Learning the network

11-785 Introduction to Deep Learning

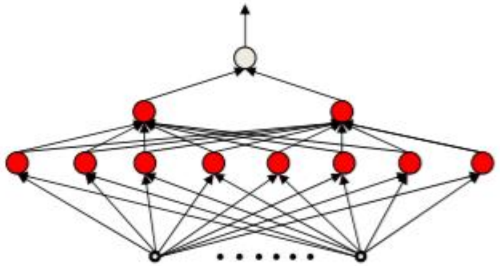
- lecture 3 -

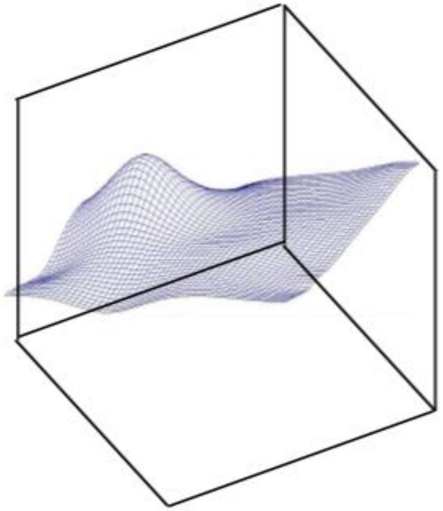
TAVE Research DL001
Heeji Won

Contents

Contents

01. How do we construct the network?

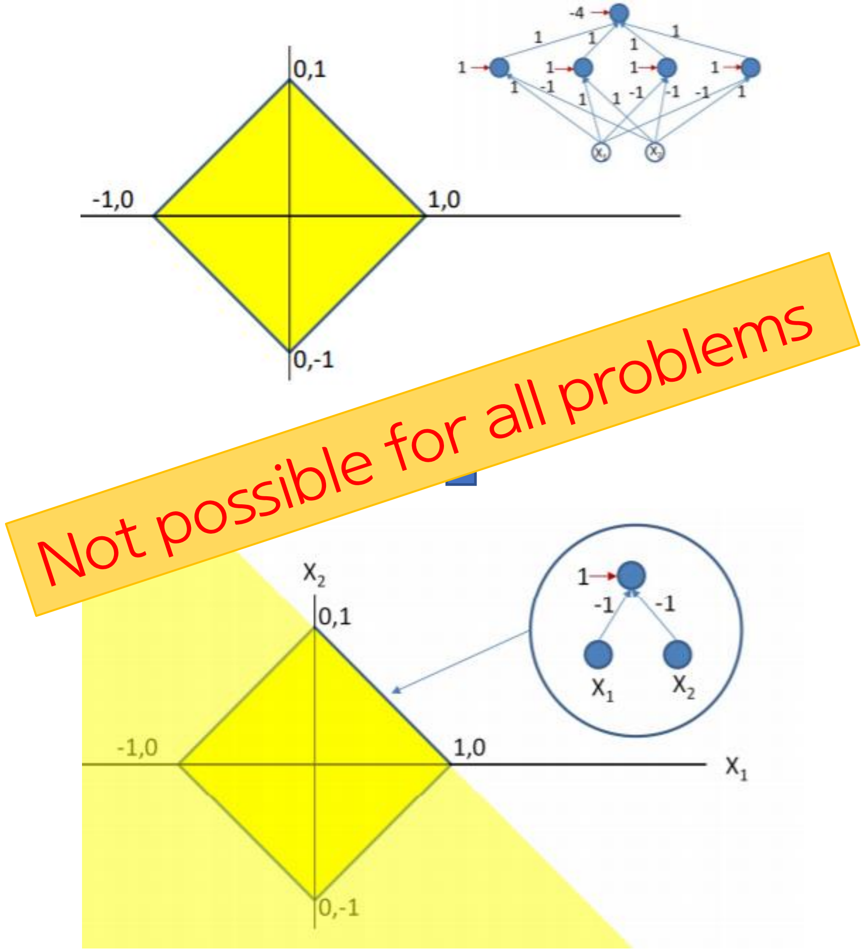




We learned …

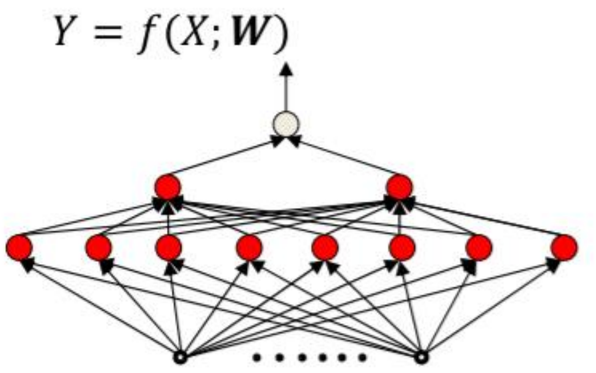
- The MLP can represent anything
