Introduction to Machine Learning – Discriminative Approach and Deep Learning

The 8th KIAS CAC Summer School
2017. 7. 28 (Wed.)

Yung-Kyun Noh
Seoul National University

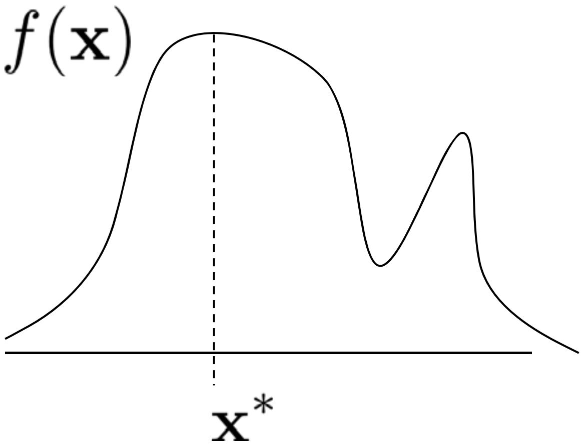
Overview

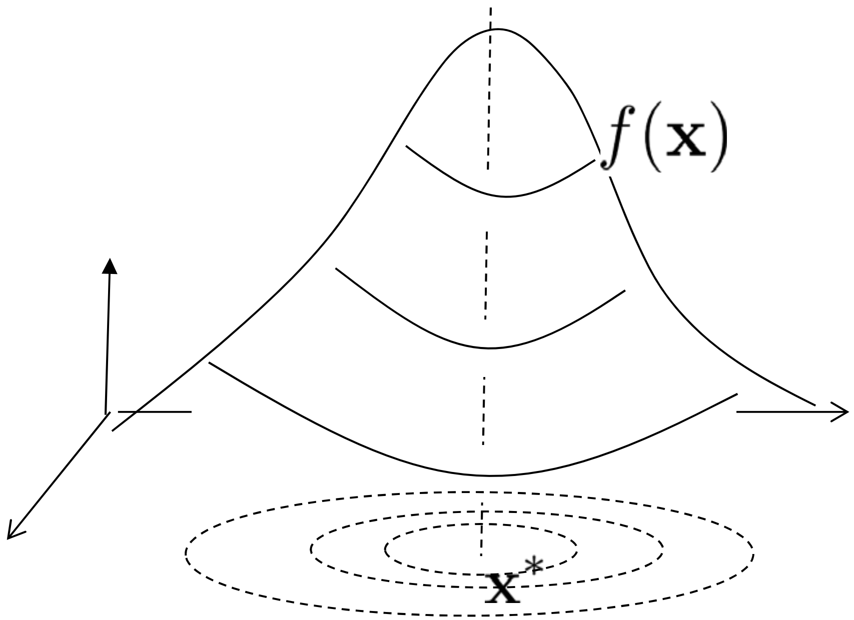
- Optimization
- Numerical methods, gradient ascent/descent
- Analytical methods
- Constrained optimization
- Convex optimization
- Backpropagation

Motivation - Optimization

- Optimization

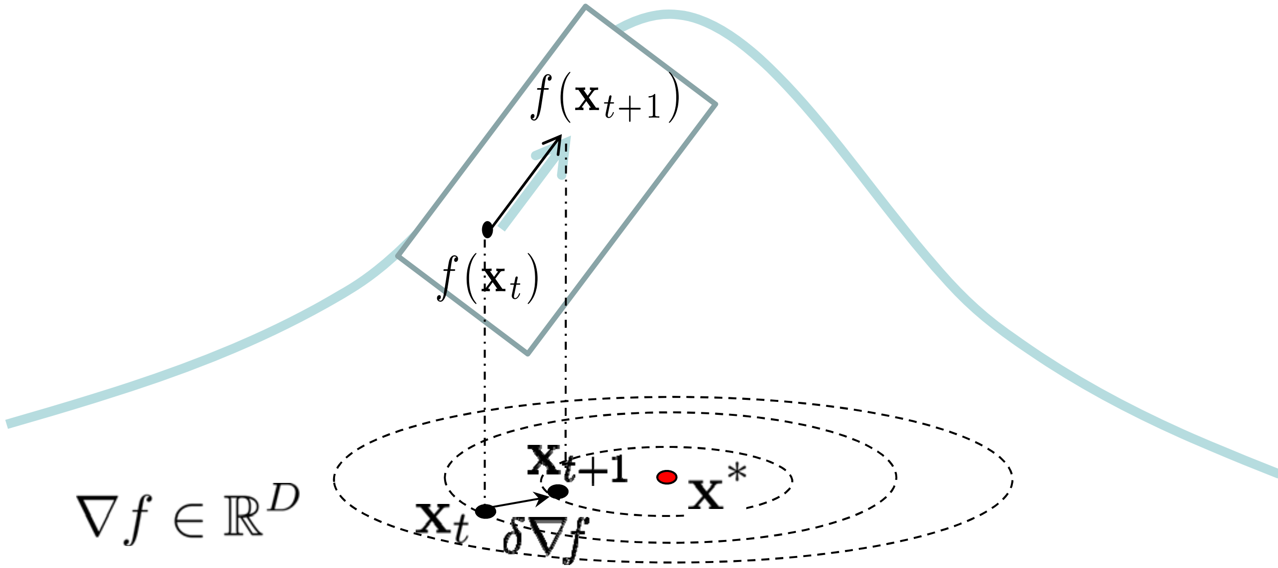
$$\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x}) \quad \mathbf{x} \in \mathbb{R}^D$$





Optimization – Using Gradient

$$f(\mathbf{x}) : \mathbb{R}^D \longrightarrow \mathbb{R}$$
$$\mathbf{x} \longmapsto f(\mathbf{x})$$

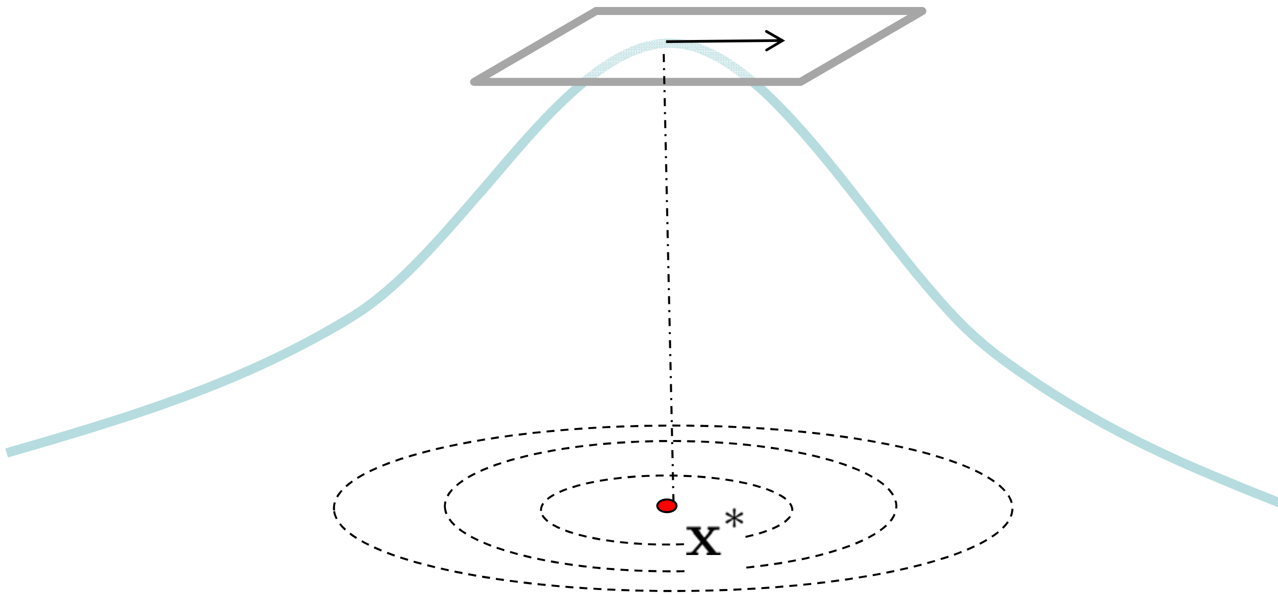


$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \delta \nabla f \Big|_{\mathbf{x}=\mathbf{x}_t}$$

Optimization – Using Gradient

- Analytic solution

$$\nabla f|_{\mathbf{x}^*} = \vec{0}$$



Differentiation w.r.t. Vector / Matrix

$$f : \mathbb{R}^D \longrightarrow \mathbb{R}$$
$$\mathbf{x} \longmapsto y$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{pmatrix} \in \mathbb{R}^D$$

$$\frac{df}{d\mathbf{x}} = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \vdots \\ \partial f / \partial x_D \end{pmatrix}$$

$$\left[\frac{df}{d\mathbf{x}} \right]_i = \frac{\partial f}{\partial x_i}$$

Differentiation w.r.t. Vector / Matrix

$$f : \mathbb{R}^{D\times D} \longrightarrow \mathbb{R}$$
$$A \longmapsto y$$

$$A = \begin{pmatrix} A_{1,1} & \cdots & A_{1,D} \\ A_{2,1} & \cdots & A_{2,D} \\ \vdots & & \\ A_{D,1} & \cdots & A_{D,D} \end{pmatrix}$$
$$\in \mathbb{R}^{D\times D}$$

$$\frac{df}{dA} = \begin{pmatrix} \frac{\partial f}{\partial A_{1,1}} & \cdots & \frac{\partial f}{\partial A_{1,D}} \\ \vdots & & \\ \frac{\partial f}{\partial A_{D,1}} & \cdots & \frac{\partial f}{\partial A_{D,D}} \end{pmatrix}$$

$$\left[\frac{df}{dA}\right]_{i,j} = \frac{\partial f}{\partial A_{i,j}}$$

Differentiation w.r.t. Vector / Matrix

- Ex)

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$
$$= \sum w_i x_i$$

$$\left(\frac{df}{d\mathbf{x}}\right)_i = w_i \qquad \frac{df}{d\mathbf{x}} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix} = \mathbf{w} \quad \in \mathbb{R}^D$$

