CMPSC 448: Machine Learning

Lecture 10. Logistic Regression

Rui Zhang

Fall 2021

PennState

Recap: linear models for binary classification

A linear classifier is specified by a weight vector $\boldsymbol{w} \in \mathbb{R}^d$

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \ \boldsymbol{w}^\top \boldsymbol{x} = \sum_{i=1}^d w_i x_i \geq 0 \\ -1 & \text{if } \ \boldsymbol{w}^\top \boldsymbol{x} = \sum_{i=1}^d w_i x_i < 0 \end{cases}$$

It will be notationally simpler to use {+1, -1} instead of {1, 0}

Recap: linear models for binary classification

A linear classifier is specified by a weight vector $\boldsymbol{w} \in \mathbb{R}^d$

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} = \sum_{i=1}^d w_i x_i \geq 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} = \sum_{i=1}^d w_i x_i < 0 \end{cases}$$

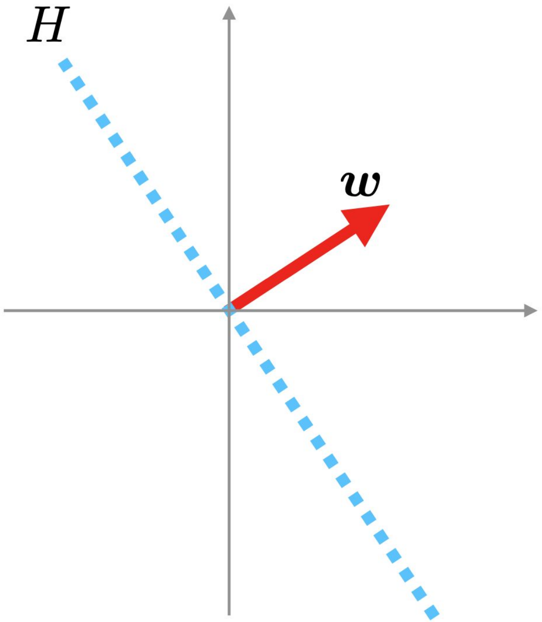
It will be notationally simpler to use {+1, -1} instead of {1, 0}

Interpretation: does a linear combination of input features exceed 0?

For $\boldsymbol{w} = [w_1, w_2, \ldots, w_d]^\top$ and $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^\top$

$$\boldsymbol{w}^\top \boldsymbol{x} = \sum_{i=1}^d w_i x_i >^{?} 0$$

Geometry of linear models for classification

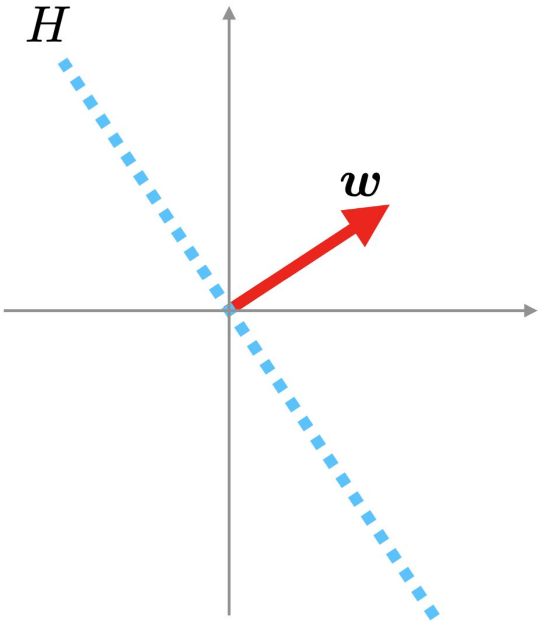


A hyperplane in $\mathbb{R}^d$ is a linear subspace:

A $\mathbb{R}^2$-hyperplane is a line

A $\mathbb{R}^3$-hyperplane is a plane

Geometry of linear models for classification



A hyperplane in $\mathbb{R}^d$ is a linear subspace:

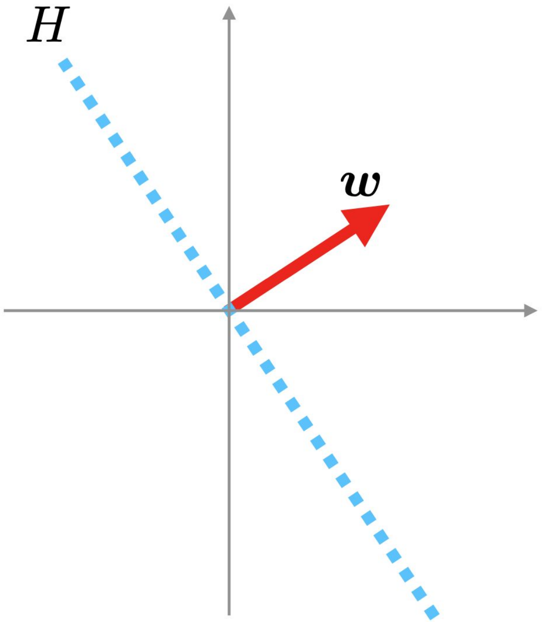
A $\mathbb{R}^2$-hyperplane is a line

A $\mathbb{R}^3$-hyperplane is a plane

A hyperplane can be specified by a (nonzero) normal vector

$$\boldsymbol{w} \in \mathbb{R}^d, \|\boldsymbol{w}\|_2 = 1$$

Geometry of linear models for classification



A hyperplane in $\mathbb{R}^d$ is a linear subspace:

A $\mathbb{R}^2$-hyperplane is a line

A $\mathbb{R}^3$-hyperplane is a plane

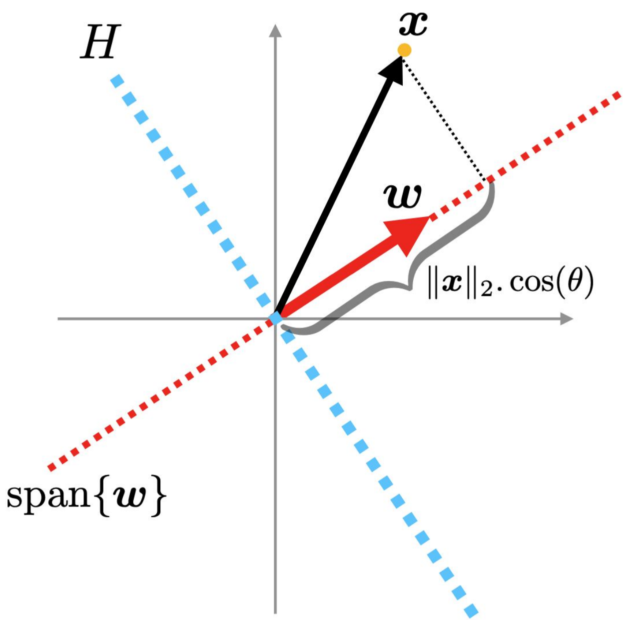
A hyperplane can be specified by a (nonzero) normal vector

$$\boldsymbol{w} \in \mathbb{R}^d, \ \|\boldsymbol{w}\|_2 = 1$$

The hyperplane with normal vector can be represented by set of points orthogonal to its normal vector:

$$H = \{\boldsymbol{x} \in \mathbb{R}^d \ : \ \boldsymbol{w}^\top \boldsymbol{x} = 0\}$$