- But how do we construct it?

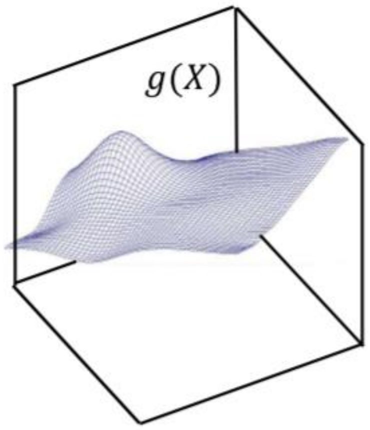
Option 1. Construct by hand



01. How do we construct the network?

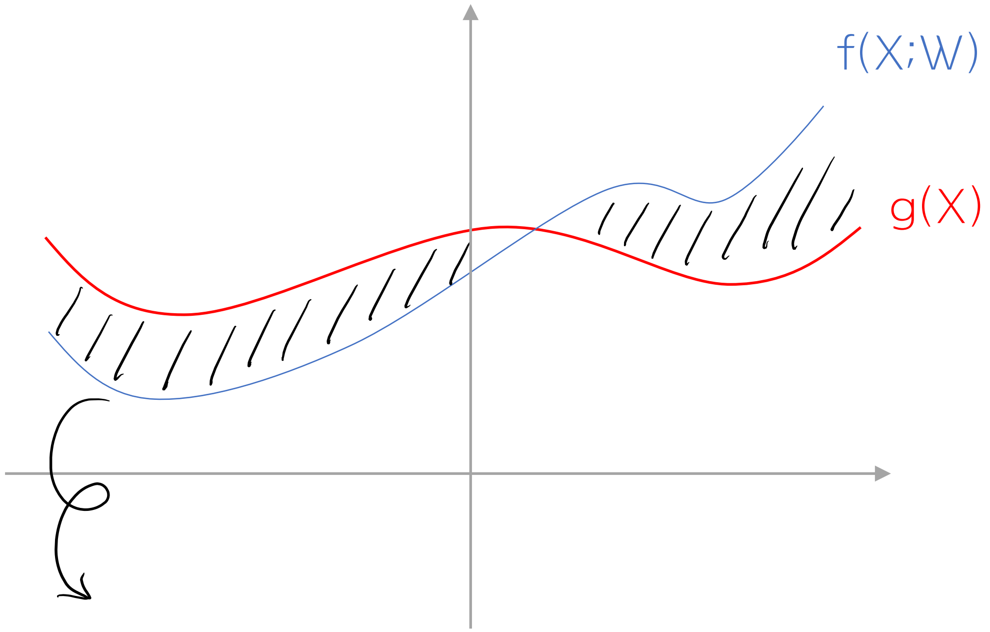
Option 2. Automatic estimation of an MLP





$$\widehat{W} = argmin_W \int_X div\big(f(X; W), g(X)\big)\, dX$$

- div() is a divergence function that goes to zero when f(X;W) = g(X)



