Learning the network

11-785 Introduction to Deep Learning

– lecture 4 –

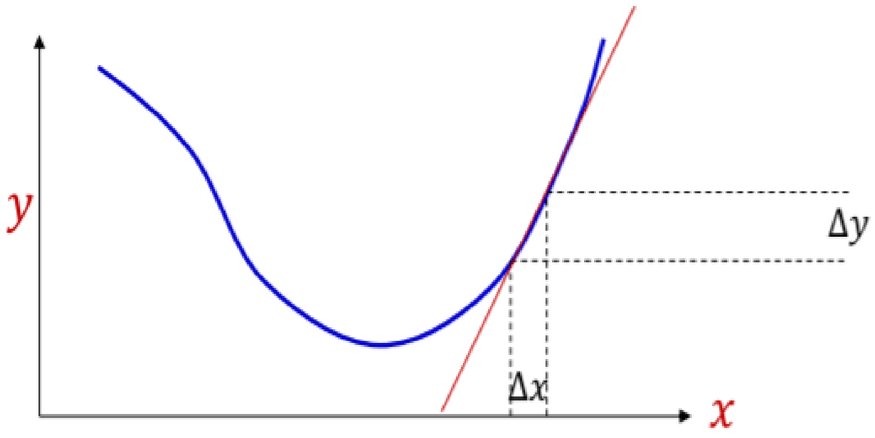
TAVE Research DL001

Heeji Won

Contents

Contents

01. Derivation



$$\Delta y = \alpha \Delta x$$

- rate of change
- how much a increment to the argument of function will increment the value of the function

- What if x is a vector? $x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}, dx = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \dots \\ \Delta x_n \end{bmatrix}$

Then, α is a row vector : $\alpha = [\alpha_1 \quad \dots \quad \alpha_n]$

$$\Delta y = \alpha \Delta x = [\alpha_1 \quad \dots \quad \alpha_n] \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \dots \\ \Delta x_n \end{bmatrix}$$

partial derivative

$$= \alpha_1 \Delta x_1 + \alpha_2 \Delta x_2 + \cdots + \alpha_n \Delta x_n$$

cf) gradient

- gradient is defined as the transpose of the derivative

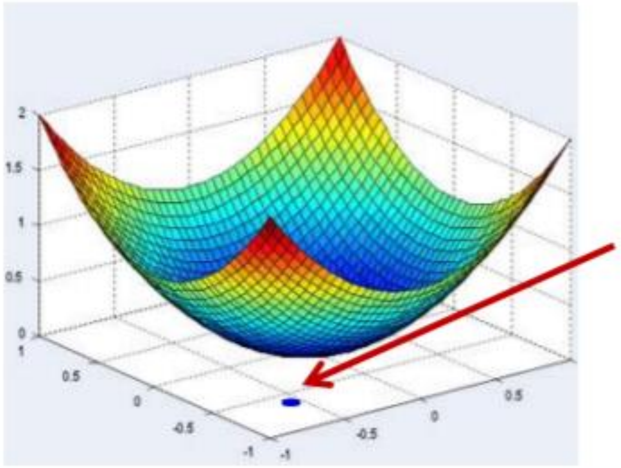
$$\Delta y = \nabla_x y \Delta x, \ \nabla_x y = \left[\frac{\partial y}{\partial x_1} \quad \dots \quad \frac{\partial y}{\partial x_D}\right]$$

