MSAI-437: Deep Learning

Winter 2025

Foundations of Machine Learning

Comparison of 349 vs. 437

349 Topics	437 Topics
Linear Regression	Foundations of ML
Linear Discriminants	
Perceptron Algorithm	
Gradient Descent	Gradient Descent
Feed Forward Neural Networks	MLPs and Backpropagation
Backpropagation	
~~Decision Trees~~	Regularization
~~K-Nearest Neighbors~~	Convolutional Neural Networks
~~K-Means Clustering~~	Adversarial Examples
~~Calculation Graphs~~	Generative Adv ersarial Networks
~~Deep Learning Platforms~~	Autoencoders
~~Probability Basics~~	Recurrent Neural Networks
~~Naïve Bayes Classifiers~~	Transformers
~~Hypothesis Testing~~	Deep Reinforcement Learning
~~Gaussian Mixture Models~~	Diffusion Models

Supervised Machine Learning in One Slide

1. Pick data **D,** model (function) $\mathbf{M}(\mathbf{w})$ and objective function $\mathbf{J}(\mathbf{D}, \mathbf{w})$
2. Initialize model weights (parameters) $\mathbf{w}$ somehow
3. Measure model performance with the objective function $\mathbf{J}(\mathbf{D}, \mathbf{w})$
4. Modify parameters $\mathbf{w}$ somehow, hoping to improve $\mathbf{J}(\mathbf{D}, \mathbf{w})$
5. Repeat 3 and 4 until you stop improving or run out of time

Classification Learning Task

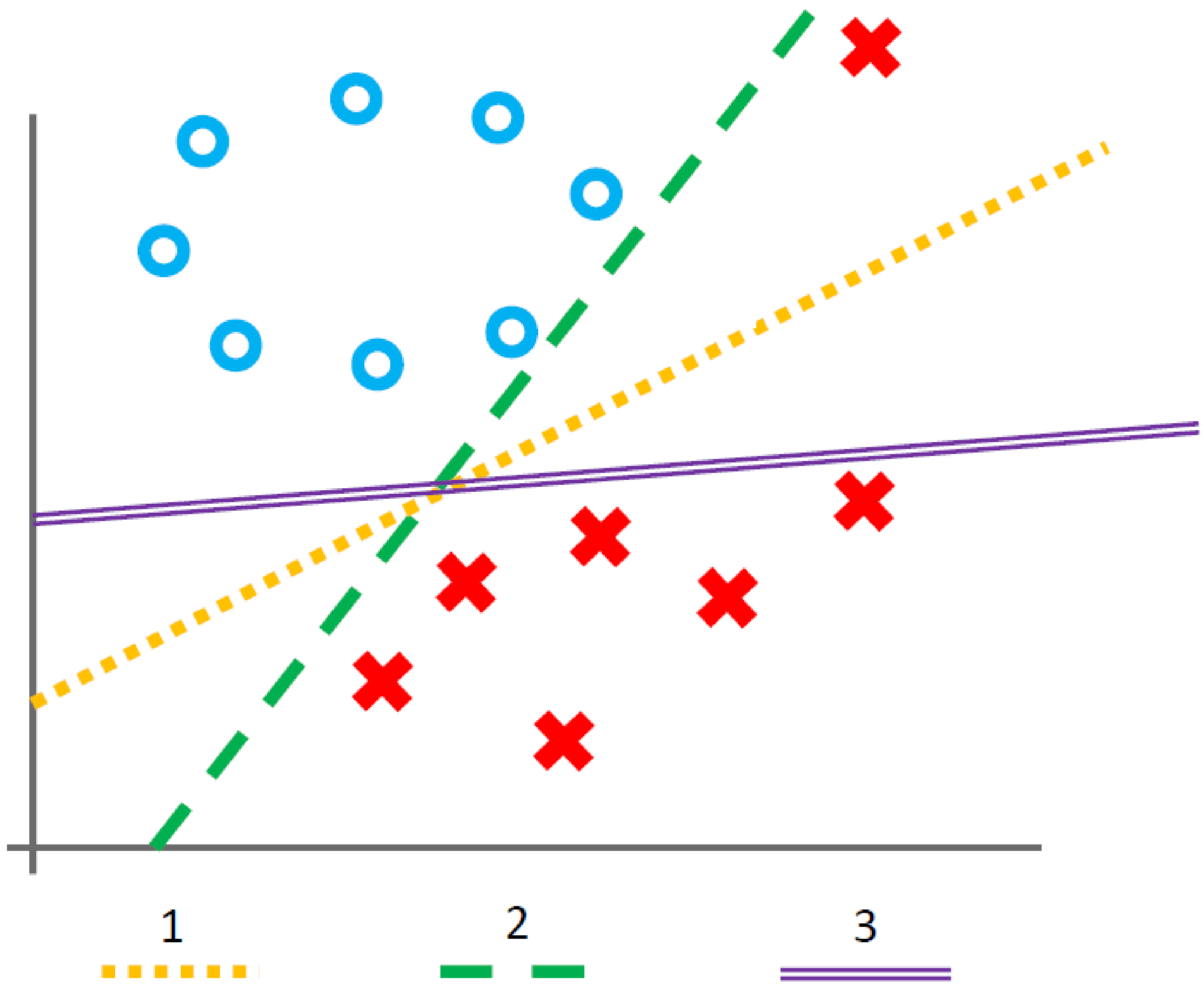
- We have a data matrix X of shape (n, d) $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$
- Each of the n rows is a d-dimensional vector $\mathbf{x_i} = \langle x_{i,1}, \ldots, x_{i,d} \rangle$
- Assume there's a true function f() that outputs a **discrete** label for each X_i

