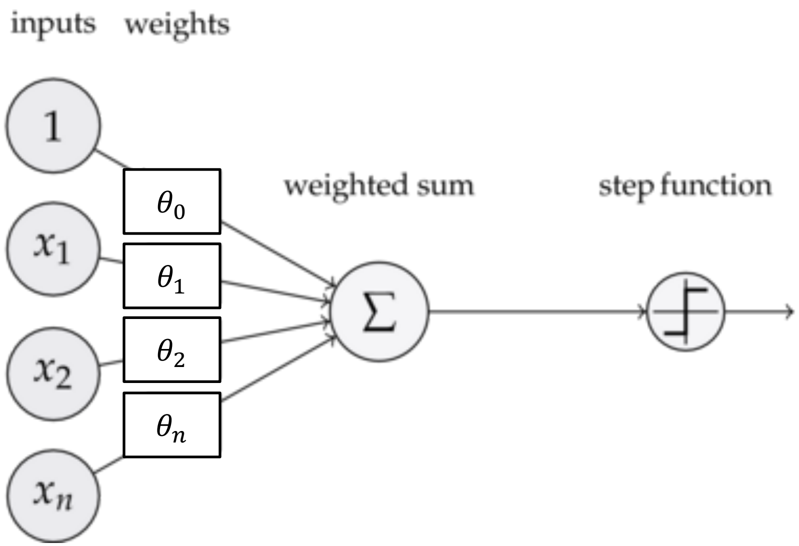
$$\mathcal{R}_1\left(\tilde{\theta}; \tilde{x}, y\right) = \left[\!\left[y \neq h\left(\tilde{x}; \tilde{\theta}\right) \right]\!\right]$$

$$\mathcal{R}\left(\tilde{\theta}; \mathcal{S}_*\right) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_*} \mathcal{R}_1\left(\tilde{\theta}; \tilde{x}, y\right)$$

Perceptron Algorithm

Perceptron

Linear classifiers are often also called perceptrons.



Perceptrons (1957) were designed to resemble neurons.

Zero-One Loss

Let $\mathcal{L}_1(\theta; x, y) = 1$ (0 otherwise) if

- $y \neq h(x; \theta)$, or [misclassified]
- (x, y) is on decision boundary [boundary]

Note that $y(\theta^\top x) \leq 0$ if

- $\theta^\top x$ and y differ in sign, or [misclassified]
- $\theta^\top x$ is zero [boundary]

$$\mathcal{L}_1(\theta; x, y) = [\![y(\theta^\top x) \leq 0]\!] = \mathrm{Loss}\big(y(\theta^\top x)\big)$$

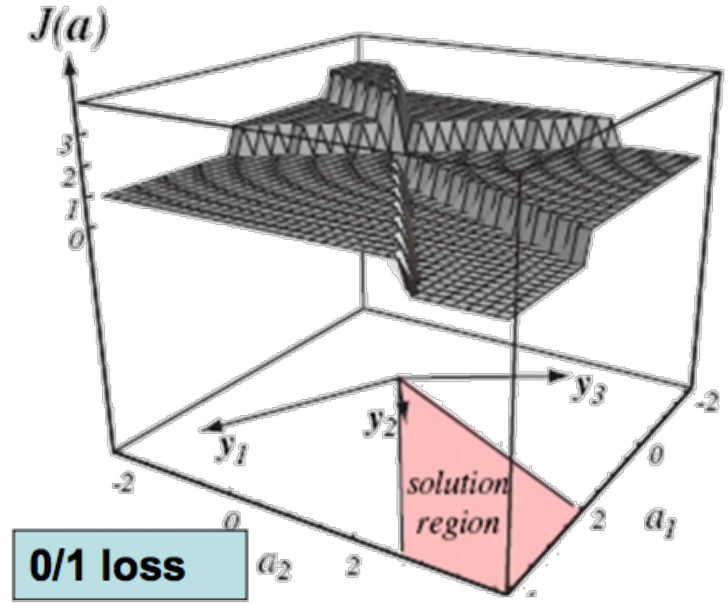
where $\mathrm{Loss}(z) = [\![z \leq 0]\!]$ is the zero-one loss.

Training Loss

$$\mathrm{Loss}(z) = [\![z \le 0]\!]$$

$$\mathcal{L}_1(\theta; x, y) = \mathrm{Loss}\big(y(\theta^\top x)\big)$$

$$\mathcal{L}_n(\theta; \mathcal{S}_n) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \mathcal{L}_1(\theta; x, y)$$



Gradient is zero almost everywhere!

Gradient descent not possible.

Mistake-Driven Algorithm

1. Initialize $\theta = 0$.
2. For each data $(x, y) \in \mathcal{S}_n$,
   a. Check if $h(x; \theta) = y$.
   b. If not, update θ to correct the mistake.
3. Repeat Step (2) until no mistakes are found.

Perceptron Algorithm

We start from 0, not some random point.

1. Initialize $\theta = 0$.
2. For each data $(x, y) \in \mathcal{S}_n$,
   a. If $y(\theta^\top x) \leq 0$,
      i. $\theta \longleftarrow \theta + yx$.
3. Repeat Step (2) until no mistakes are found.

Due to the constant feature trick $x_0 = 1$, update for θ_0 will be $\theta_0 \longleftarrow \theta_0 + yx_0 = \theta_0 + y$.