find W minimizing this area

✓ In practice, g(X) is unknown

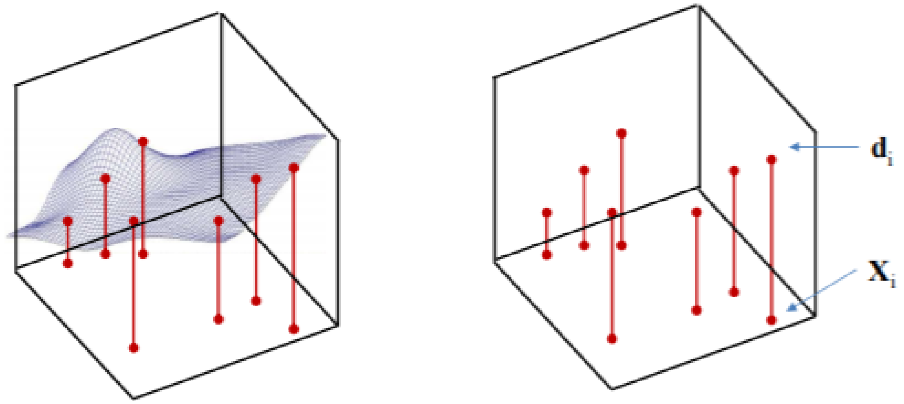
=> use training samples

01. How do we construct the network?

Option 2. Automatic estimation of an MLP

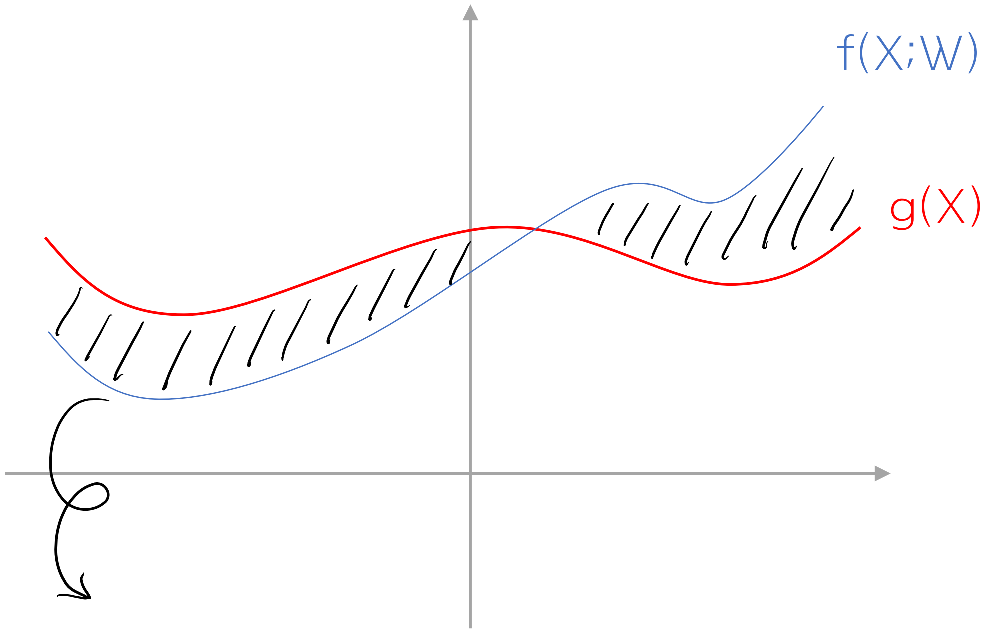
Sample g(X)

- get input–output pairs (X_i, d_i)