Differentiation w.r.t. Vector / Matrix

- Ex)

$$f(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}_i - b)^2$$

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

$$= \sum_{i=1}^{N} (\sum_{j=1}^{D} w_j x_{ij} - b)^2$$

$$\frac{\partial f}{\partial w_k} = \sum_{i=1}^{N} 2 \left(\sum_{j=1}^{D} w_j x_{ij} - b \right) \cdot x_{ik}$$

$$\frac{\partial f}{\partial \mathbf{w}} = 2 \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}_i - b) \cdot \mathbf{x}_i \quad \in \mathbb{R}^D$$

Differentiation w.r.t. Vector / Matrix

- Ex)

$$f(B) = tr[ABC] = \sum_{ijk} A_{ij} B_{jk} C_{ki}$$

$$ABC = \begin{array}{c} \\ i \\ \\ \\ \end{array} \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,D} \\ A_{2,1} & \cdots & & A_{2,D} \\ \cdots & & & \\ A_{D,1} & \cdots & & A_{D,D} \end{pmatrix} \begin{pmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,D} \\ B_{2,1} & \cdots & & B_{2,D} \\ \cdots & & & \\ B_{D,1} & \cdots & & B_{D,D} \end{pmatrix} \begin{array}{c} \begin{pmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,D} \\ C_{2,1} & \cdots & & C_{2,D} \\ \cdots & & & \\ C_{D,1} & \cdots & & C_{D,D} \end{pmatrix} \\ \begin{array}{cccc} \phantom{C_{1,1}} & j & & \phantom{C_{1,D}} \end{array} \end{array}$$

$$[ABC]_{ij} = \sum_{lm} A_{il} B_{lm} C_{mj}$$

$$tr[ABC] = \sum_{i} [ABC]_{ii} = \sum_{ijk} A_{ij} B_{jk} C_{ki}$$

Differentiation w.r.t. Vector / Matrix

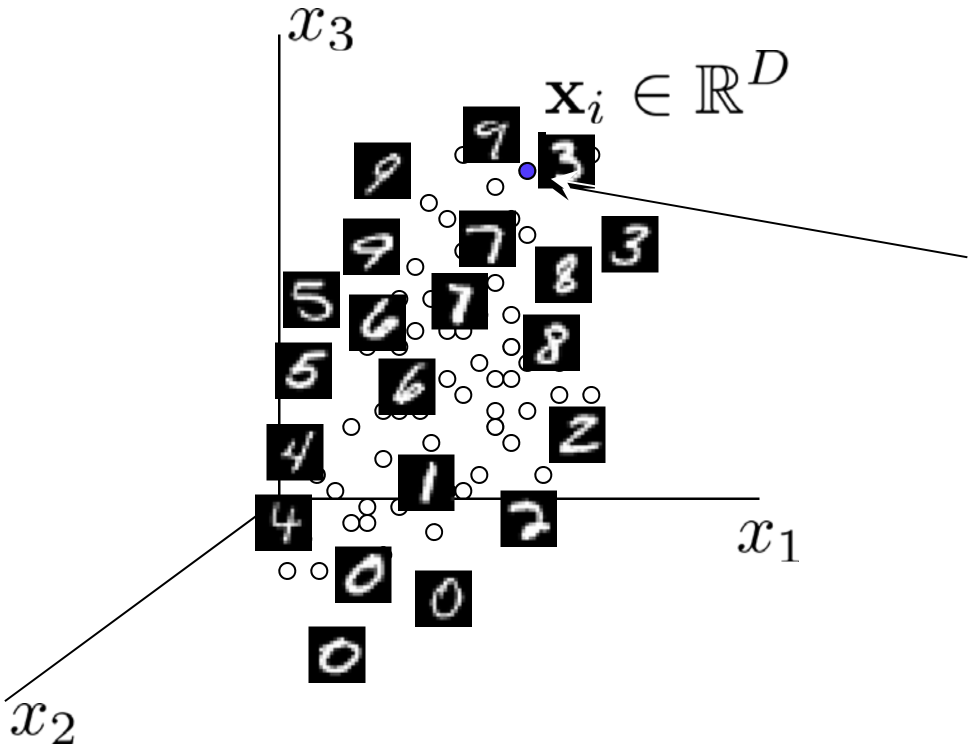
- Ex)

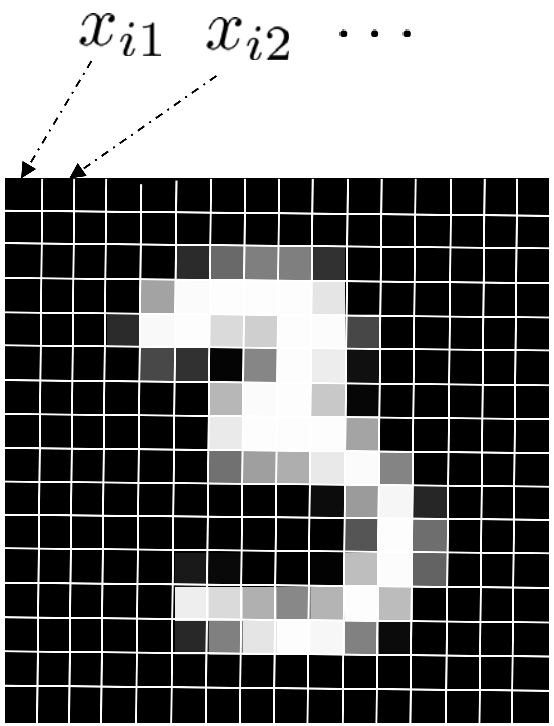
$$f(B) = tr[ABC] = \sum_{ijk} A_{ij} B_{jk} C_{ki}$$

$$\left[\frac{df(B)}{dB}\right]_{lm} = \sum_i A_{il} C_{mi} = \sum_i A_{li}^\top C_{im}^\top$$

$$\frac{df}{dB} = A^\top C^\top \quad \in \ \mathbb{R}^{D \times D}$$

Data Space

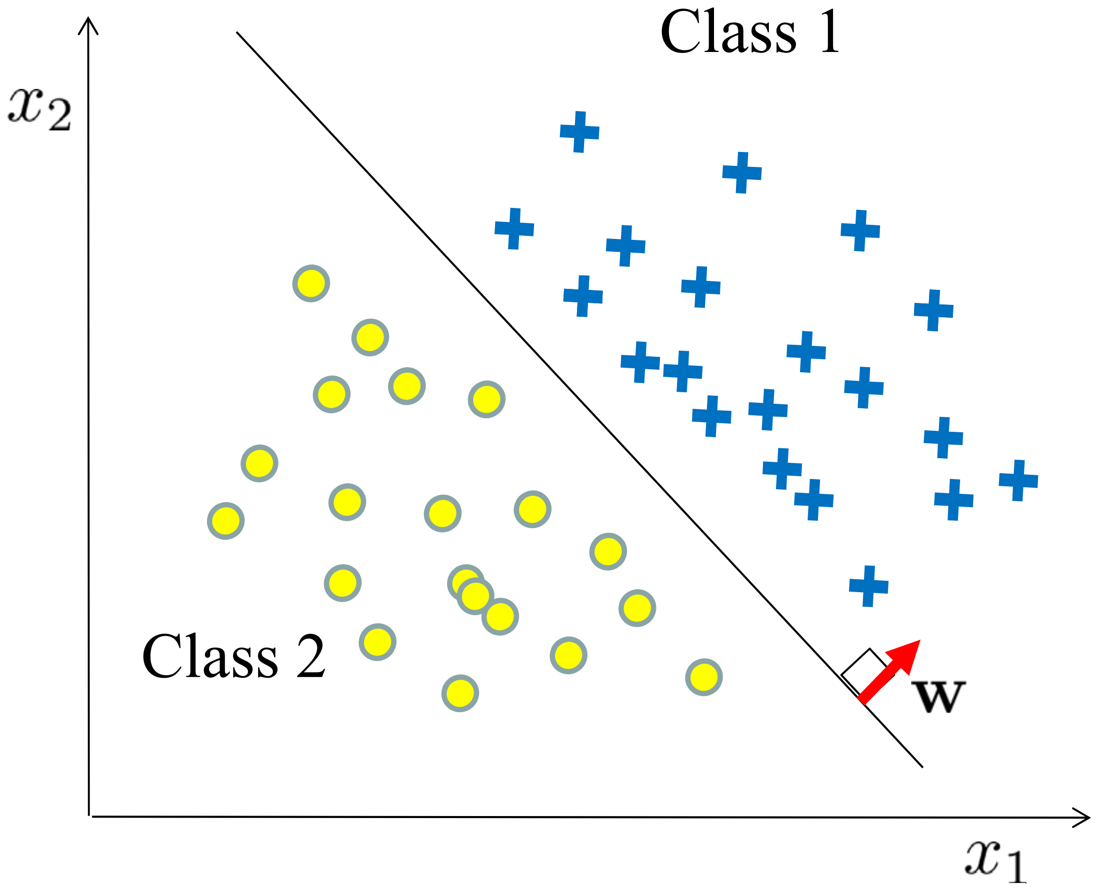




$$\mathbf{x}_i = [x_{i1}, \ldots, x_{iD}]$$
$$= [1, 2, 5, 12, 10, \ldots]$$

- Each datum is one point in a data space

Linear Classifier



Given

$$\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$$

Learn a prediction function from data

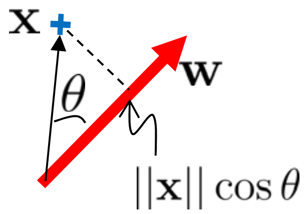
$$y = f(\mathbf{x}) = f(\mathbf{x}; \mathbf{w})$$