Example (Activity)

Training data

- $\left(x^{(1)}, y^{(1)}\right) = \left((2,\ \ 2)^\top, +1\right)$
- $\left(x^{(2)}, y^{(2)}\right) = \left((2, -1)^\top, -1\right)$

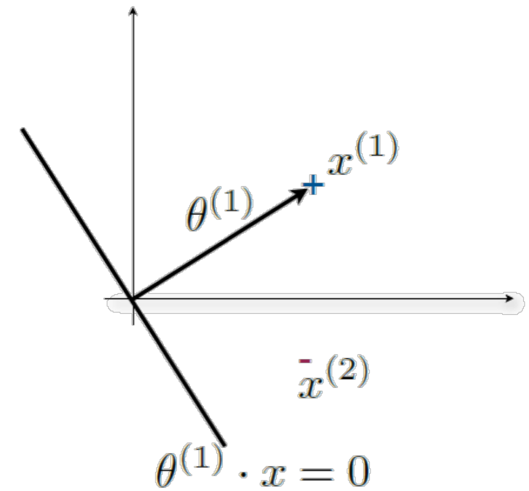
Apply the perceptron algorithm to the data to find a classifier $h(x;\theta), \theta = (\theta_1, \theta_2)$, that separates the data.

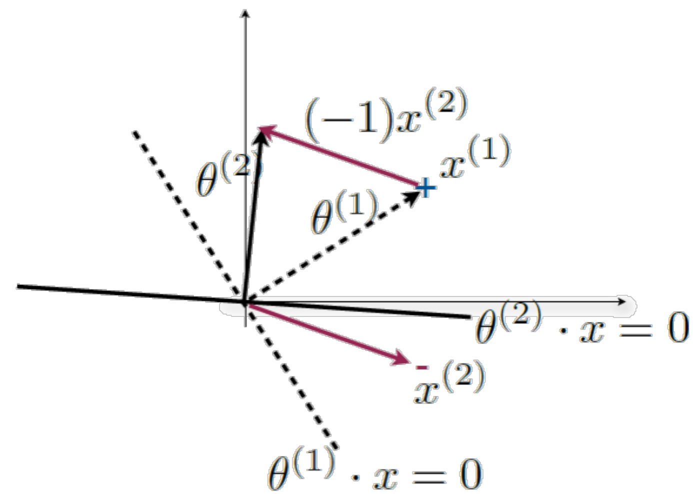
Example

Training data

- $(x^{(1)}, y^{(1)}) = ((2,\ \ 2)^\top, +1)$
- $(x^{(2)}, y^{(2)}) = ((2, -1)^\top, -1)$

- Initialize $\theta = (0,0)$.
- Since $y^{(1)}\theta^\top x^{(1)} = 0$,
  set $\theta = (0,0) + (2,2)^\top = (2,2)^\top$.
- Since $y^{(2)}\theta^\top x^{(2)} = -2$,
  set $\theta = (2,2)^\top - (2,-1)^\top = (0,3)^\top$.
- $y^{(1)}\theta^\top x^{(1)} = 6 > 0$.
- $y^{(2)}\theta^\top x^{(2)} = 3 > 0$.
- No more mistakes, so we are done.





Perceptron Algorithm

1. Training Set (Linearly Separable)

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$$

2. Model (Set of Perceptrons)

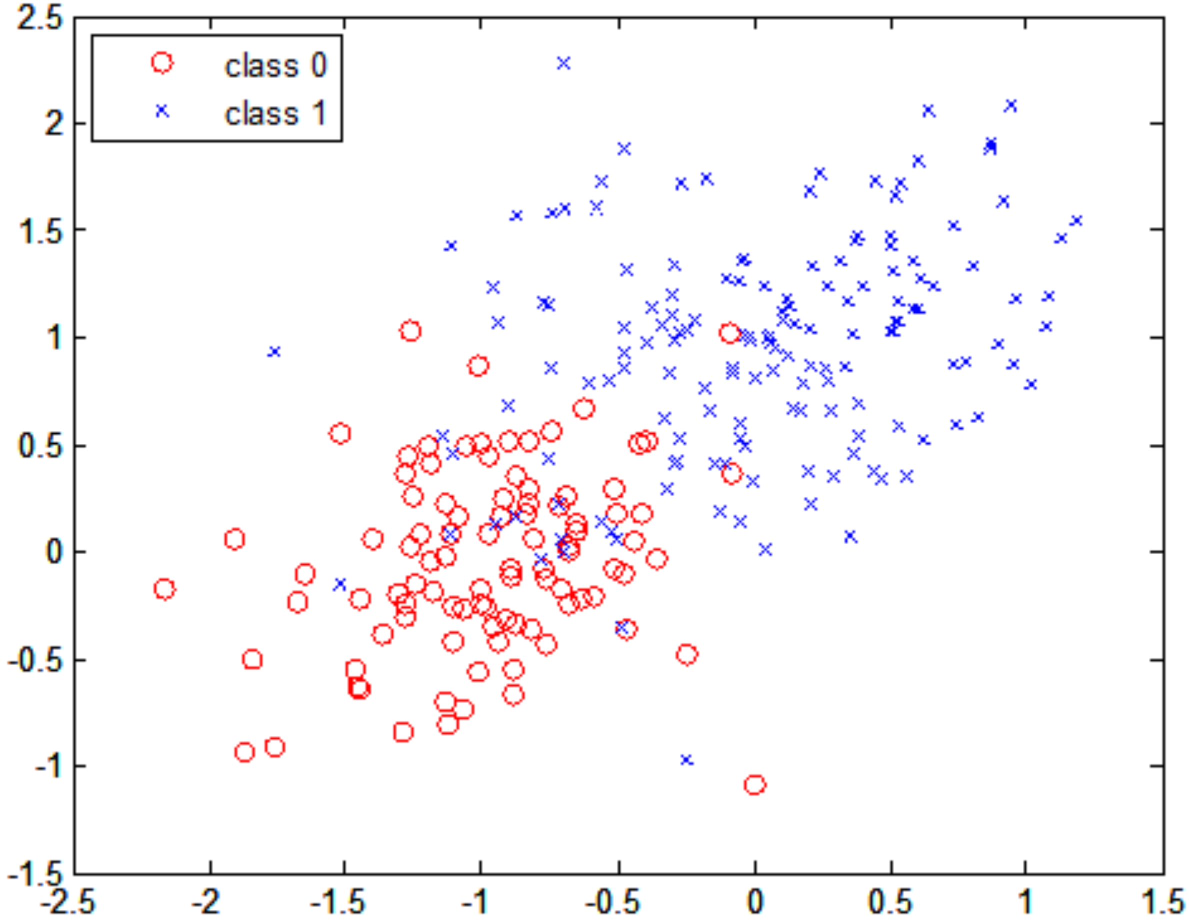
$$h(x; \theta) = \operatorname{sign}(\theta_1 x_1 + \cdots + \theta_d x_d)$$

