COGS 118A, Winter 2020

Supervised Machine Learning Algorithms

Lecture 08: Perceptron and Logistic Regression

Zhuowen Tu

Perceptron

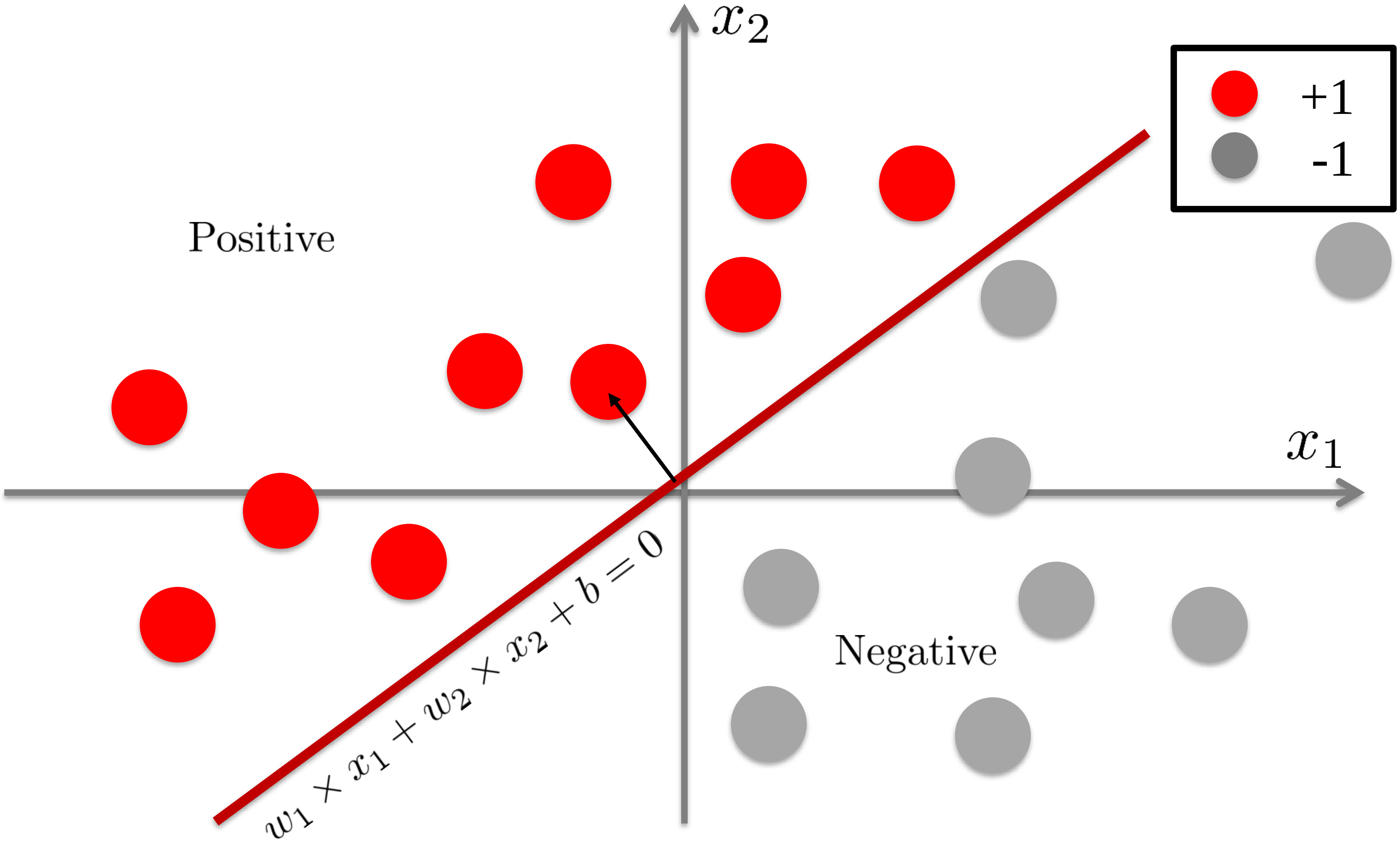
$$\mathbf{x} = (x_1, x_2, ...) \qquad \mathbf{w} = (w_1, w_2, ...)$$

$$b$$

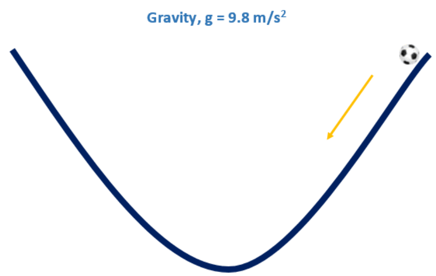
Perceptron:

$$f(\mathbf{x}|\mathbf{w}; b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

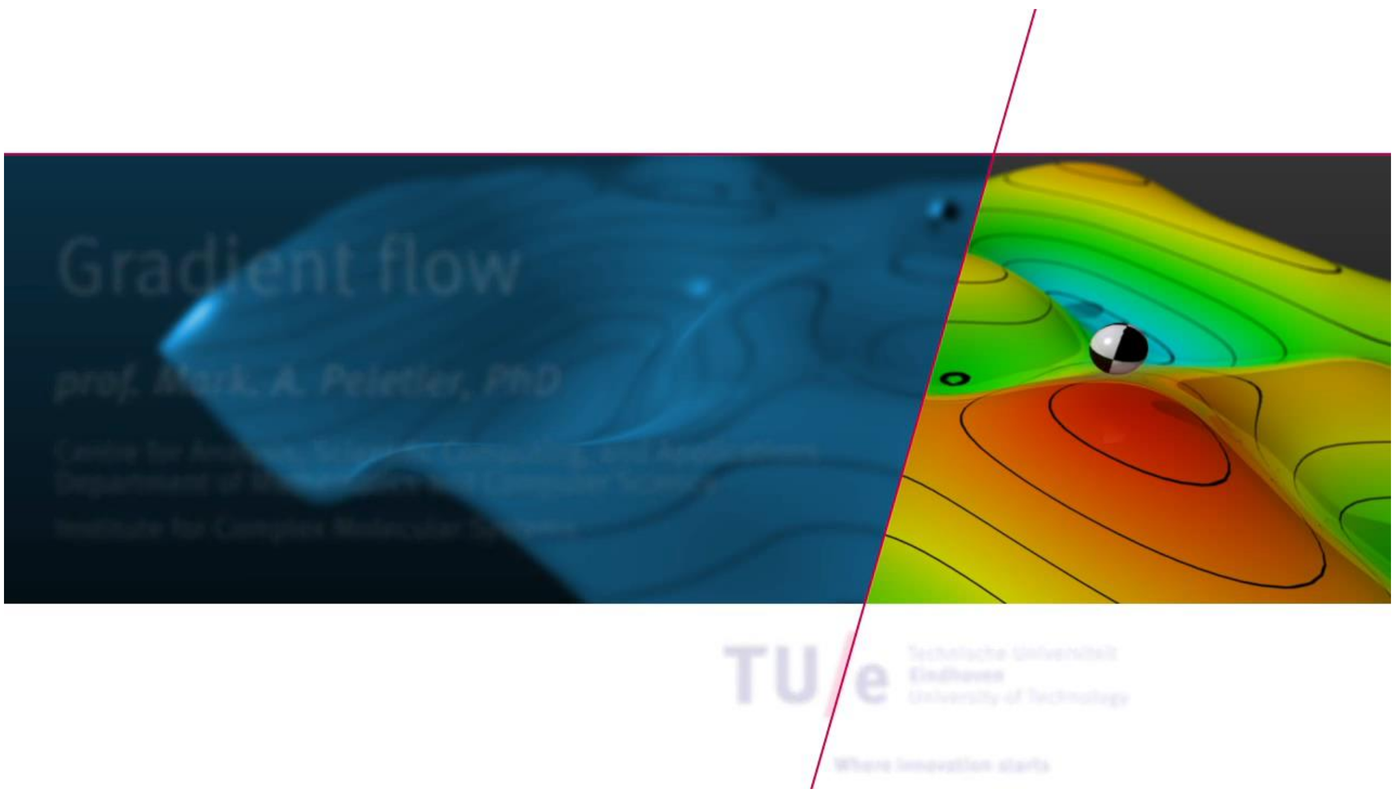
Perceptron classifier: $f(x_1, x_2; w_1, w_2, b) = sign(w_1 \times x_1 + w_2 \times x_2 + b)$



Gradient decent animation



https://www.kaggle.com/abdalimran/intuition-of-gradient-descent-for-machine-learning



https://www.youtube.com/watch?v=vWFjqgb-ylQ

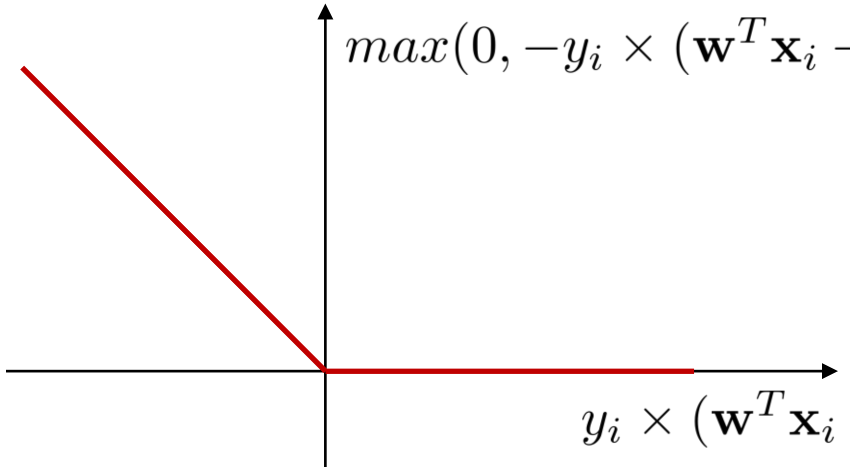
Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\} \quad y_i \in -1, +1$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$



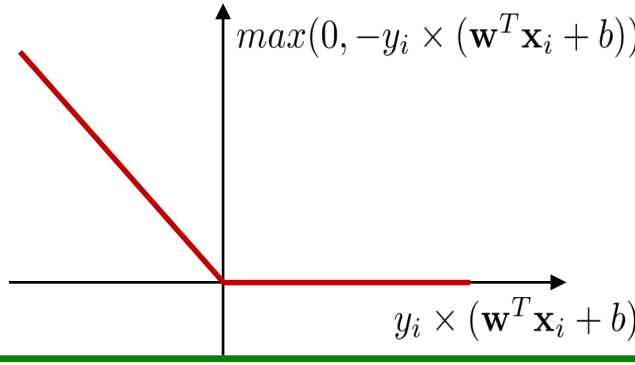
$$\frac{\mathcal{L}(\mathbf{w},\mathbf{b})}{\partial \mathbf{w}} = \begin{cases} 0 & if\ y_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \\ -y_i\mathbf{x}_i & otherwise \end{cases}$$

$$y_i = \frac{1}{2}(y_i - sign(\mathbf{w}^T\mathbf{x} + b)) = \frac{1}{2}(target_i - output_i)$$

Main motivation

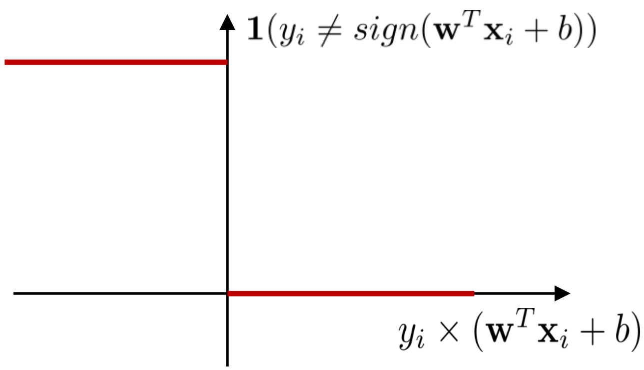
Loss implicitly used in the perceptron algorithm: with gradient feedback when the target (ground-truth label) and the output (classification) are different).

