Linear Classifiers, Logistic Regression

CSC 461: Machine Learning

Fall 2022

Prof. Marco Alvarez
University of Rhode Island

Linear classifiers

Linear classifiers

- Discriminative
  - ✓ Perceptron
  - ✓ Logistic regression
  - ✓ Support vector machines
- Generative
  - ✓ Linear discriminant analysis
  - ✓ Naive bayes

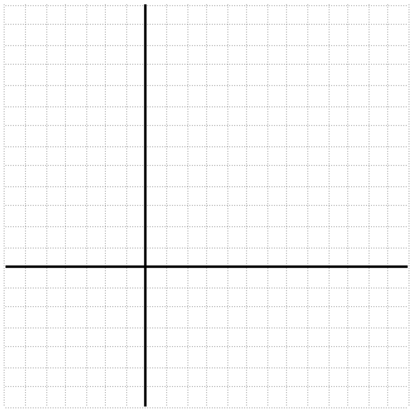
Binary classification

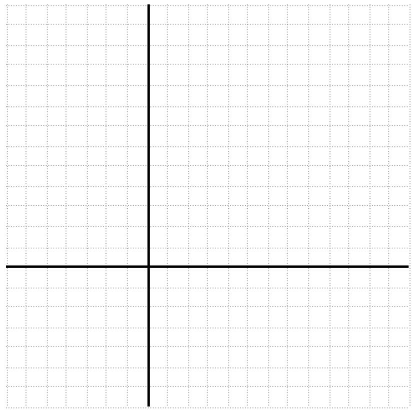
$$\mathscr{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$$

$\mathbf{x}^{(i)} \in \mathbb{R}^d$ $\qquad$ $y^{(i)} \in \{-1, +1\}$

X1	X2	Y
0.5	0.1	+1
0.3	0.9	−1
0.3	0.875	−1
0.45	0.15	+1
…	…	…

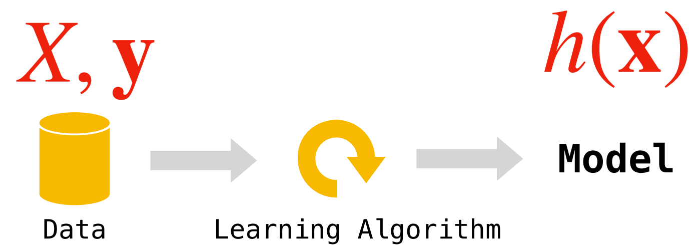
Plots (regression x classification)





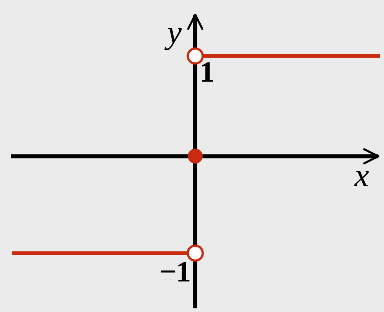
Binary classification goal

- Learn a **decision boundary** such that two classes can be separated



The *sign* function

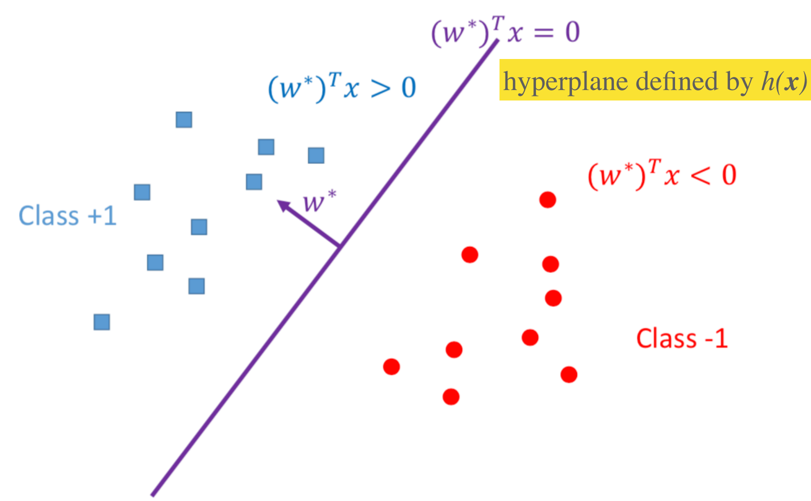
$$sign(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ +1 & \text{if } x > 0 \end{cases}$$