3. Training Loss (Fraction of Misclassified/Boundary Points)

$$\mathcal{L}_n(\theta) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} [\![\, y(\theta^\top x) \leq 0 \,]\!]$$

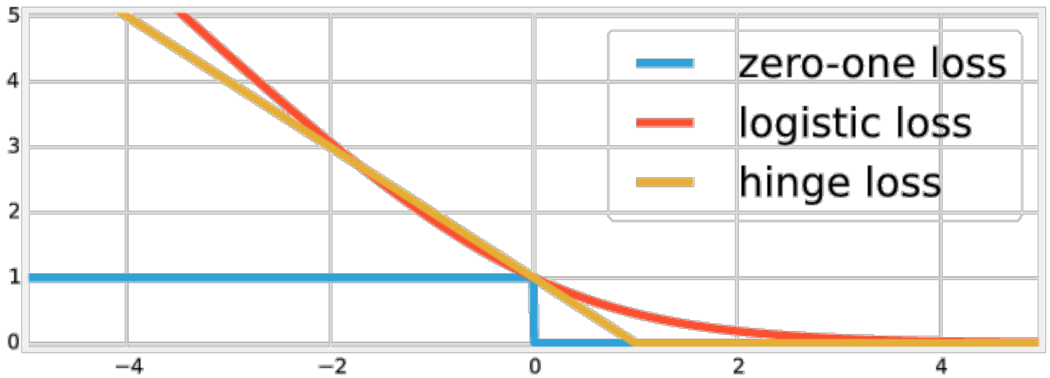
3. Algorithm (Mistake-Driven Algorithm)

Hinge Loss



Perceptron algorithm does not converge for training sets that are not linearly separable.

Loss Functions



Training Loss

$$\mathcal{L}_n(\theta) = \frac{1}{n}\sum_{\text{data }(x,y)} \text{Loss}\big(y(\theta^\top x)\big)$$

Zero-One Loss

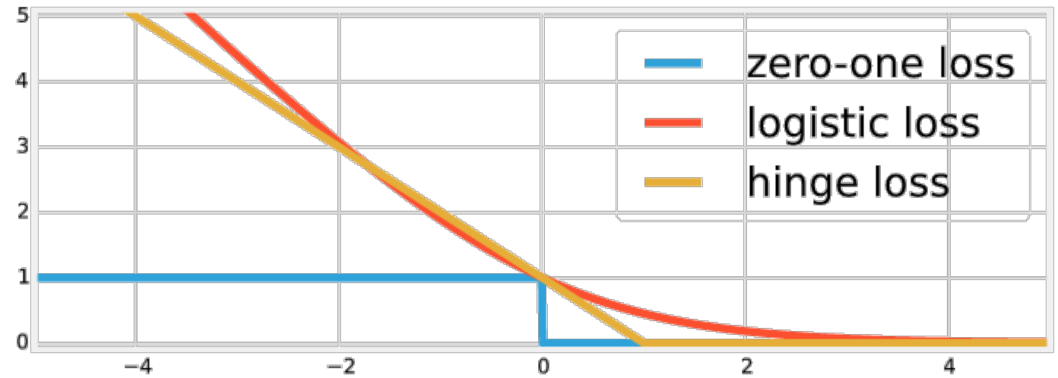
$$\text{Loss}_{01}(z) = [\![\, z \leq 0 \,]\!]$$

Hinge Loss

$$\text{Loss}_{\text{H}}(z) = \max\{1 - z, 0\}$$

CONVEX!
Penalize large mistakes more.
Penalize near-mistakes, i.e. $0 \leq z \leq 1$.