$$\mathbf{y} = \{y_1, \ldots, y_n\}$$

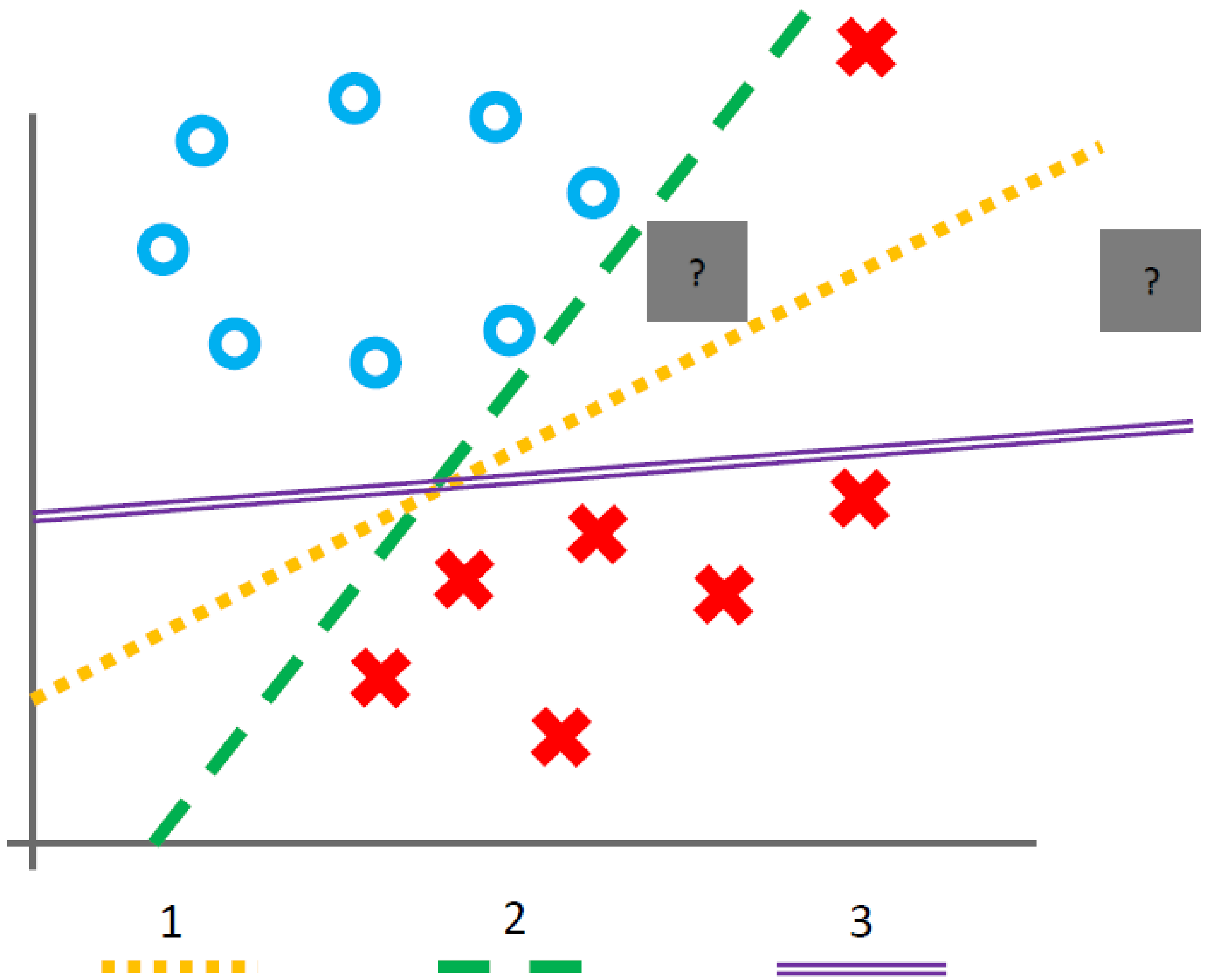
$$y_i = f(\mathbf{x_i})$$

- Our goal is to find h() that approximates f() $\mathcal{D} = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$

Classification Example

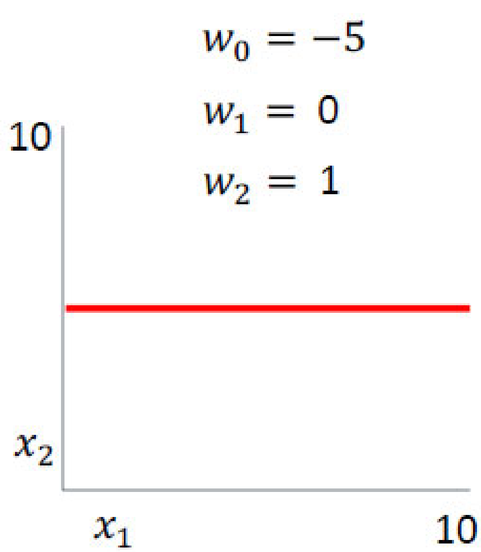


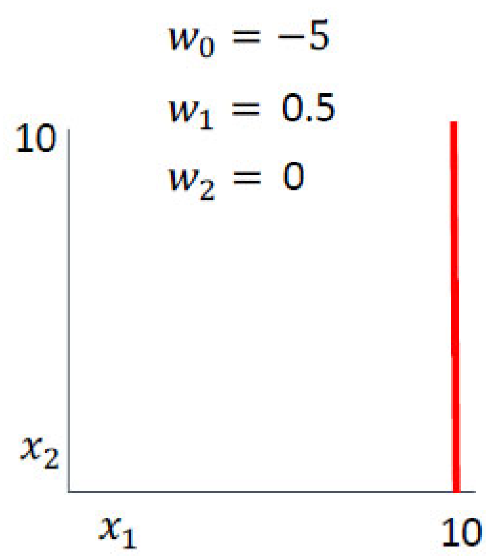
Classification Example

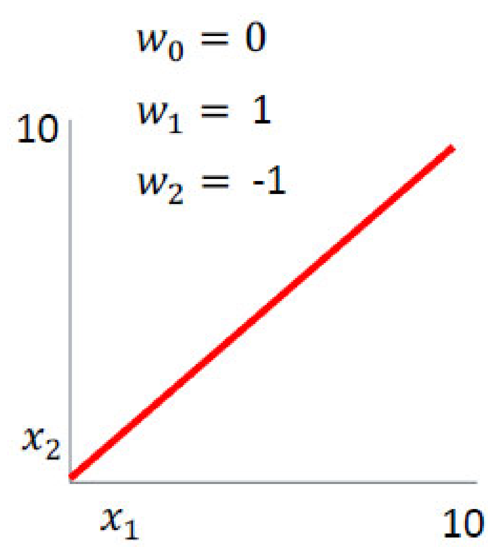


Decision Boundary

$$0 = g(x) = w_0 + w_1 x_1 + w_2 x_2 = \mathbf{xw}$$



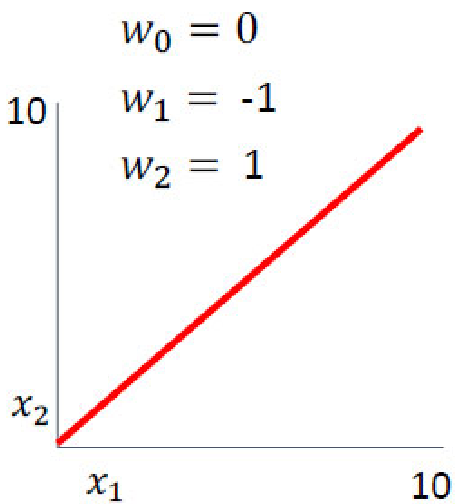


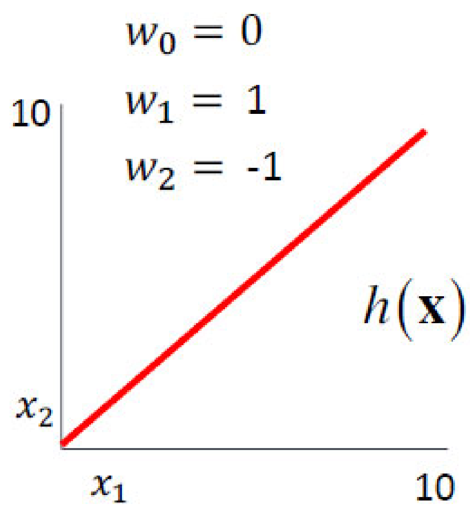


Decision Boundary (*Cont.*)

$$0 = g(x) = w_0 + w_1 x_1 + w_2 x_2 = \mathbf{xw}$$

What's the difference between these two?