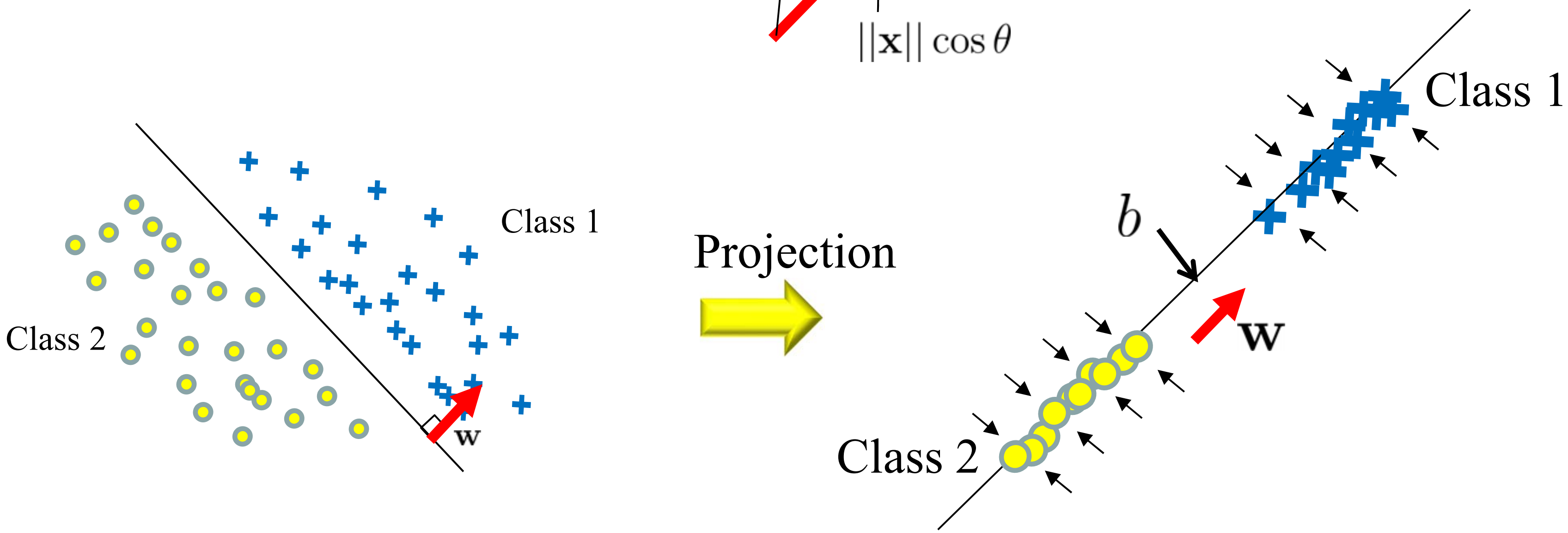
Linear Classifier

Introduce a vector $\mathbf{w}$

$$\mathbf{w}^\top \mathbf{x} \begin{cases} \geq b \rightarrow f(\mathbf{x}) = \text{Class 1} \\ < b \rightarrow f(\mathbf{x}) = \text{Class 2} \end{cases}$$

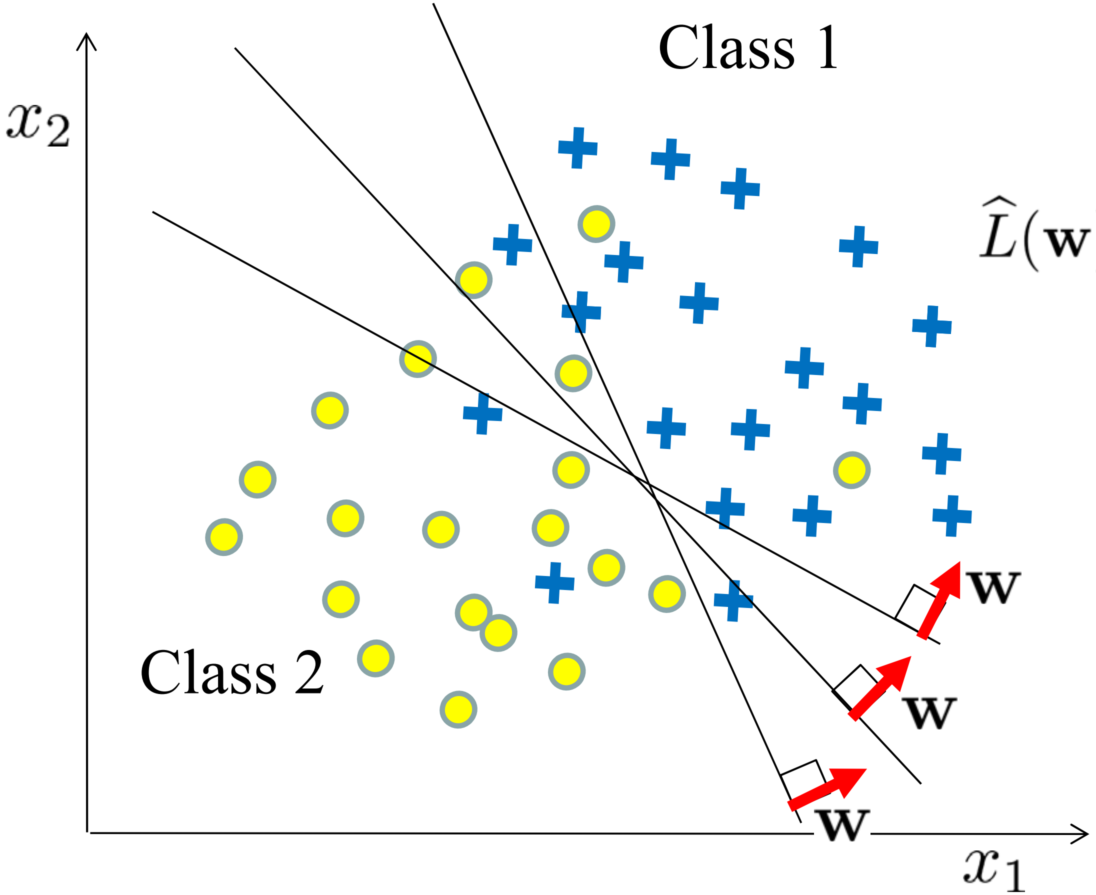
$$\mathbf{w}^\top \mathbf{x} = ||\mathbf{w}||||\mathbf{x}|| \cos\theta$$





Linear Classifier

- Find $\mathbf{w} \in \mathbb{R}^D$ which classifies training data correctly as many as possible.



$$\widehat{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} ||f(\mathbf{x}_i;\mathbf{w}) - y_i||^2$$

$$\mathbf{x}_i \in \mathbb{R}^D$$

$$y_i, f(\mathbf{x}_i;\mathbf{w}) \in \{1,2\}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^D} \widehat{L}(\mathbf{w})$$

Linear Classifier

- A naïve one $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$

$$\frac{\partial \widehat{L}}{\partial \mathbf{w}} = \frac{1}{2} \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^{N} ||f(\mathbf{x}_i; \mathbf{w}) - y_i||^2$$

$$= \sum_{i=1}^{N} (f(\mathbf{x}_i; \mathbf{w}) - y_i) \frac{\partial f(\mathbf{x}_i; \mathbf{w})}{\partial \mathbf{w}}$$

$$= \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}_i - y_i) \mathbf{x}_i = 0$$

has a closed-form solution: $\mathbf{w} = ?$

Linear Classifier

$$\sum_{i=1}^{N}(\mathbf{w}^\top \mathbf{x}_i - y_i)\mathbf{x}_i = 0$$

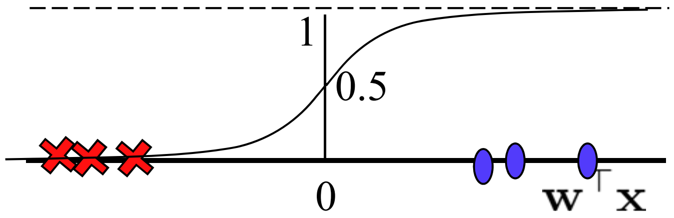
$$X = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_N \\ | & & | \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

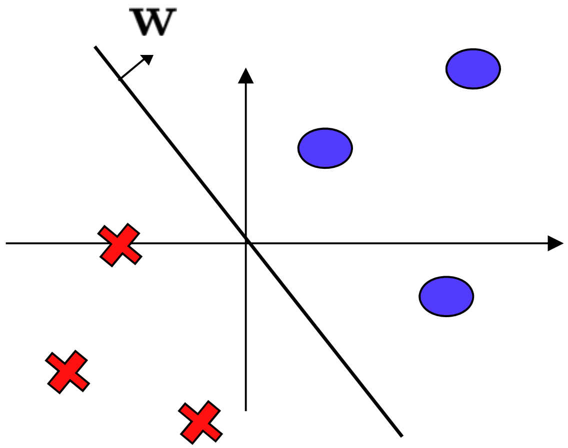
$$\sum_{i=1}^{N}\mathbf{w}^\top \mathbf{x}_i \mathbf{x}_i - \sum_{i=1}^{N} y_i \mathbf{x}_i = 0 \longrightarrow \sum_{i=1}^{N}\mathbf{x}_i \mathbf{x}_i^\top \mathbf{w} - \sum_{i=1}^{N} y_i \mathbf{x}_i = 0$$

$$XX^\top \mathbf{w} - X\mathbf{y} = 0 \longrightarrow \mathbf{w} = (XX^\top)^{-1}Xy$$

Logistic Regression

$$f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$





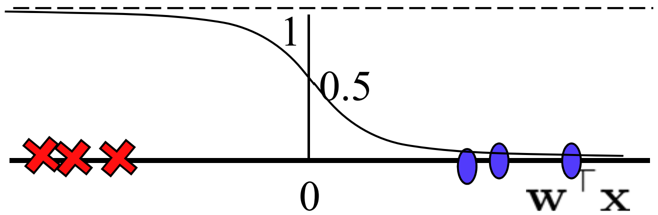
$$\widehat{L}(\mathbf{w}) = \prod_{i=1}^{N} f(\mathbf{x}_i)^{y_i} (1 - f(\mathbf{x}_i))^{1-y_i}$$

Probability for $y_i = 1$ Probability for $y_i = 0$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \widehat{L}(\mathbf{w}) = \arg\max_{\mathbf{w}} \ln \widehat{L}(\mathbf{w})$$

Logistic Regression

$$1 - f(\mathbf{x}) = 1 - \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})} = \frac{\exp(-\mathbf{w}^\top \mathbf{x})}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$



$$\widehat{L}(\mathbf{w}) = \prod_{i=1}^{N} f(\mathbf{x}_i)^{y_i} (1 - f(\mathbf{x}_i))^{1-y_i}$$

Probability for $y_i = 1$ Probability for $y_i = 0$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \widehat{L}(\mathbf{w}) = \arg\max_{\mathbf{w}} \ln \widehat{L}(\mathbf{w})$$

Logistic Regression

$$\ln\left(f(\mathbf{x}_i)^{y_i}(1-f(\mathbf{x}_i)^{1-y_i}\right)$$
$$= y_i \ln f(\mathbf{x}_i) + (1-y_i)\ln(1-f(\mathbf{x}_i)) \qquad \cdots (*)$$