Classification using linear models



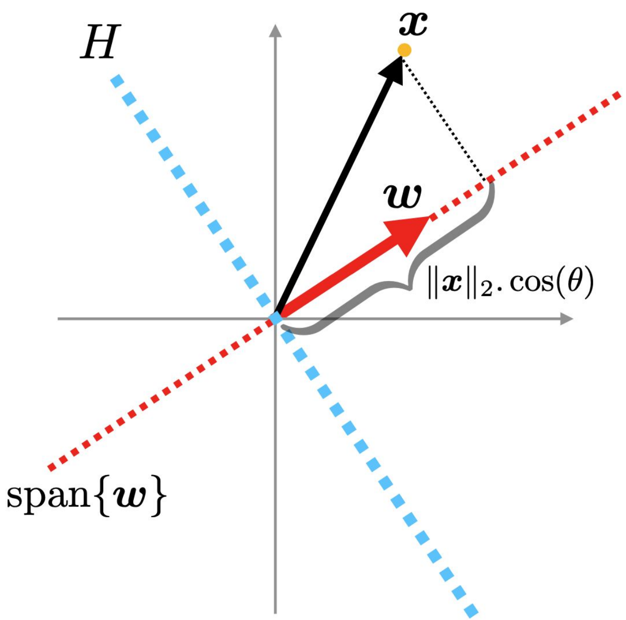
- Projection of $\boldsymbol{x}$ onto $\mathrm{span}\{\boldsymbol{w}\}$ (a line) has coordinate

$$\|\boldsymbol{x}\|_2 \cos(\theta)$$

where

$$\cos(\theta) = \frac{\boldsymbol{w}^\top \boldsymbol{x}}{\|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2}$$

Classification using linear models



- Projection of $\boldsymbol{x}$ onto $\mathrm{span}\{\boldsymbol{w}\}$ (a line) has coordinate

$$\|\boldsymbol{x}\|_2 \cos(\theta)$$

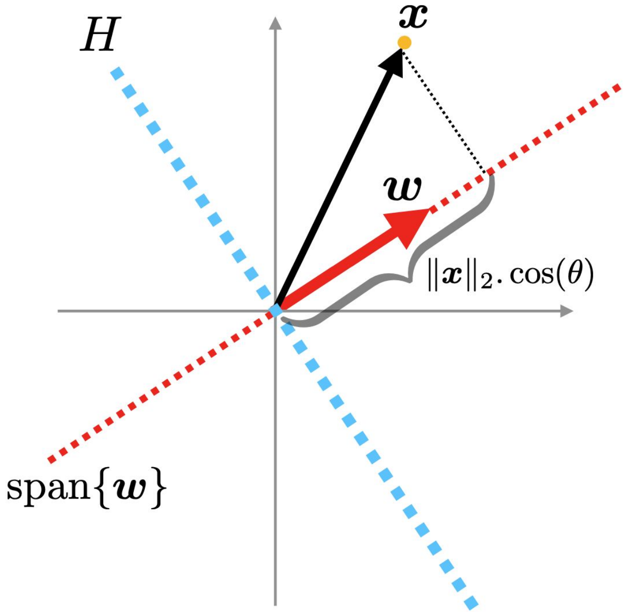
where

$$\cos(\theta) = \frac{\boldsymbol{w}^\top \boldsymbol{x}}{\|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2}$$

- Decision boundary is hyperplane oriented by $\boldsymbol{w}$

$$\begin{aligned} & \boldsymbol{w}^\top \boldsymbol{x} > 0 \\ \iff & \|\boldsymbol{x}\|_2 \cos(\theta) > 0 \\ \iff & \boldsymbol{x} \text{ on same side of } H \text{ as } \boldsymbol{w} \end{aligned}$$

Classification using linear models



- Projection of $\boldsymbol{x}$ onto $\mathrm{span}\{\boldsymbol{w}\}$ (a line) has coordinate

$$\|\boldsymbol{x}\|_2 \cos(\theta)$$

where

$$\cos(\theta) = \frac{\boldsymbol{w}^\top \boldsymbol{x}}{\|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2}$$

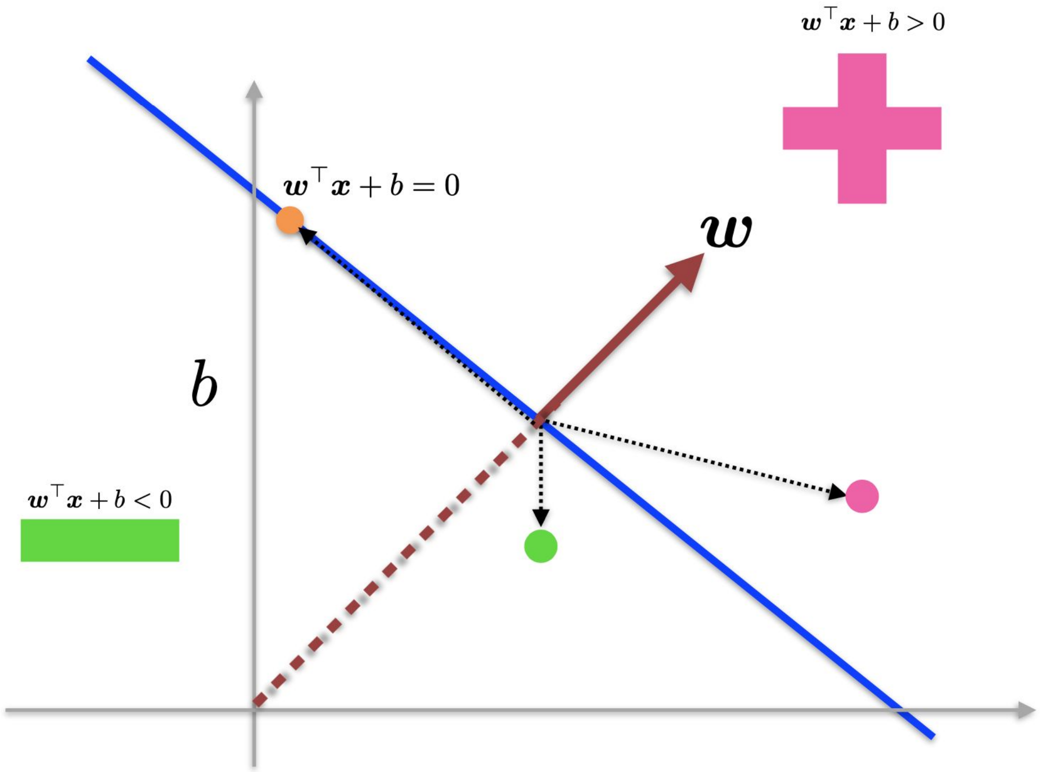
- Decision boundary is hyperplane oriented by $\boldsymbol{w}$

$$\begin{aligned} &\boldsymbol{w}^\top \boldsymbol{x} > 0 \\ \iff\ & \|\boldsymbol{x}\|_2 \cos(\theta) > 0 \\ \iff\ & \boldsymbol{x} \text{ on same side of } H \text{ as } \boldsymbol{w} \end{aligned}$$

Q: What should we do if we want hyperplane decision boundary that does not (necessarily) go through origin?
A: Add bias.