- We must learn the entire function from these few samples

✓ Estimate the network parameters to fit the training points exactly



find W minimizing this area

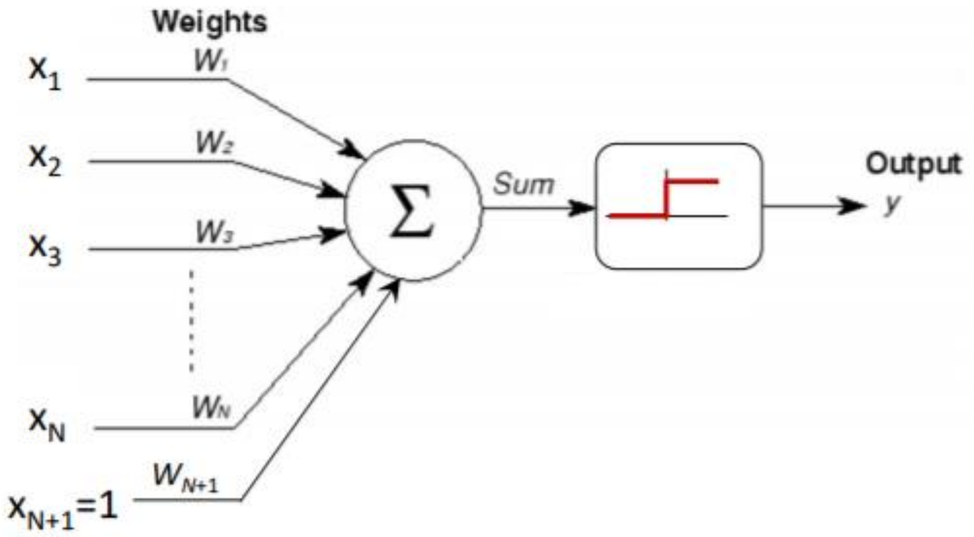
✓ In practice, g(X) is unknown

=> use training samples

Contents

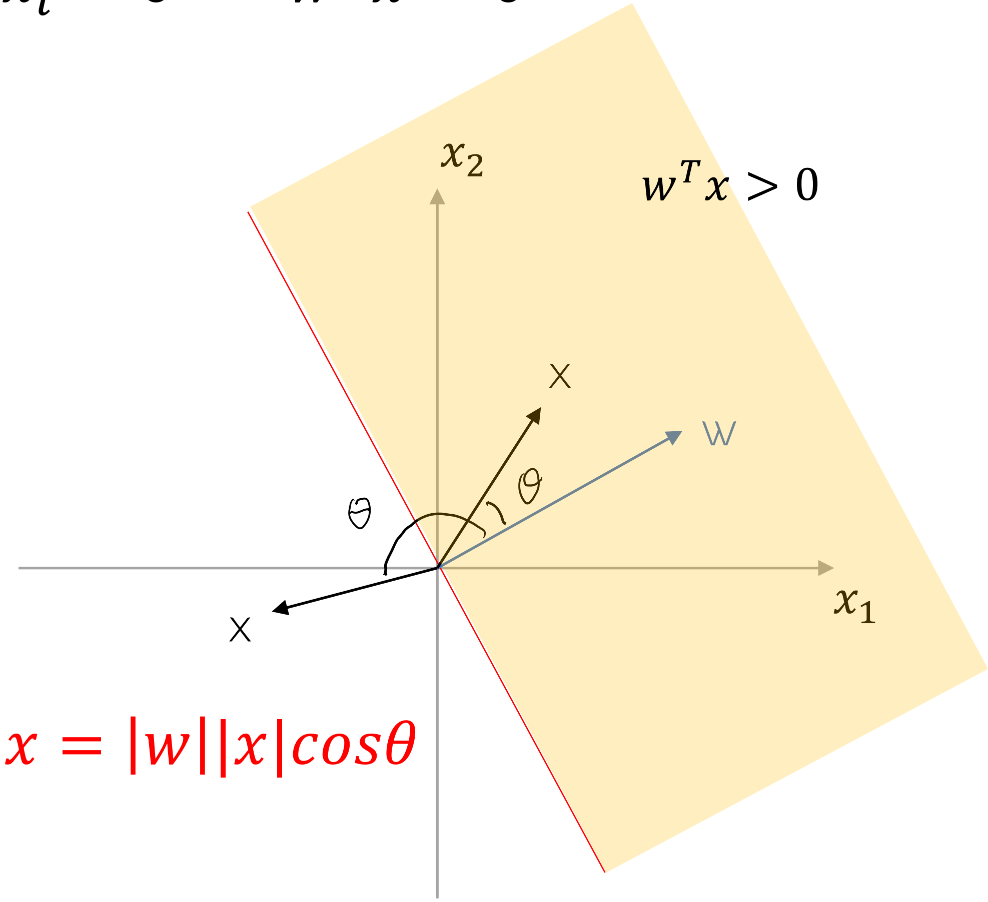
02. Perceptron Algorithm

- Learning the perceptron



✓ Learning the perceptron is the same as learning the hyperplane

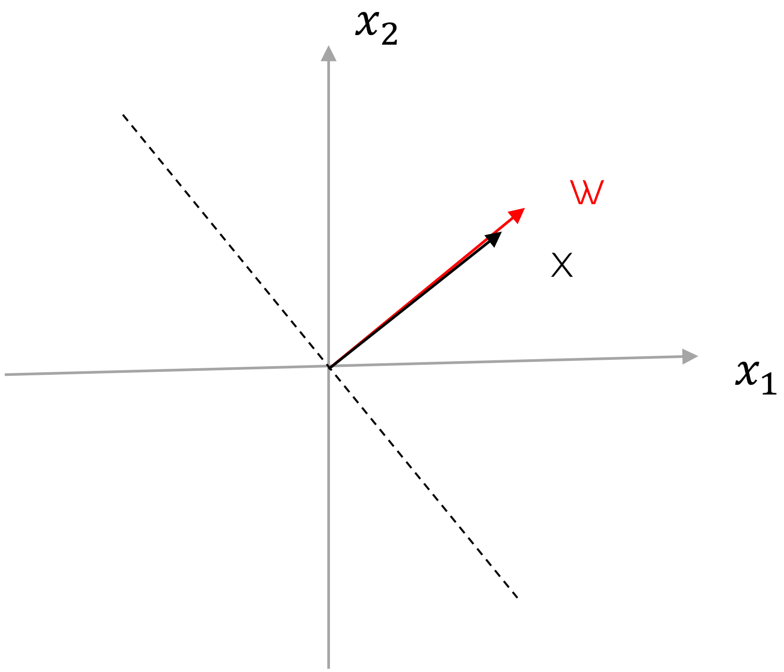
$$\sum_i w_i x_i = 0 \Leftrightarrow w^T x = 0$$