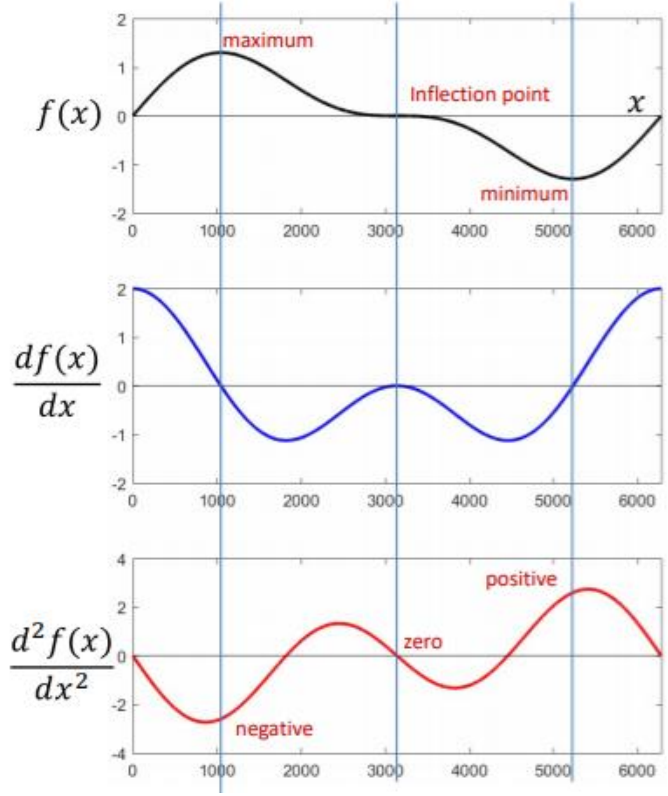
Contents

02. Optimization

General problem of optimization

: find the value of x where f(x) is minimum





- ✓ To be a minimum,

  $f'(x_0) = 0$ and $f''(x_0) > 0, \quad at\ any\ x_0$

- ✓ At a **inflection point**, the derivative is zero,
  But, second derivative is also zero

- ✓ Checking the second derivative may often not be sufficient and for **functions of multiple variable** it gets much more complex

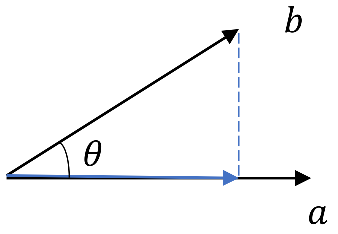
02. Optimization

- For functions of multiple variables

– The optimum point is still 'turning' point

$$dy = \nabla_x y \, dx$$
$$= \langle \nabla_x y, dx \rangle$$ inner product

✓ inner product : $a \cdot b = |a||b|cos\theta$



maximum, when $\theta = 0°$
minimum, when $\theta = 180°$

– f increase fastest when $\nabla_x y$ and dx has same direction
– **f decrease fastest when $\nabla_x y$ and dx has opposite direction**

- To be a minimum,

$\nabla_X f(X) = 0$ and

$\nabla_X^2 f(X)$ (Hessian Matrix) is **positive definite** (eigenvalues positive)

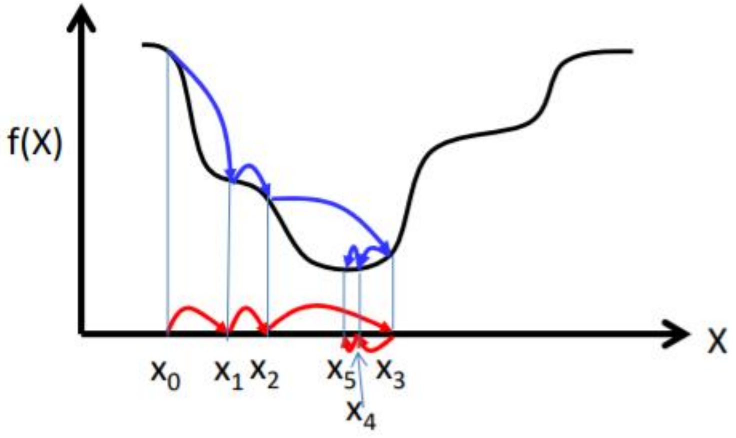
$$\nabla_x^2 f(x_1, \ldots, x_n) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

✓ Often not possible to solve $\nabla_X f(X) = 0$

=> **Iterative solution**

Contents

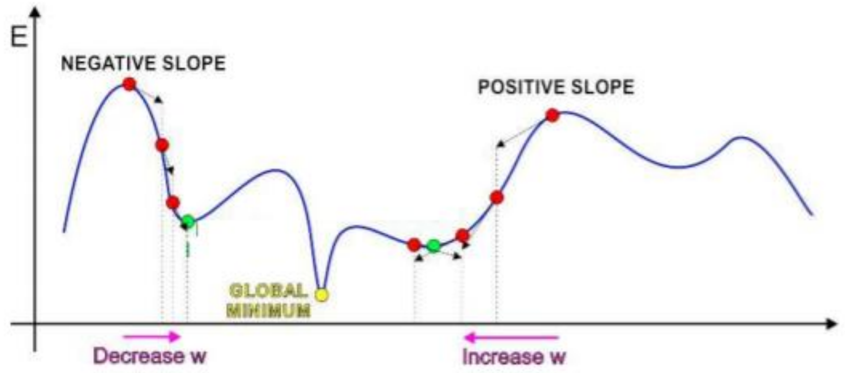
03. Iterative solutions



1. Start from an initial guess X_0 for optimal X
2. Update the guess towards a 'better' value of $f(X)$
3. Stop when $f(X)$ no longer decreases