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T \mathbf{x}_i + b))$



Standard 0/1 loss (gradient 0 nearly everywhere, no gradient feedback):

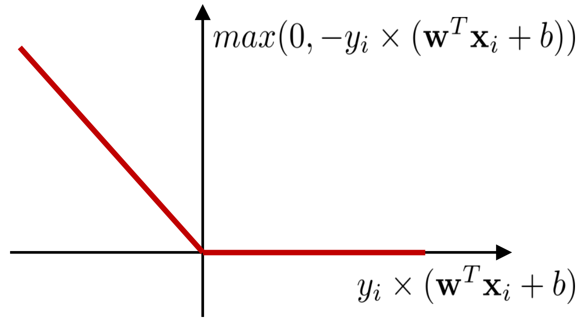
Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i \mathbf{1}(y_i \neq sign(\mathbf{w}^T \mathbf{x}_i + b))$



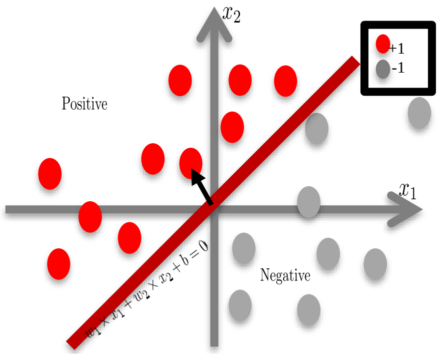
Main motivation

Loss implicitly used in the perceptron algorithm: with gradient feedback when the target (ground-truth label) and the output (classification) are different).

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$



Perceptron classifier: $f(x_1, x_2; w_1, w_2, b) = sign(w_1 \times x_1 + w_2 \times x_2 + b)$



- Perceptron is a linear classifier.
- It replaces the 0/1 loss by a relaxed loss.
- It updates the model parameters (**w**,b) based on a single sample, whereas standard gradient decent algorithm computes the gradient by taking ALL the training samples into account.

Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \boxed{\sum_i} max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial\mathbf{w}} = \boxed{\sum_i} -\frac{1}{2}(target_i - output_i)\mathbf{x}_i$$

$$target_i = y_i$$
$$output_i = sign(\mathbf{w}^T\mathbf{x}_i + b)$$

Gradient decent: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \lambda\frac{\mathcal{L}(\mathbf{w},b)}{\mathbf{w}}$

$\lambda = 2$ (learning rate)

Note the update rule for the perceptron is:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i$$

Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial b} = \boxed{\sum_i} -\frac{1}{2}(target_i - output_i)$$

$target_i = y_i$

$output_i = sign(\mathbf{w}^T\mathbf{x}_i + b)$

Gradient decent: $b_{t+1} \leftarrow b_t - \lambda\frac{\mathcal{L}(\mathbf{w},b)}{b}$

$\lambda = 2$ (learning rate)

Note the update rule for the perceptron is:

$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

Perceptron Learning Algorithm

- Initialize the weights (however you choose)
  - $w_1x_1 + w_2x_2 + b$ (initialize w_1, w_2, and b)
- Step 1: Choose a data point.
- Step 2: Compute the model output for the data point.
- Step 3: Compare model output to the target output.
  - If correct classification, go to Step 5!
  - If not, go to Step 4.

Perceptron Learning Algorithm

- Step 4: Update weights using perceptron learning rule.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

- Step 5: If you have visited all the data points and they are all corrected classifier, then exit; otherwise visit next data point and go back to step 2.

Initialize the weights

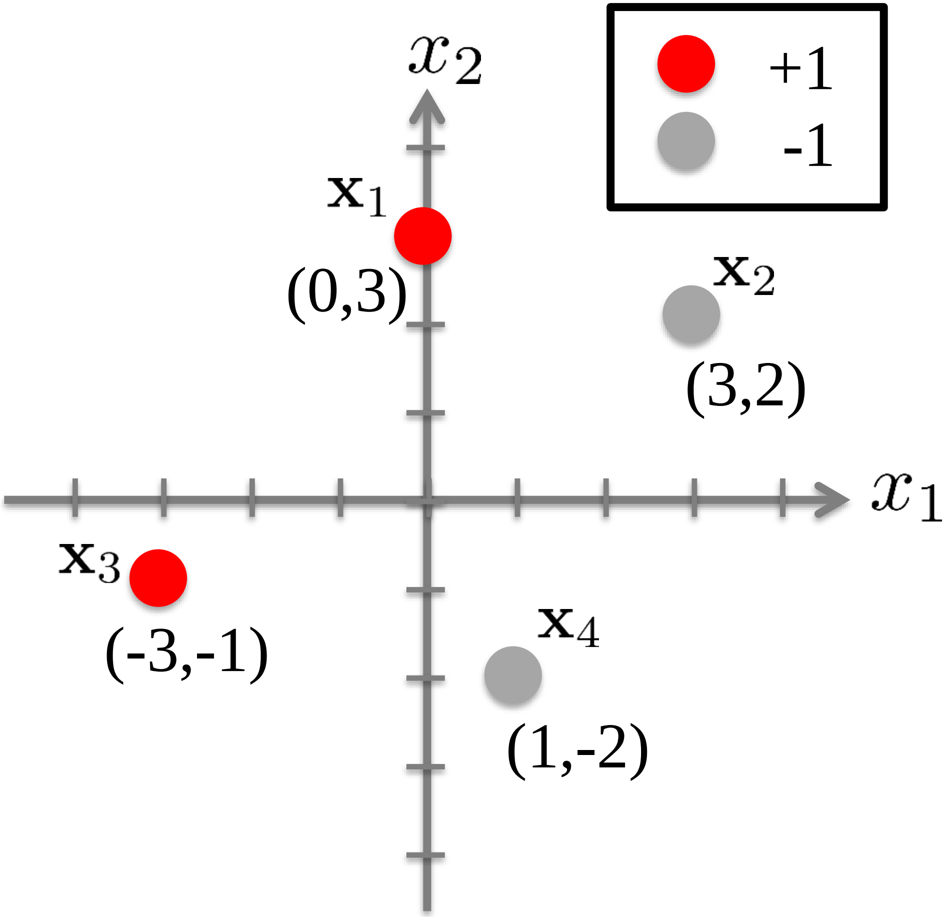
Given a training dataset:

$$S = \{(\mathbf{x}_1 = (0,3), y_1 = +1),$$

$$(\mathbf{x}_2 = (3,2), y_2 = -1),$$

$$(\mathbf{x}_3 = (-3,-1), y_3 = +1),$$

$$(\mathbf{x}_4 = (1,-2), y_4 = -1)\}$$



Initialize the weights

- Choose randomly – but start the weights small!

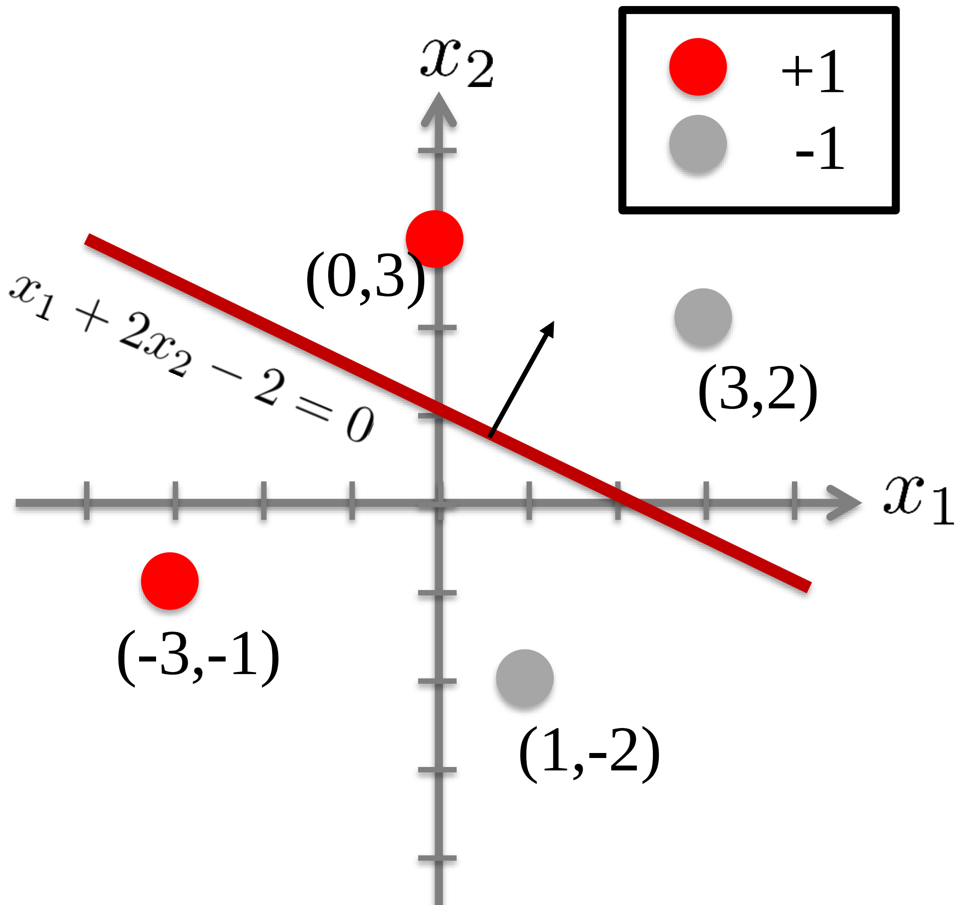
Decision boundary:
$w_1 x_1 + w_2 x_2 + b = 0$

- We'll choose:

$$w_1 = 1 \qquad w_2 = 2 \qquad b = -2$$

$$\Downarrow$$

$$x_1 + 2x_2 - 2 = 0$$

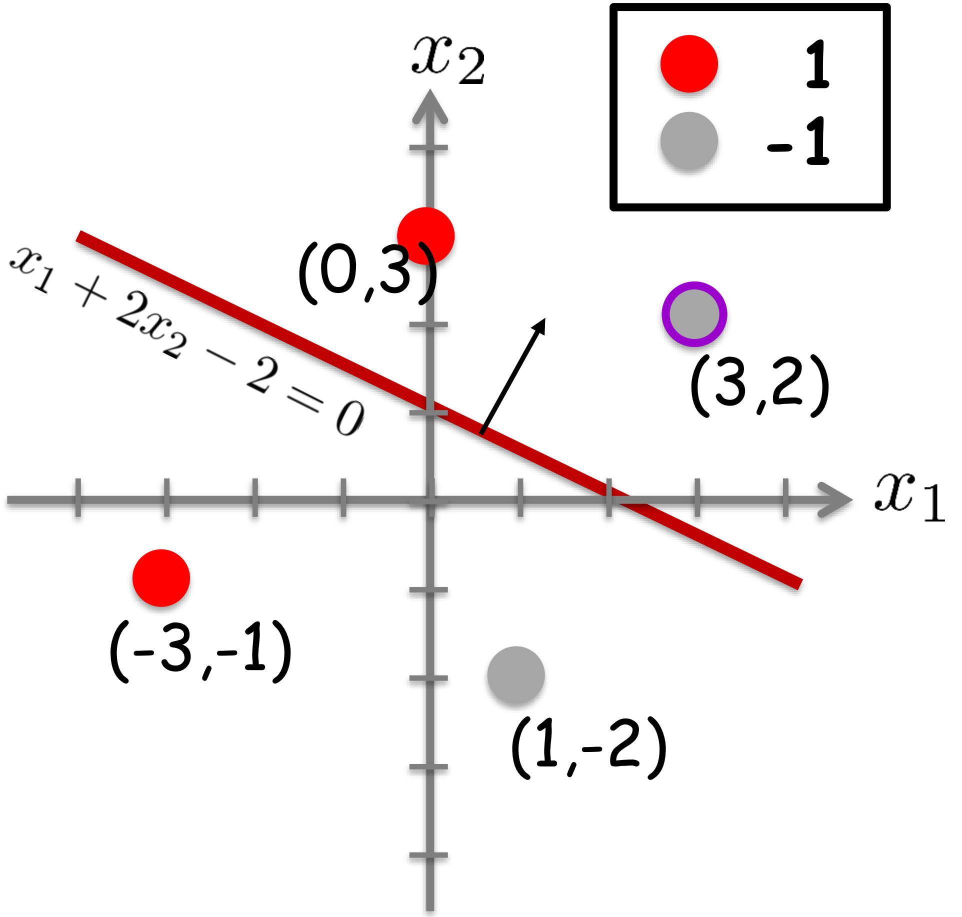


Step 1: Choose a point

- Choose randomly (or sequentially), but remember which you choose!