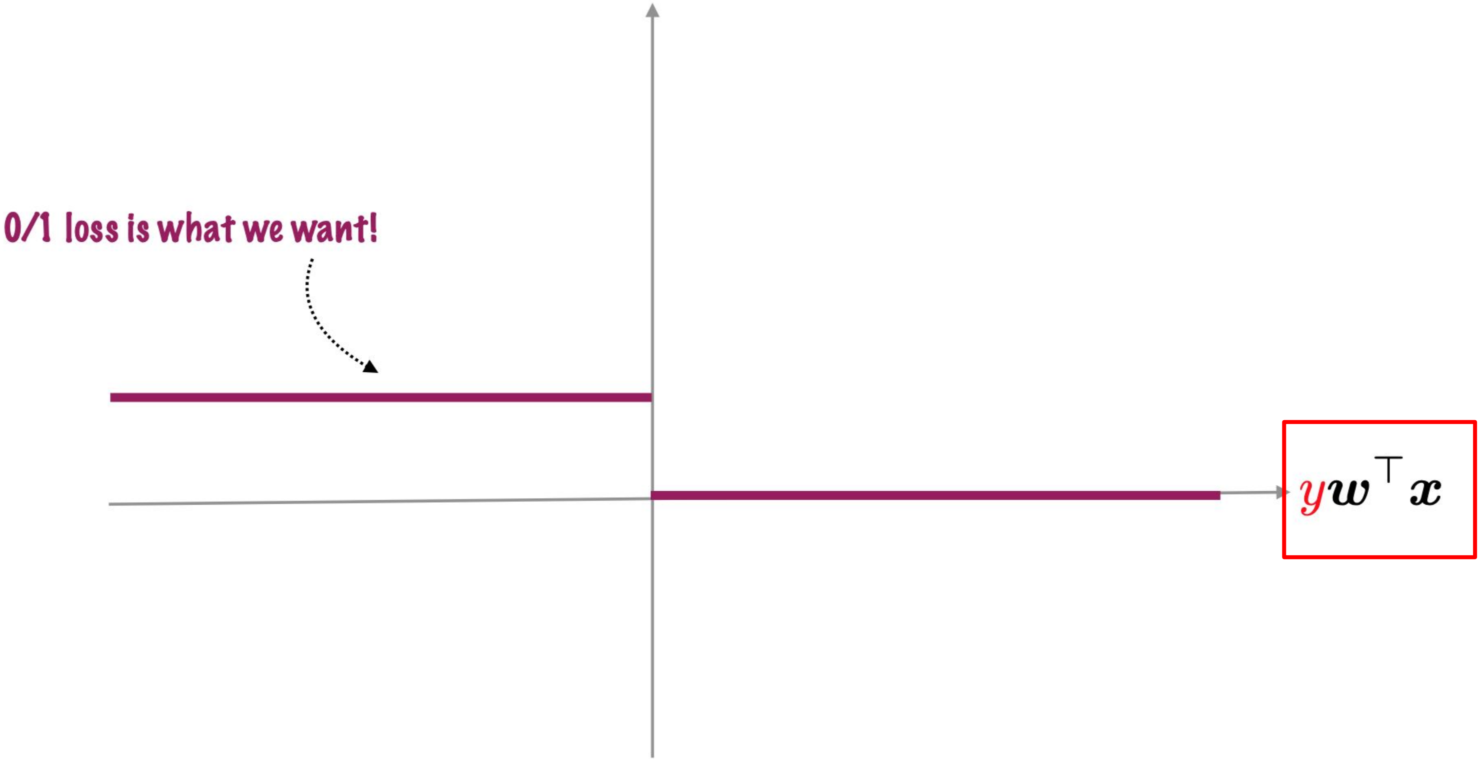
Linear models and hyperplanes

Every hyperplane is a decision boundary that splits the space into two subspaces:



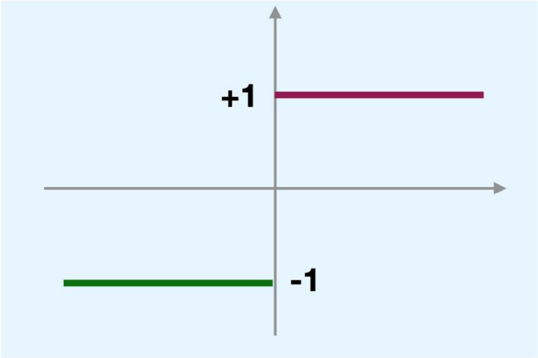
0/1 loss for classification

Combining two cases results in the 0/1 loss function for classification:



Perceptron as an artificial neuron

Perceptron is an **algorithm** to <u>learn</u> **linear models**!

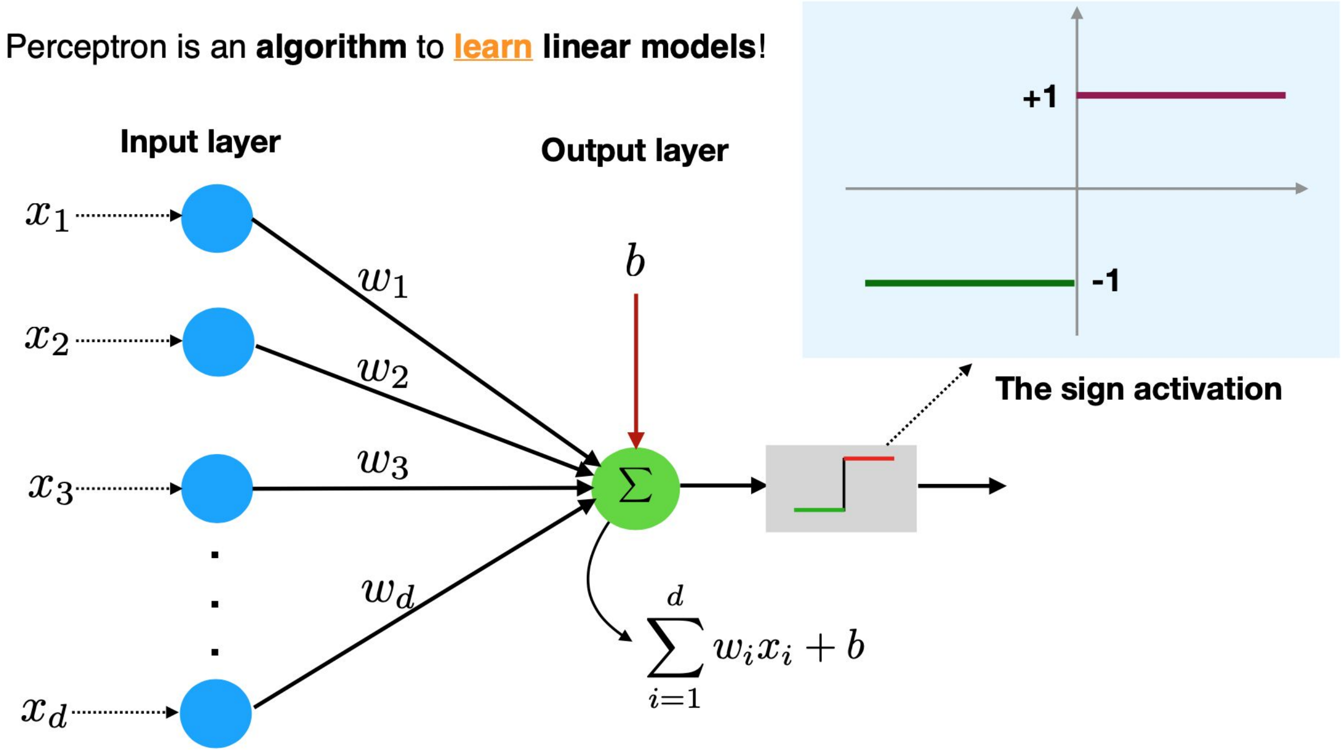


The sign activation

$$\sum_{i=1}^{d} w_i x_i + b$$

It only works if the data is linearly separable (an assumption that does not hold for many real data)

Perceptron as an artificial neuron



It only works if the data is linearly separable (an assumption that does not hold for many real data)

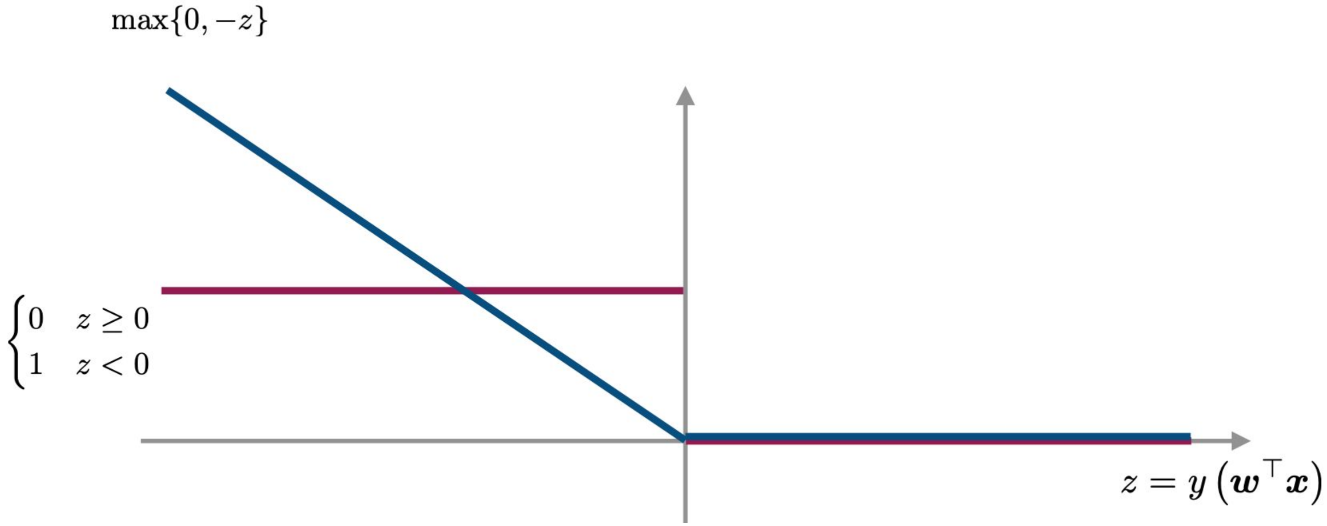
Convex approximation of 0/1 loss

- We want 0/1 loss for classification
- Unfortunately 0/1 loss function is hard to optimize
  - is a non-convex function
  - The gradient is zero
- Let's replace 0/1 loss with approximate convex functions!