✓ Which direction to step in
✓ How big must the steps be

- The Approach of Gradient Descent



Trivial algorithm

- Initialize x_0
- While $f'(x^k) \neq 0$

step size

$$x^{k+1} = x^k - \eta^k f'(x^k)$$

For multivariate,

$$x^{k+1} = x^k - \eta^k \nabla_x f(x^k)^T$$

03. Iterative solutions

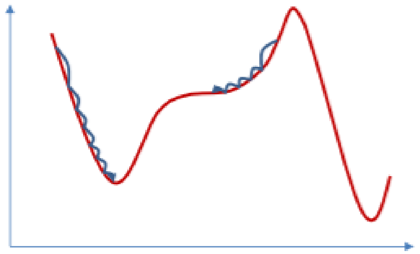
- Convergence criteria

$$|f(x^{k+1}) - f(x^k)| < \epsilon_1$$

or

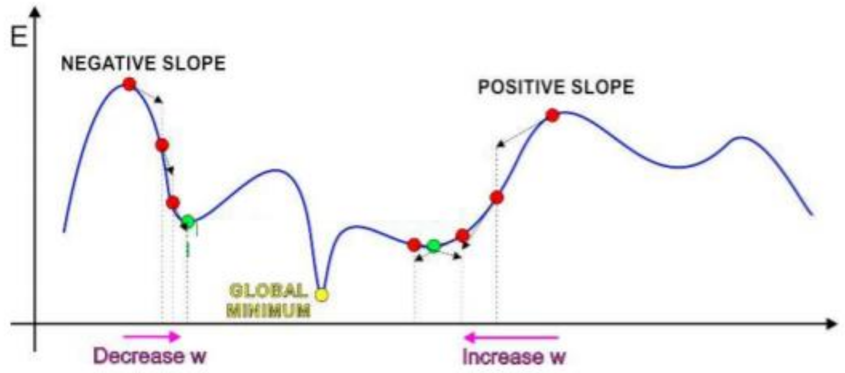
$$\|\nabla_x f(x^k)\| < \epsilon_2$$ dangerous

- For non-convex functions



✓ For non-convex functions it will find a local minimum or an inflection point

- The Approach of Gradient Descent



Trivial algorithm

- Initialize x_0
- While $f'(x^k) \neq 0$

step size

$$x^{k+1} = x^k - \eta^k f'(x^k)$$

For multivariate,

$$x^{k+1} = x^k - \eta^k \nabla_x f(x^k)^T$$

Contents

04. Things to define

- Problem Statement
  - Given a training set of input-output pairs
    $(X_1, d_1), (X_2, d_2), \dots, (X_T, d_T)$
  - Minimize the following function

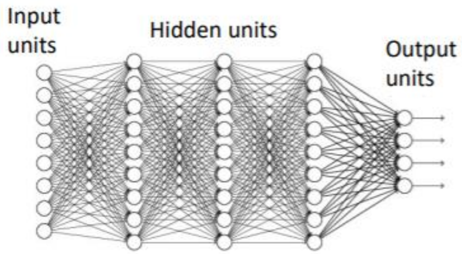
$$Loss(W) = \frac{1}{T}\sum_i div(f(X_i; W), d_i)$$

✓ What is f() and W?

✓ What is the div()?

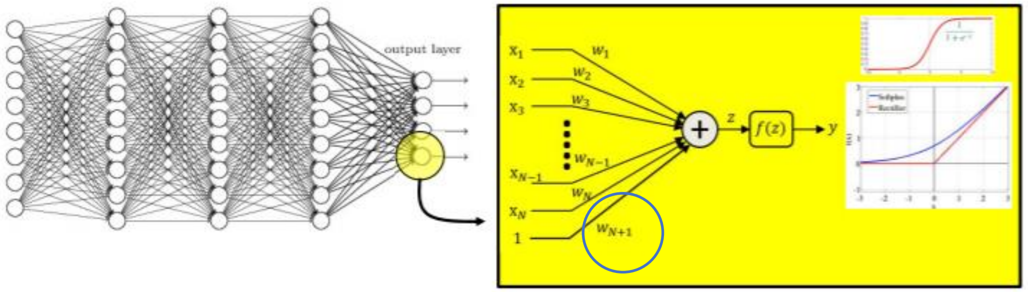
✓ What are those input-output pairs?

- What is f()?



  - Multi-layer perceptron
  - A directed network with a set of inputs and outputs

- What is W?