$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$

Matrix Notation

Create a feature column of all ones that we'll implicitly multiply by w_0 :

$$\mathbf{X} = \begin{bmatrix} 1, x_{1,1} & \dots & x_{1,d} \\ 1, x_{2,1} & \dots & x_{2,d} \\ \vdots & \ddots & \vdots \\ 1, x_{n,1} & \dots & x_{n,d} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

For our linear classification hypothesis class:

$$g(x) = w_0 + w_1 x_1 + w_2 x_2 = \boldsymbol{xw}$$

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$

Loss/Cost/Objective Function

- To train a model (e.g. learn the weights of a useful line) we define a measure of the ”goodness” of that model. (e.g. the number of misclassified points).

- We make that measure a function of the parameters of the model (and the data).

- This is called a loss function, or an objective function.

- We want to minimize the loss (or maximize the objective) by picking good model parameters.

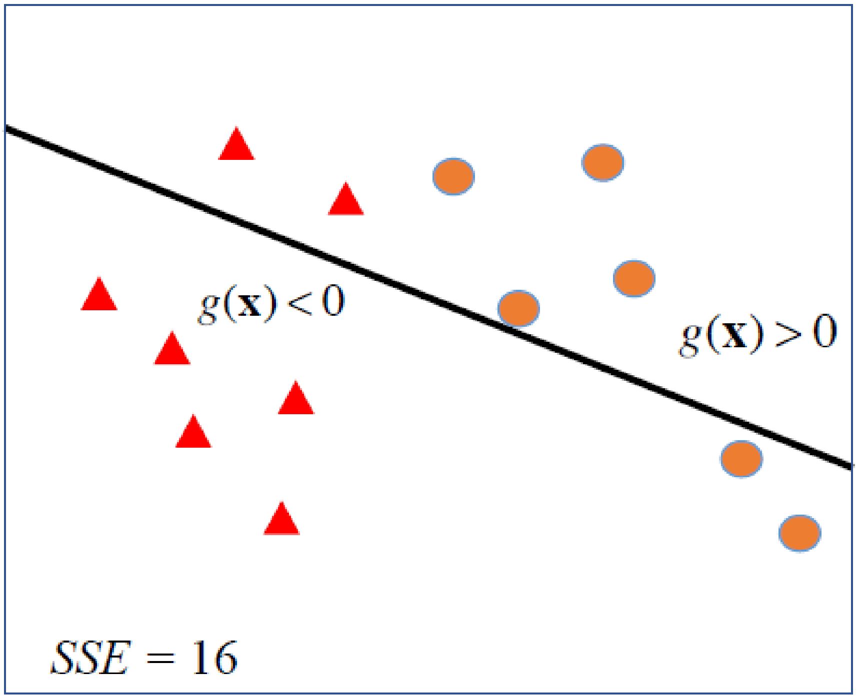
Loss/Cost/Objective Function

- Let’s define an objective (aka “loss”) function that directly measures the thing we want to get right

- Then let’s try and find the line that minimizes the loss.

- How about basing our loss function on the number of misclassifications?

Objective Function: Sum of Squared Errors (SSE)

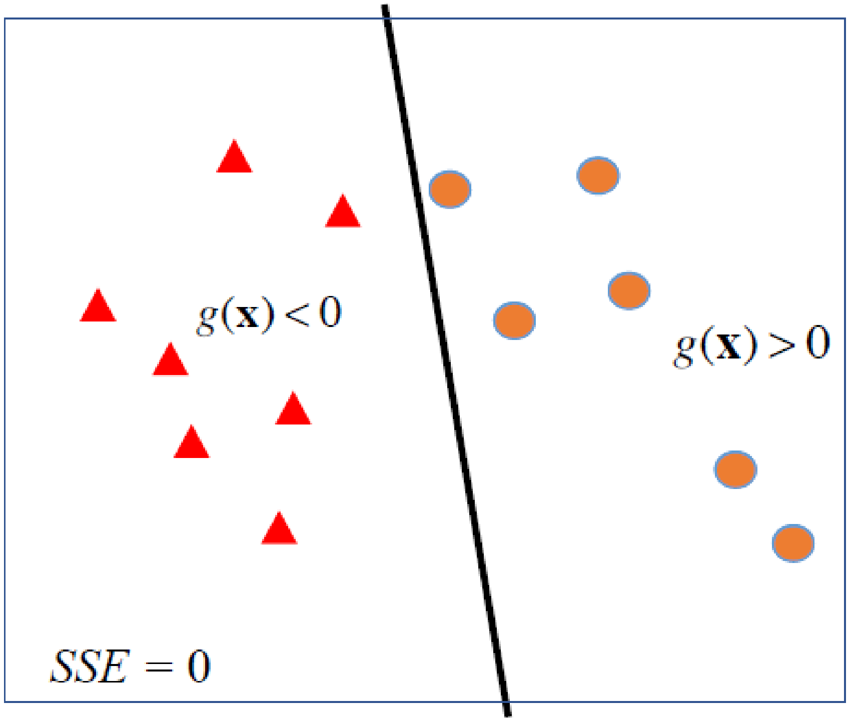


$$g(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 = 0$$
$$= \boldsymbol{xw}$$

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$SSE = \sum_{i}^{n} (y_i - h(\mathbf{x}_i))^2$$

Objective Function: Sum of Squared Errors (SSE)



$$g(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 = 0$$
$$= \boldsymbol{xw}$$

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$SSE = \sum_{i}^{n} (y_i - h(\mathbf{x}_i))^2$$

No Closed-Form Solution

- For many objective (aka loss) functions we can’t find a formula to to get the best model parameters, like one can with regression.
- The objective function from the previous slide is one of those ”no closed form solution” functions.
- This means we must try various guesses for what the weights should be.
- Let’s look at the perceptron approach.

Perceptron Classifier

- Binary classifier
- We'll learn a linear decision boundary

$$0 = g(x) = \mathbf{xw}$$

- Things on each side of 0 get their class labels according to the sign of what g(x) outputs.