$$h(\mathbf{x}) = sign\left(\mathbf{w}^T\mathbf{x}\right)$$

Image credit: Wikipedia

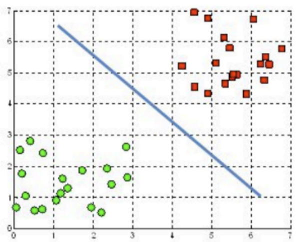
Decision boundary



credit: yingyu liang, cos 495, princeton

Decision boundary

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



$$0 = b + w_1 x_1 + w_2 x_2$$

$$x_2 = -\frac{b}{w_2} - \frac{w_1}{w_2} x_1$$

Image credit: https://mc.ai/why-activation-function-is-used-in-neural-network/

Absorbing the bias

$$h(\mathbf{x}) = sign\left(\mathbf{w}^T\mathbf{x} + b\right)$$

$$= sign\left(\sum_{i=1}^{d} w_i x_i + b\right)$$

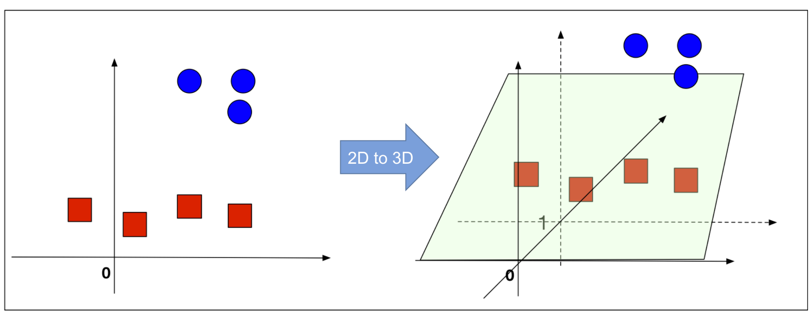
$$x_0 = 1, \quad w_0 = b$$

$$h(\mathbf{x}) = sign\left(\sum_{i=0}^{d} w_i x_i\right)$$

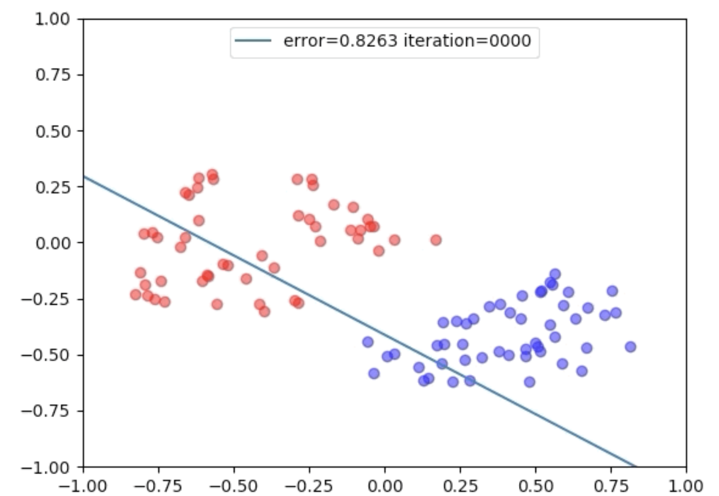
$$= sign\left(\mathbf{w}^T\mathbf{x}\right)$$

X0	X1	X2	Y
1	0.5	0.1	+1
1	0.3	0.9	−1
1	0.3	0.875	−1
1	0.25	0.561	−1
1	0.45	0.15	+1
…	…	…	…

Absorbing the bias



Learning



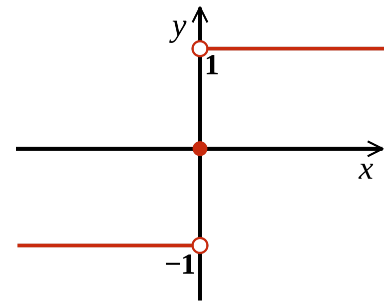
Example

- Provide a solution (weight vector)

x_0	x_1	x_2	y
1	0	0	-1
1	0	1	-1
1	1	0	-1
1	1	1	$+1$

The *sign* function (again)

$$sign(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ +1 & \text{if } x > 0 \end{cases}$$

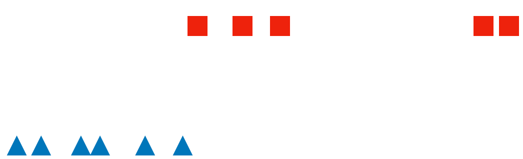


Note that the gradient is zero almost everywhere and the gradient is undefined at $x = 0$.