  - A continuous activation function applied to an affine combination of the input

$$y = f\Big(\sum_i w_i x_i + b\Big)$$

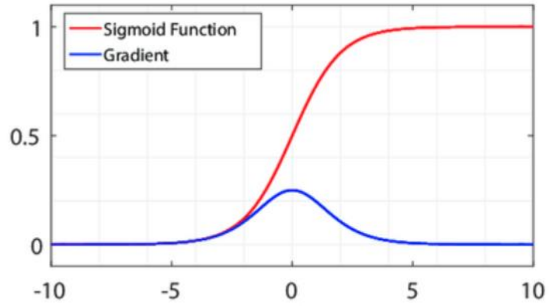
bias

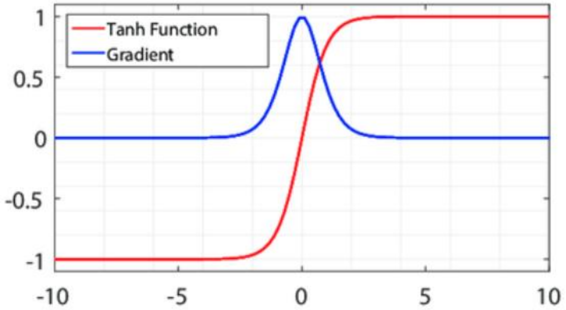
weights

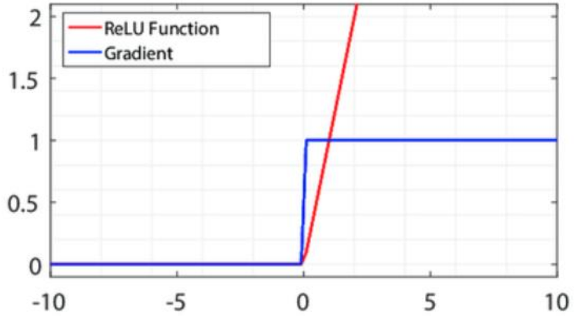
04. Things to define

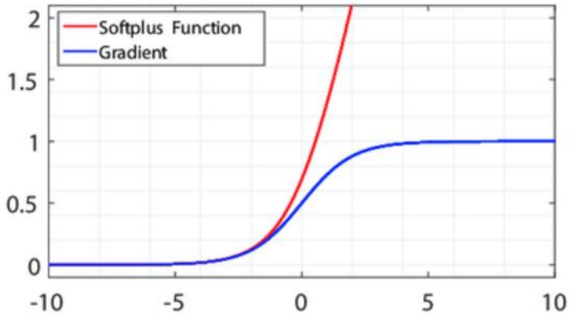
- Activations and their derivatives

sigmoid	$f(z) = \dfrac{1}{1 + \exp(-z)}$	$f'(z) = f(z)(1 - f(z))$
tanh	$f(z) = \tanh(z)$	$f'(z) = (1 - f^2(z))$
ReLU	$f(z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases}$	$f'(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$
Softplus	$f(z) = \log(1 + \exp(z))$	$f'(z) = \dfrac{1}{1 + \exp(-z)}$