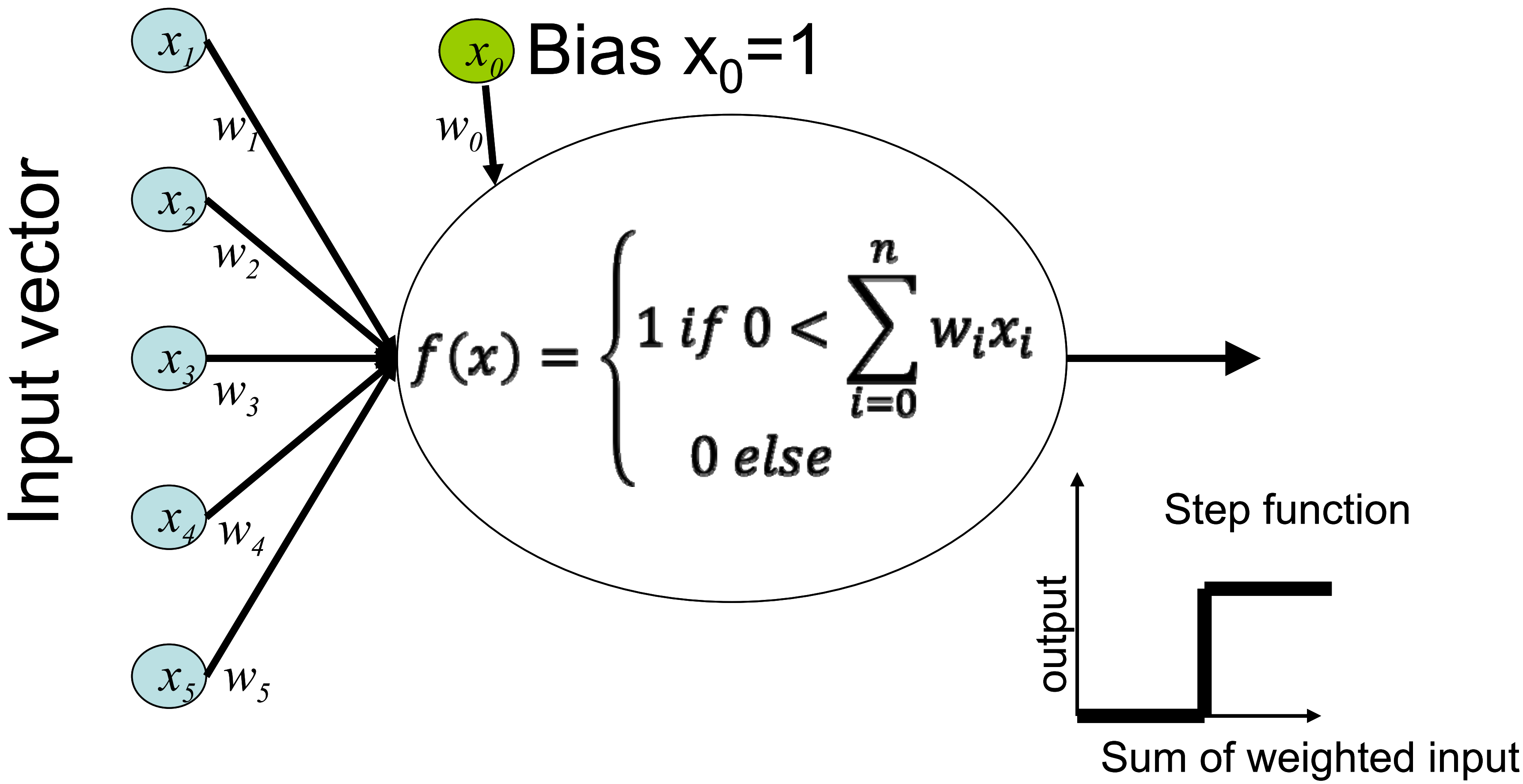
$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$

Perceptron Classifier

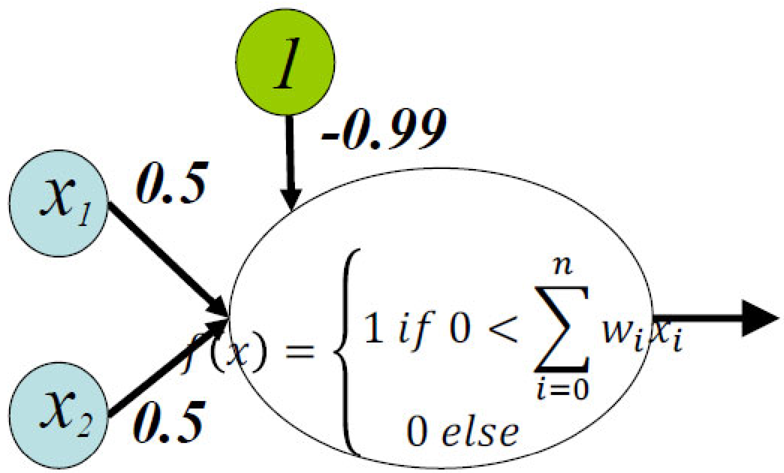
The perceptron: a probabilistic model for information storage and organization in the brain.
F Rosenblatt - Psychological review, 1958 - psycnet.apa.org
To answer the questions of how information about the physical world is sensed, in what form is information remembered, and how does information retained in memory influence recognition and behavior, a theory is developed for a hypothetical nervous system called a …
Cited by 13410 Related articles All 44 versions