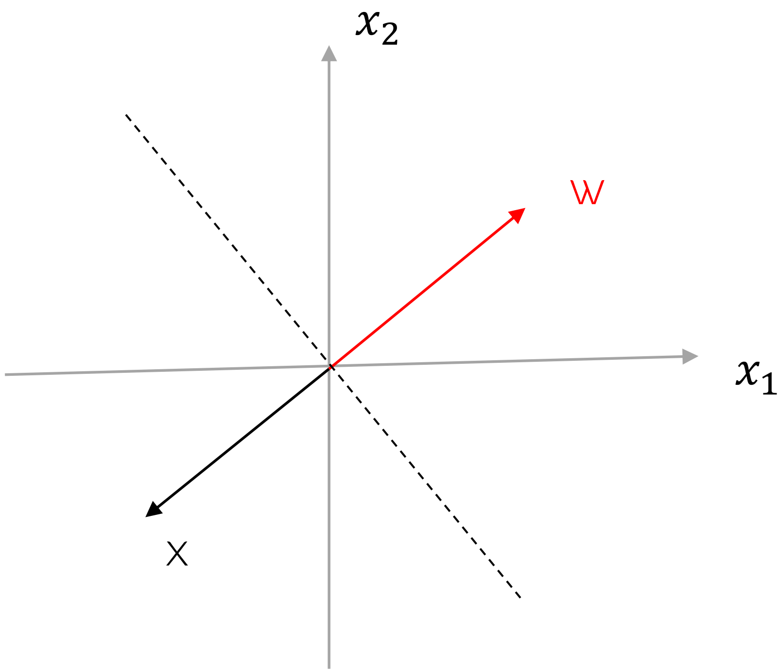
$$w^T x = |w||x| cos\theta$$

02. Perceptron Algorithm

- Learning the perceptron



$x \in +1$이면, $W = x$



$x \in -1$이면, $W = -x$

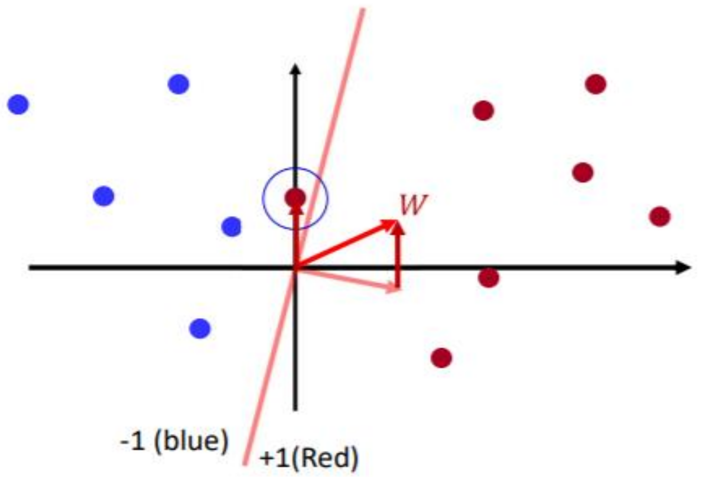
02. Perceptron Algorithm

- Perceptron Algorithm
  - Cycle through the training instances
  - **Only update W on misclassified** instances
  - If instance misclassified
    - If instance is **positive class** (positive misclassified as negative)

$$W = W + X_i$$

    - If instance is **negative class** (negative misclassified as positive)