Image credit: Wikipedia

Can we use the squared loss?

- Treat target labels (binary) as continuous
  - ✓ final prediction decided by checking $h(\mathbf{x}) > 0$



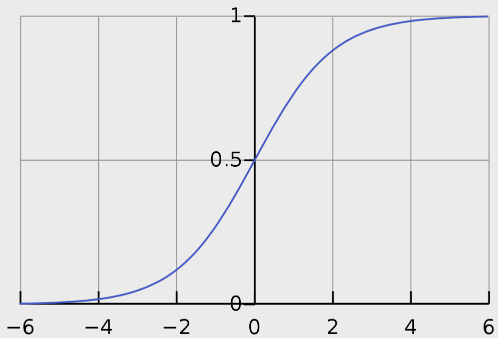
Predicted values can fall outside $[-1,1]$ range

Square loss penalizes correct predictions with large losses

Logistic regression

Logistic function

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



mapping $\mathbb{R}$ to [0,1]

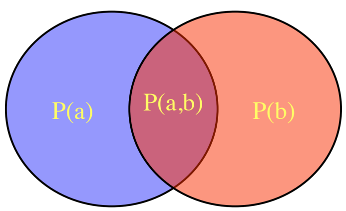
continuous and differentiable

https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.Logistic

Logistic regression

- Binary classifier
  - ✓ uses a **logistic function** (type of sigmoid function, S-shaped)
  - ✓ models **probability** of output in terms of input
- It is considered a **linear classifier**
  - ✓ even though the *activation function* is non-linear
- It is a **discriminative model**
  - ✓ models decision boundary directly, $P(y \mid \mathbf{x})$ in this case

Conditional probabilities



$$P(a|b) = \frac{P(a,b)}{P(b)}$$

$P(X, Y)$

X	Y	P
+x	+y	0.2
+x	-y	0.3
-x	+y	0.4
-x	-y	0.1

P(+x|+y)?.2/.6   P(−x|+y)?.4/.6   P(−y|+x)?.3/.5

https://inst.eecs.berkeley.edu/~cs188/fa19/assets/slides/lec13.pdf

Set up

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$$

$$\mathbf{x}^{(i)} \in \mathbb{R}^d$$

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

Probabilistic interpretation

$$h(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} = \frac{e^{\mathbf{w}^T\mathbf{x}}}{e^{\mathbf{w}^T\mathbf{x}} + 1}$$

(probability of class +1) $P(y = +1 \mid \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} = \sigma(\mathbf{w}^T\mathbf{x})$

$$P(y = -1 \mid \mathbf{x}) = 1 - P(y = +1 \mid \mathbf{x})$$

(probability of class -1) $P(y = -1 \mid \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \sigma(-\mathbf{w}^T\mathbf{x})$

Probabilistic interpretation

(probability of class +1) $P(y = +1 \mid \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} = \sigma(\mathbf{w}^T\mathbf{x})$

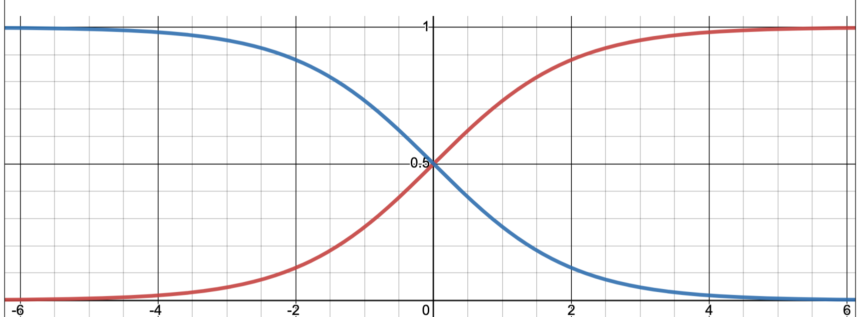
(probability of class -1) $P(y = -1 \mid \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \sigma(-\mathbf{w}^T\mathbf{x})$

$$P(y \mid \mathbf{x}) = \frac{1}{1 + e^{-y\mathbf{w}^T\mathbf{x}}} = \sigma(y\mathbf{w}^T\mathbf{x})$$