Consider

$$f(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})} \longrightarrow \frac{df}{d\mathbf{w}} = \frac{\exp(-\mathbf{w}^\top \mathbf{x})\ \mathbf{x}}{(1+\exp(-\mathbf{w}^\top \mathbf{x}))^2} = f(1-f)\mathbf{x}$$

$$\frac{d(*)}{d\mathbf{w}} = \frac{y_i}{f} f(1-f)\mathbf{x}_i - \frac{1-y_i}{1-f} f(1-f)\mathbf{x}_i$$
$$= y_i(1-f)\mathbf{x}_i - (1-y_i) f \mathbf{x}_i$$
$$= (y_i - f)\mathbf{x}_i$$

Logistic Regression

- Update rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))\ \mathbf{x}_i$$

small constant

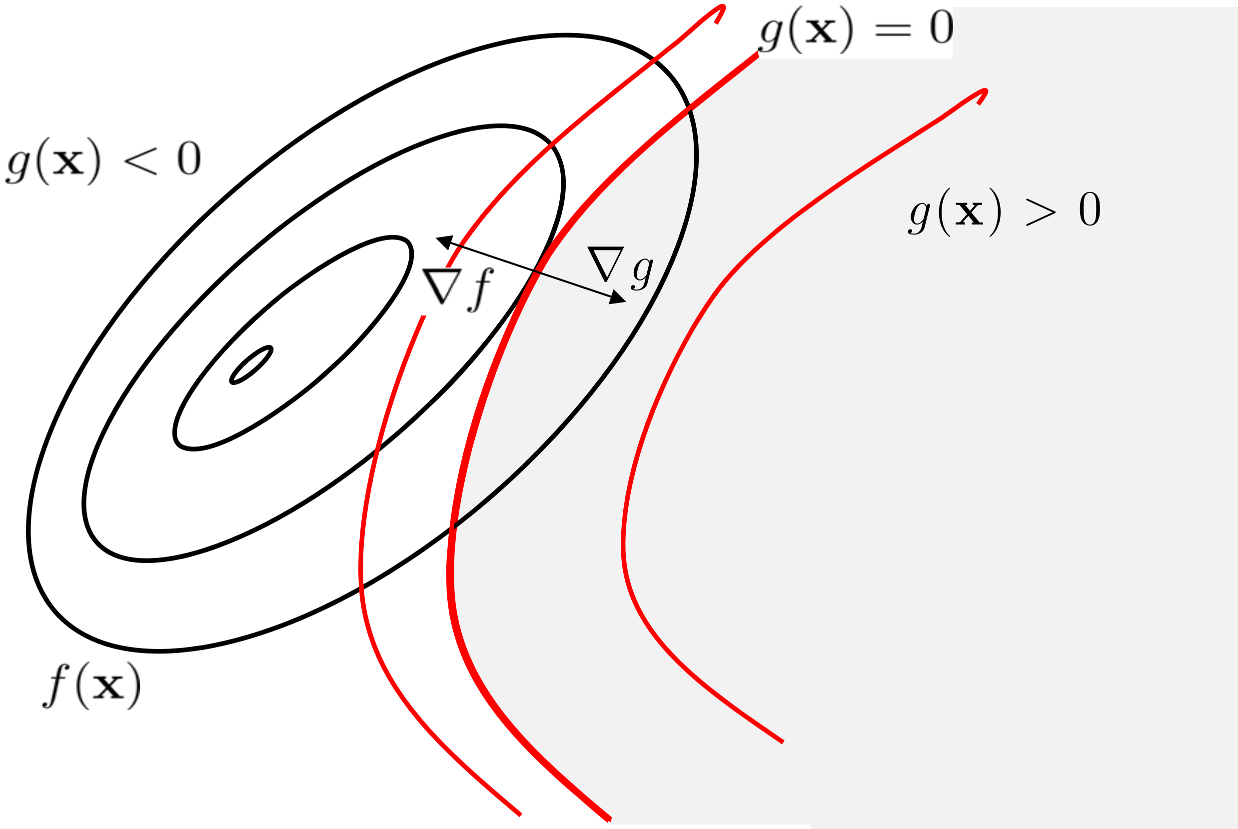
  – Perform this operation until converge

- Online algorithm

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda (y_i - f(\mathbf{x}_i))\ \mathbf{x}_i$$

Optimization with Constraints

$$\max f(\mathbf{x}) \quad s.t. \quad g(\mathbf{x}) \geq 0$$



$\nabla f = -\lambda \nabla g$ for a positive constant λ

Karush–Kuhn–Tucker condition for Optimal Solution

- For $\max f(\mathbf{x}) \quad s.t. \quad g(\mathbf{x}) \geq 0$

$$L(\mathbf{x}) = f(\mathbf{x}) + \lambda g(\mathbf{x}) \Longrightarrow \left.\frac{dL(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} = 0$$

- For $\min f(\mathbf{x}) \quad s.t. \quad g(\mathbf{x}) \geq 0$

$$L(\mathbf{x}) = f(\mathbf{x}) - \lambda g(\mathbf{x}) \Longrightarrow \left.\frac{dL(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} = 0$$

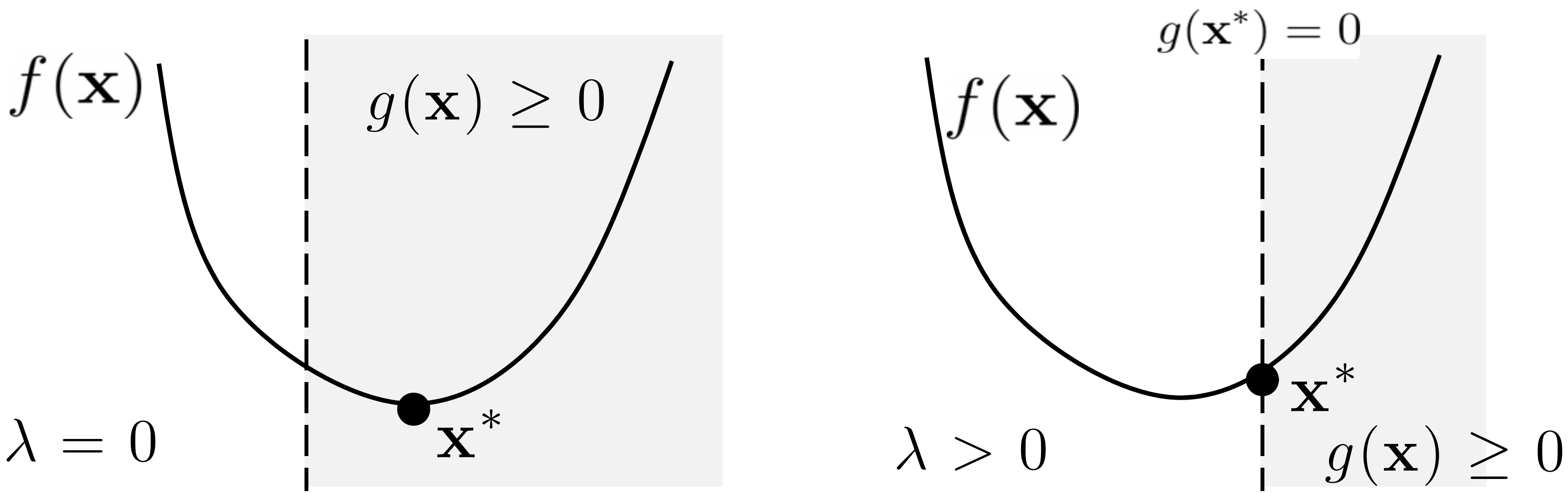
KKT Condition for Inequality Constraint

- Karush–Kuhn–Tucker condition
  - The solution satisfies either

$$\lambda = 0 \quad \& \quad g(\mathbf{x}^*) \geq 0$$

or

$$\lambda > 0 \quad \& \quad g(\mathbf{x}^*) = 0$$



KKT Condition for Inequality Constraint

- To find analytic solution, we have to search for the solution that satisfies KKT condition – 2^k cases in the worst case for k constraints.

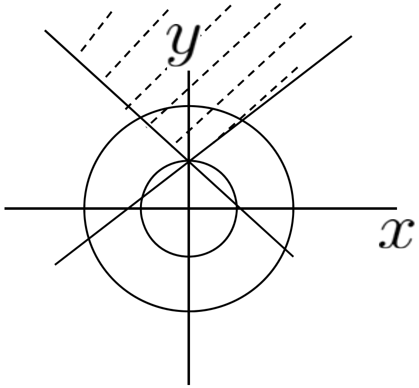
$$g_1(\mathbf{x}) \geq 0, \ldots, g_k(\mathbf{x}) \geq 0$$

$$L(\mathbf{x}) = f(\mathbf{x}) + \lambda_1 g_1(\mathbf{x}) + \ldots + \lambda_k g_k(\mathbf{x})$$

$$\underbrace{\begin{matrix} \lambda_1 = 0, g_1(\mathbf{x}^*) \geq 0 \\ \lambda_2 = 0, g_2(\mathbf{x}^*) \geq 0 \\ \vdots \\ \lambda_k = 0, g_k(\mathbf{x}^*) \geq 0 \end{matrix} \quad \begin{matrix} \lambda_1 > 0, g_1(\mathbf{x}^*) = 0 \\ \lambda_2 = 0, g_2(\mathbf{x}^*) \geq 0 \\ \vdots \\ \lambda_k = 0, g_k(\mathbf{x}^*) \geq 0 \end{matrix} \quad \begin{matrix} \lambda_1 > 0, g_1(\mathbf{x}^*) = 0 \\ \lambda_2 > 0, g_2(\mathbf{x}^*) = 0 \\ \vdots \\ \lambda_k = 0, g_k(\mathbf{x}^*) \geq 0 \end{matrix} \quad \cdots}_{2^k \text{ cases}}$$

Optimization with Constraints

- Ex)



$$f(\mathbf{x}) = x^2 + y^2$$
$$g_1(\mathbf{x}) = y - x - 1 \geq 0$$
$$g_2(\mathbf{x}) = y + x - 1 \geq 0$$

$$L = x^2 + y^2 - \lambda_1(y - x - 1) - \lambda_2(y + x - 1)$$

$$\frac{dL}{dx} = 2x + \lambda_1 - \lambda_2 = 0$$

$$\frac{dL}{dy} = 2y - \lambda_1 - \lambda_2 = 0$$

KKT condition (visit all cases)