HINGE LOSS



Find θ that minimizes

$$\mathcal{L}_n(\theta) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \mathrm{Loss_H}(z)$$
$$= \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \max\{1 - y(\theta^\top x), 0\}$$

Gradient

$$\nabla_z \mathrm{Loss_H}(z) = \begin{cases} 0 & \text{if } z > 1, \\ -1 & \text{otherwise.} \end{cases}$$

$$\nabla_\theta \mathrm{Loss_H}\big(y(\theta^\top x)\big) = \begin{cases} 0 & \text{if } y(\theta^\top x) > 1, \\ -yx & \text{otherwise.} \end{cases}$$

Stochastic Gradient Descent

1. Initialize $\theta = 0$.
2. Select data $(x, y) \in \mathcal{S}_n$ at random.
    a. If $y(\theta^\top x) \leq 1$, then
        i. $\theta \longleftarrow \theta + \eta_k \, yx$.
3. Repeat Step (2) until convergence.
   (e.g. when improvement in $\mathcal{L}_n(\theta)$ is small enough)

Differences from Perceptron Algorithm

- Check $z \leq 1$ rather than $z \leq 0$
- Decreasing η_k rather than $\eta = 1$
- Selecting data at random rather than in sequence

Hinge Loss Algorithm

1. Training Set (Not Necessarily Linearly Separable)

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \dots, \left(x^{(n)}, y^{(n)}\right)$$

2. Model (Set of Perceptrons)

$$h(x; \theta) = \mathrm{sign}(\theta_1 x_1 + \cdots + \theta_d x_d)$$

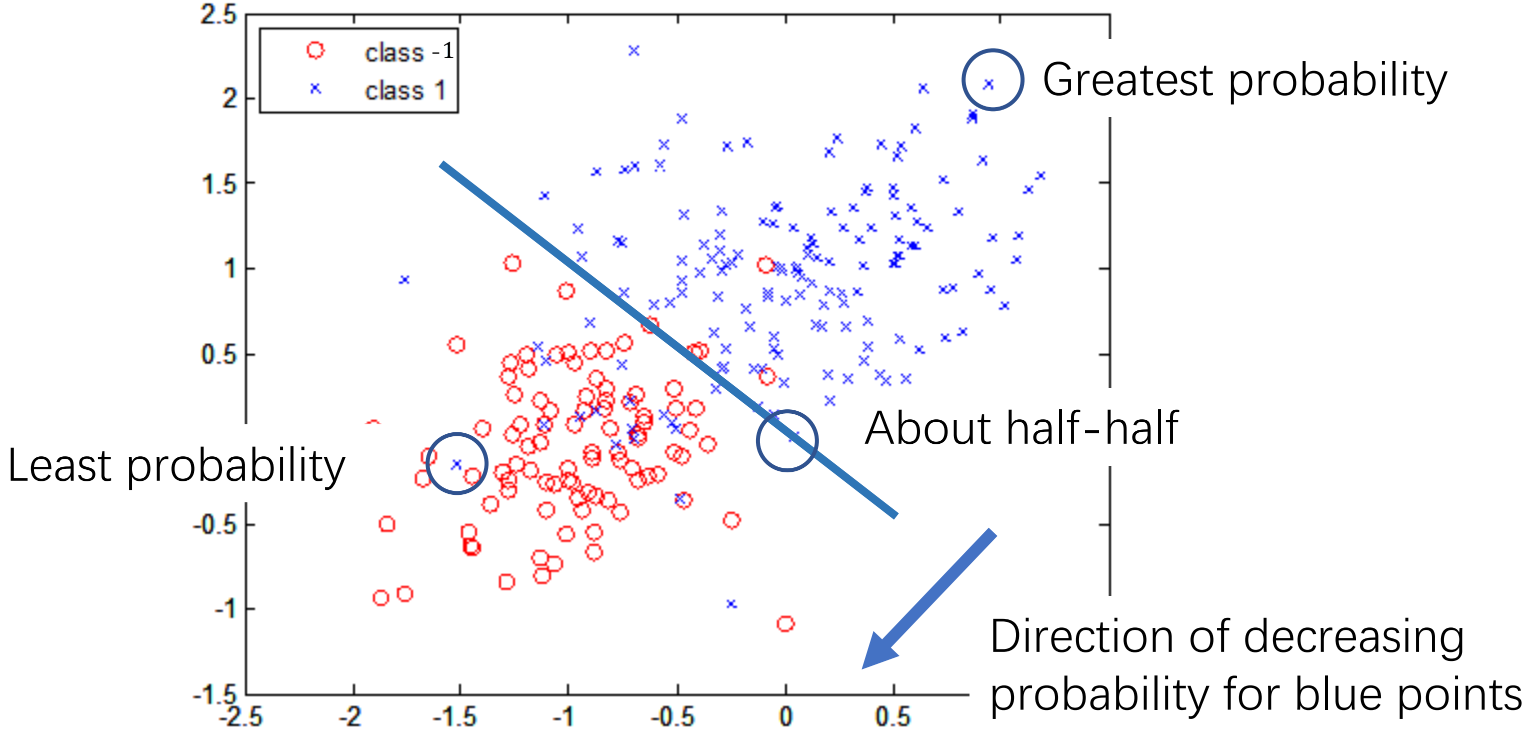
3. Training Loss (Hinge Loss)

$$\mathcal{L}_n(\theta) = \frac{1}{n} \sum_{(x,y)\in \mathcal{S}_n} \max\{1 - y(\theta^\top x), 0\}$$

3. Algorithm (Gradient Descent)

$$\theta \longleftarrow \theta + \frac{\eta_k}{n} \sum_{(x,y)\in \mathcal{S}_n} yx$$

Logistic Regression



class -1
class 1
Greatest probability
About half-half
Least probability
Direction of decreasing
probability for blue points