Say: $(\mathbf{x}_2 = (3,2), y_2 = -1)$

It's ground-truth label (target) $= -1$: a negative sample.

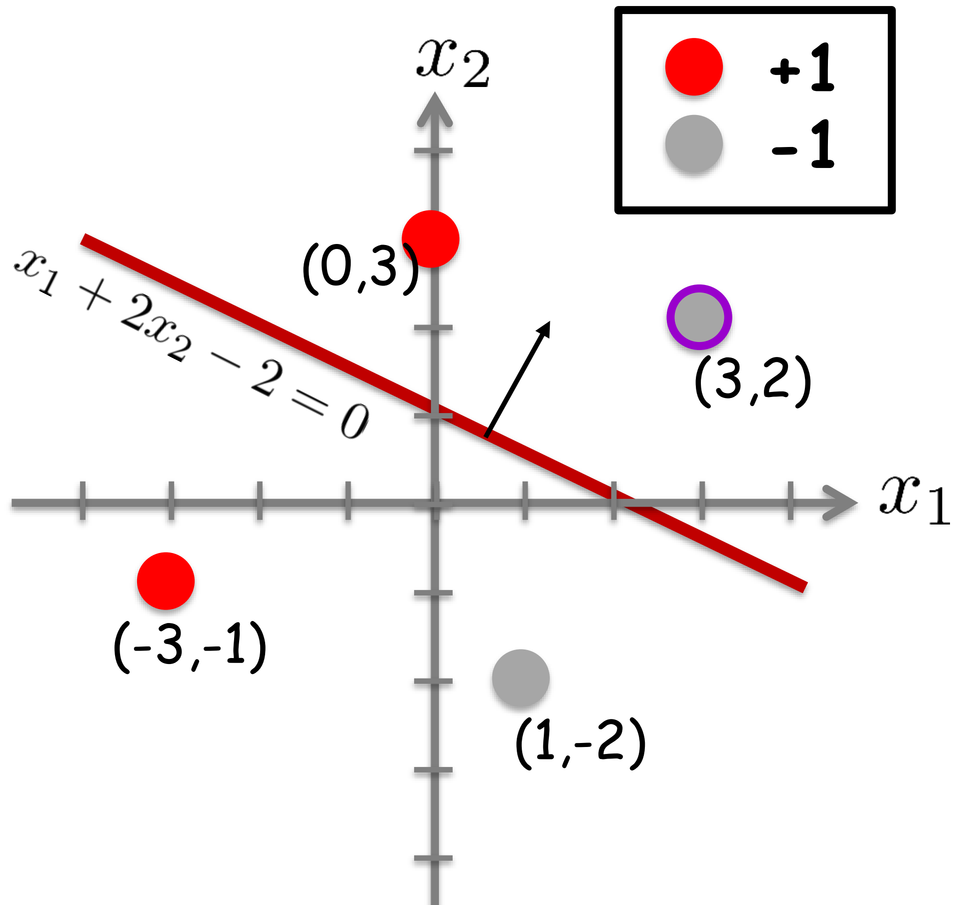


Step 1: Choose a point

Say: $(\mathbf{x}_2 = (3, 2), y_2 = -1)$

$w_1 = 1 \quad w_2 = 2 \quad b = -2$

It's ground-truth label (target) $= -1$: a negative sample.



Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$

Step 1: Choose a point at random

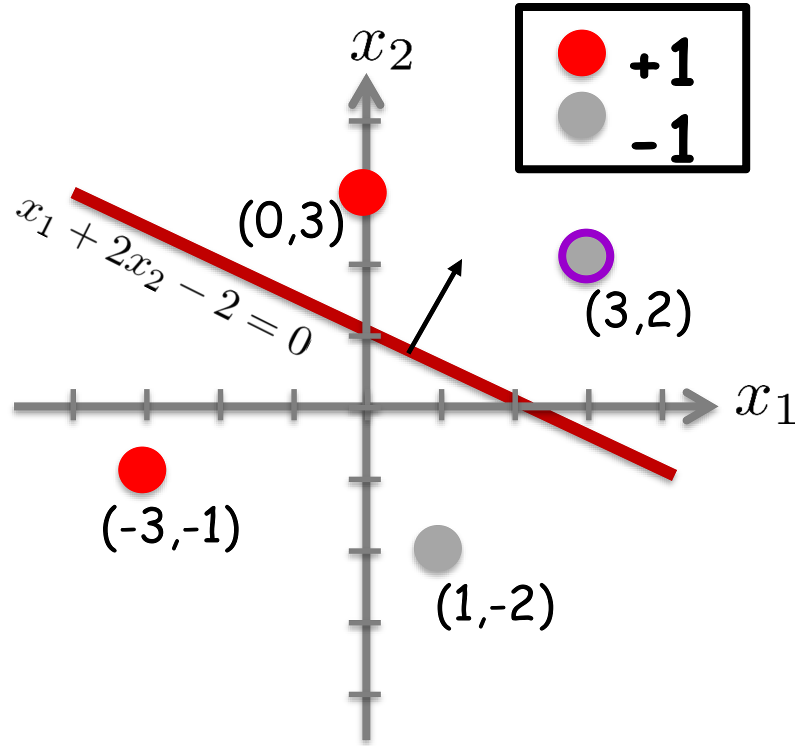
Say: $(\mathbf{x}_2 = (3,2), y_2 = -1)$

$w_1 = 1 \quad w_2 = 2 \quad b = -2$

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$

We plug in $\mathbf{x}_2 = (3,2)$:
$3 + 2 \times 2 - 2 = 3 + 4 - 2 = 5$



Classification:

$$f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2) = sign(5) = +1$$

$+1 \neq -1$

$y_2 \neq f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2)$

We need to change (**w**,b)!

Step 1: Update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

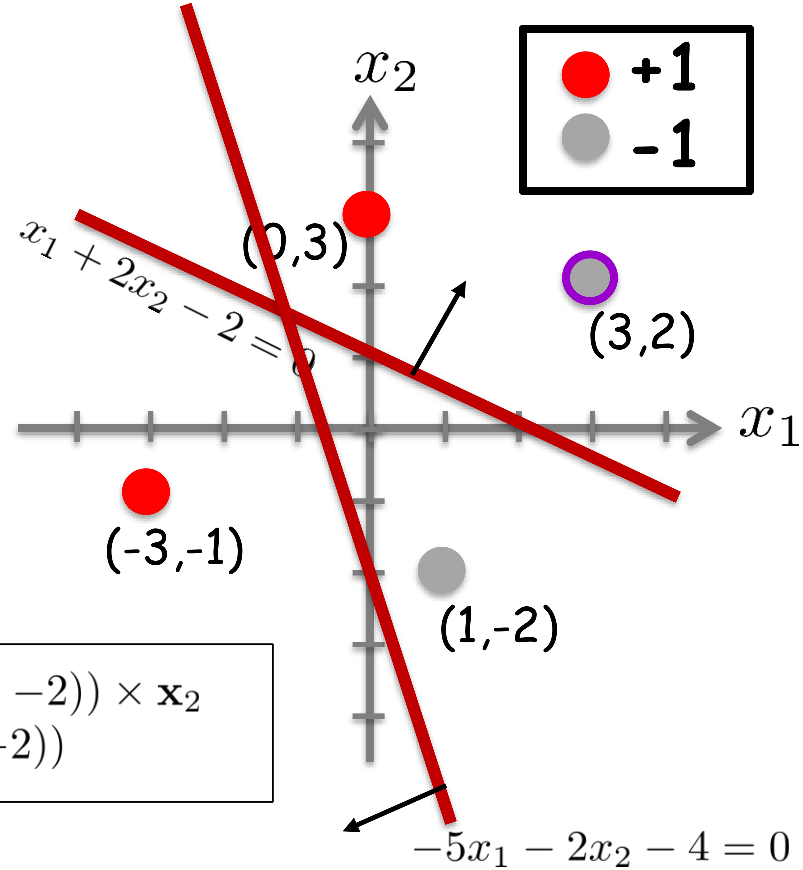
$$\text{Say: } (\mathbf{x}_2 = (3,2), y_2 = -1)$$

$$w_1 = 1 \quad w_2 = 2 \quad b = -2$$

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$

$$y_2 \neq f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2)$$



Updating rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (y_2 - f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2)) \times \mathbf{x}_2$$
$$b_{t+1} \leftarrow b_t + (y_2 - f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2))$$

$$\mathbf{w}_{t+1} \leftarrow (1,2) + (-1-1) \times (3,2)$$
$$b_{t+1} \leftarrow -2 + (-1-1)$$

$$\mathbf{w}_{t+1} = (-5, -2)$$
$$b_{t+1} = -4$$

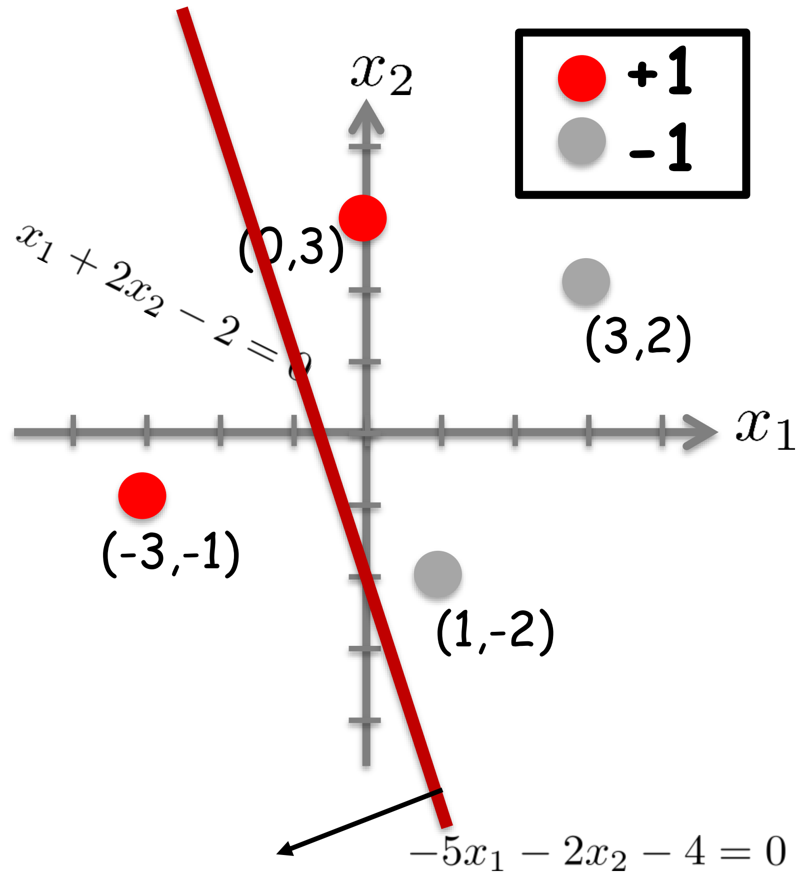
Step 1: Update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