Decision boundary

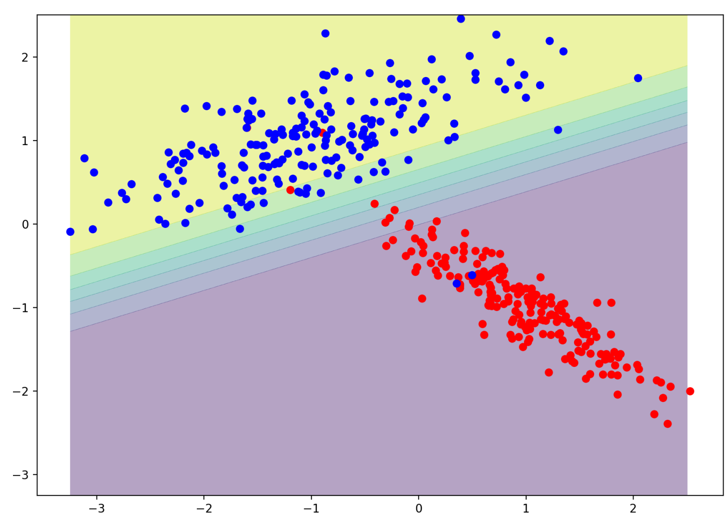
$$P(y = +1 \mid \mathbf{x}) = P(y = -1 \mid \mathbf{x}) = 0.5$$

$\sigma(\mathbf{w}^T\mathbf{x})$ $\sigma(-\mathbf{w}^T\mathbf{x})$

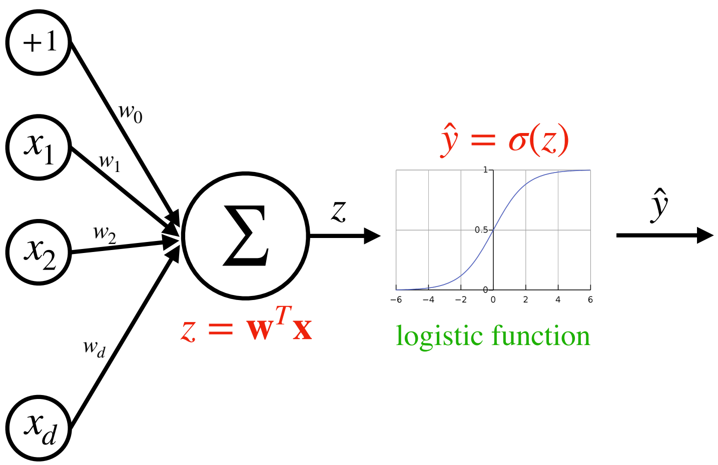


Logistic regression has a linear decision boundary $\mathbf{w}^T\mathbf{x} = 0$

Linear decision boundary



Logistic regression



Solving logistic regression

Maximum likelihood estimation (MLE)

- MLE estimates **parameters** based on the principle that if we observe $\mathscr{D}$, we should choose the parameters that make $\mathscr{D}$ most probable
- We can derive formulas for W that maximize $p(\mathscr{D}; W)$
  - ✓ many machine learning algorithms follow this **maximum likelihood** principle
  - ✓ **want** $P(y \mid \mathbf{x}; W)$
  - ✓ **learn** $W^* = \arg\max_{\mathbf{W}} P(y \mid \mathbf{x}; W)$

Maximum likelihood estimation (MLE)

- The **conditional data likelihood** $\mathscr{L}(\mathbf{w})$ is the probability of the observed labels y conditioned on the feature values $\mathbf{x}$
  - ✓ weights can be learned by maximizing this likelihood

$$\mathscr{L}(\mathbf{w}) = P(y^{(1)}, \ldots, y^{(n)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}; \mathbf{w}) = \prod_{i=1}^{n} P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w})$$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w})$$

Maximum likelihood estimation

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \log \left(\prod_{i=1}^{n} P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) \right)$$

$$= \arg\max_{\mathbf{w}} \frac{1}{n} \log \left(\prod_{i=1}^{n} P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) \right) \qquad \frac{1}{1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}}}$$

$$= \arg\max_{\mathbf{w}} -\frac{1}{n} \sum_{i=1}^{n} \log \left(1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}} \right)$$

negative log likelihood $$= \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}} \right)$$ error (loss) $e\left(h(\mathbf{x}), y\right)$