The Perceptron loss versus 0/1 loss

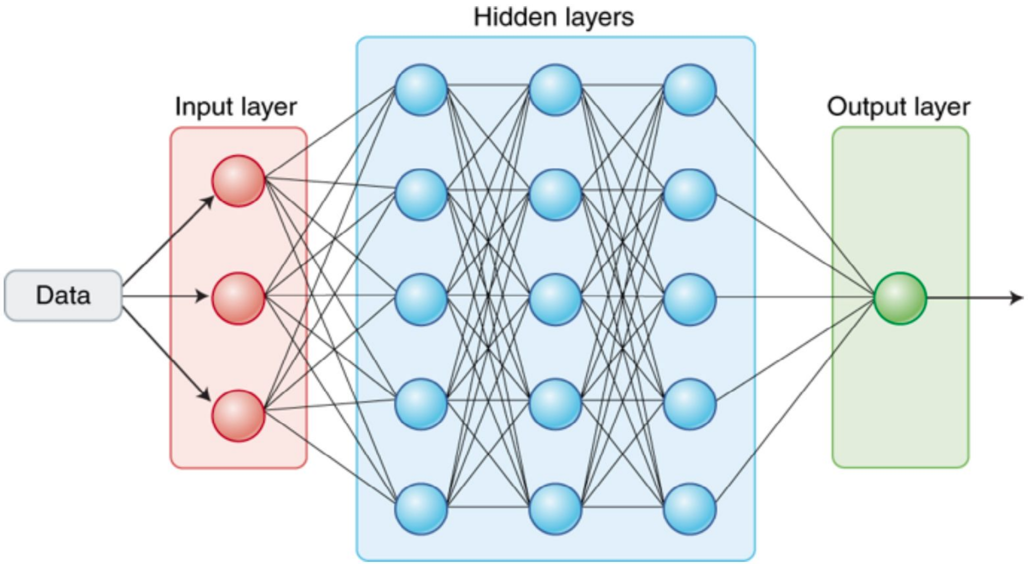
The Perceptron algorithm is optimizing the following loss function in training:

$$\max\{0, -y\left(\boldsymbol{w}^{\top}\boldsymbol{x}\right)\}$$



Computational virtues (convex versus non-convex)

Multiple layer perceptron



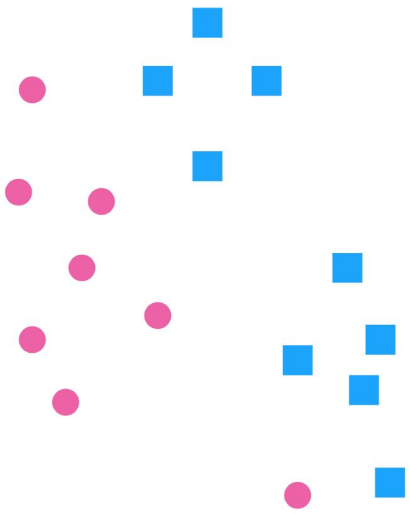
Training: use back-propagation (gradient descent + change rule) to learn the parameters for all layers

Introduces new challenges: huge amount of data + hyperparameter + vanishing gradient issue + high-computational power + more effective regularization + etc

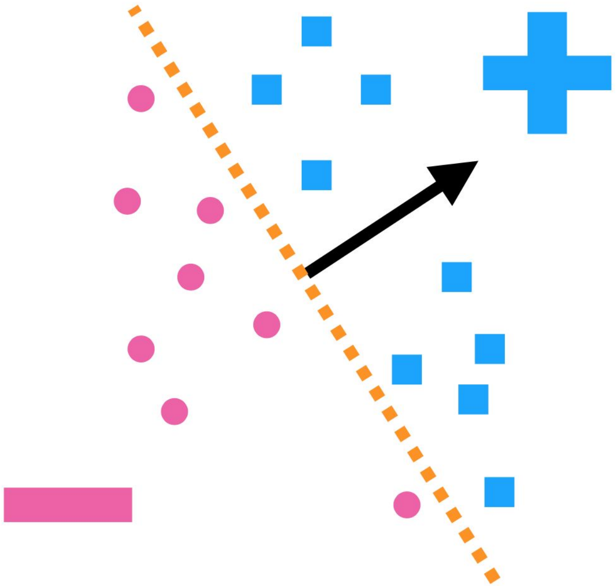
Outline

- Predicting probability
- Logistic regression for binary classification
- MLE for Logistic Regression
- The cross-entropy loss function
- Regularized logistic regression
- The decision boundary of logistic regression

The lack of confidence



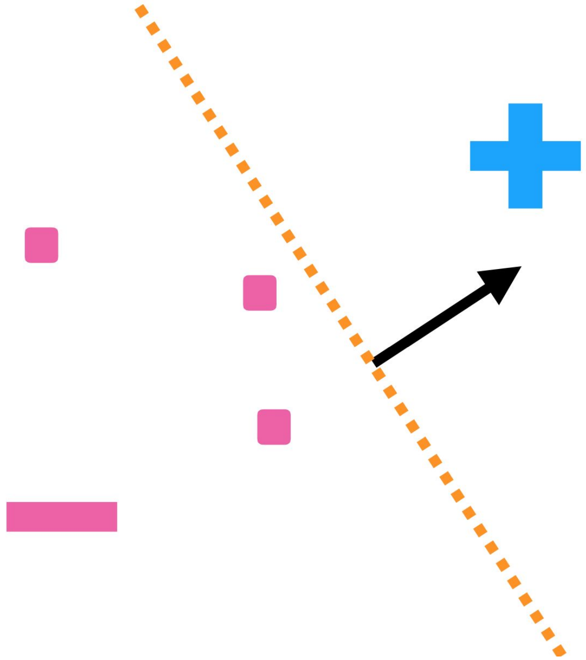
Data



Training

Assume we learn a linear classifier for data given here.

The lack of confidence

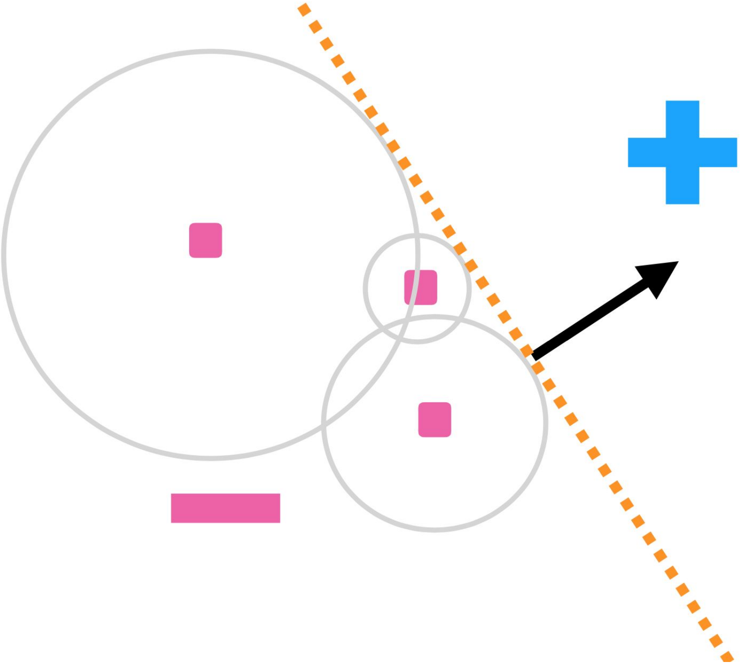


Three test points are given. All will be classifies as -1.