- The "first wave" in neural networks
- A linear classifier

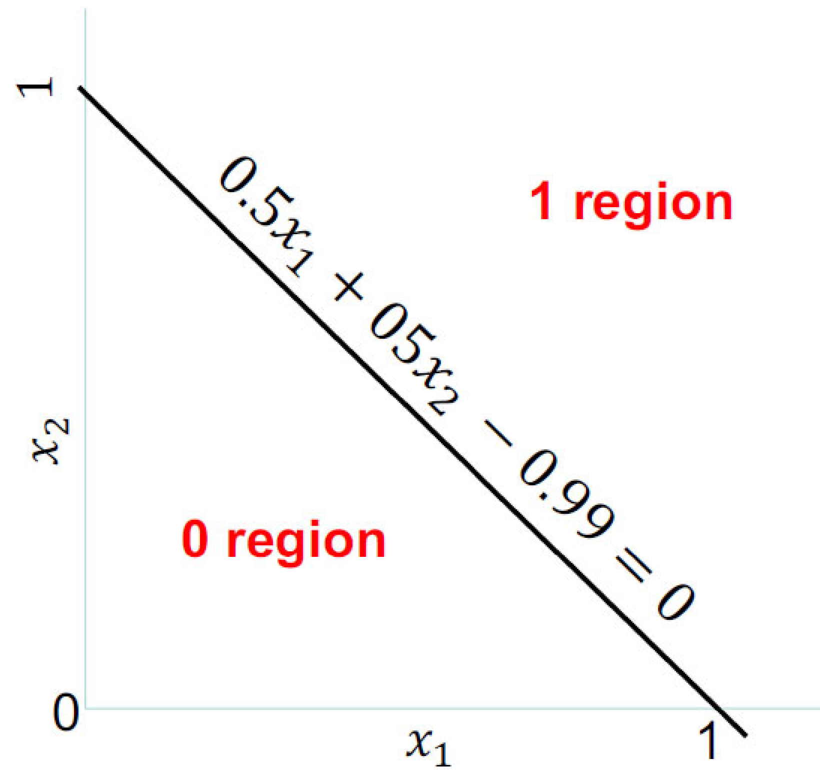
A Single Perceptron



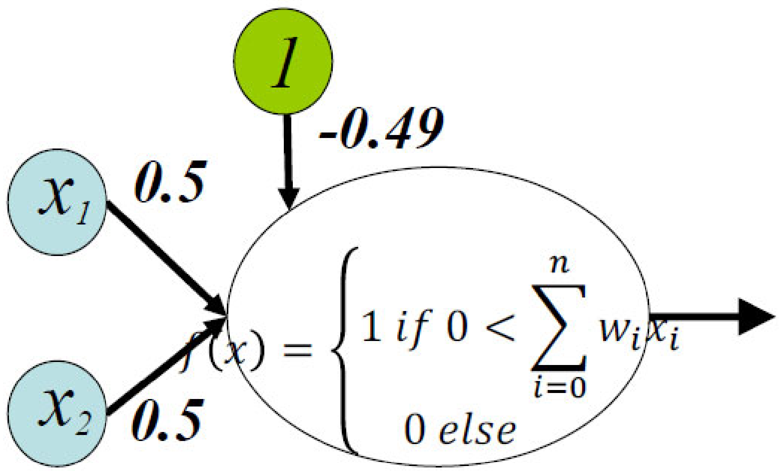
A Single Perceptron and Logical Functions



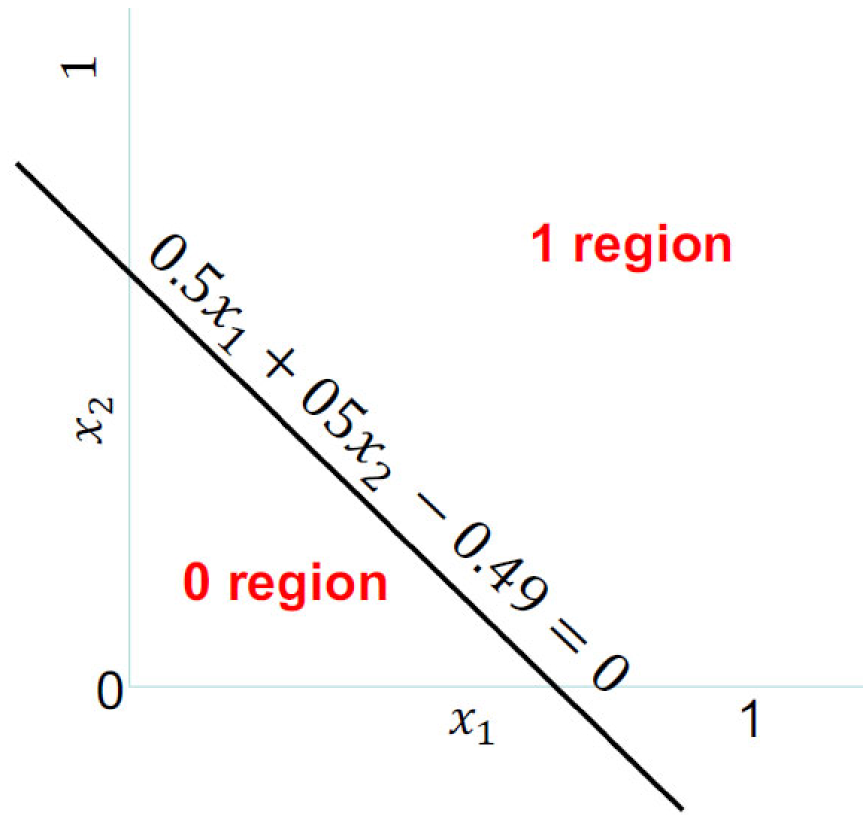
AND



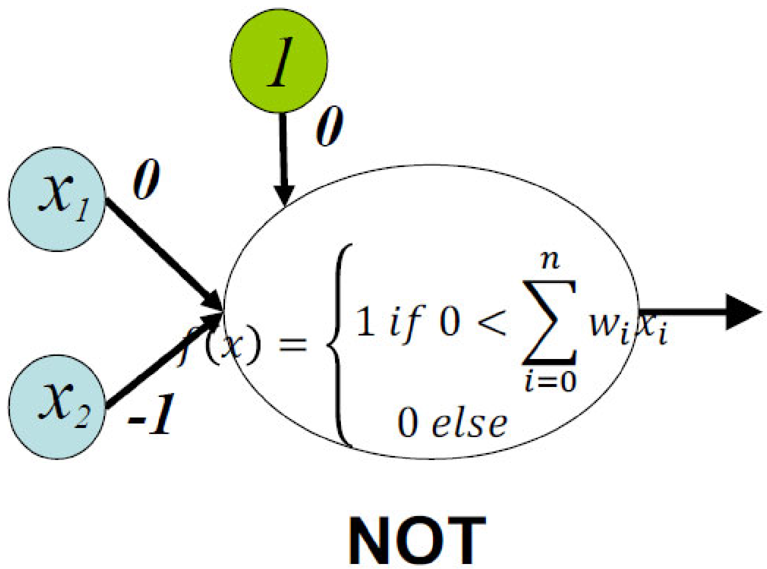
A Single Perceptron and Logical Functions (*Cont.*)

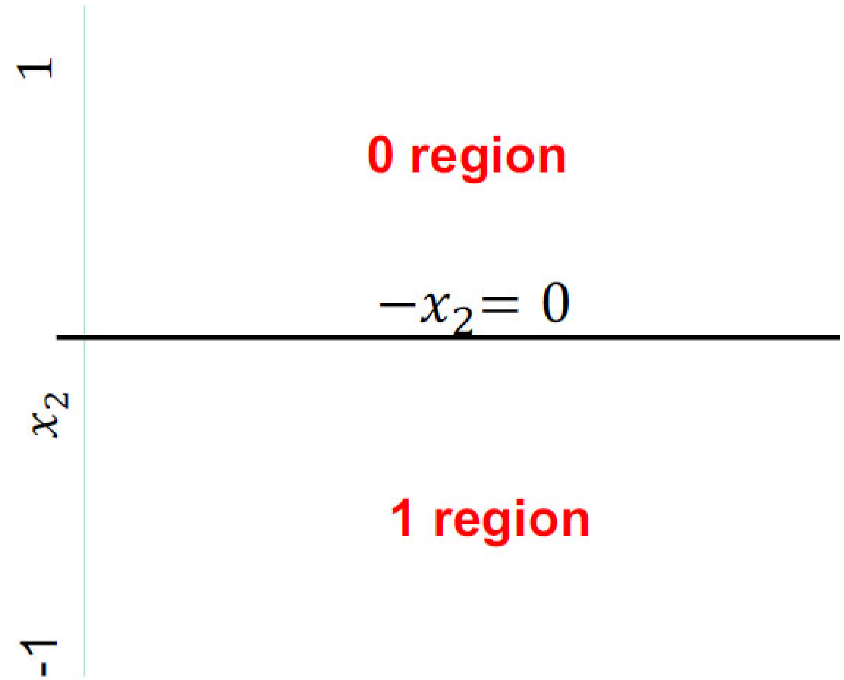


OR



A Single Perceptron and Logical Functions (Cont.)