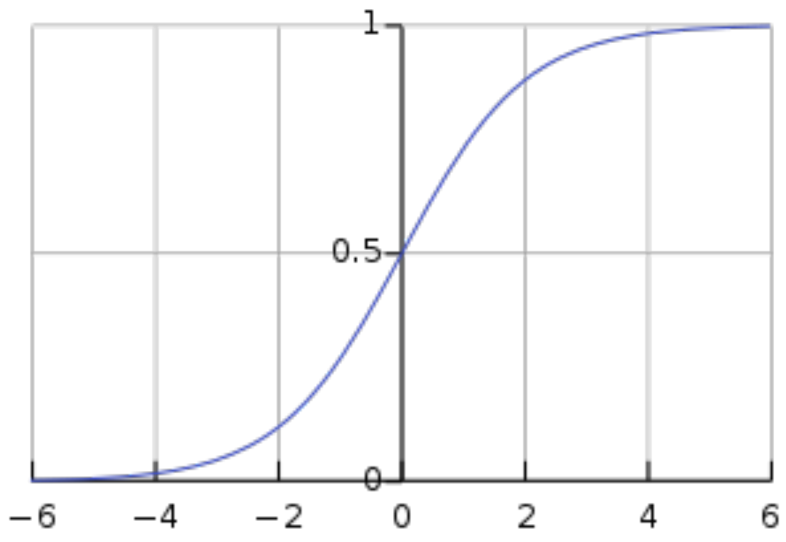
Probabilistic Model

$[0, 1]$ 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.



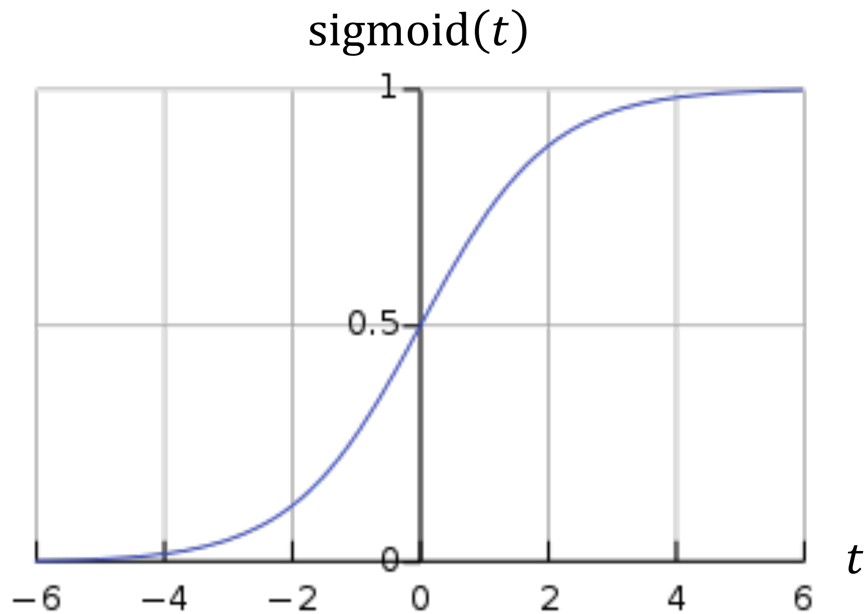
For large $\theta^\top x$, the label is very likely to be $+1$.

Sigmoid function

$$\text{sigmoid}(t) = \frac{1}{1 + e^{-t}}$$

$$\text{sigmoid}: \mathbb{R} \to [0, 1]$$

Sometimes also known as the *logistic* function.



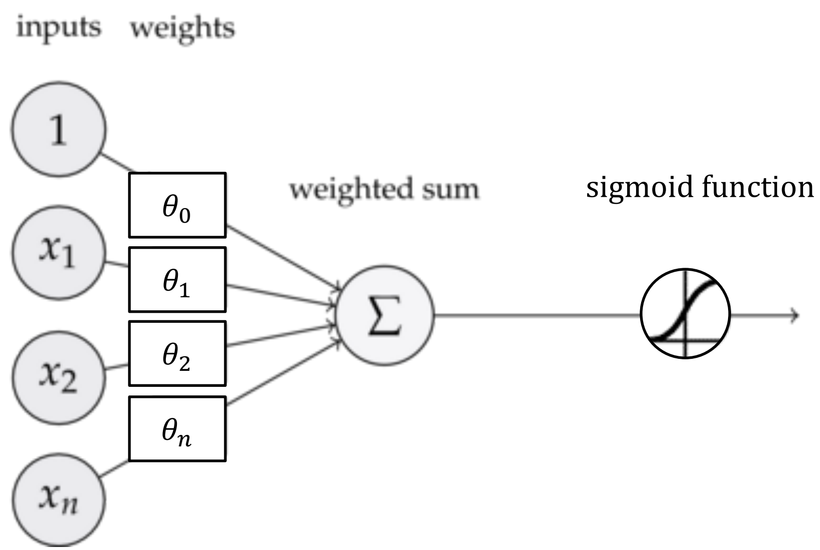
Super useful formula

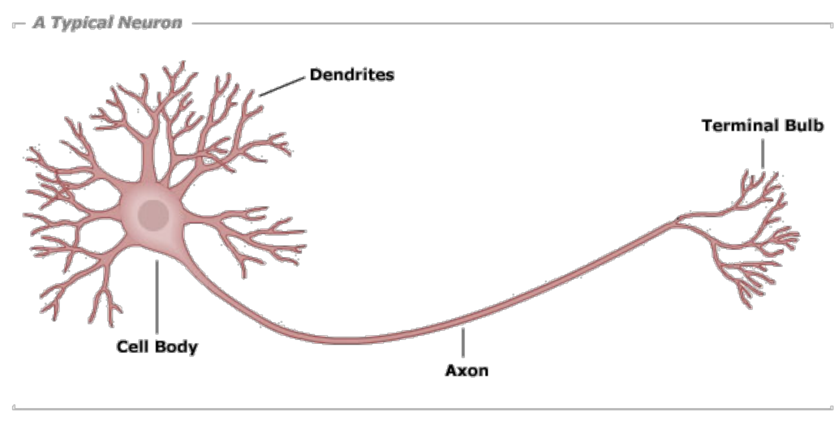
$$\text{sigmoid}(-t) = \frac{1}{1+e^{t}} = \frac{e^{-t}}{e^{-t}+1} = 1 - \frac{1}{e^{-t}+1} = 1 - \text{sigmoid}(t)$$

Sigmoid Neurons

Model consists of sigmoid neurons.