But, which one we are more confident in its label?

Is there any measure of this confidence/probability?

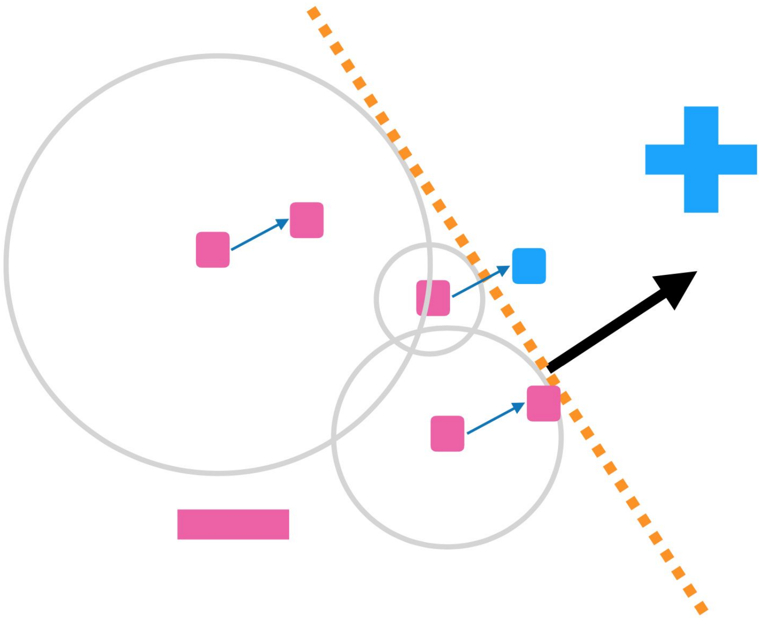
The measure of confidence



The larger the margin (distance from hyperplane), more confidence the algorithm will be in predicted label, which can be computed as

$$y(\boldsymbol{w}^\top \boldsymbol{x})$$

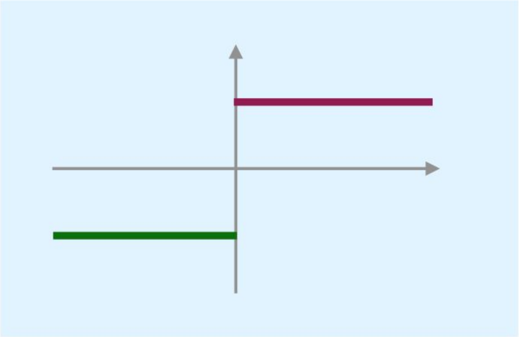
The measure of confidence



Noise tolerance: the amount of noise we can add to each data point such that the label stays same!

From score to probability

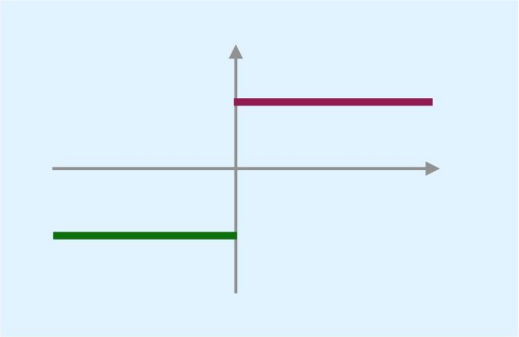
$\boldsymbol{w}^\top \boldsymbol{x}$ →



→ $$y = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} \geq 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases}$$

From score to probability

$$\boldsymbol{w}^\top \boldsymbol{x} \longrightarrow$$



$$\longrightarrow \quad y = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} \geq 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases}$$

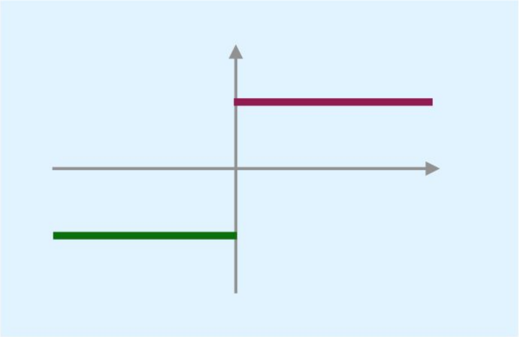
$$\boldsymbol{w}^\top \boldsymbol{x} \longrightarrow \quad \longrightarrow$$

$$\mathbb{P}[y = +1]$$

$$\mathbb{P}[y = -1] = 1 - \mathbb{P}[y = +1]$$

From score to probability

$\boldsymbol{w}^\top \boldsymbol{x} \longrightarrow$



$$y = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} \geq 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases}$$

$\boldsymbol{w}^\top \boldsymbol{x} \longrightarrow$

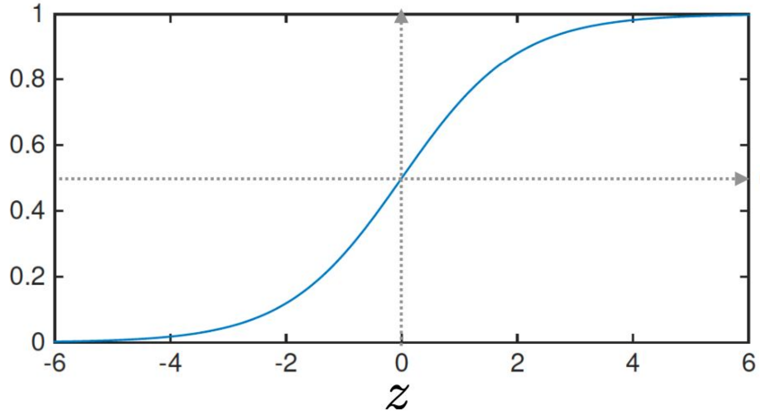
?

$$\mathbb{P}[y = +1]$$

$$\mathbb{P}[y = -1] = 1 - \mathbb{P}[y = +1]$$

From score to probability

$\boldsymbol{w}^\top \boldsymbol{x}$

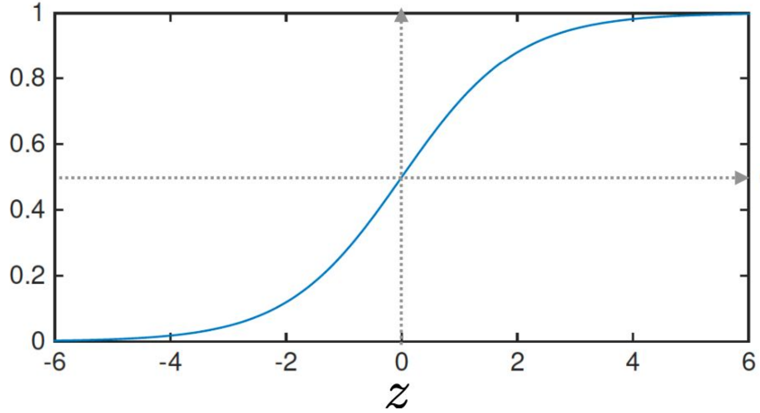


$\mathbb{P}[y = +1 \mid \boldsymbol{x}; \boldsymbol{w}]$

Logistic Function (aka Sigmoid Function)