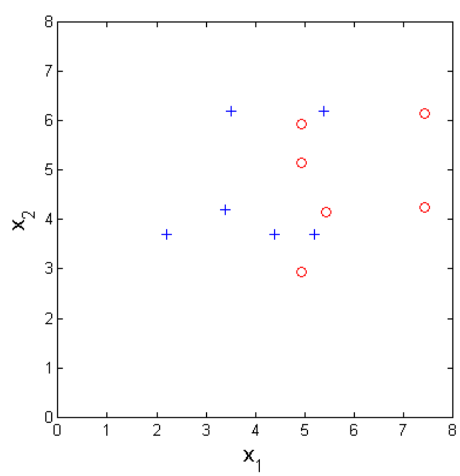
Loss function

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}} \right)$$

cross-entropy loss (over a dataset)

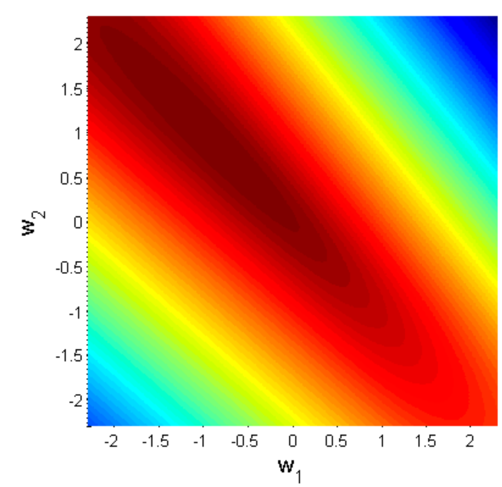
- no closed-form solution (non-linear function), but loss is convex
- can use gradient descent or second-order methods

Example: 2d dataset



https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.Logistic

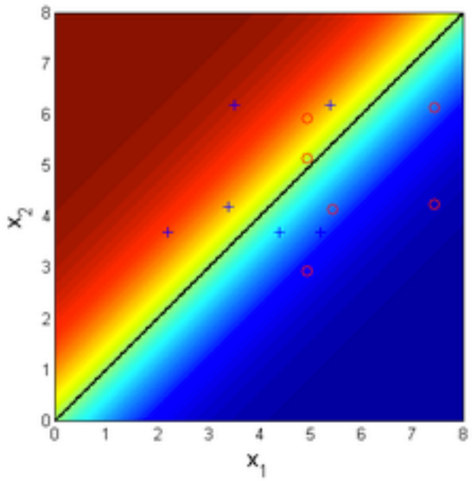
Example: loss function



plot shows contour lines in the space of parameters w_1 and w_2, w_0 is omitted

https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.Logistic

Solution



https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.Logistic

Logistic function (derivative)

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

$$\begin{aligned}\sigma'(x) &= \frac{(1+e^{-x})(0) - (1)(-e^{-x})}{(1+e^{-x})^2} \\ &= \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \sigma(x)(1-\sigma(x))\end{aligned}$$

Gradient

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = \left[\frac{\partial L(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial L(\mathbf{w})}{\partial w_d}\right]$$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}}\right)$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \sigma\left(-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}\right) y^{(i)} x_j^{(i)}$$

How to classify new data?

- Once the final hypothesis $h(\mathbf{x})$ is known …
  - ✓ $h(\mathbf{x}) = p(+1 \mid \mathbf{x})$
  - ✓ predict label $+1$ to input instance $\mathbf{x}$
    - ✓ if $p(+1 \mid \mathbf{x}) \geq 0.5$
  - ✓ predict label -1 to input instance $\mathbf{x}$
    - ✓ if $p(+1 \mid \mathbf{x}) < 0.5$

Final remarks

- Simple classifier with **probabilistic outputs**
- Loss function is convex and can be trained with GD methods (**no closed-form**)
- Robust to overfitting
- Offers interpretability to weights (feature importance)
- However, **decision boundary is still linear**

Colab notebook

https://colab.research.google.com/drive/1In0fRIDMQGzVQoOwTHrcyTF8G1xtmeBU?usp=sharing