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





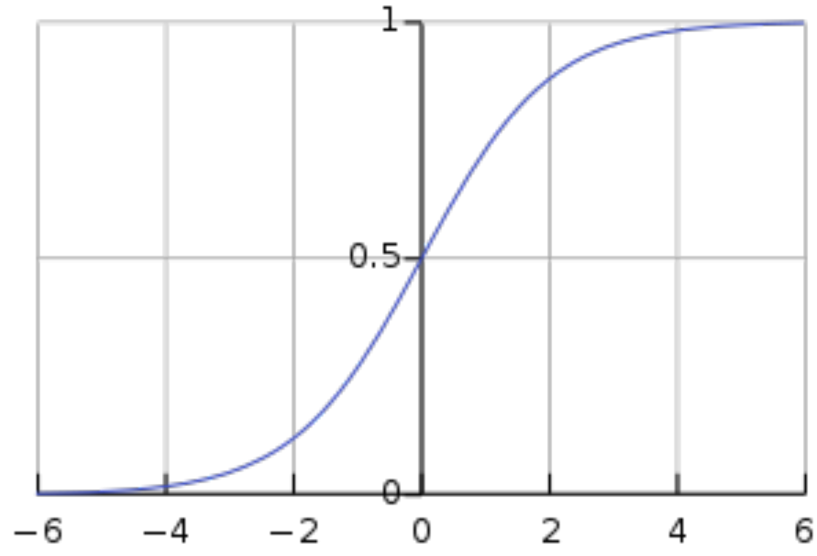
Label Probabilities

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

To get the other label probability,

$$\begin{aligned}\mathbb{P}(\, y = -1 \mid x\,) \quad &= 1 - \mathbb{P}(\, y = +1 \mid x\,) \\ &= 1 - \text{sigmoid}(\theta^\top x) \\ &= \text{sigmoid}(-\theta^\top x) = \text{sigmoid}\big(\mathrm{y}(\theta^\top x)\big)\end{aligned}$$

Thus, $\mathbb{P}(y|x) = \text{sigmoid}\big(\mathrm{y}(\theta^\top x)\big)$ for both $y \in \{+1, -1\}$.



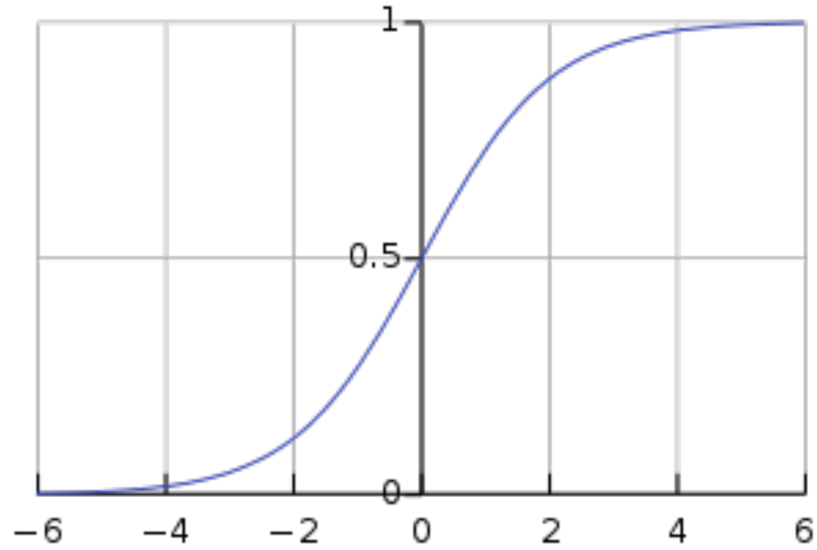
Label Predictions

Find out which label is more probable.

$$\mathbb{P}(\, y = +1 \mid x \,) \geq \mathbb{P}(\, y = -1 \mid x \,) \iff h(x;\theta) \geq \frac{1}{2}$$

If $h(x;\theta) \geq \frac{1}{2}$, then we predict the label of x is $y = +1$.

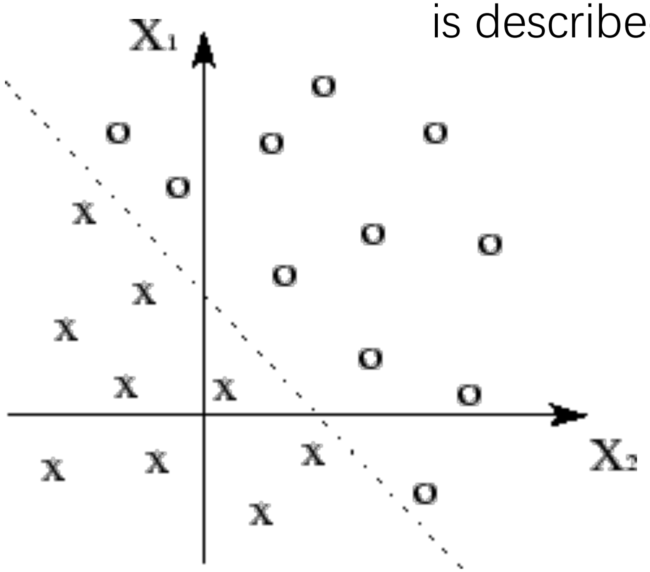
If $h(x;\theta) < \frac{1}{2}$, then we predict the label of x is $y = -1$.