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

$$\boldsymbol{w}^\top \boldsymbol{x} \longrightarrow$$



$$\longrightarrow \mathbb{P}[y = +1 \mid \boldsymbol{x}; \boldsymbol{w}]$$

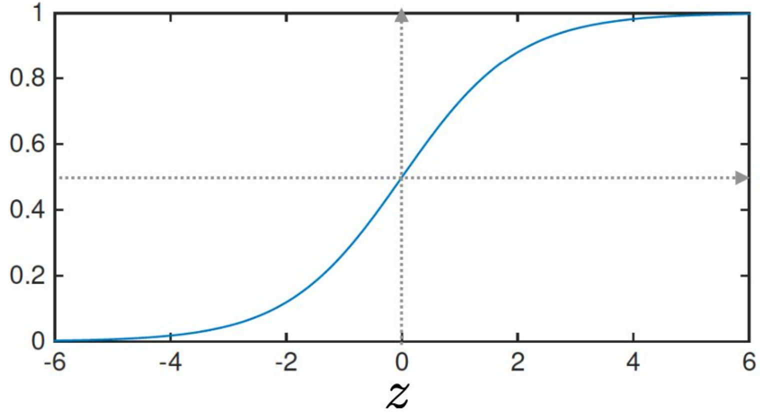
$$\begin{aligned} &\mathbb{P}\left[y = +1 | \boldsymbol{x}; \boldsymbol{w}\right] \\ &= \sigma(\boldsymbol{w}^\top \boldsymbol{x}) \\ &= \frac{1}{1 + e^{-\boldsymbol{w}^\top \boldsymbol{x}}} \end{aligned}$$

$$\begin{aligned} &\mathbb{P}\left[y = -1 | \boldsymbol{x}; \boldsymbol{w}\right] \\ &= 1 - \mathbb{P}\left[y = +1 | \boldsymbol{x}; \boldsymbol{w}\right] \\ &= \sigma(-\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}}} \end{aligned}$$

Logistic Function (aka Sigmoid Function)

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

$\boldsymbol{w}^\top \boldsymbol{x} \rightarrow$



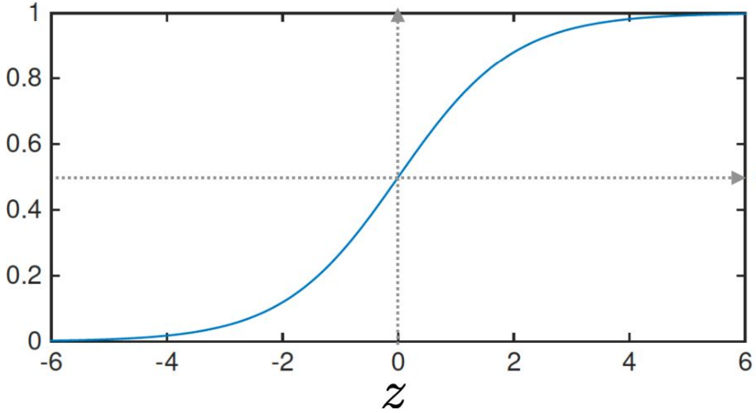
$\rightarrow \mathbb{P}[y = +1 \mid \boldsymbol{x}; \boldsymbol{w}]$

$$\boxed{\mathbb{P}\left[y|\boldsymbol{x};\boldsymbol{w}\right] = \frac{1}{1+e^{-y\boldsymbol{w}^\top \boldsymbol{x}}}}$$

Logistic Function (aka Sigmoid Function)

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

$\boldsymbol{w}^\top \boldsymbol{x}$ →



→ $\mathbb{P}[y = +1 \mid \boldsymbol{x}; \boldsymbol{w}]$

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

This “squashing function” known as the **logistic function** (or **sigmoid function**) turns linear prediction score into probability!

Logistic Regression

- Given a data for binary classification: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1 = \pm 1), (\boldsymbol{x}_2, y_2 = \pm 1), \ldots, (\boldsymbol{x}_n, y_n = \pm 1)\}$
- The conditional probability of the label given the input example $\boldsymbol{x}$ for a classifier with parameters vector $\boldsymbol{w}$ is expressed as:

$$\mathbb{P}\left[y|\boldsymbol{x};\boldsymbol{w}\right] = \frac{1}{1 + e^{-y\boldsymbol{w}^\top \boldsymbol{x}}}$$

where the $\boldsymbol{w} = \left[w_1, w_2, \ldots, w_d\right]^\top$ are the adjustable parameters.

Logistic Regression

- Given a data for binary classification: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1 = \pm 1), (\boldsymbol{x}_2, y_2 = \pm 1), \ldots, (\boldsymbol{x}_n, y_n = \pm 1)\}$
- The conditional probability of the label given the input example $\boldsymbol{x}$ for a classifier with parameters vector $\boldsymbol{w}$ is expressed as:

$$\mathbb{P}[y|\boldsymbol{x};\boldsymbol{w}] = \frac{1}{1 + e^{-y\boldsymbol{w}^\top \boldsymbol{x}}}$$

where the $\boldsymbol{w} = [w_1, w_2, \ldots, w_d]^\top$ are the adjustable parameters.

- If the probability of y being 1 is greater than 0.5, then we can predict $y = +1$
- This is the "logistic function" and model is called "logistic regression".

Logistic Regression

- Given a data for binary classification: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1 = \pm 1), (\boldsymbol{x}_2, y_2 = \pm 1), \ldots, (\boldsymbol{x}_n, y_n = \pm 1)\}$
- The conditional probability of the label given the input example $\boldsymbol{x}$ for a classifier with parameters vector $\boldsymbol{w}$ is expressed as:

$$\mathbb{P}\left[y|\boldsymbol{x};\boldsymbol{w}\right] = \frac{1}{1 + e^{-y\boldsymbol{w}^\top \boldsymbol{x}}}$$

where the $\boldsymbol{w} = \left[w_1, w_2, \ldots, w_d\right]^\top$ are the adjustable parameters.

- If the probability of y being 1 is greater than 0.5, then we can predict $y = +1$
- This is the "logistic function" and model is called "logistic regression".

In the terminology of statistics, this model is known as logistic "regression", although it should be emphasized that this is a model for classification rather than regression.

MLE for logistic regression

We can fit the logistic models using the maximum likelihood estimation (MLE).

$$\mathbb{P}\left[y|\boldsymbol{x};\boldsymbol{w}\right] = \frac{1}{1+e^{-y\boldsymbol{w}^\top \boldsymbol{x}}}$$

Recall training data are independently generated (i.i.d.), so the likelihood is:

$$\mathbb{P}\left[y_1, y_2, \ldots, y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \boldsymbol{w}\right] = \prod_{i=1}^{n} \mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]$$

MLE for logistic regression

We can fit the logistic models using the maximum likelihood estimation (MLE).

$$\mathbb{P}\left[y|\boldsymbol{x};\boldsymbol{w}\right]=\frac{1}{1+e^{-y\boldsymbol{w}^{\top}\boldsymbol{x}}}$$

Recall training data are independently generated (i.i.d.), so the likelihood is:

$$\mathbb{P}\left[y_1,y_2,\ldots,y_n\mid\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n;\boldsymbol{w}\right]=\prod_{i=1}^{n}\mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]$$

The likelihood measures the probability that the data were generated if $\boldsymbol{w}\in\mathbb{R}^d$ was the model to generate the labels in training data!

MLE for logistic regression

We can fit the logistic models using the maximum likelihood estimation (MLE).

$$\mathbb{P}\left[y|\boldsymbol{x};\boldsymbol{w}\right] = \frac{1}{1+e^{-{\color{red}y}\boldsymbol{w}^\top\boldsymbol{x}}}$$

Recall training data are independently generated (i.i.d.), so the likelihood is:

$$\mathbb{P}\left[y_1, y_2, \ldots, y_n \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \boldsymbol{w}\right] = \prod_{i=1}^{n} \mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]$$

The likelihood measures the probability that the data were generated if $\boldsymbol{w} \in \mathbb{R}^d$ was the model to generate the labels in training data!