Perceptron Classifier

Model $\mathbf{M}(\mathbf{w})$

$$h(\mathbf{x}_i) = \begin{cases} 1, & \text{if } \mathbf{x}_i\mathbf{w} > 0 \\ -1, & \text{otherwise} \end{cases}$$

Objective function $\mathbf{J}(\mathbf{D}, \mathbf{w})$

$$SSE = \sum_{i}^{n} (y_i - h(\mathbf{x}_i))^2$$

Let's parameters $\mathbf{w}$ somehow, hoping to improve $\mathbf{J}(\mathbf{D}, \mathbf{w})$.

How might we do this?

Reframing Our Objective Function

We want to get all positively-labeled points on the positive side of the boundary, all negatively-labeled points on the negative side.

$$SSE = \sum_{i}^{n} (y_i - h(\mathbf{x}_i))^2$$

Can phrase this as: $(\boldsymbol{xw})y > 0$ for all $(\mathbf{x}, y) \in D$

Why does this equation capture our goal?

Modifying Parameters

Model $\mathbf{M}(\mathbf{w})$

$$h(\mathbf{x}_i) = \begin{cases} 1, & \text{if } \mathbf{x}_i \mathbf{w} > 0 \\ -1, & \text{otherwise} \end{cases}$$

Objective function $\mathbf{J}(\mathbf{D}, \mathbf{w})$

$(\mathbf{x}\mathbf{w})y > 0$ for all $(\mathbf{x}, y) \in D$

Proposal:

$$\mathbf{w}^{(t+1)} = \begin{cases} \mathbf{w}^{(t)}, & \text{if } (\mathbf{x}_i \mathbf{w}^{(t)}) y_i > 0 \\ \mathbf{w}^{(t)} + \mathbf{x}_i y_i, & \text{otherwise} \end{cases}$$

Perceptron Algorithm Example

Perceptron Algorithm

$$h(\mathbf{x}_i) = \begin{cases} 1, & \text{if } \mathbf{x}_i \mathbf{w} > 0 \\ -1, & \text{otherwise} \end{cases}$$

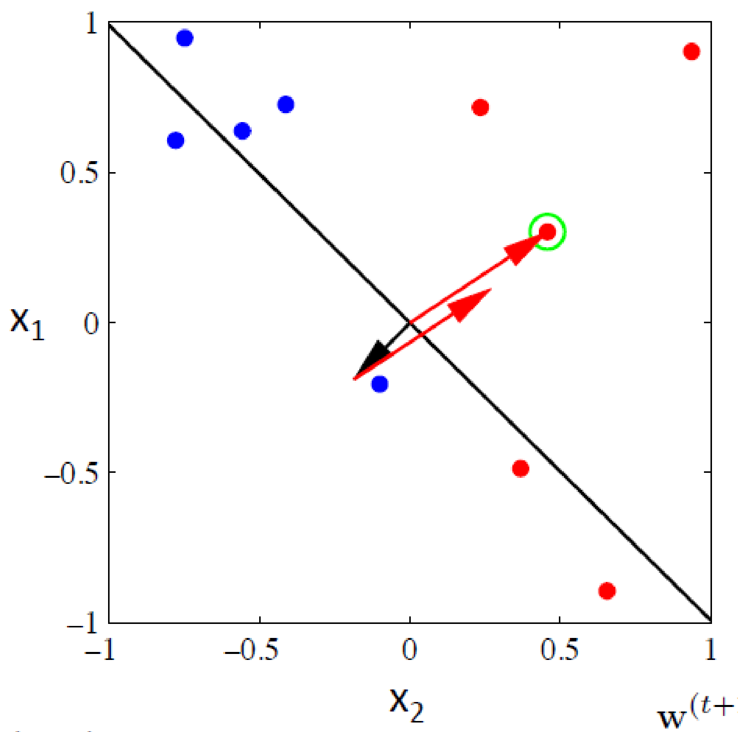
w_0	w_1	w_2
0*	-0.2	-0.1

$w^{(1)}$

x_0	x_1	x_2
0*	0.3	0.5

w_0	w_1	w_2
0*	0.1	0.4

$w^{(2)}$



+1: Red is the positive class

-1: Blue is the negative class

$$\mathbf{w}^{(t+1)} = \begin{cases} \mathbf{w}^{(t)}, & \text{if } (\mathbf{x}_i \mathbf{w}^{(t)}) y_i > 0 \\ \mathbf{w}^{(t)} + \mathbf{x}_i y_i, & \text{otherwise} \end{cases}$$

* Intercept is fixed to 0 for this example only

Regression Learning Task

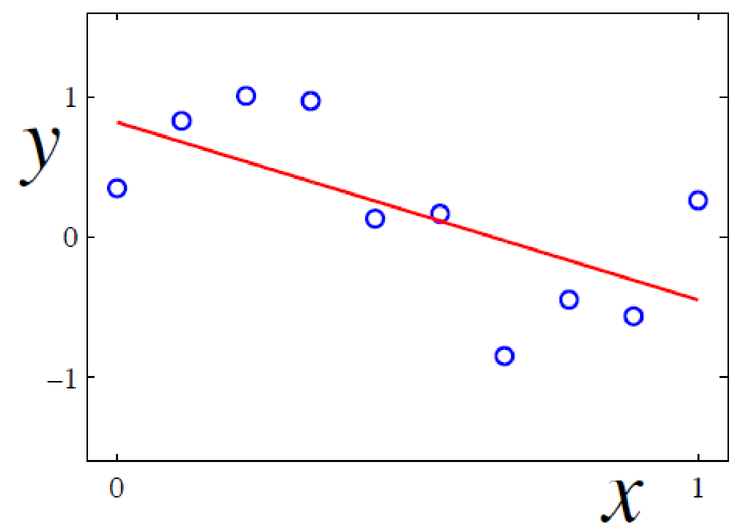
- We have a data matrix X of shape (n, d) $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$
- Each of the n rows is a d-dimensional vector $\mathbf{x_i} = \langle x_{i,1}, \ldots, x_{i,d} \rangle$
- Assume there's a true function f() that $\mathbf{y} = \{y_1, \ldots, y_n\}$

  outputs a **real-valued** label for each X_i $y_i = f(\mathbf{x_i})$
- Our goal is to find h() that approximates f() $\mathcal{D} = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$