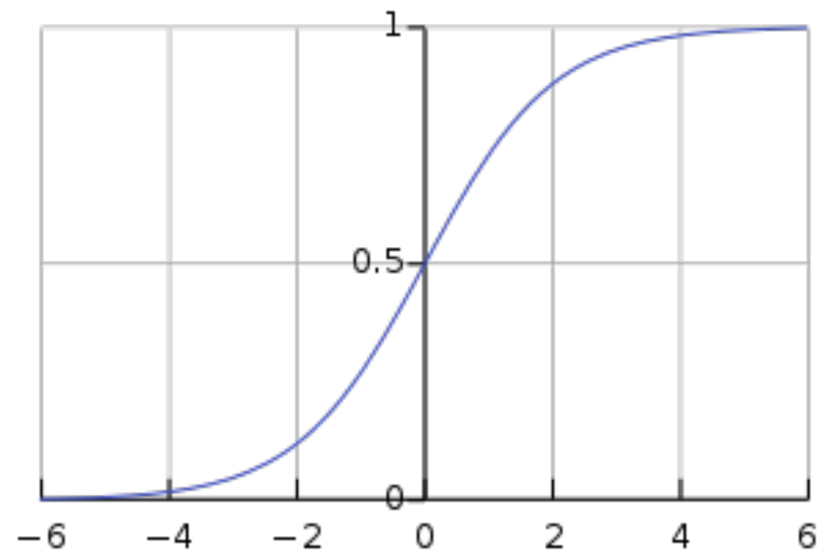
Decision Boundary

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

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



$\theta^\top x = 0$



Likelihood

$L(\theta; \mathcal{S}_n)$ is called the *likelihood* of the data $\mathcal{S}_n$.

Probability of label y given feature x

$$\mathbb{P}(y|x) = \text{sigmoid}\big(y(\theta^\top x)\big).$$

Probability of labels $y^{(1)}, \dots, y^{(n)}$ given features $x^{(1)}, \dots, x^{(n)}$

$$\begin{aligned} L(\theta; \mathcal{S}_n) &= \mathbb{P}\big(y^{(1)}, \dots, y^{(n)} | x^{(1)}, \dots, x^{(n)}\big) \\ &= \mathbb{P}\big(y^{(1)} | x^{(1)}\big) \times \cdots \times \mathbb{P}\big(y^{(n)} | x^{(n)}\big) \\ &= \prod_{(x,y) \in \mathcal{S}_n} \mathbb{P}(y|x) \end{aligned}$$

Maximizing $L(\theta; \mathcal{S}_n)$ is the same as maximizing $\log L(\theta; \mathcal{S}_n)$,
which is the same as minimizing $-\frac{1}{n} \log L(\theta; \mathcal{S}_n)$.

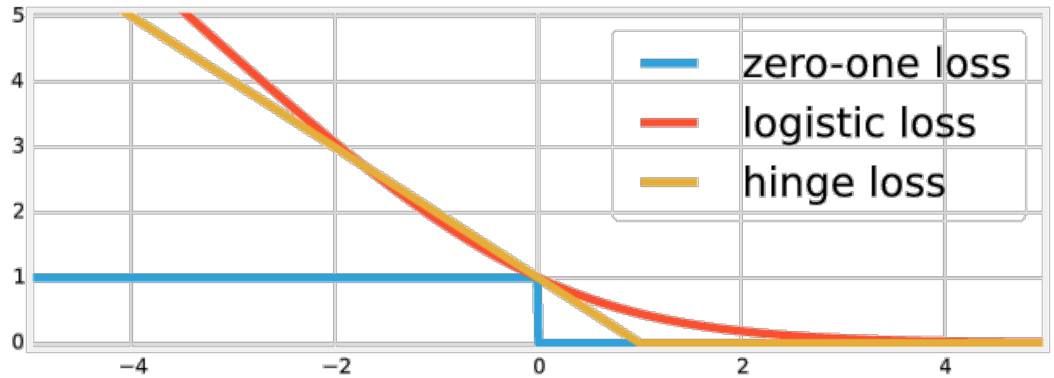
Logistic Loss

Minimize the training loss

$$
\begin{aligned}
\mathcal{L}_n(\theta; \mathcal{S}_n) &= -\frac{1}{n} \log L(\theta; \mathcal{S}_n) \\
&= -\frac{1}{n} \log \prod_{(x,y)\in\mathcal{S}_n} \mathbb{P}(y|x) \\
&= -\frac{1}{n} \sum_{(x,y)\in\mathcal{S}_n} \log \mathbb{P}(y|x) \\
&= -\frac{1}{n} \sum_{(x,y)\in\mathcal{S}_n} \log \frac{1}{1+e^{-y(\theta^\top x)}} \\
&= \frac{1}{n} \sum_{(x,y)\in\mathcal{S}_n} \log\left(1 + e^{-y(\theta^\top x)}\right) \\
&= \frac{1}{\mathrm{n}} \sum_{(x,y)\in\mathcal{S}_n} \mathrm{Loss}\left(y(\theta^\top x)\right)
\end{aligned}
$$

$\mathrm{Loss}(z) = \log(1 + e^{-z})$
is the *logistic loss*.