By maximizing the likelihood over all parameter vectors $\boldsymbol{w} \in \mathbb{R}^d$ we can find the optimal classifier:

$$\arg\max_{\boldsymbol{w}} \quad \prod_{i=1}^{n} \mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]$$

Fitting logistic regression models using MLE

$$\log\left(\prod_{i=1}^{n}\mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]\right) = \sum_{i=1}^{n}\log\mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]$$

$$= \sum_{i=1}^{n}\log\frac{1}{1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}}$$

$$\log\frac{1}{z} = -\log z$$

$$= \sum_{i=1}^{n}-\log\left(1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}\right)$$

$$= -\sum_{i=1}^{n}\log\left(1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}\right)$$

Fitting logistic regression models using MLE

$$\log\left(\prod_{i=1}^{n}\mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]\right) = \sum_{i=1}^{n}\log\mathbb{P}\left[y_i|\boldsymbol{x}_i;\boldsymbol{w}\right]$$

$$= \sum_{i=1}^{n}\log\frac{1}{1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}}$$

$$\log\frac{1}{z} = -\log z \qquad = \sum_{i=1}^{n}-\log\left(1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}\right)$$

$$= -\sum_{i=1}^{n}\log\left(1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}\right)$$

The final MLE objective becomes (or minimizing the negative likelihood):

$$\arg\max_{\boldsymbol{w}\in\mathbb{R}^d}\ -\sum_{i=1}^{n}\log\left(1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}\right) \qquad \arg\min_{\boldsymbol{w}\in\mathbb{R}^d}\ \sum_{i=1}^{n}\log\left(1+e^{-y_i\boldsymbol{w}^\top\boldsymbol{x}_i}\right)$$

The cross-entropy loss function

The cross-entropy function to measure the error of model on a training sample:

$$\log\left(1+e^{-y\boldsymbol{w}^\top\boldsymbol{x}}\right)$$

The cross-entropy loss function

The cross-entropy function to measure the error of model on a training sample:

$$\log\ \left(1 + e^{-y\boldsymbol{w}^\top \boldsymbol{x}}\right)$$

It looks complicated and ugly, but,

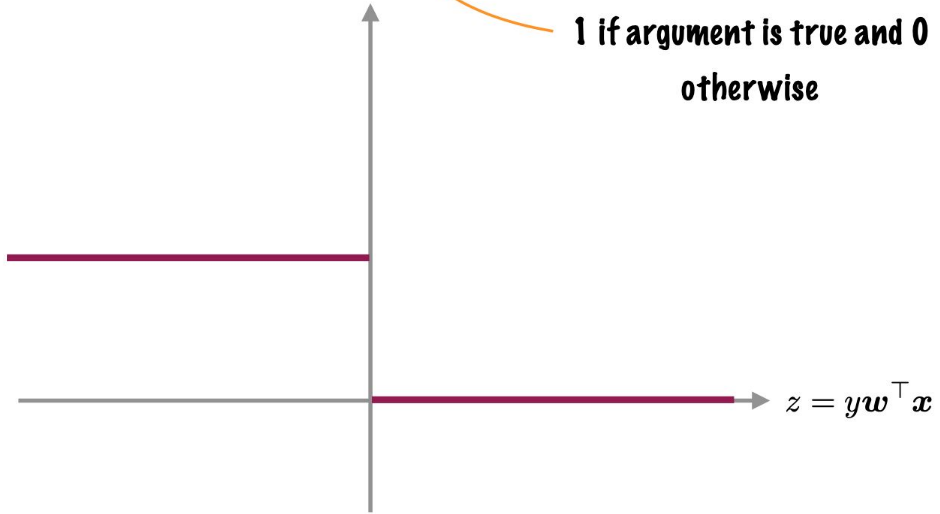
- It is based on an intuitive probabilistic interpretation.
- It is convex.
- However, unlike linear regression, there is no closed-form solution (even though it is convex).
- It is very convenient and mathematically easy to minimize using Gradient Descent (GD).

Optimization viewpoint of linear classifiers

The original goal of linear binary classification is to find the parameters of a linear model that minimizes the number of mistakes over training data (0/1 loss):

$$\arg \min_{\boldsymbol{w} \in \mathbb{R}^d,\ b \in \mathbb{R}} \sum_{i=1}^{n} \mathbb{I}\left[\text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) \neq y_i\right]$$

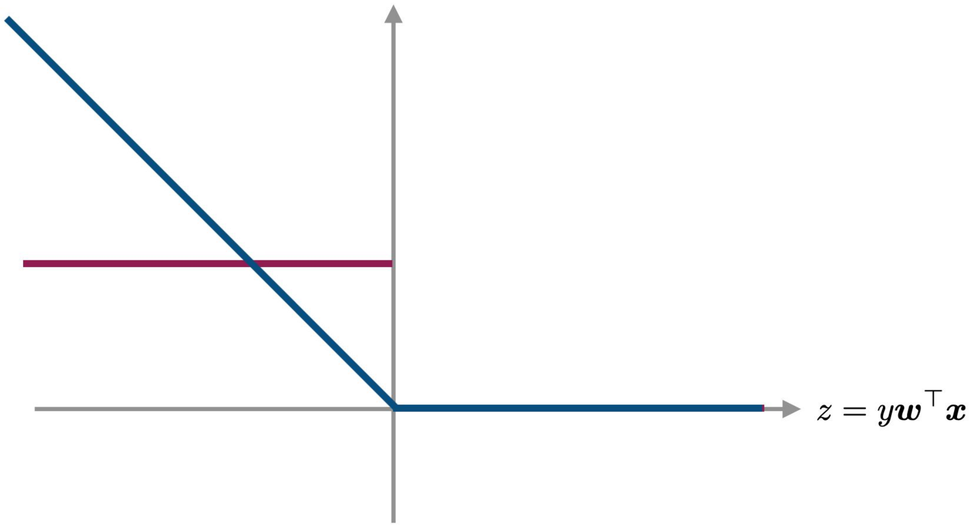
Indicator functions: returns 1 if argument is true and 0 otherwise



Optimization viewpoint of linear classifiers

Perceptron minimizes a convex relaxation version of 0/1 loss:

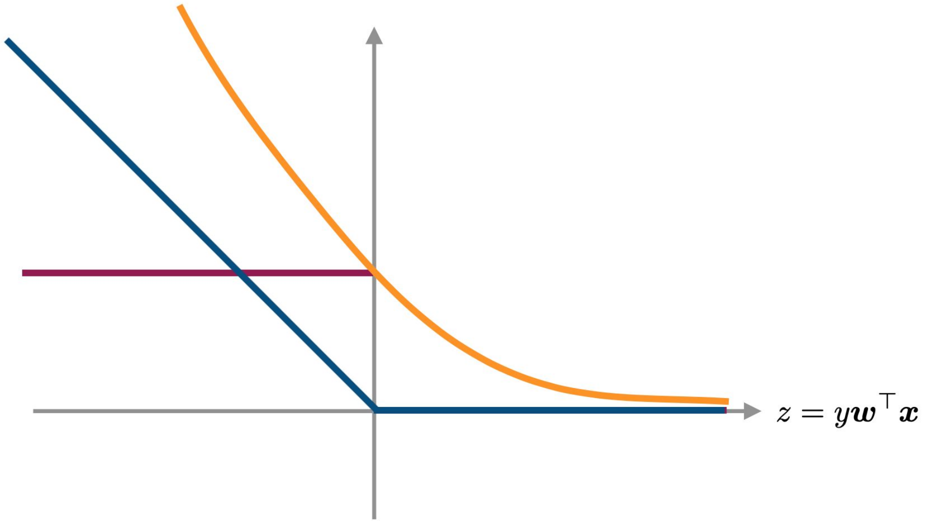
$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d,\ b\in\mathbb{R}} \sum_{i=1}^{n} \max\left(0, -y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\right)$$



Optimization viewpoint of linear classifiers

Logistic regression minimizes another convex relaxation version of 0/1 loss, and it is a smooth version of loss used in Perceptron:

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d,\ b\in\mathbb{R}} \sum_{i=1}^{n} \log\left(1+e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)}\right)$$