Simple Linear Regression

- x and y both have a single dimension
- Hypothesis function is a straight line

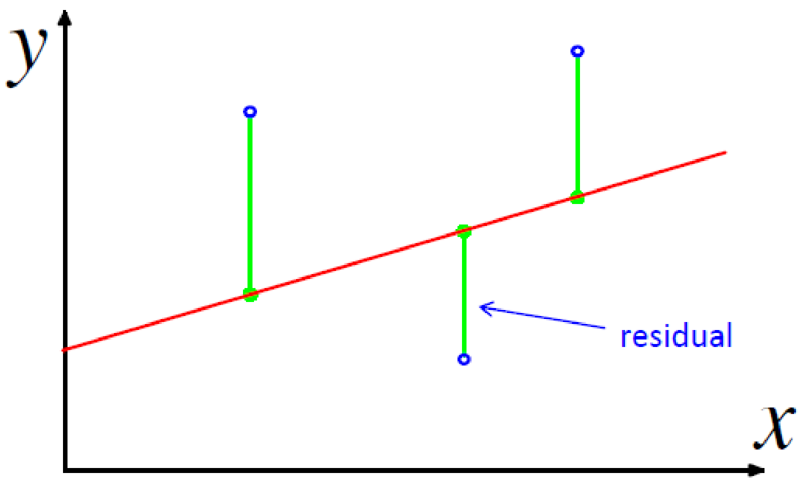
$$\hat{y} = h(x) = w_0 + w_1 x$$

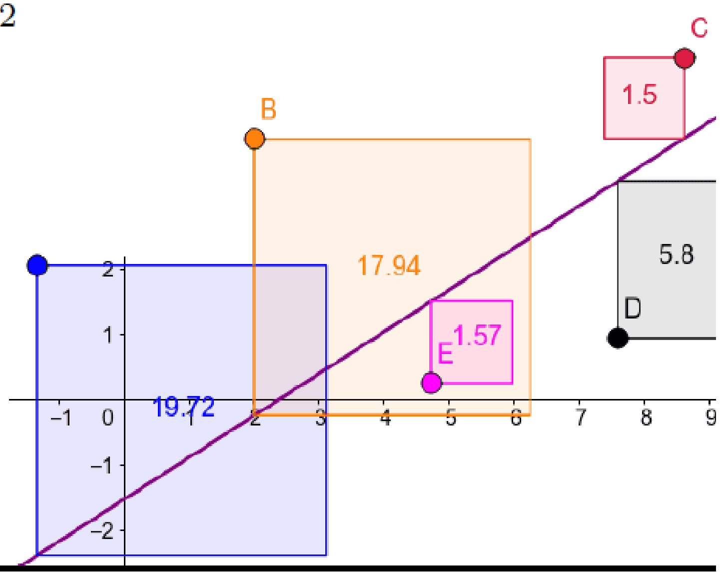


Objective Function: Mean Squared Error (MSE)

- Minimize the mean squared error (MSE)
  - Also called sum of squared residuals, sum of squared errors

$$\mathrm{MSE} = \sum_{i=1}^{N} (y_i - h(\mathbf{x_i}))^2$$

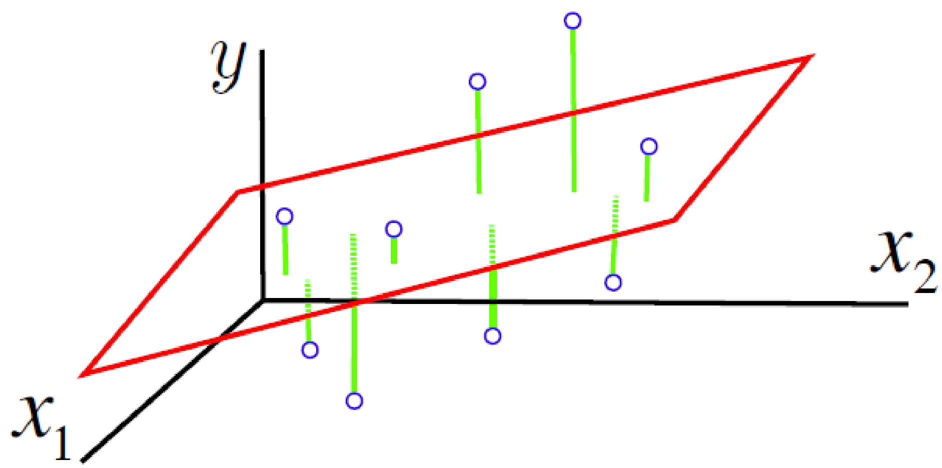




Multiple Linear Regression

- Many features, not just one
- Hypothesis space now of hyperplanes, not lines

$$h(\mathbf{x_i}) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \ldots + w_d x_{i,d}$$



Matrix Notation

Recall our hypothesis class:

$$h(\mathbf{x_i}) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \ldots + w_d x_{i,d}$$

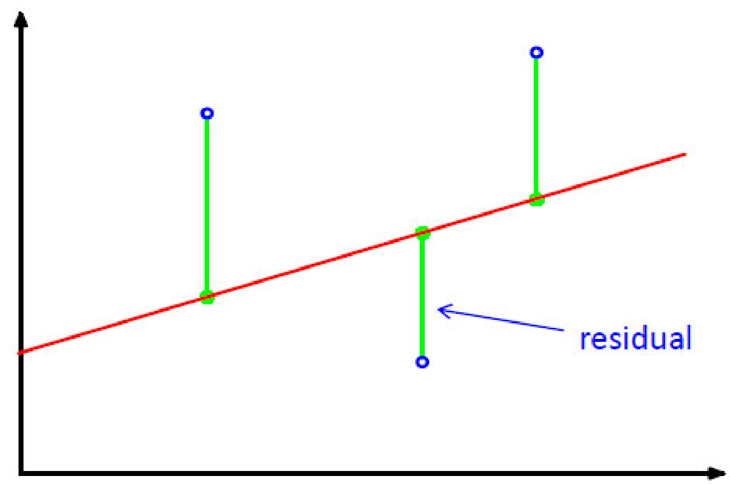
Create a feature column of all ones that we'll implicitly multiply by w_0 :

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ x_{2,1} & \cdots & x_{2,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \longrightarrow \mathbf{X} = \begin{bmatrix} 1, x_{1,1} & \cdots & x_{1,d} \\ 1, x_{2,1} & \cdots & x_{2,d} \\ \vdots & \ddots & \vdots \\ 1, x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$h(\mathbf{x_i}) = \mathbf{x_i}\mathbf{w}$$

Matrix Notation (*Cont.*)

$$
\begin{aligned}
\mathrm{MSE} &= \sum_{i=1}^{N} (y_i - h(\mathbf{x_i}))^2 \\
&= \sum_{i=1}^{N} (y_i - \mathbf{x_i w})^2 \\
&= \sum_{i=1}^{N} (y_i - \mathbf{x_i w})(y_i - \mathbf{x_i w}) \\
&= (\mathbf{y} - \mathbf{X w})^\top (\mathbf{y} - \mathbf{X w}) \\
&= (\mathbf{y} - \mathbf{X w})^{\mathbf{2}}
\end{aligned}
$$



$$
\frac{\partial \mathrm{MSE}}{\partial \mathbf{w}} = -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X w})
$$

Closed-From Solution

- Given the hypothesis class:

$$h(\mathbf{x_i}) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \ldots + w_d x_{i,d}$$

- And objective function:

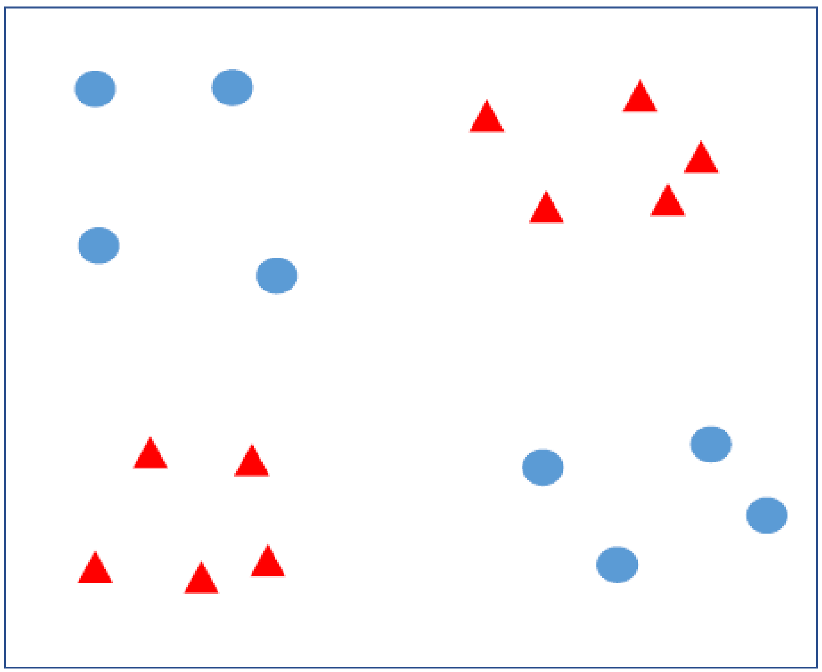
$$\mathrm{MSE} = \sum_{i=1}^{N} (y_i - h(\mathbf{x_i}))^2$$

- For any dataset there is a single best solution given by:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X} \mathbf{y}$$

Limitations

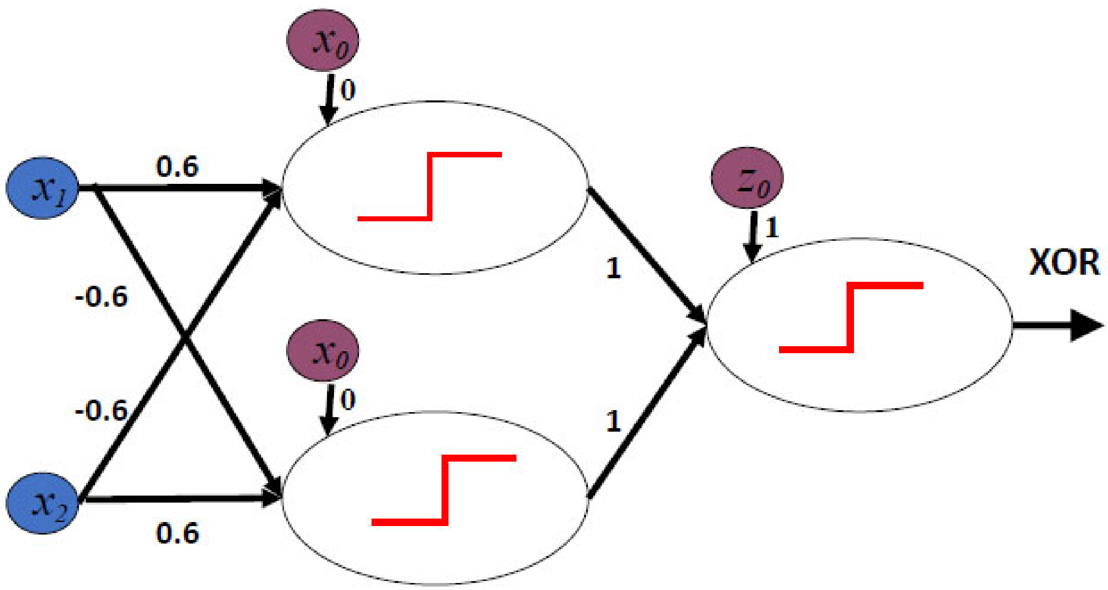
One perceptron: Only linear decisions



This is XOR.

It can't learn XOR.

Limitations (Cont.)



MLPs can fit any (Boolean) function

...if you can set the weights & connections right

Limitations (*Cont.*)

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X} \mathbf{y}$$

Limitations (Cont.)

349 Topics

- Linear Regression
- Linear Discriminants
- Perceptron Algorithm
- Gradient Descent
- Feed Forward Neural Networks
- Backpropagation
- ~~Decision Trees~~
- ~~K-Nearest Neighbors~~
- ~~K-Means Clustering~~
- ~~Calculation Graphs~~
- ~~Deep Learning Platforms~~
- ~~Probability Basics~~
- ~~Naïve Bayes Classifiers~~
- ~~Hypothesis Testing~~
- ~~Gaussian Mixture Models~~

ID3(D, Attributes, Target)

1. Create a node t for the tree.
2. Label t with the most common value of Target in D.
3. If all examples in D are positive, return the single-node tree t, with label "+".
   If all examples in D are negative, return the single-node tree t, with label "–".
4. If Attributes is empty, return the single-node tree t.

- Otherwise:
  5. Let A* be the attribute from Attributes that best classifies examples in D.
     Assign t the decision attribute A*.
  6. For each possible value "a" in A* do:
     - Add a new tree branch below t, corresponding to the test A* = "a".
     - Let D_a be the subset of D that has value "a" for A*.
     - If D_a is empty:
       Then add a leaf node with label of the most common value of Target in D.
       Else add the subtree ID3(D_a, Attributes \ {A*}, Target).

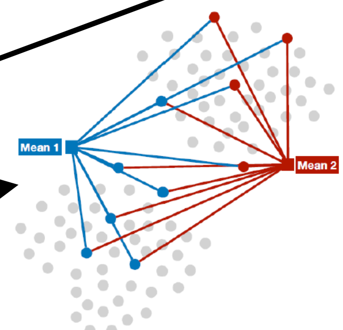
7. Return t.

distance function

$$\vec{x}_{nn} = \arg\min_{\vec{x}\in D}(d(\vec{x}, \vec{x}_q))$$

$$h(\vec{x}_q) = f(\vec{x}_{nn})$$

Our hypothesis



$$v_{MAP} = \operatorname*{argmax}_{v_j\in V} P(v_j|a_1, a_2 \ldots a_n) = \operatorname*{argmax}_{v_j\in V} \frac{P(a_1, a_2 \ldots a_n|v_j)P(v_j)}{P(a_1, a_2 \ldots a_n)}$$

$$= \operatorname*{argmax}_{v_j\in V} P(a_1, a_2 \ldots a_n|v_j)P(v_j) = \operatorname*{argmax}_{v_j\in V} \prod_i P(a_i|v_j)P(v_j)$$

$$\text{new } w_j = \frac{\Gamma_j}{N}$$

$$\text{new } \mu_j = \frac{\sum_{i=1}^{N} \gamma_{j,i} x_i}{\Gamma_j}$$

$$\text{new } \sigma_j^2 = \frac{\sum_{i=1}^{N} \gamma_{j,i}(x_i - \mu_j)^2}{\Gamma_j}$$