$$\boxed{w_1 = -5 \quad w_2 = -2 \quad b = -4}$$

Updated Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$

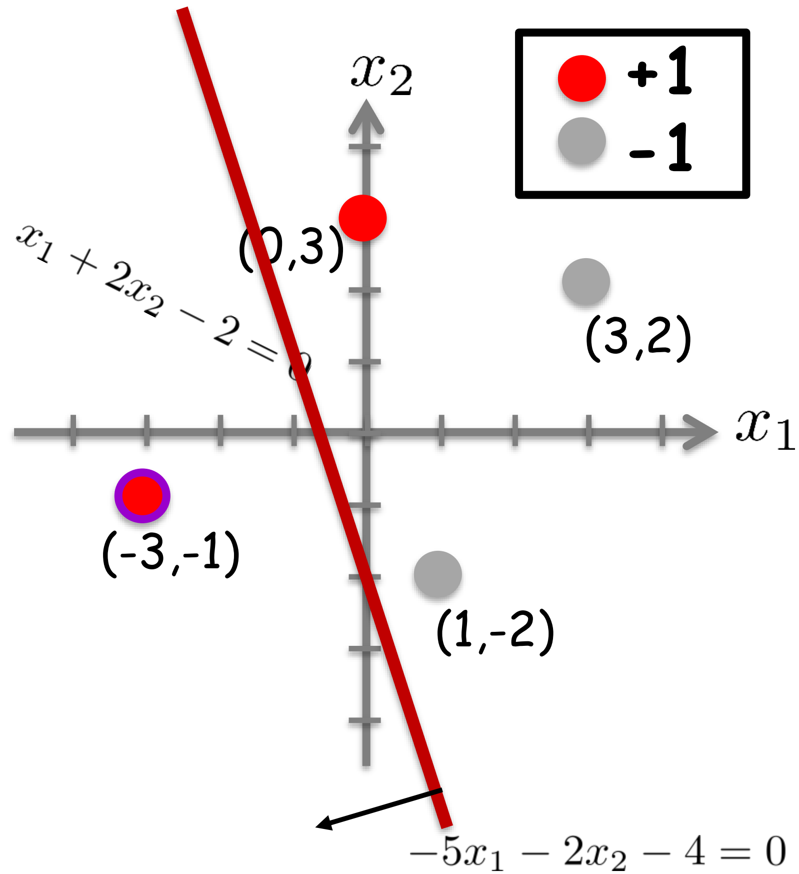


Step 2: Choose another point

$$w_1 = -5 \quad w_2 = -2 \quad b = -4$$

Perceptron:

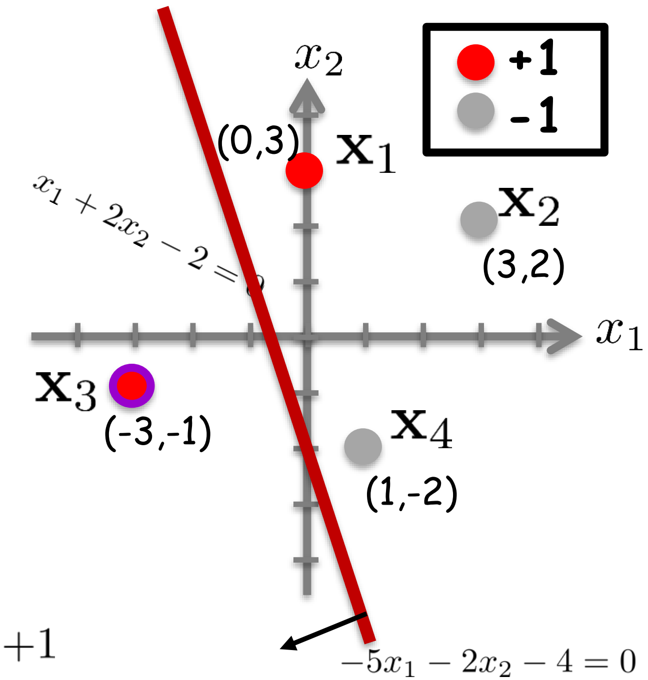
$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$



Step 2: Choose another point

Say: $(\mathbf{x}_3 = (-3,-1), y_3 = +1)$

$w_1 = -5 \;\; w_2 = -2 \quad b = -4$



Perceptron:

$$f(\mathbf{x}|w_1,w_2,b) = \begin{cases} +1 & if \;\; -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$

We plug in $\mathbf{x}_3 = (-3,-1)$:
$-5 \times (-3) + (-2) \times (-1) - 4 = 15 + 2 - 4 = 13$

Classification:

$$f(\mathbf{x}_2 = (-3,-1)|w_1 = -5, w_2 = -2, b = -4) = sign(13) = +1$$

$+1 = +1$

$y_3 == f(\mathbf{x}_3 = (-3,-1)|w_1 = -5, w_2 = -2, b = -4)$

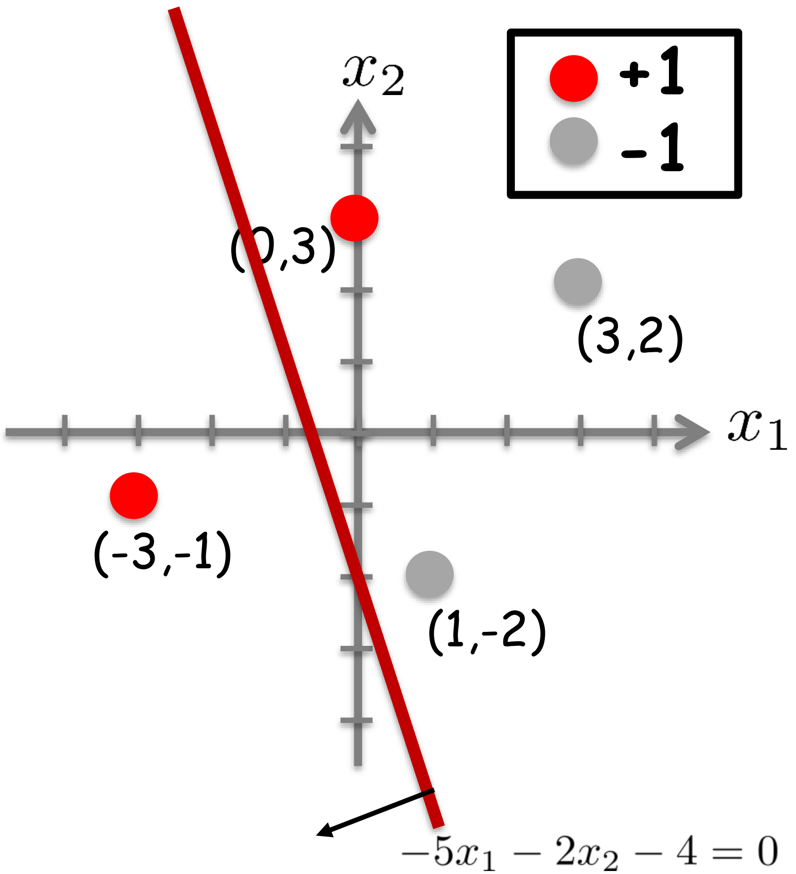
We don’t need to change (**w**,b)!

Step 2: Keep

$$w_1 = -5 \quad w_2 = -2 \quad b = -4$$

Updated Perceptron:

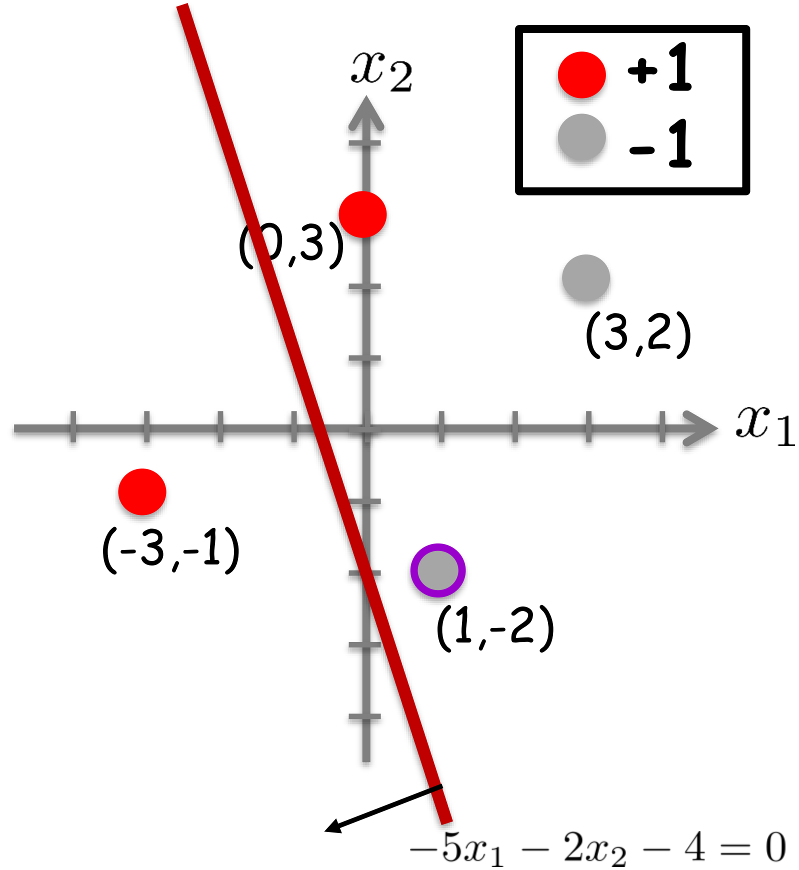
$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \; -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$