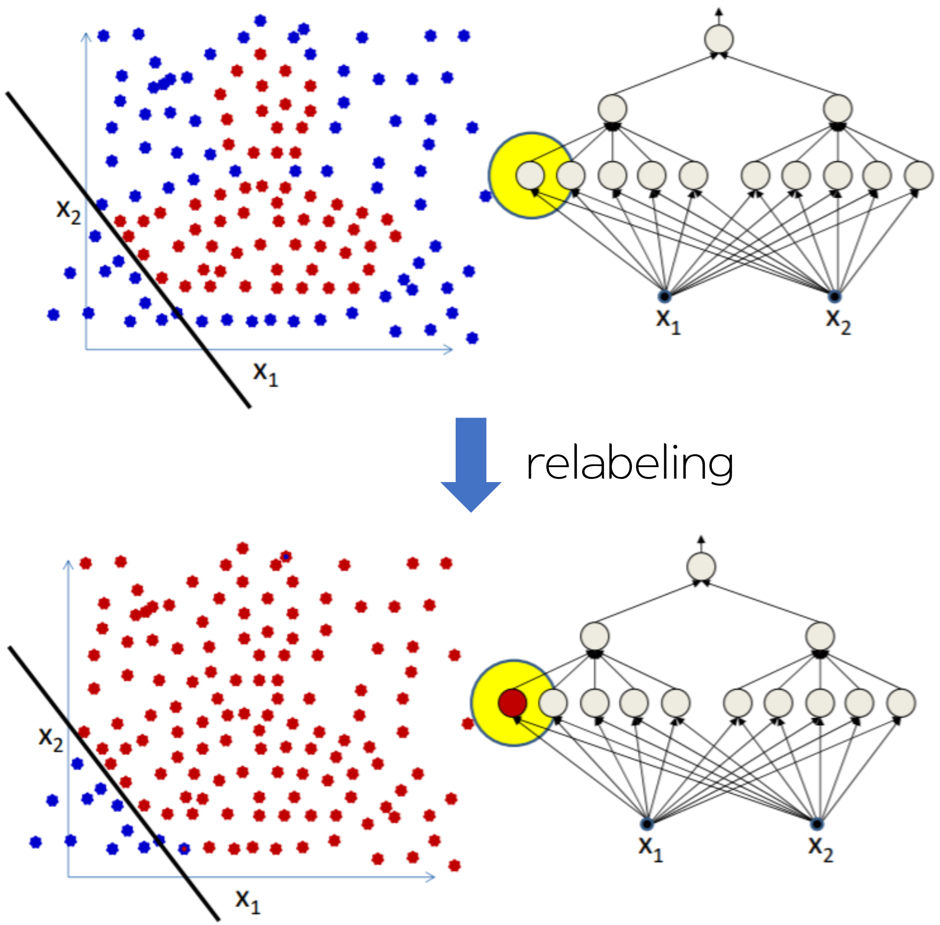
$$W = W - X_i$$



The new weight vector

02. Perceptron Algorithm

- Perceptron Algorithm for complex problems



– We have to check every possible way of relabeling for each neuron