(1) $\lambda_1 = \lambda_2 = 0$ (2) $\lambda_1 = 0, y + x - 1 = 0$

(3) $y - x - 1 = 0, \lambda_2 = 0$ (4) $y - x - 1 = 0, y + x - 1 = 0$

Optimization with Constraints

$$f(\mathbf{x}) = x^2 + y^2 \quad g_1(\mathbf{x}) = y - x - 1 \geq 0$$
$$g_2(\mathbf{x}) = y + x - 1 \geq 0$$

$$\frac{dL}{dx} = 2x + \lambda_1 - \lambda_2 = 0 \qquad \frac{dL}{dy} = 2y - \lambda_1 - \lambda_2 = 0$$

(1) $\lambda_1 = \lambda_2 = 0$

$$x = y = 0$$ *Not feasible*

(2) $\lambda_1 = 0, y + x - 1 = 0$

$$\left.\begin{aligned} 2x - \lambda_2 &= 0 \\ 2y - \lambda_2 &= 0 \end{aligned}\right] \longrightarrow \quad x - y = 0$$

$$x = y = \frac{1}{2}$$ *Not feasible*

Optimization with Constraints

(3) $y - x - 1 = 0, \lambda_2 = 0$

$$\left.\begin{aligned} 2x + \lambda_1 &= 0 \\ 2y - \lambda_1 &= 0 \end{aligned}\right] \longrightarrow \quad x + y = 0$$

$$x = -\frac{1}{2}, y = \frac{1}{2}$$

Not feasible

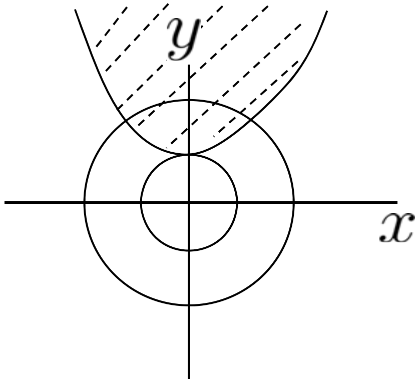
(4) $y - x - 1 = 0, y + x - 1 = 0$

$$x = 0, y = 1$$

Feasible

Optimization with Constraints

- Ex)



$$f(\mathbf{x}) = x^2 + y^2$$
$$g(\mathbf{x}) = y - x^2 - 1 \geq 0$$

$$L = x^2 + y^2 - \lambda(y - x^2 - 1)$$

$$\left.\begin{aligned} \frac{dL}{dx} &= 2x + 2\lambda x = 0 \quad \rightarrow \quad (1+\lambda)x = 0 \\ \frac{dL}{dy} &= 2y - \lambda = 0 \quad \rightarrow \quad \lambda = 2y \end{aligned}\right\} \quad (1+2y)x = 0 \quad \cdots (1)$$

We assume that the solution is at $g(\mathbf{x}) = 0$

From (1), $\quad (2x^2 + 3)x = 0 \quad \rightarrow \quad y = 1$

Optimization with Constraints

- Continued…
  - Candidate solution: (0, 1)
  - The candidate solution $(x_1, x_2) = (0, 1)$ satisfies $g(\mathbf{x}) \geq 0$
- Solution:

$$\mathbf{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Convex Optimization

- If the objective function is convex & if the feasible region is also convex.
  - Objective function is convex

$$\alpha_1 + \alpha_2 = 1, \quad \alpha_1, \alpha_2 \geq 0$$

$$\frac{\alpha_1 f(\mathbf{x}_1) + \alpha_2 f(\mathbf{x}_2)}{\alpha_1 + \alpha_2} \geq f(\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2)$$

  - Domain is convex

$$\mathbf{x}_1, \mathbf{x}_2 \in Domain$$

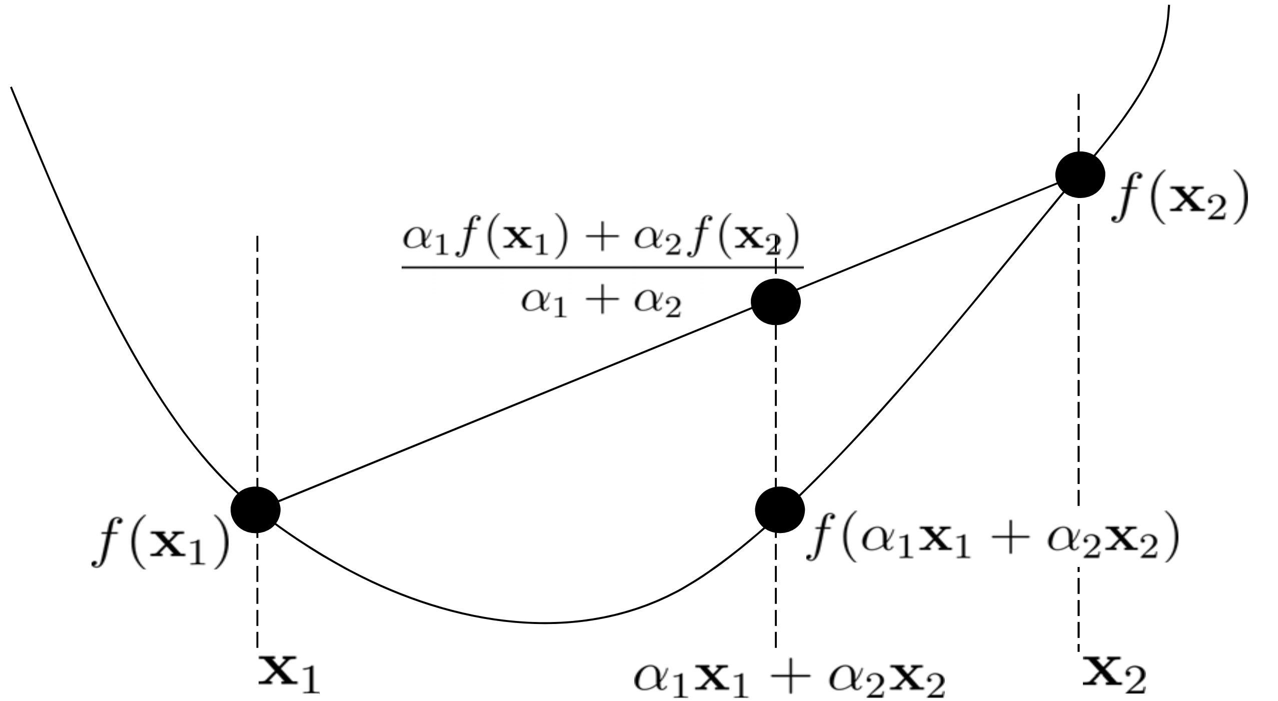
$$\alpha_1 + \alpha_2 = 1, \quad \alpha_1, \alpha_2 \geq 0$$

$$\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 \in Domain$$

Convex Optimization

- Function convexity

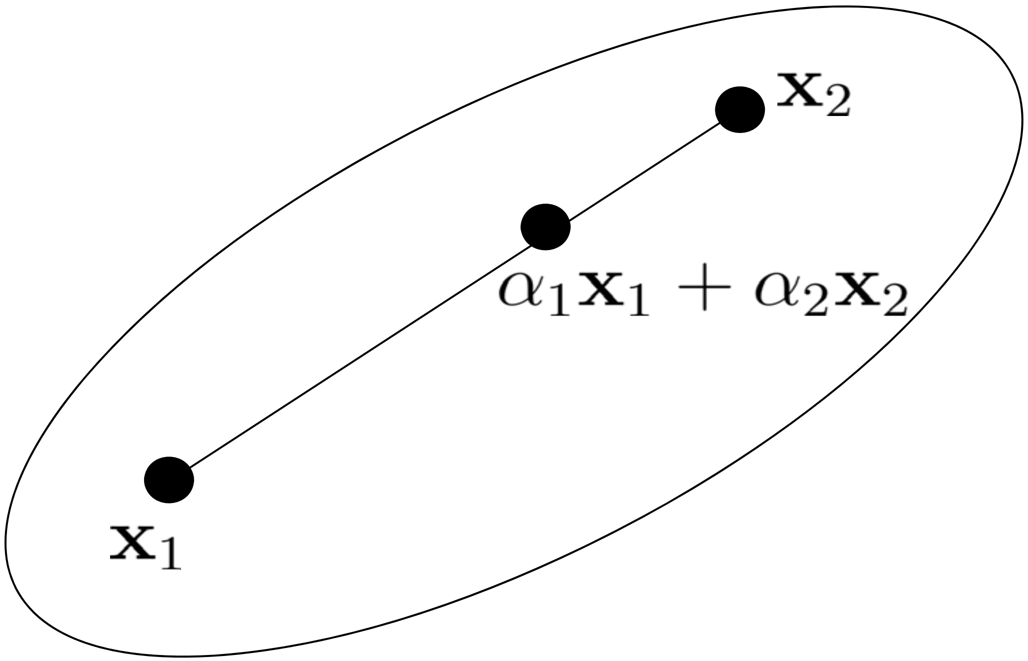
$$\frac{\alpha_1 f(\mathbf{x}_1) + \alpha_2 f(\mathbf{x}_2)}{\alpha_1 + \alpha_2} \geq f(\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2)$$