Step 3: Choose another point

Say: $(\mathbf{x}_4 = (1, -2), y_4 = -1)$

$w_1 = -5 \quad w_2 = -2 \quad b = -4$

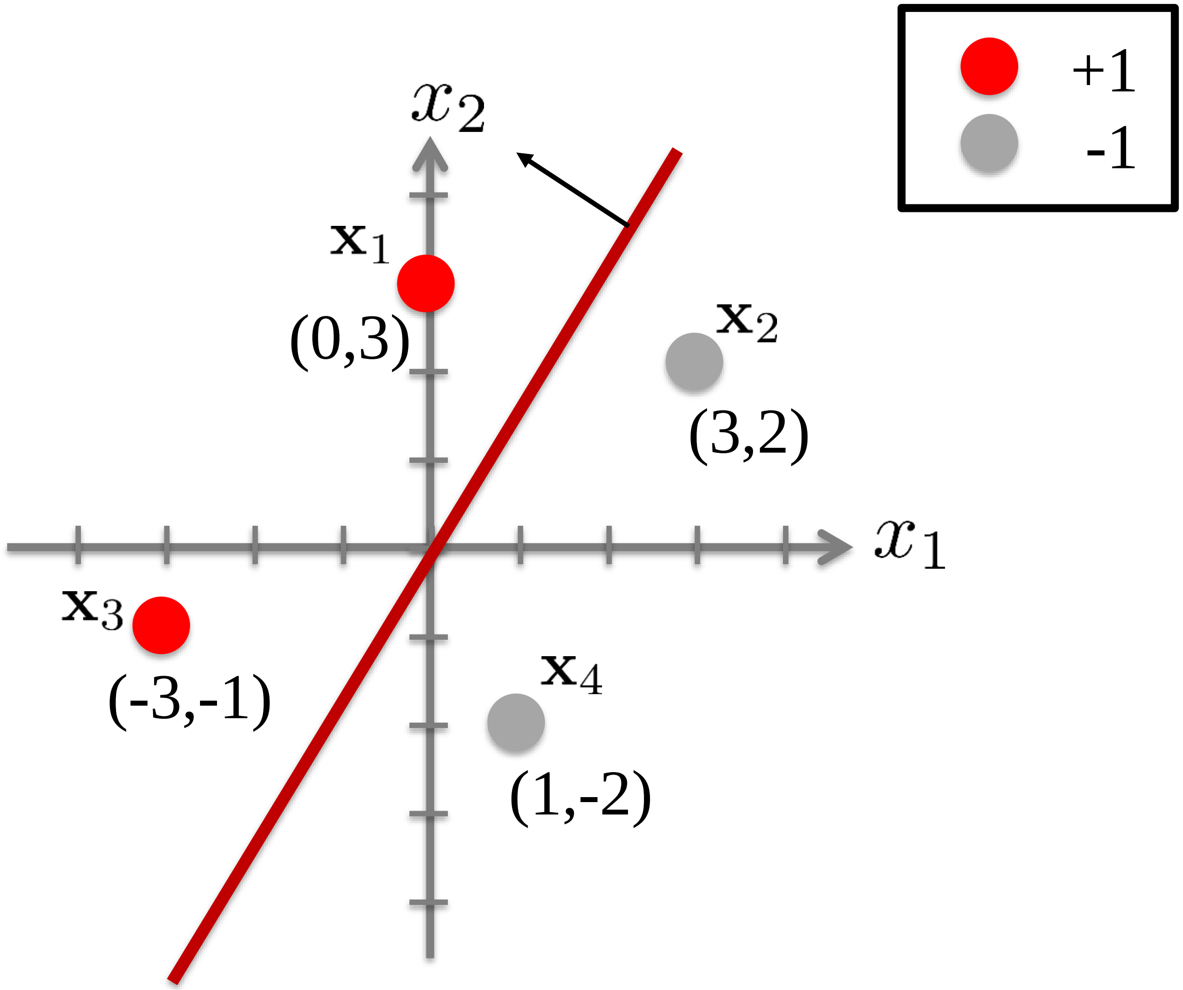


Updated Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \; -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$

Iterate Until Converge

. . . .



Perceptron Learning Algorithm

- Initialize the weights (however you choose)
  - $w_1x_1 + w_2x_2 + b$ (initialize w_1, w_2, and b)
- Step 1: Choose a data point.
- Step 2: Compute the model output for the datapoint.
- Step 3: Compare model output to the target output.
  - If correct classification, go to Step 5!
  - If not, go to Step 4.
- Step 4: Update weights using perceptron learning rule. Start over on Step 1 with the first data point.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i$$
$$b_{t+1} = b_t + (target_i - output_i)$$

Or

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(target_i - output_i)\mathbf{x}_i$$
$$b_{t+1} = b_t + \lambda(target_i - output_i)$$

- Step 5: Go to the next data point. If you have gone through them all, you have found the solution!

Perceptron Learning

Perceptron:

Note that the learning process is not strictly gradient decent.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i$$
$$b_{t+1} = b_t + (target_i - output_i)$$

It's a stochastic gradient decent algorithm!

Standard gradient decent:

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \boxed{\sum_i} max(0, -y_i \times (\mathbf{w}^T \mathbf{x}_i + b))$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial \mathbf{w}} = \boxed{\sum_i} -\frac{1}{2}(target_i - output_i)\mathbf{x}_i$$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial b} = \boxed{\sum_i} -\frac{1}{2}(target_i - output_i)$$

$$(\mathbf{w}, b)_{t+1} = (\mathbf{w}, b)_t - \lambda_t \frac{\partial \mathcal{L}(\mathbf{w},b)}{\partial (\mathbf{w},b)}$$

Perceptron

The perceptron algorithm mainly consist of an updating process:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$

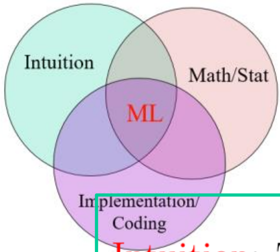
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

$$\text{if } target_i \neq output_i$$

In essence, it is a gradient decent algorithm that minimizes:

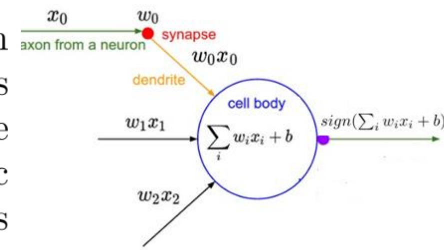
Training: Minimize $\sum_i max(0, -y_i \times (\mathbf{w}^T \mathbf{x}_i + b))$

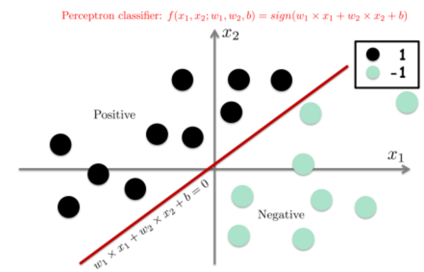
It is an algorithm that is easy to implement and works reasonably well.



Recap: Perceptron

Intuition: The perceptron classifier itself is still a linear classifier using the sign (positive or negative) of the output as the prediction. Training a perceptron is done through an iterative procedure by visiting each individual sample to update the modal until convergence. This can be viewed as an extreme case of stochastic gradient descent with the batch size being 1. The perceptron classifier itself has limited classification power and cannot well classify non-separable samples (e.g. sample distribution as XOR). Therefore, it has been ignored for many years in the computing field. Although perceptron itself is limited, adding perceptrons with an activation function (e.g. sigmoid or ReLU) and building layers of them have led to significantly enhanced power in the modern deep learning era.

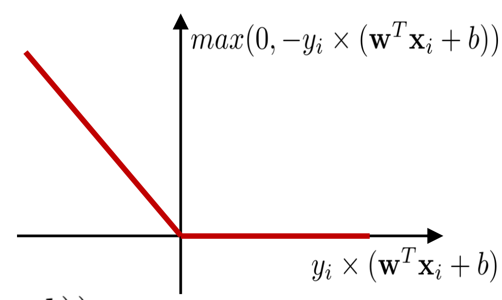




Math:

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$



Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

$$\boxed{\begin{array}{ll} \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i & target_i = y_i \\ b_{t+1} \leftarrow b_t + (target_i - output_i) & output_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \end{array}}$$

Recap: Perceptron

Implementation:

Initialize the weights $\mathbf{w} \in \mathbb{R}$ and $b \in \mathbb{R}$ for

$$f(\mathbf{x}|\mathbf{w}; b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

- Step 1: Choose a data point $\mathbf{x}_i$.
- Step 2: Compute the model output for the data point.

  $output_i = f(\mathbf{x}_i; \mathbf{w}; b)$
- Step 3: Compare model output to the target output.
- If correct classification, go to Step 5; if not, go to Step 4.
- Step 4: Update weights using perceptron learning rule.

  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i$

  $b_{t+1} \leftarrow b_t + (target_i - output_i)$
- Step 5: If you have visited all the data points and they are all corrected classifier, then exit; otherwise visit next data point and go back to step 2.

Perceptron

The perceptron algorithm ignited some initial excitement in artificial intelligence (not machine learning since the term didn’t exist back then).

However, it quickly shows its limitation as a classifier due to an idea assumption that the positives and negatives are well separated.

The problem of XOR is a particular hit to perceptron making people loose confidence in it.

The XOR problem that kills the Perceptron algorithm

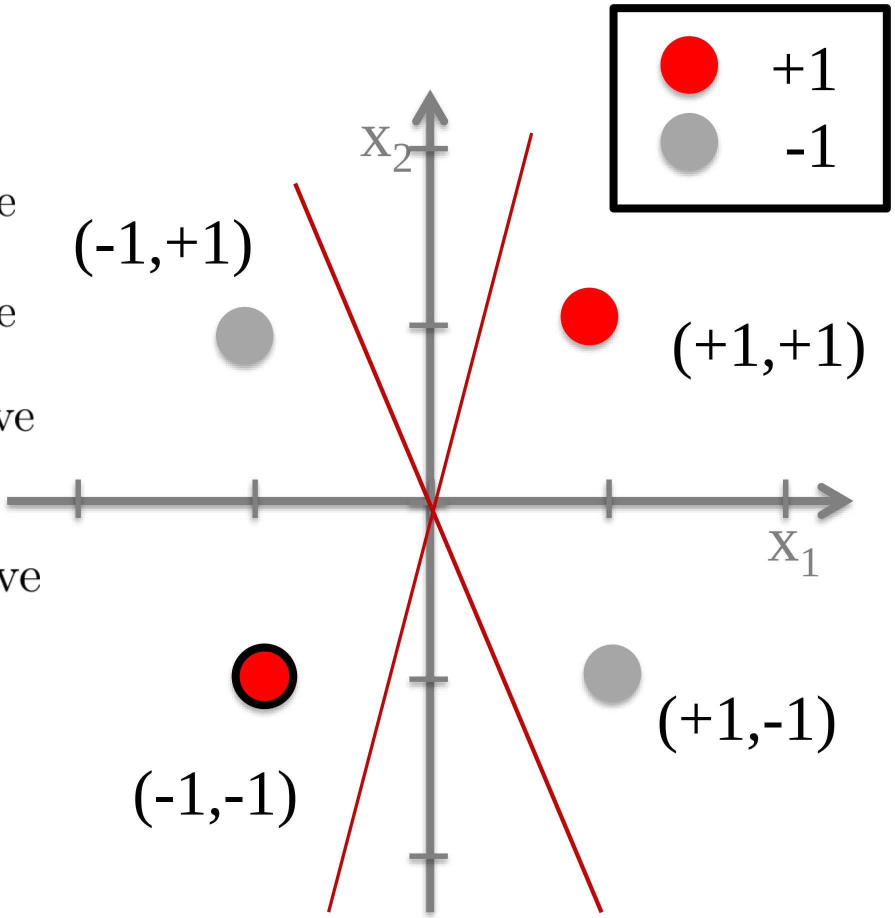
The XOR operator: $\bigoplus$

Positive (+1) $\bigoplus$ Negative (-1) = Negative

Positive (+1) $\bigoplus$ Positive (+1) = Positive

Negative (-1) $\bigoplus$ Positive (+1) = Negative

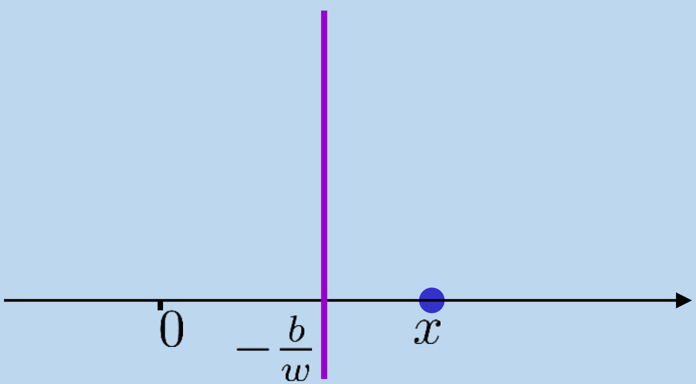
Neative (-1) $\bigoplus$ Negative (-1) = Positive



Logistic regression classifier

Logistic regression classifier

$x, w, b \in \mathbb{R}$

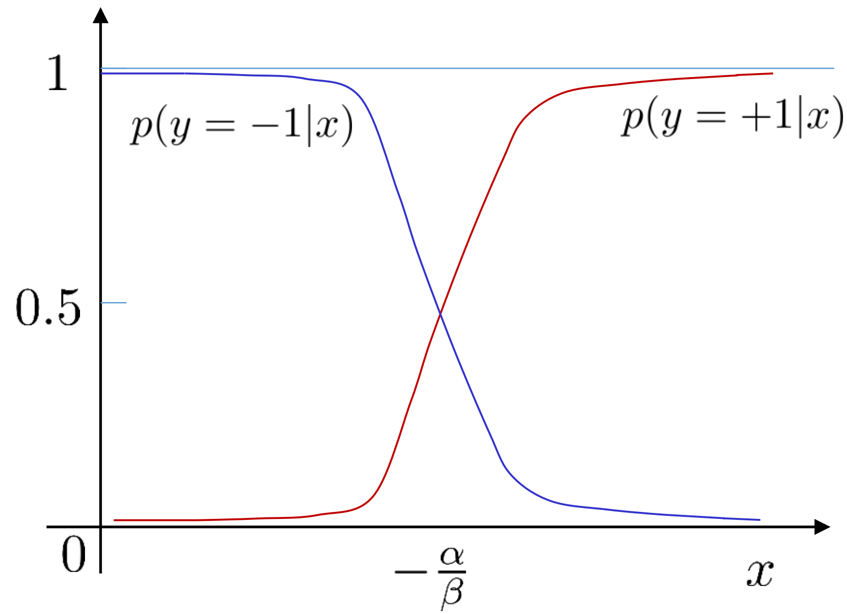


$$w \times x + b \overset{?}{\geq} 0$$

$$w \times (\frac{b}{w} + x) \overset{?}{\geq} 0$$

Let's look at the simplest case where x is a scalar:

Probability of being positive



We have: $f(x; w, b) = \begin{cases} +1 & if\ w \times x + b \geq 0 \\ -1 & otherwise \end{cases}$.

$$p(y=+1|x) = \frac{e^{w \times x + b}}{1 + e^{w \times x + b}}$$

$$p(y=-1|x) = \frac{1}{1 + e^{w \times x + b}}$$

Decision boundary for a logistic regression classifier?