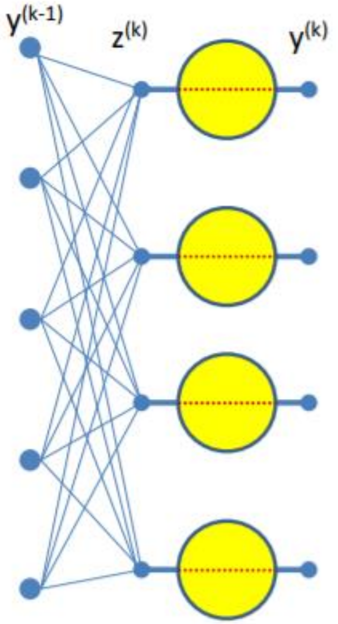


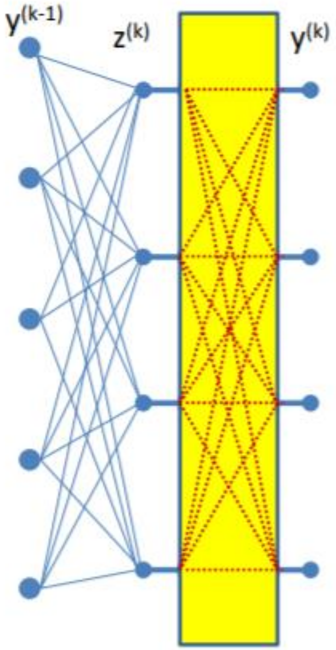




04. Things to define

- Vector Activation





< Scalar activation >

- Each z_i
- influences one y_i

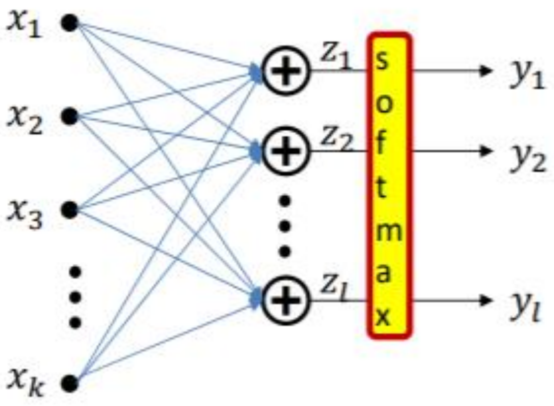
< Vector activation >

- Each z_i
- influences all, $y_1, \dots, y_M$

- Example – Softmax

$$z_i = \sum_i w_{ji} x_j + b_j$$

$$y = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



- Notation

- $\boldsymbol{w_{ij}^{(k)}}$: the weight of the connection between the i–th unit of the k–1th layer and the j–th unit of the k–th layer
- $\boldsymbol{y_i^{(k)}}$: output of the i–th perceptron of the kth layer

04. Things to define

- Input-output pairs

- Given a training set of input-output pairs $(X_1, d_1), (X_2, d_2), \dots, (X_T, d_T)$

- $X_n = [x_{n1}, x_{n2}, \dots, x_{nD}]^T$ is the nth input vector
- $d_n = [d_{n1}, d_{n2}, \dots, d_{nL}]^T$ is the nth desired vector
- $Y_n = [y_{n1}, y_{n2}, \dots, y_{nL}]^T$ is the nth vector of actual outputs of the network

- Output vector

- For binary classification, the desired output is 0 or 1
  => Actual outputs mean $\boldsymbol{P(X = 1|0)}$
- For multi-class classification, the desired output is one-hot vector
  => Actual outputs mean **probability vector**

- What is the div()?

- For real-valued output vectors, the **L_2 divergence** is popular

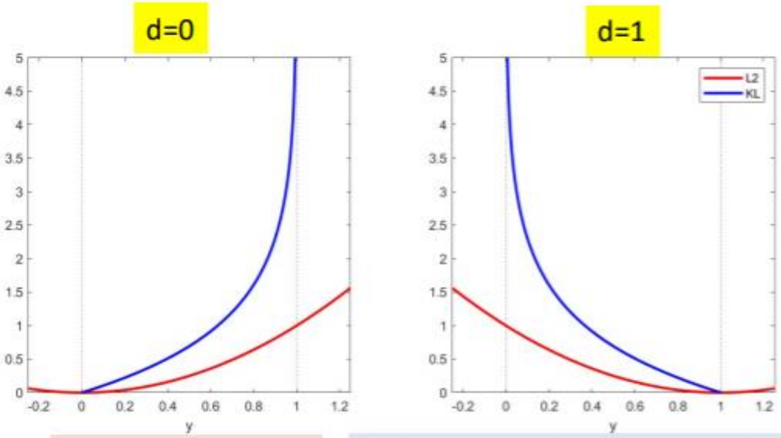
$$Div(Y, d) = \frac{1}{2}\|Y - d\|^2 = \frac{1}{2}\sum_i (y_i - d_i)^2$$

- For binary classifier, the **KL(Kullback Leibler) divergence** is popular

$$Div(Y, d) = -d\log Y - (1 - d)\log(1 - Y)$$

04. Things to define

- KL vs L2



$L2(Y,d) = (y-d)^2$ $\quad$ $KL(Y,d) = -d\log Y - (1-d)\log(1-Y)$

- Derivative of KL

$$\frac{dDiv(Y,d)}{dY} = \begin{cases} -\frac{1}{Y} & if\ d = 1 \\ \frac{1}{1-Y} & if\ d = 0 \end{cases}$$

✓ when y = d, **the derivative is not 0**

This is why we must not use $\|\nabla_x f(x^k)\| < \epsilon_2$

- For multi-class classifier,

$$Div(Y,d) = \sum_i d_i log d_i - \sum_i d_i log y_i = -log y_c$$

$$\frac{dDiv(Y,d)}{dY_i} = \begin{cases} -\frac{1}{y_c} & for\ the\ c-th\ component \\ 0 & for\ remaining\ component \end{cases}$$

$$\nabla_Y Div(Y,d) = \left[0\ 0\ \dots \frac{-1}{y_c} \dots 0\ 0\right]$$

✓ The slope is negative w.r.t y_c

✓ Indicates increasing y_c will reduce divergence

- Cross entropy $\quad Xent(Y,d) = -\sum_i d_i \log y_i$

cf) $KL(Y,d) = \sum_i d_i \log d_i - \sum_i d_i \log y_i$

✓ The W that **minimizes cross-entropy** will **minimize the KL**

Thank you