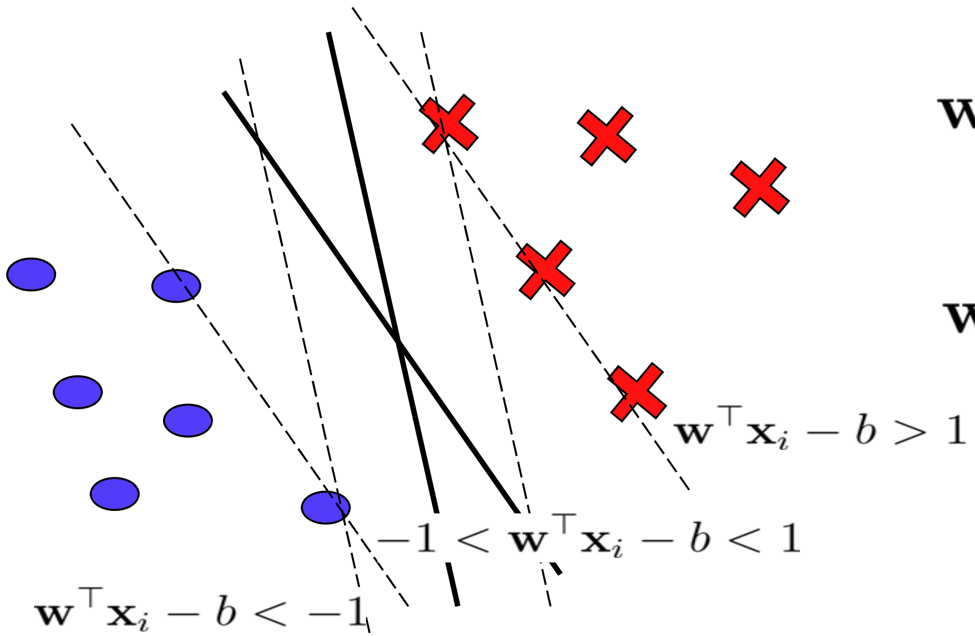
Convex Optimization

- Domain convexity

$$\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 \in Domain$$



Large Margin Classifier



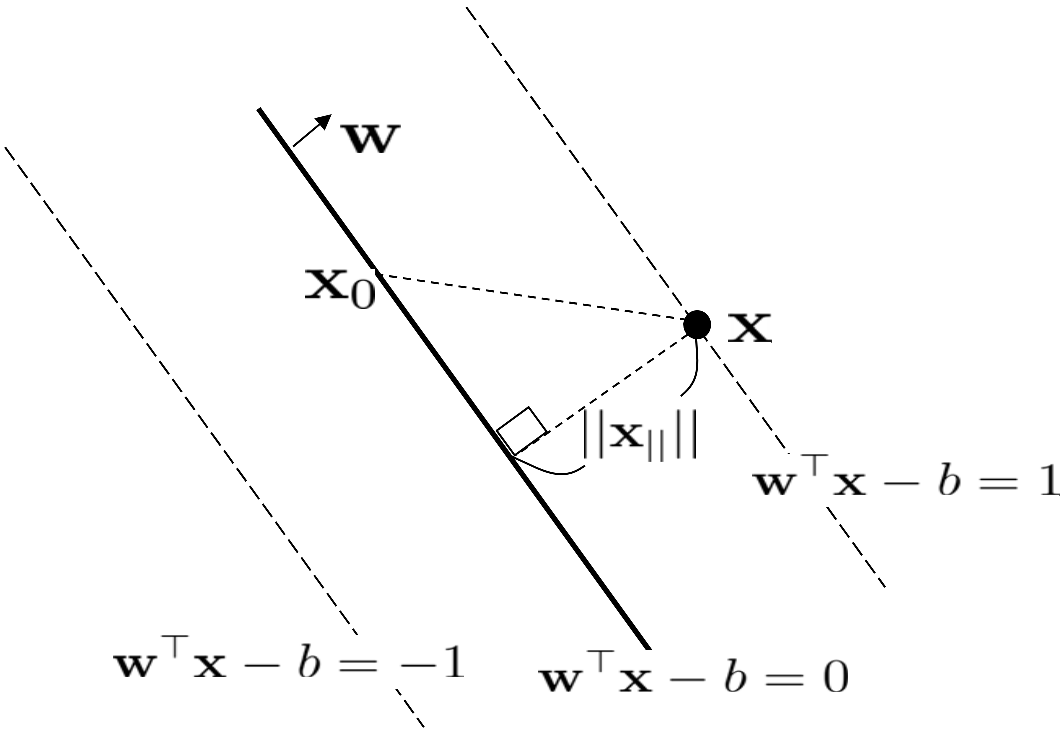
$$\mathbf{w}^\top \mathbf{x}_i - b > 0 \quad \rightarrow f(\mathbf{x}_i) = 1$$

$$\mathbf{w}^\top \mathbf{x}_i - b < 0 \quad \rightarrow f(\mathbf{x}_i) = -1$$

- We let all data outside margin

$$y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 \qquad y \in \{-1, 1\}$$

Large Margin Classifier



$$\mathbf{w}^\top(\mathbf{x} - \mathbf{x}_0) = ||\mathbf{w}||||\mathbf{x}_{||}|| = 1$$

Margin: $||\mathbf{x}_{||}|| = \dfrac{1}{||\mathbf{w}||}$

Large Margin Classifier

- Support vector machines (SVMs) *Convex problem*

$$\min ||\mathbf{w}||^2 \quad s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 \quad \text{for all } i$$

  – *D+1* number of parameters, *N* constraints

$$L(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1 \right)$$

$$\alpha_i \geq 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \rightarrow \quad \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L}{\partial \mathbf{b}} = 0 \quad \rightarrow \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

Large Margin Classifier

- Maximize

$$L = \sum_i \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\alpha_i \geq 0, \qquad \sum \alpha_i y_i = 0$$

orthant *simplex* *Convex dual*

- With obtained $\alpha_1, \ldots, \alpha_N$,

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$b = -1 + \mathbf{w}^\top \mathbf{x}^{(s)}$$

$\mathbf{x}^{(s)}$: a support vector

- Classification:

$$y = \operatorname{sgn}\left[\mathbf{w}^\top \mathbf{x}_{\text{new}} - b\right]$$

Kernel Idea

- For training:

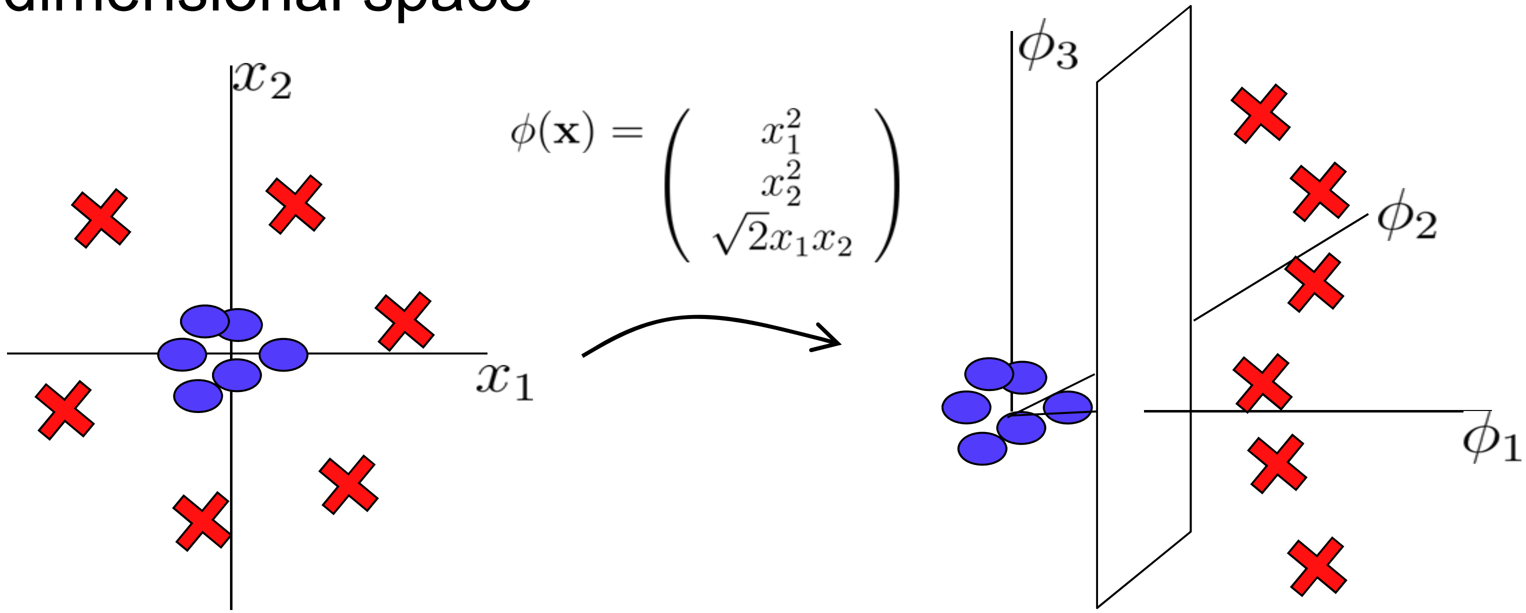
$$L = \sum_i \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underline{\mathbf{x}_i^\top \mathbf{x}_j}$$

- For classification

$$\begin{aligned} y &= \operatorname{sgn}\left[\mathbf{w}^\top \mathbf{x}_{\text{new}} - b\right] \qquad \left(\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i\right) \\ &= \operatorname{sgn}\left[\sum_i \alpha_i y_i \underline{\mathbf{x}_i^\top \mathbf{x}_{\text{new}}} - b\right] \end{aligned}$$

Mapping Data to a High Dimensional Space

- Consider a nonlinear mapping to a higher-dimensional space



- And consider a function

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^2 = (x_1z_1 + x_2z_2)^2 = x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$

$$= \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}^\top \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix} = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$

Mapping Data to High Dimensional Space

- Consider functions *k(.,.)* of which the corresponding mapping *Φ(.)* exists:
- Condition: if a function *k(.,.)* is **positive definite (P.D.)**, there exists a mapping function *Φ(.)* satisfying

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$ (Mercer theorem)

- Such functions include

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^d$$ (Polynomial kernel)

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{(\mathbf{x} - \mathbf{z})^2}{\gamma}\right)$$ (Gaussian kernel)

$$\vdots$$

Replace All Inner Products with Kernels

- For training:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underline{k(\mathbf{x}_i, \mathbf{x}_j)}$$

- For classification

$$\begin{aligned} y &= \operatorname{sgn}\left[\mathbf{w}^\top \mathbf{x}_{\text{new}} - b\right] \\ &= \operatorname{sgn}\left[\sum_i \alpha_i y_i \underline{k(\mathbf{x}_i, \mathbf{x}_{\text{new}})} - b\right] \end{aligned}$$

Positive Definiteness of a Matrix

- Positive definiteness of a matrix $A \in \mathbb{R}^{D\times D}$

$$A \succ 0$$

- For any vector $\mathbf{c} \in \mathbb{R}^D$

$$\mathbf{c}^\top A\mathbf{c} > 0$$

Positive semi-definiteness $A \succeq 0$ is similarly defined with the corresponding condition $\mathbf{c}^\top A\mathbf{c} \geq 0$ for any vector $\mathbf{c} \in \mathbb{R}^D$.