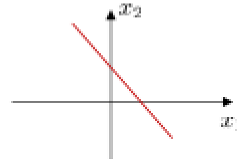
$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

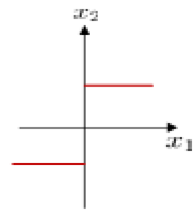
$$f(\mathbf{x};\mathbf{w};b) = \begin{cases} +1 & if\ \frac{1}{1+e^{-(\mathbf{x}\cdot\mathbf{w}+b)}} \geq 0.5 \\ -1 & otherwise \end{cases}$$

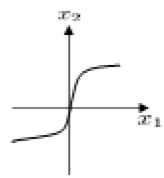
$$= \begin{cases} +1 & w\cdot x+b \geq 0 \\ -1 & \text{else} \end{cases}$$

$$\frac{1}{1+e^{-(w^Tx+b)}}$$

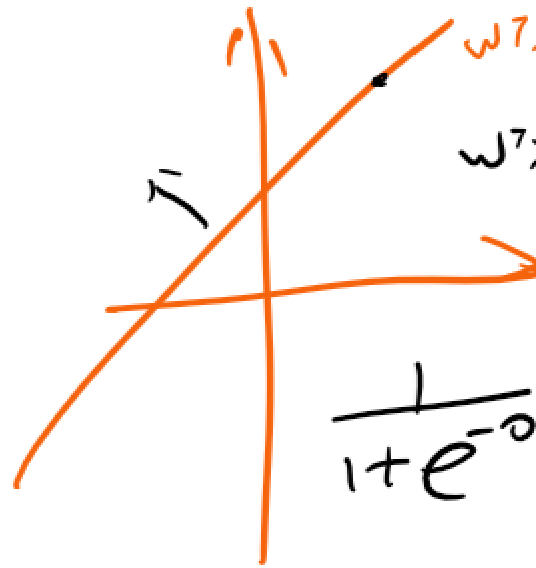
$$w^Tx+b=0$$

A. 

B. 

C. 

D. None of above.



$w^Tx+b=0$

$$\frac{1}{1+e^{-0}} = \frac{1}{1+1} = \frac{1}{2}$$

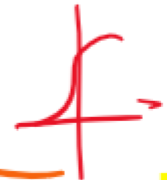
$w^Tx+b=100$ $\quad \frac{1}{1+e^{-100}} = \frac{1}{1} = 1$

$w^Tx+b=-100$

$$\frac{1}{1+e^{-(-100)}} = \frac{1}{1+e^{100}} \approx 0$$

$$\frac{1}{1+e^{-(-100)}} = \frac{1}{1+e^{100}} \approx 0$$

$$p(y=+1|x) = \frac{1}{1+e^{-(w\cdot x+b)}}$$



+

$$p(y=-1|x) = 1 - p(y=+1|x) = \frac{1}{1+e^{(w\cdot x+b)}}$$

$$= 1 - \frac{1}{1+e^{-(w\cdot x+b)}} = \frac{\cancel{1}+e^{-(w\cdot x+b)}\cancel{-1}}{1+e^{-(w\cdot x+b)}}$$

$$= \frac{e^{-(w\cdot x+b)} \times e^{(w\cdot x+b)}}{\left(1+e^{-(w\cdot x+b)}\right)\times e^{(w\cdot x+b)}} = \frac{1}{e^{w\cdot x+b}+1}$$

$$= \frac{1}{1+e^{w\cdot x+b}}$$

$$p(y|x) = \frac{1}{1+e^{-y(w\cdot x+b)}}$$

$$y \in \{-1,+1\}$$

Logistic regression function

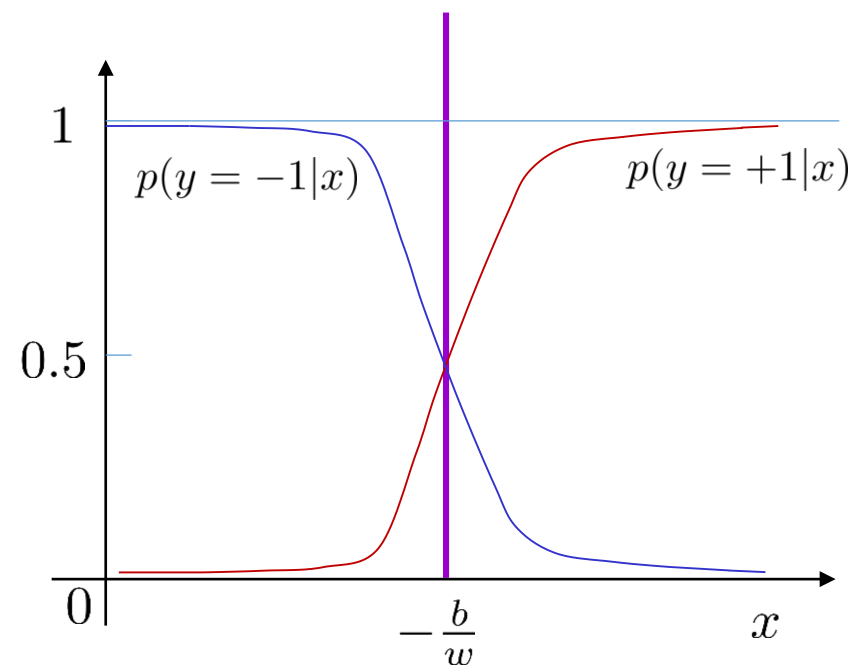
$$p(y=+1|x) = \frac{1}{1+e^{-(w\times x+b)}} \qquad x, w, b \in \mathbb{R}$$

$$p(y=-1|x) = \frac{1}{1+e^{(w\times x+b)}} \qquad y \in \{-1,+1\}$$

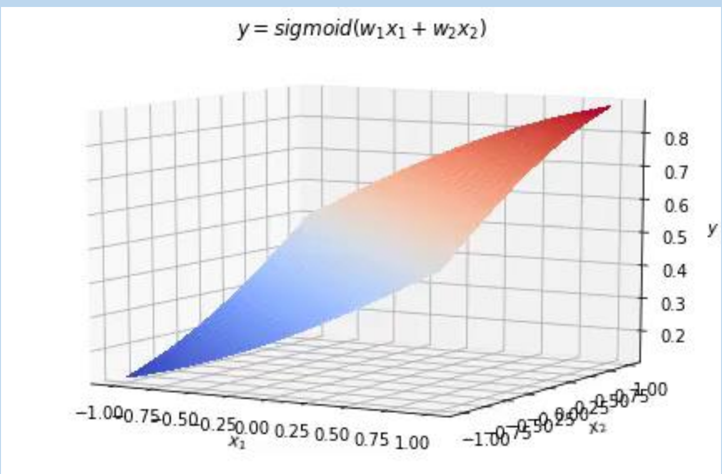
$$p(y=+1|x) + p(y=-1|x) = 1$$

$$\frac{1}{1+e^{-(w\times x+b)}} + \frac{1}{1+e^{(w\times x+b)}}$$

$$= \frac{e^{(w\times x+b)}}{e^{(w\times x+b)}+1} + \frac{1}{1+e^{(w\times x+b)}} \quad = \frac{e^{(w\times x+b)}+1}{1+e^{(w\times x+b)}} = 1$$

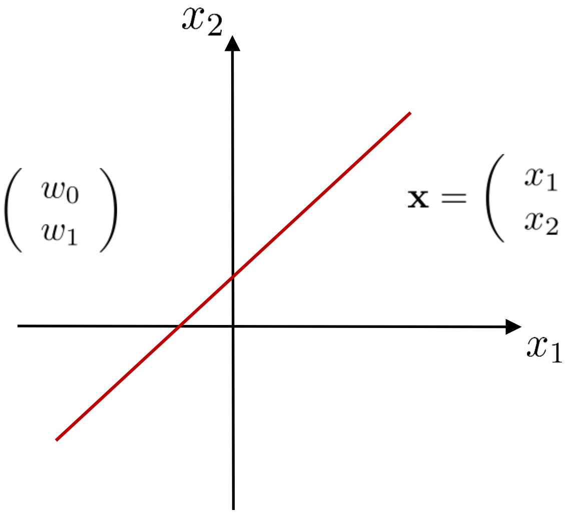


Logistic regression classifier (2D case)



$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

b



$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

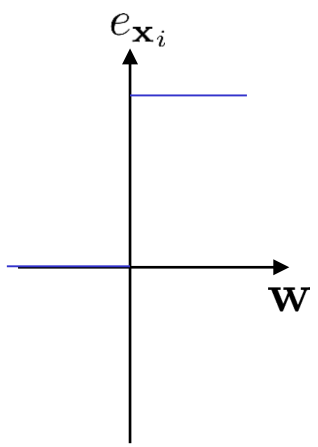
We have: $f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if \;\; \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}.$

sigmoid function: $\sigma(v) = \frac{1}{1+e^{(-v)}}.$

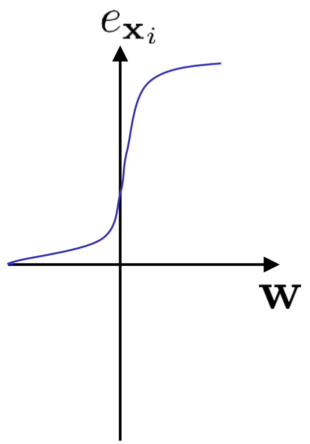
$$p(y = +1|\mathbf{x}) = \sigma\big(\mathbf{w} \cdot \mathbf{x} + b\big)$$

$$p(y = -1|\mathbf{x}) = \sigma\big(-(\mathbf{w} \cdot \mathbf{x} + b)\big)$$

From a **hard** to a **soft** function



$$e_{training} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left(y_i \neq \begin{cases} +1 & if \;\; \mathbf{w}\cdot\mathbf{x}_i + b \geq 0 \\ -1 & otherwise \end{cases}\right)$$



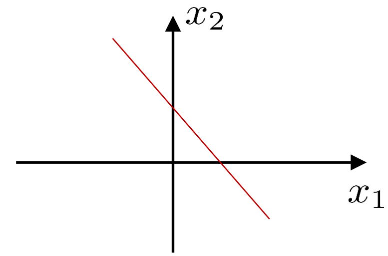
$$e_{training} = \frac{1}{n}\sum_{1=1}^{n} -\ln\left(\frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}\right)$$

Decision boundary for a logistic regression classifier?