Loss Functions



Training Loss

$$\mathcal{L}_n(\theta) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \mathrm{Loss}\big(y(\theta^\top x)\big)$$

Zero-One Loss

$$\mathrm{Loss}_{01}(z) = [\![\, z \le 0 \,]\!]$$

Perceptron Algorithm

Hinge Loss

$$\mathrm{Loss}_{\mathrm{H}}(z) = \max\{1 - z, 0\}$$

Hinge Loss Algorithm

Logistic Loss

$$\mathrm{Loss}_{\mathrm{L}}(z) = \log(1 + e^{-z})$$

Logistic Regression

Some folks use $\log_2$ instead of $\log_e$ so that $\mathrm{Loss}_{\mathrm{L}}(0) = 1$.

Gradient

$$\text{Loss}_\text{L}(z) = \log(1 + e^{-z})$$

$$\nabla_z \text{Loss}_\text{L}(z) = \frac{-e^{-z}}{1+e^{-z}} = \frac{-1}{e^{z}+1} = -\text{sigmoid}(-z) = \text{sigmoid}(z) - 1$$

By chain rule, the point gradient $\nabla_\theta \mathcal{L}_1(\theta; x, y)$ is

$$\begin{aligned}
\nabla_\theta \text{Loss}_\text{L}\left(y(\theta^\top x)\right) \quad &= yx(\text{sigmoid}(z) - 1) \\
&= \begin{cases} x(h(x;\theta) - 1) & \text{if } y = +1, \\ x(h(x;\theta) - 0) & \text{if } y = -1. \end{cases} \\
&= x(h(x;\theta) - [\![y = 1]\!])
\end{aligned}$$

Training Gradient

$$\begin{aligned}\nabla_\theta \mathcal{L}_n(\theta; \mathcal{S}_n) \quad &= \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \nabla_\theta \mathcal{L}_1(\theta; x, y) \\ &= \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} x(h(x;\theta) - [\![y = 1]\!])\end{aligned}$$

Compare this with the training gradient for least squares

$$\nabla_\theta \mathcal{L}_n(\theta; \mathcal{S}_n) \quad = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} x(f(x;\theta) - y)$$

Gradient Descent

1. Initialize θ randomly.
2. Update $\theta \longleftarrow \theta - \frac{\eta_k}{n} \sum_{(x,y)\in\mathcal{S}_n} x(h(x;\theta) - [\![y = 1]\!])$
3. Repeat (2) until convergence.
   (e.g. when improvement in $\mathcal{L}_n(\theta)$ is small enough)

Logistic Regression

1. Training Set (Not Necessarily Linearly Separable)

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \dots, \left(x^{(n)}, y^{(n)}\right)$$

2. Model (Set of Sigmoid Neurons)

$$h(x;\theta) = \mathrm{sigmoid}(\theta_1 x_1 + \cdots + \theta_d x_d)$$

3. Training Loss (Logistic Loss)

$$\mathcal{L}_n(\theta) = \frac{1}{n} \sum_{(x,y)\in\mathcal{S}_n} \log\left(1 + e^{-y(\theta^\top x)}\right)$$

3. Algorithm (Gradient Descent)

$$\theta \longleftarrow \theta - \frac{\eta_k}{n} \sum_{(x,y)\in\mathcal{S}_n} x\left(h(x;\theta) - [\![y = 1]\!]\right)$$

Summary

- Linear Classification
  - Test Loss
  - Zero-One Loss
  - Decision Region
  - Decision Boundary
  - Linearly Separable
- Perceptron Algorithm
  - Perceptron
  - Training Loss
  - Mistake-Driven
  - Perceptron Algorithm
  - Only for Linearly Separable Data

- Hinge Loss
  - Hinge Loss
  - Gradient Descent
  - Hinge Loss SGD Algorithm
  - Differences with Perceptron Algorithm
  - OK for Non Linearly Separable Data

Summary

- Logistic Regression
  - Sigmoid Function
  - Sigmoid Neuron
  - Label Probabilities
  - Label Predictions
  - Decision Boundary
  - Likelihood
  - Logistic Loss
  - Training Loss
  - Training Gradient

- Extensions
  - Categorical Features
  - Multiclass Classification
  - Multilayer Neural Network
  - Kaggle

Intended Learning Outcomes

Linear Classification

- Write down the form of the data, the model and the test loss.
- Given a classifier, identify its decision regions and boundaries.
- Define what it means for a data set to be linearly separable. Give an example of a data set that is not linearly separable.

Perceptron Algorithm

- Write down the zero-one loss, and plot its graph. Explain why we cannot apply gradient descent to the training loss.
- Describe what it means for an algorithm to be mistake-driven.
- Describe the perceptron algorithm, and apply it to a given data set. Explain why it only applies to linearly separable data.

Intended Learning Outcomes

Hinge Loss

- Write down the hinge loss, and plot its graph. Write down the training loss and its gradient.
- Describe the hinge loss SGD, and apply it to a given data set.
- List differences between the hinge loss SGD algorithm and the perceptron algorithm.
- Explain why the hinge loss SGD applies to non linearly separable data while the perceptron algorithm does not.

Intended Learning Outcomes

Logistic Regression

- Write down the sigmoid function and logistic loss function.
- Describe the model as a set of sigmoid neurons which predict the probability of the labels given the features.
- Derive the label predictions from the label probabilities. Describe the decision boundary.
- Derive the training loss from the likelihood of the data, and write it in terms of the logistic loss.
- Derive the training gradient, and describe an algorithm for performing logistic regression.