Positive Definiteness of a Matrix

- Matrix *A* is P.D. if (and only if)
  - All eigenvalues are positive $U = \left(\mathbf{u}_1, \cdots, \mathbf{u}_D\right)$

$$\mathbf{c}^\top A\mathbf{c} = \mathbf{c}^\top U\Lambda U^\top \mathbf{c} = \sum_{i=1}^{D} \lambda_i (\mathbf{u}_i^\top \mathbf{c})^2 > 0$$

  - A matrix can be decomposed as $A = Z^\top Z$ using a full rank matrix Z.

$$\mathbf{c}^\top A\mathbf{c} = \mathbf{c}^\top Z^\top Z\mathbf{c} = \sum_{i=1}^{D} (\mathbf{z}_i^\top \mathbf{c})^2 > 0$$

$$Z = \left(\mathbf{z}_1, \cdots, \mathbf{z}_D\right)$$

Positive Definiteness of a Function

- Positive definiteness of a function $k(\mathbf{x}_1, \mathbf{x}_2)$ of two variables $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$,
  - For any set of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ for arbitrary N, the matrix of size $N \times N$ with function values is positive definite.

$$\begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & \cdots & k(\mathbf{x}_2, \mathbf{x}_N) \\ & \cdots & \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix} \succ 0$$

Support Vectors

$$L(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1\right)$$

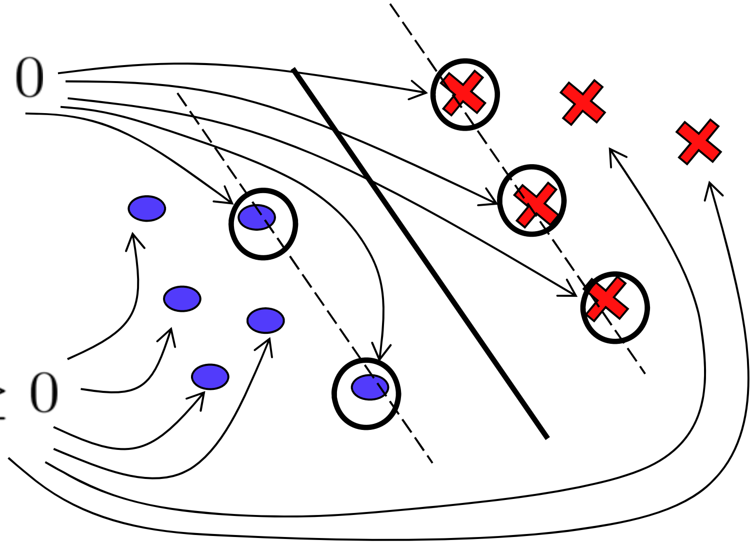
$$y = \operatorname{sgn}\left[\sum_i \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} - b\right]$$

- From KKT conditions, for *i = 1,…,N*,

$$\alpha_i \geq 0 \quad \& \quad y_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1 = 0$$

or

$$\alpha_i = 0 \quad \& \quad y_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1 \geq 0$$



Support Vectors and Classification

- Classification function

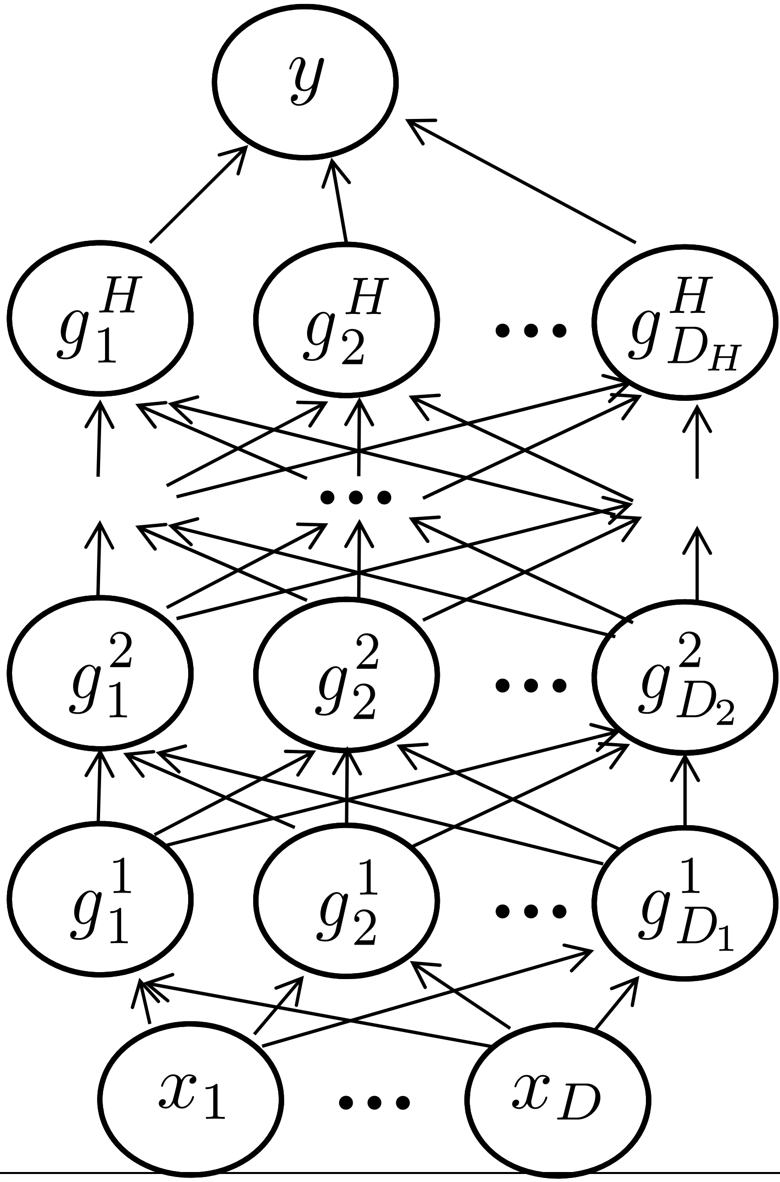
$$y = \mathrm{sgn}\left[\sum_{i \in SupportVectors} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_{\mathrm{new}}) - b\right]$$

- Keep data $\mathbf{x}_i$ having nonzero α_i .
- We can keep support vectors, and others do not contribute to classification

Advanced Topics

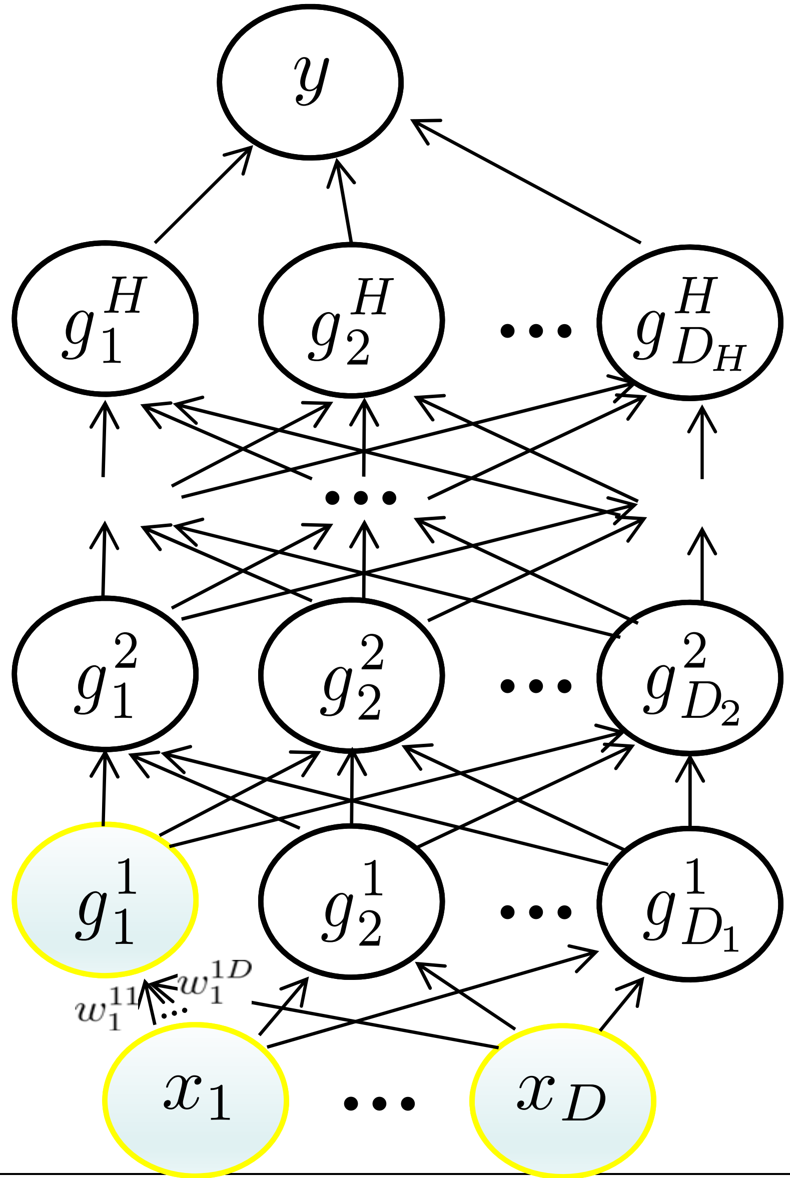
- Reproducing Kernel Hilbert Space (RKHS)
- Representer Theorem
- Kernelization of other algorithms

Artificial Neural Networks

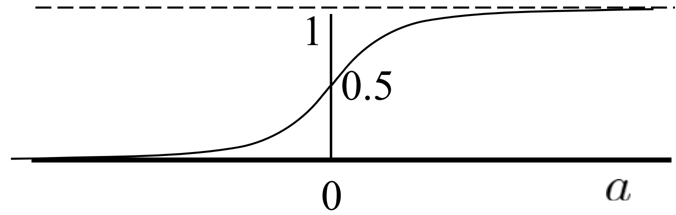


- Feedforward Neural Network
  - No loop (no recurrency)