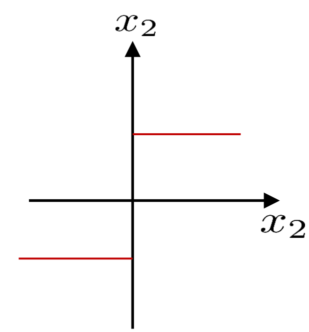
$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$f(\mathbf{x}; \mathbf{w}; b) = \begin{cases} +1 & if \ \frac{1}{1+e^{-(\mathbf{x}\cdot\mathbf{w}+b)}} \geq 0.5 \\ -1 & otherwise \end{cases}.$$

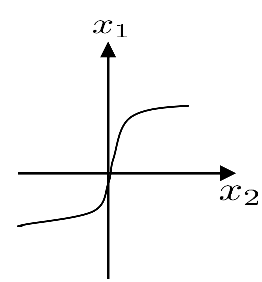
A.



B.



C.



D. None of above.

Logistic regression function

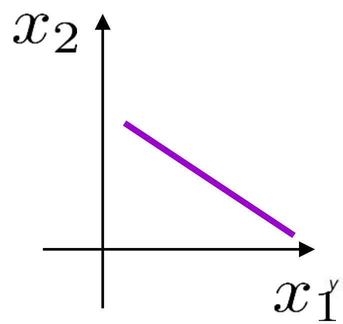
$$p(y=+1|\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x}+b)}}$$

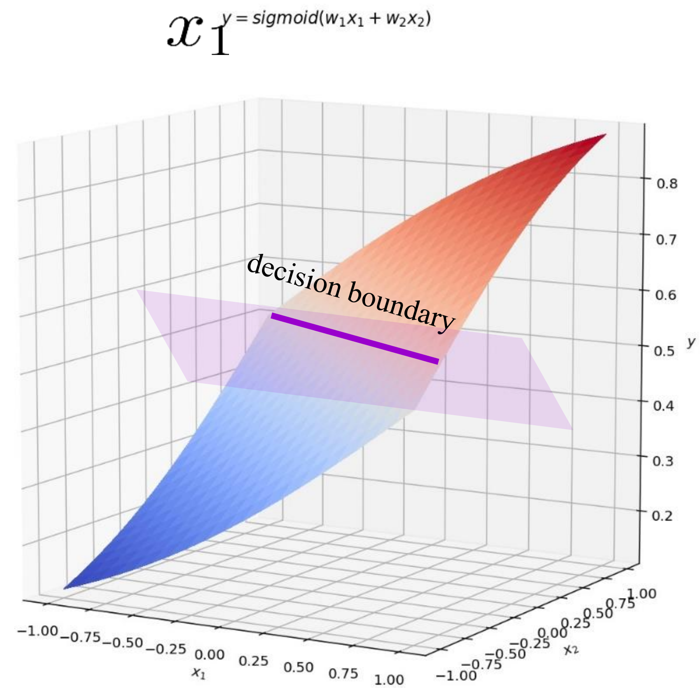
$$p(y=-1|\mathbf{x}) = \frac{1}{1+e^{(\mathbf{w}\cdot\mathbf{x}+b)}}$$

$$\mathbf{x}, \mathbf{w} \in \mathbb{R}^m$$

$$b \in \mathbb{R}$$

$$y \in \{-1, +1\}$$





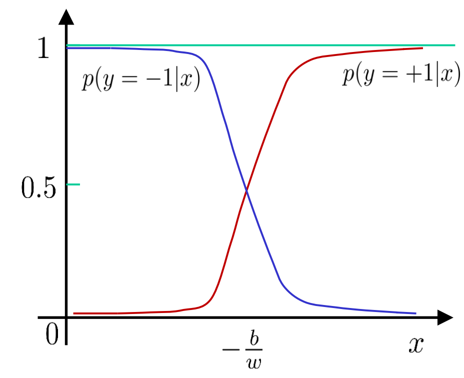
Training a logistic regression classifier

$S_{training} = \{(-1.1, -1), (3.2, +1), (2.5, -1), (5.0, +1), (4.3, +1)\}$

Train a logistic regression classifier $f(x) = \begin{cases} +1 & if \ \frac{1}{1+e^{-(w \times x + b)}} \geq 0.5 \\ -1 & otherwise \end{cases}$:

$$p(y = +1|x) = \frac{1}{1+e^{-(w \times x + b)}}$$

$$p(y = -1|x) = \frac{1}{1+e^{(w \times x + b)}}$$



$$p(y_i|x_i) = \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}$$

Intuition: find the best parameters $(w, b)^*$ to maximize the probabilities of fitting the ground-truth label y_i for each x_i.

Math: $(w, b)^* = \arg\max_{(w,b)} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}$

Training a logistic regression classifier

$$(\alpha, \beta)^* = \arg\max_{(w,b)} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}$$

$$= \arg\max_{(w,b)} \ln\left(\prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$$

$$= \arg\min_{(w,b)} -\ln\left(\prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$$

Question: which is the correct answer for the optimal solution?

Answer A: $(w,b)^* = \arg\min_{(w,b)} \sum_{i=1}^{n} -\ln\left(\frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$

Answer B: $(w,b)^* = \arg\max_{(w,b)} \sum_{i=1}^{n} -\ln\left(\frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$

Answer C: $(w,b)^* = \arg\min_{(w,b)} -\ln\left(\sum_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$

Training a logistic regression classifier

Intuition: find the best parameters $(w, b)^*$ to maximize the probabilities of fitting the ground-truth label y_i for each x_i.

Math: $(w, b)^* = \arg\max_{(w,b)} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}$

$$(w, b)^* = \arg\max_{(w,b)} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}$$

$$= \arg\max_{(w,b)} \ln\left(\prod_{i=1}^{n} \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$$

$$= \arg\min_{(w,b)} \sum_{i=1}^{n} -\ln\left(\frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right)$$

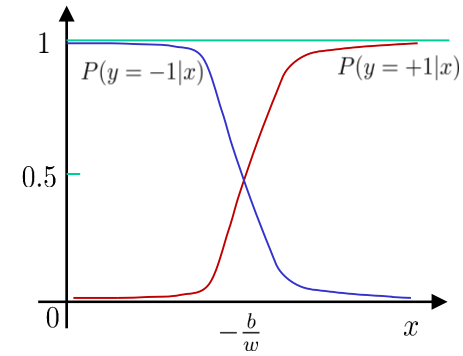
Training a logistic regression classifier

$$S_{training} = \{(-1.1, -1), (3.2, +1), (2.5, -1), (5.0, +1), (4.3, +1)\}$$

$$x_i \in \mathbb{R}, i = 1..n$$

$$y_i \in \{-1, +1\}, i = 1..n$$

$$p(y_i|x_i) = \frac{1}{1+e^{-y_i \times (w \times x_i + b)}}$$



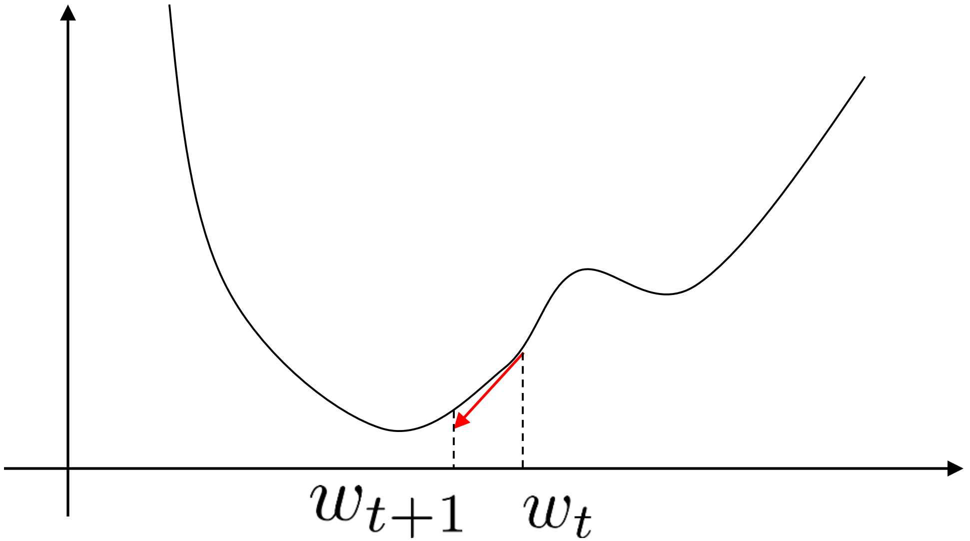
$$(w, b)^* = \arg\max_{(w,b)} \prod_{i=1}^{n} [p(y_i|x_i)]$$

$$(w, b)^* = \arg\min_{(w,b)} -\sum_{i=1}^{n} \ln\left(\frac{1}{1+e^{-y_i \times (w \times x_i + b)}}\right) = \arg\min_{(w,b)} \sum_{i=1}^{n} \ln(1+e^{-y_i \times (w \times x_i + b)})$$

$$(w, b)^* = \arg\min_{(w,b)} [\ln(1+e^{(-1.1w+b)}) + \ln(1+e^{-(3.2w+b)}) +$$

$$\ln(1+e^{(2.5w+b)}) + \ln(1+e^{-(5.0w+b)}) + \ln(1+e^{-(4.3w+b)})]$$

Gradient descent (ascent)



Gradient Descent Direction

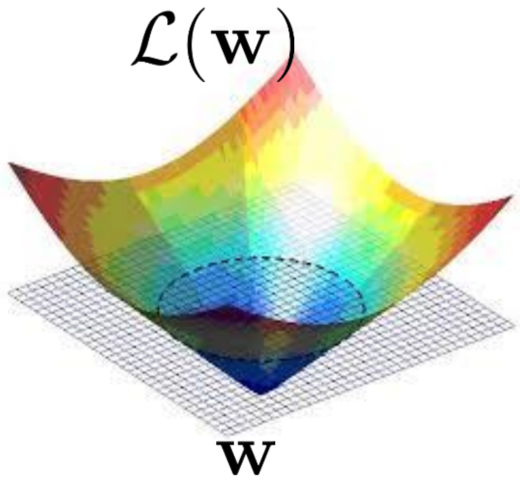
(a) Pick a direction $\nabla \mathcal{L}(w_t)$

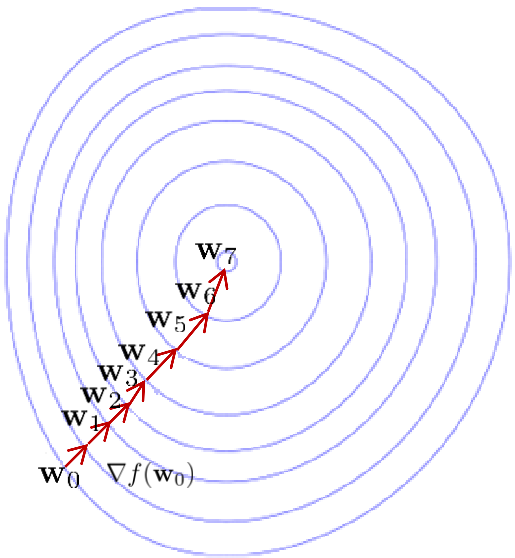
(b) Pick a step size λ_t

(c) $w_{t+1} = w_t - \lambda_t \times \nabla \mathcal{L}(w_t)$ such that function decreases

(d) Repeat

Gradient descent





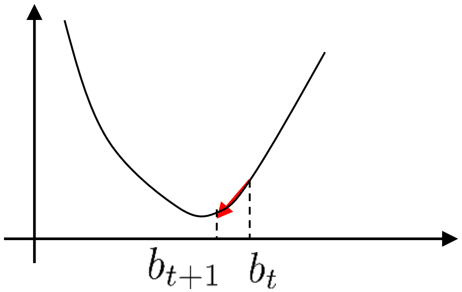
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \lambda_t \nabla \mathcal{L}(\mathbf{w}_t) \quad \lambda_t : stepsize$$

Theorem:

For a continuous differentiable function $\mathcal{L}$ on a neighborhood of $\mathbf{w}_0$, if $v^T \nabla \mathcal{L}(\mathbf{w}) < 0$, then there exists $T > 0$ such that $\mathcal{L}(\mathbf{w}_0 + tv) < \mathcal{L}(\mathbf{w}_0), \forall t \in (0, T]$.