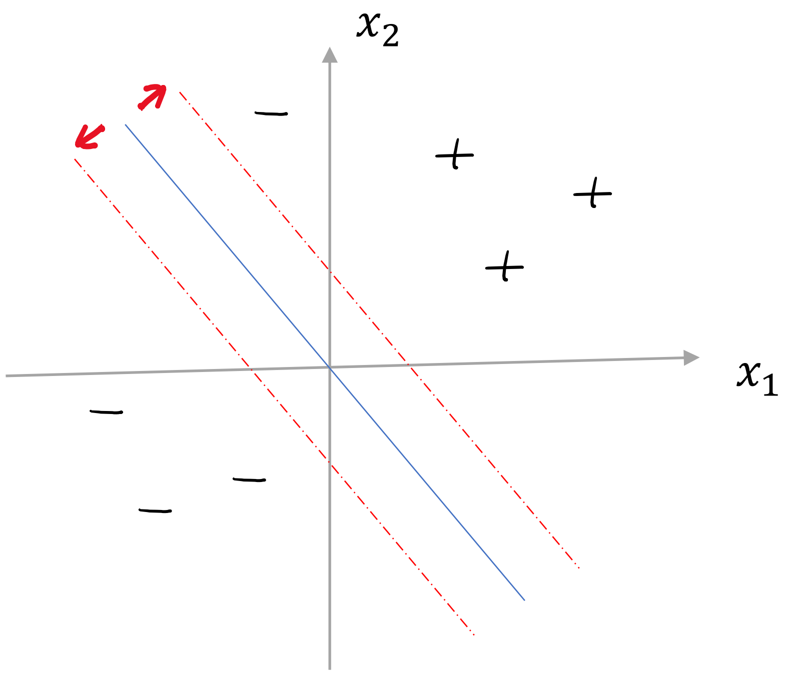
– The perceptron learning algorithm cannot directly be used to learn a MLP

=> Greedy algorithms

Contents

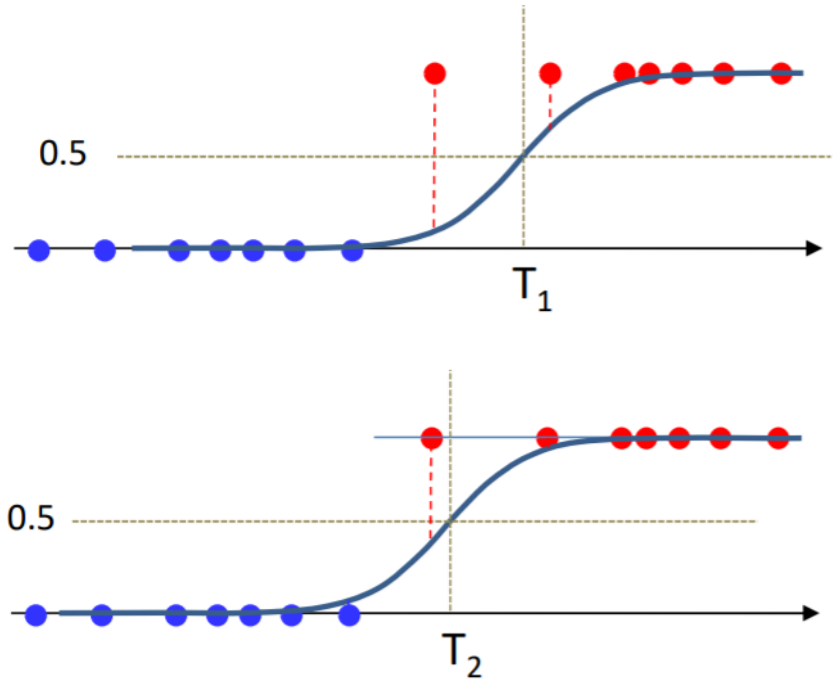
03. Perceptron with differentiable function