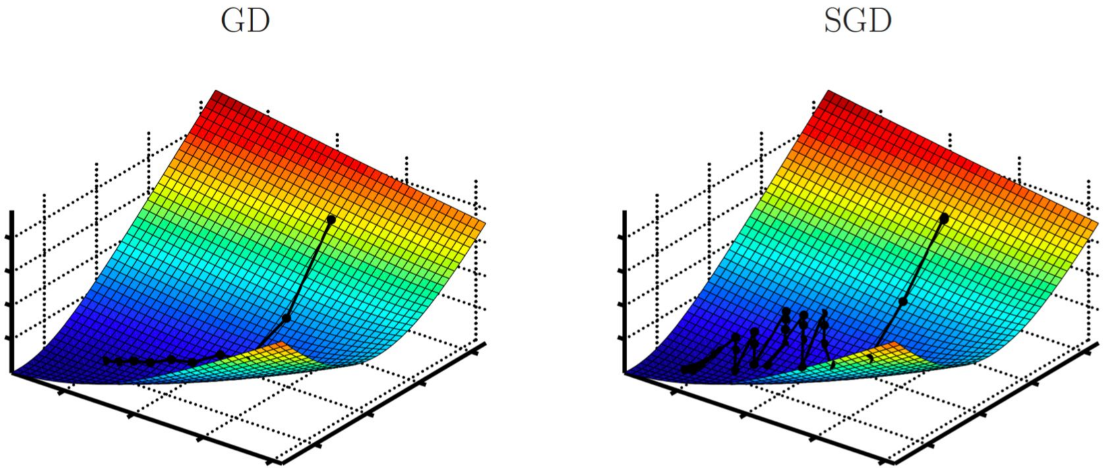


GD and SGD

Our task now is to solve the following optimization problem to find optimal model:

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{w}^\top \boldsymbol{x}_i}\right)$$

The function is a convex function of the parameters and we can use GD or SGD to find the minimizer.



Overfitting in Logistic Regression

MLE can exhibit severe **overfitting** for data sets that are **linearly separable**.

- In fact, there are infinite number separating hyperplanes
- MLE will give solution for weights go to infinity!

$$|\boldsymbol{w}| \to \infty, \quad \text{then we have} \quad P(y_i|\boldsymbol{w}, \boldsymbol{x}_i) = \sigma(y_i \boldsymbol{w}^\top \boldsymbol{x}_i) \to 1$$

Regularization

MLE can exhibit severe **overfitting** for data sets that are **linearly separable**.

- In fact, there are infinite number separating hyperplanes
- MLE will give solution for weights go to infinity!

$$|\boldsymbol{w}| \to \infty, \quad \text{then we have} \quad P(y_i|\boldsymbol{w}, \boldsymbol{x}_i) = \sigma(y_i \boldsymbol{w}^\top \boldsymbol{x}_i) \to 1$$

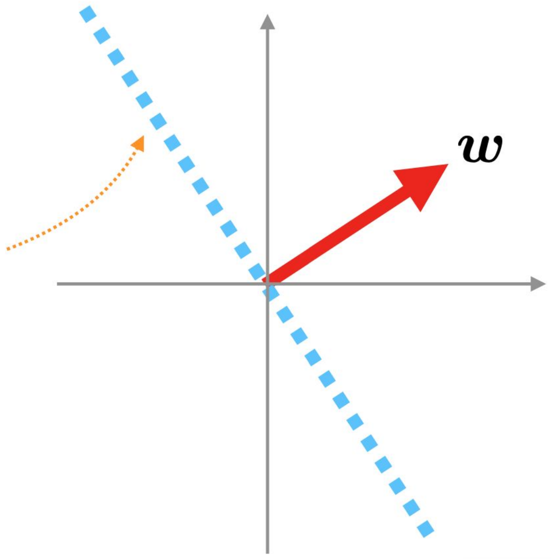
Penalize the model parameters to avoid overfitting:

$$\arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \quad \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{w}^\top \boldsymbol{x}_i}\right) + \lambda \|\boldsymbol{w}\|_2^2$$

The decision boundary

Recall that for Perceptron the decision boundary of a model parametrized by $\boldsymbol{w}$ is the set of all points for which:

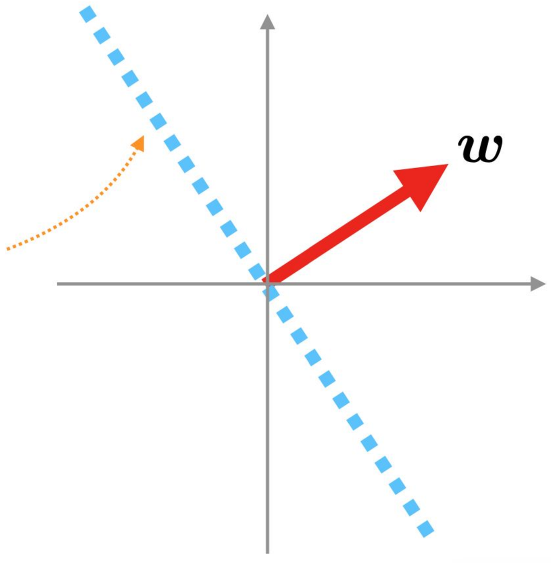
$$\boldsymbol{w}^\top \boldsymbol{x} = 0$$



The decision boundary

Recall that for Perceptron the decision boundary of a model parametrized by $\boldsymbol{w}$ is the set of all points for which:

$$\boldsymbol{w}^\top \boldsymbol{x} = 0$$



For logistic regression, the decision boundary is the set of all points for which

$$\mathbb{P}[y = -1] = \mathbb{P}[y = +1] = \frac{1}{2}$$

How does the decision boundary for logistic regression look like?

The decision boundary

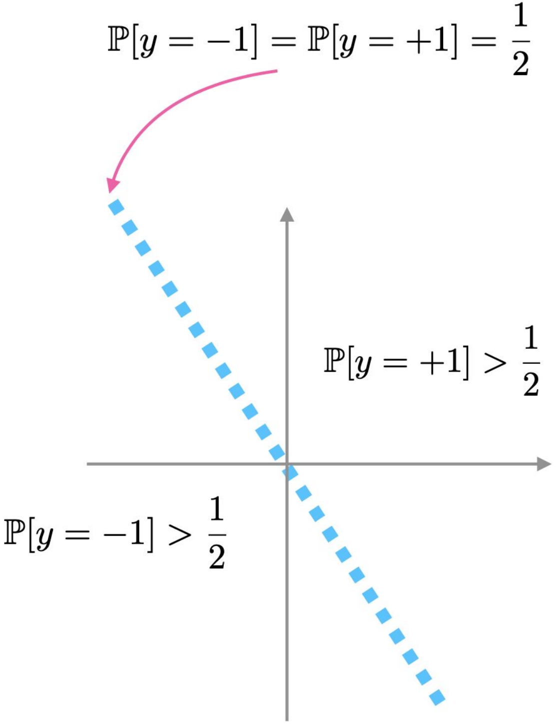
How does the decision boundary for logistic regression look like?

$$\mathbb{P}[y=+1] = \frac{1}{1+e^{-\boldsymbol{w}^\top \boldsymbol{x}}} = \frac{1}{2}$$

$$e^{-\boldsymbol{w}^\top \boldsymbol{x}} = 1$$

$$-\boldsymbol{w}^\top \boldsymbol{x} = 0$$

$$\boldsymbol{w}^\top \boldsymbol{x} = 0$$



The decision boundary of logistic regression is linear!

Note on Label Notations

In this class note, we use

$$\mathcal{Y} = \{-1, +1\}$$

It is also possible (and actually, more popular) to use

$$\mathcal{Y} = \{0, 1\}$$

You will work with this setting in your HW3.

Perceptron versus logistic regression

	Perceptron	Logistic regression
Model	$\boldsymbol{w}^\top \boldsymbol{x} + b$	$\boldsymbol{w}^\top \boldsymbol{x} + b$
Prediction	$\mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$	$\sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$
Training loss function	$\max\left(0, -y(\boldsymbol{w}^\top \boldsymbol{x} + b)\right)$	$\log\left(1 + e^{-y(\boldsymbol{w}^\top \boldsymbol{x} + b)}\right)$
Decision boundary	Linear	Linear
Regularization	NA (data needs to be separable)	$\|\boldsymbol{w}\|_2^2$ or $\|\boldsymbol{w}\|_1$
Optimization	GD + SGD	GD + SGD

Logistic regression in scikit-learn

```
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split

# Assume data are in X (features) and y (binary labels)
# Scikit Logistic Regression
scikit_log_reg = LogisticRegression()
scikit_log_reg.fit(X, y)
```