Training a logistic regression classifier

$$\mathcal{L}(w,b) = \sum_{i=1}^{n} \ln(1 + e^{-y_i \times (w \times x_i + b)})$$



$$b_{t+1} = b_t - \lambda_t \times \nabla_b \mathcal{L}(w,b)$$

$$\nabla_b \mathcal{L}(w,b) = \sum_i -y_i \times (1 - p(y_i|x_i)) \times 1$$

$$w_{t+1} = w_t - \lambda_t \times \nabla_w \mathcal{L}(w,b)$$

$$\nabla_w \mathcal{L}(w,b) = \sum_i -y_i \times (1 - p(y_i|x_i)) \times x_i$$

Training a logistic regression classifier

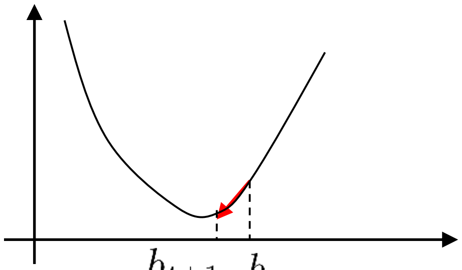
Intuition: find the best parameters $(w, b)^*$ to maximize the probabilities of fitting the ground-truth label y_i for each x_i.

Math: $(w, b)^* = \arg\min_{(w,b)} \sum_{i=1}^{n} -\ln(\frac{1}{1+e^{-y_i \times (w \times x_i + b)}})$

$$= \arg\min_{w,b} \mathcal{L}(w, b)$$

Training a logistic regression classifier

$$\mathcal{L}(w,b) = \sum_{i=1}^{n} \ln(1 + e^{-y_i \times (w \times x_i + b)})$$



$$b_{t+1} = b_t - \lambda_t \times \nabla_b \mathcal{L}(w,b)$$

$$\nabla_b \mathcal{L}(w,b) = \sum_i -y_i \times (1 - p(y_i|x_i)) \times 1$$

$$w_{t+1} = w_t - \lambda_t \times \nabla_w \mathcal{L}(w,b)$$

$$\nabla_w \mathcal{L}(w,b) = \sum_i -y_i \times (1 - p(y_i|x_i)) \times x_i$$

Training a logistic regression classifier

$\mathbf{x}_i \in \mathbb{R}^m, i = 1..n \qquad y_i \in \{-1, +1\}, i = 1..n$

$$p(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}$$

Model parameters:

$\mathbf{w} \in \mathbb{R}^m$

$b \in \mathbb{R}$

Training a logistic regression classifier

$$\mathbf{x}_i \in \mathbb{R}^m, i = 1..n \qquad y_i \in \{-1, +1\}, i = 1..n$$

Model parameters: $\mathbf{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$

$$p(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}$$

Intuition: find the best parameters $(\mathbf{w}, b)^*$ to maximize the probabilities of fitting the ground-truth label y_i for each x_i.

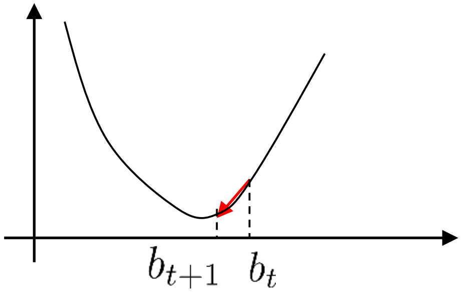
Math: $(\mathbf{w}, b)^* = \arg\min_{(\mathbf{w},b)} \sum_{i=1}^{n} -\ln(\frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}})$

$$= \arg\min_{\mathbf{w},b} \mathcal{L}(b, \mathbf{w})$$

Multivariate input

$$p(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}$$

Train a logistic regression classifier $f(x) = \begin{cases} +1 & if\ \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x}+b)}} \geq 0.5 \\ -1 & otherwise \end{cases}$:



$$(b, \mathbf{w})^* = \arg\min_{(b,\mathbf{w})} \mathcal{L}(b, \mathbf{w})$$

$$\mathcal{L}(b, \mathbf{w}) = \sum_{i=1}^{n} \ln(1 + e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)})$$

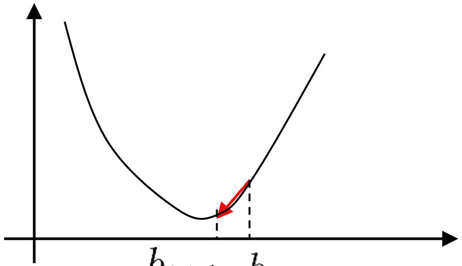
$$\nabla_b \mathcal{L}(b, \mathbf{w}) = \sum_i \frac{-y_i e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}} \quad = \sum_i -y_i(1 - p(y_i|\mathbf{x}_i))$$

$$\nabla_{\mathbf{w}} \mathcal{L}(b, \mathbf{w}) = \sum_i \frac{-y_i \mathbf{x}_i e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i b)}} \quad = \sum_i -y_i \mathbf{x}_i(1 - p(y_i|\mathbf{x}_i))$$

Multivariate input

$$p(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}$$

$$(b, \mathbf{w})^* = \arg\min_{(b,\mathbf{w})} \mathcal{L}(b, \mathbf{w})$$



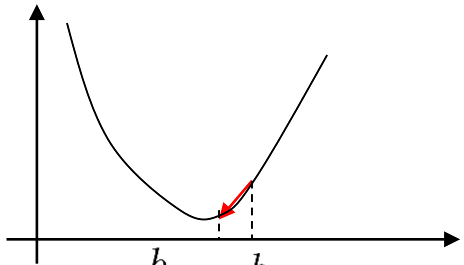
$$\mathcal{L}(b, \mathbf{w}) = \sum_{i=1}^{n} \ln(1 + e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)})$$

$$\nabla_b \mathcal{L}(b, \mathbf{w}) = \sum_i \frac{-y_i e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -y_i(1 - p(y_i|\mathbf{x}_i))$$

$$\nabla_{\mathbf{w}} \mathcal{L}(b, \mathbf{w}) = \sum_i \frac{-y_i \mathbf{x}_i e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -y_i\mathbf{x}_i(1 - p(y_i|\mathbf{x}_i))$$

Multivariate input

$$\mathcal{L}(b, \mathbf{w}) = \sum_{i=1}^{n} \ln(1 + e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i + b)})$$



$$\nabla_b \mathcal{L}(b, \mathbf{w}) = \sum_i \frac{-y_i e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -y_i(1 - p(y_i|\mathbf{x}_i))$$

$$\nabla_{\mathbf{w}} \mathcal{L}(b, \mathbf{w}) = \sum_i \frac{-y_i \mathbf{x}_i e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -y_i\mathbf{x}_i(1 - p(y_i|\mathbf{x}_i))$$

$$b_{t+1} = b_t - \lambda_t \times \nabla_b \mathcal{L}(b_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \times \nabla_{\mathbf{w}} \mathcal{L}(b_t, \mathbf{w}_t)$$

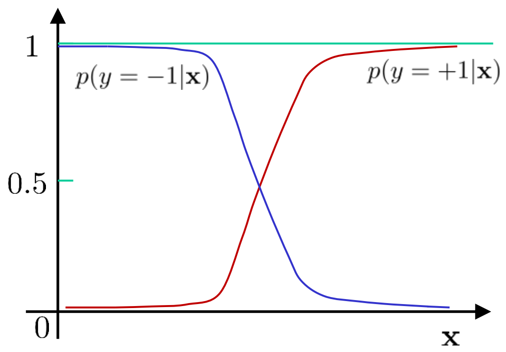
Logistic regression classifier

$$p(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)}}$$

$\mathbf{x} \in \mathbb{R}^m$

$y \in \{-1, +1\}$

$$f(\mathbf{x}) = \begin{cases} +1 & if \ \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x}+b)}} \geq 0.5 \\ -1 & otherwise \end{cases}$$



Pros:

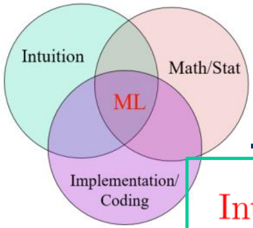
1. It is well-normalized.
2. Easy to turn into probability.
3. Easy to implement.

Cons:

1. Indirect loss function.
2. Dependent on good feature set.
3. Weak on feature selection.

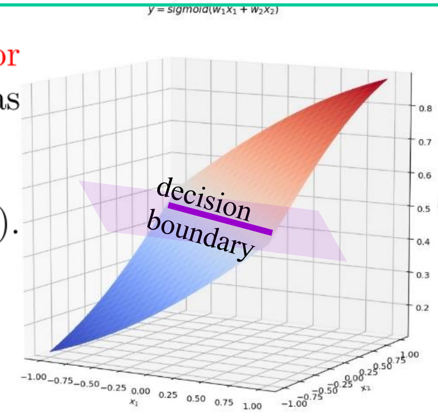
Take home message

- Logistic regression classifier is still a linear classifier but with a probability output.
- It can be trained using a gradient descent algorithm.
- The "regression" refers to fitting the discriminative probabilities: $p(y|\mathbf{x})$
- It has been widely adopted in practice, especially in the modern deep learning era.



Recap: Logistic Regression Classifier

Intuition: Logistic regression classifier nicely turns a hard classification error (0 *or* 1) into a soft measure using the sigmoid function $\sigma(v) = \frac{1}{1+e^{-v}}$ which has three particularly appealing properties:



- A soft measure that maps any value $v \in (-\infty, \infty)$ to a normalized $\rightarrow (0, 1)$.
- Nice gradient form.
- Convex function for the objective function in training.

Math:

$$p(y|\mathbf{x}) = \frac{1}{1+e^{-y(\mathbf{w}\cdot\mathbf{x}+b)}}$$

$Training:$

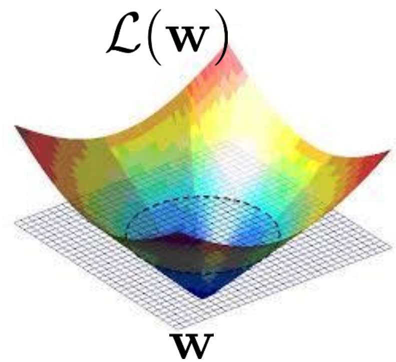
$$(b, \mathbf{w})^* = \arg\min_{(b,\mathbf{w})} \mathcal{L}(b, \mathbf{w}) = \arg\min_{(b,\mathbf{w})} \sum_{i=1}^{n} \ln(1 + e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i+b)})$$

$$\nabla_b \mathcal{L}(b, \mathbf{w}) = \sum_i -y_i \times (1 - p(y_i|\mathbf{x}_i))$$

$$\nabla_{\mathbf{w}} \mathcal{L}(b, \mathbf{w}) = \sum_i -y_i \times \mathbf{x}_i(1 - p(y_i|\mathbf{x}_i))$$

Recap: Logistic Regression Classifier

Implementation:



Gradient Descent Direction

(a) Pick a direction $\nabla \mathcal{L}(\mathbf{w}_t, b_t)$

(b) Pick a step size λ_t

(c) $\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \times \nabla \mathcal{L}_{\mathbf{w}_t}(\mathbf{w}_t, b_t)$ such that function decreases;
$b_{t+1} = b_t - \lambda_t \times \nabla \mathcal{L}_{b_t}(\mathbf{w}_t, b_t)$

(d) Repeat