Step function



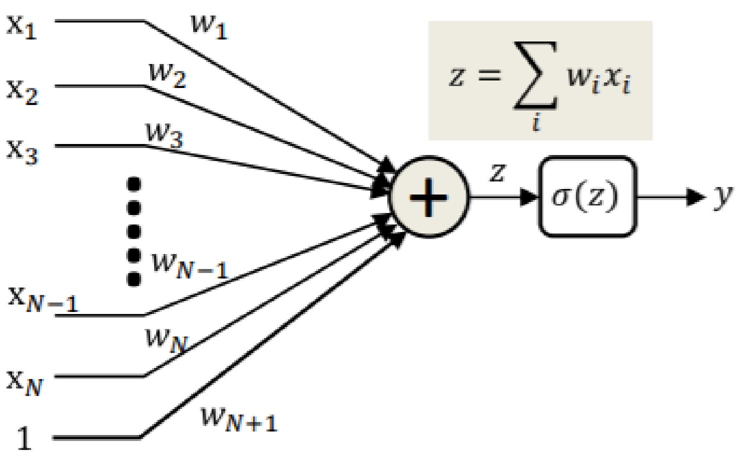
- ✓ same # of errors
- ✓ Step function doesn't tell how far the data is from the boundary

Differentiable function



⇒ Can now quantify **how much** the output differs from the desired target

03. Perceptron with differentiable function



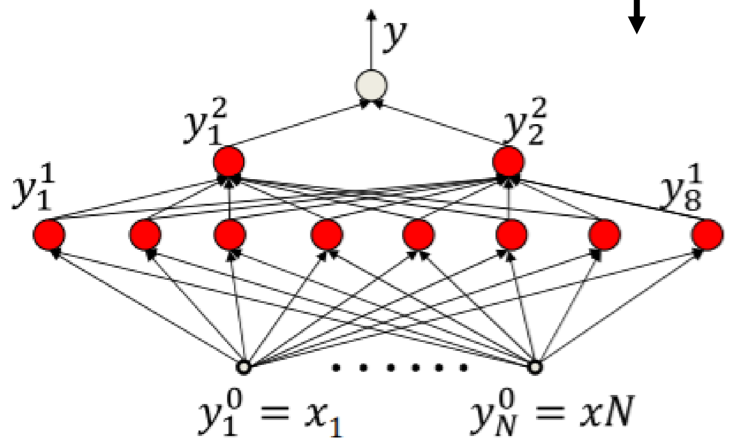
$$z = \sum_i w_i x_i$$

$$\frac{dy}{dz} = \sigma'(z)$$

$$\frac{dy}{dw_i} = \frac{dy}{dz}\frac{dz}{dw_i} = \sigma'(z)x_i$$

$$\frac{dy}{dx_i} = \frac{dy}{dz}\frac{dz}{dx_i} = \sigma'(z)w_i$$

generalization



$$y_j^k = \sigma\left(\sum_i w_{i,j}^{k-1} y_i^{k-1}\right)$$

- ✓ Using the chain rule, y is a differentiable function of every inputs x_i and weights w_i
- ✓ This means that we can compute the change in the output for small changes in either the input or the weights

Contents

04. Empirical risk minimization

- Learn through Empirical risk minimization
  - The expected divergence(or risk) is the average divergence over the entire input space

$$E\left[div\left(f(X;W),g(X)\right)\right] = \int_x div\left(f(X;W),g(X)\right)P(X)\,dX$$

  - Given a training set of input-output pairs, $(X_1, d_1), \ldots, (X_N, d_N)$
  - The empirical estimate of expected risk is **the average divergence over the samples (unbiased estimator)**

$$E\left[div\left(f(X:W),g(X)\right)\right] \approx \frac{1}{N}\sum_{i=1}^{N} div\left(f(X_i;W),d_i\right)$$

$$Loss(W) = \frac{1}{N}\sum_i div\left(f(X_i;W),d_i\right)$$

$$\widehat{W} = argmin_W Loss(W)$$

Thank you