Artificial Neural Networks

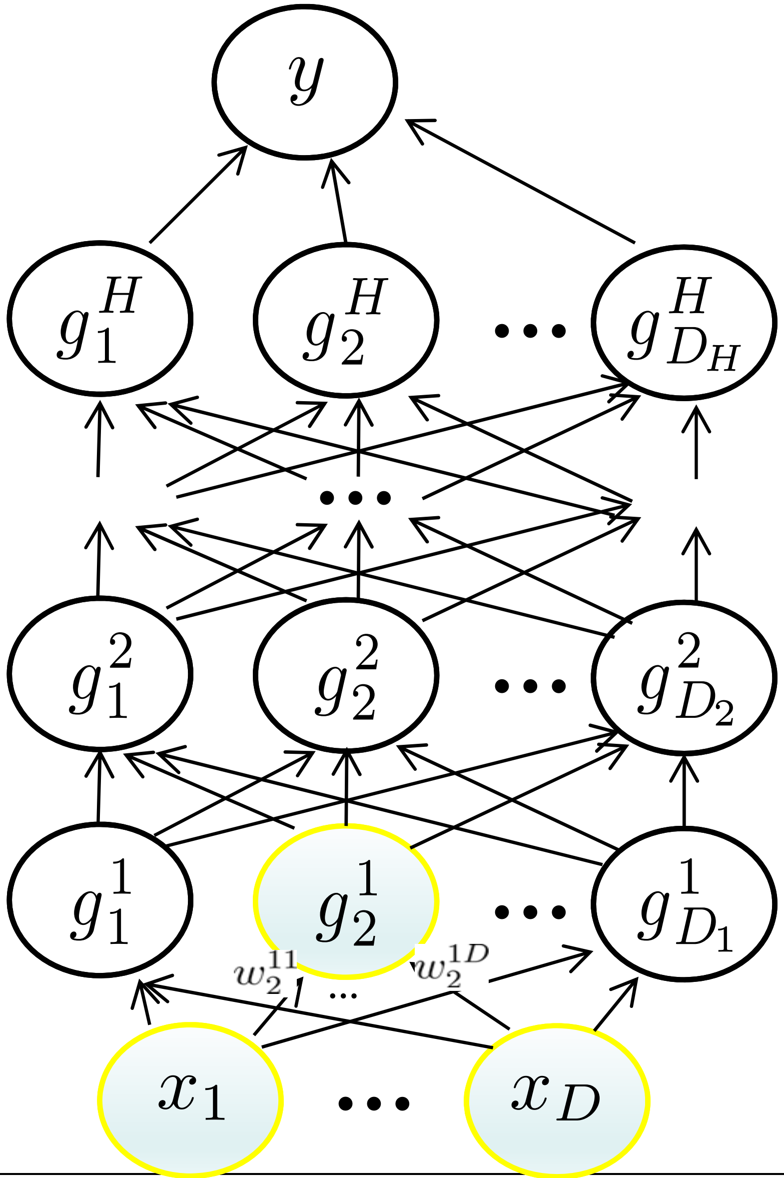


$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

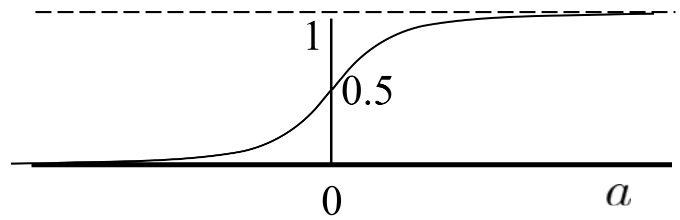


$$g_1^1 = \sigma\left(\sum_{i=1}^{D} w_1^{1i} x_i\right)$$

Artificial Neural Networks

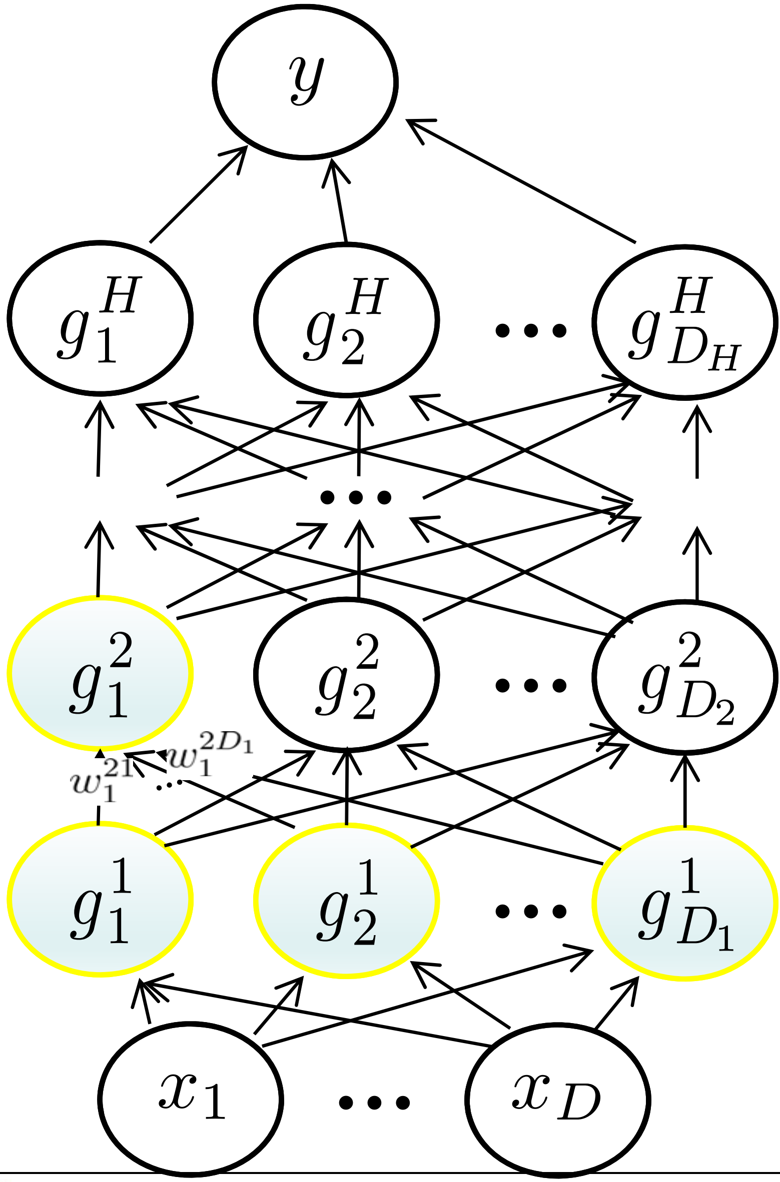


$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

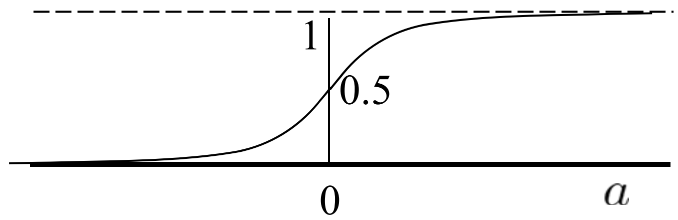


$$g_2^1 = \sigma\left(\sum_{i=1}^{D} w_2^{1i} x_i\right)$$

Artificial Neural Networks

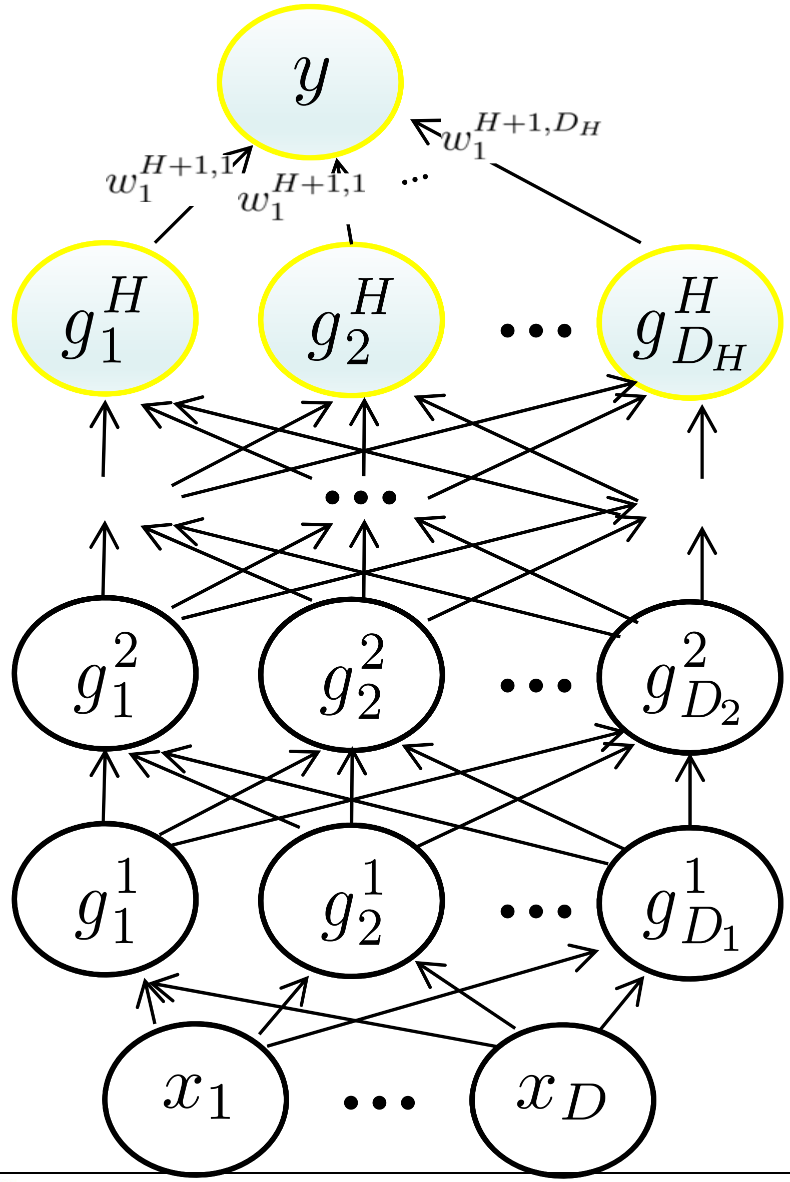


$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



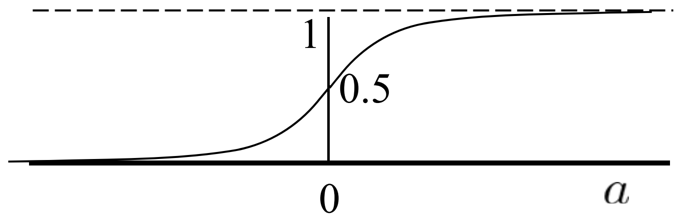
$$g_1^2 = \sigma\left(\sum_{i=1}^{D_1} w_1^{2i} g_i^1\right)$$

Artificial Neural Networks

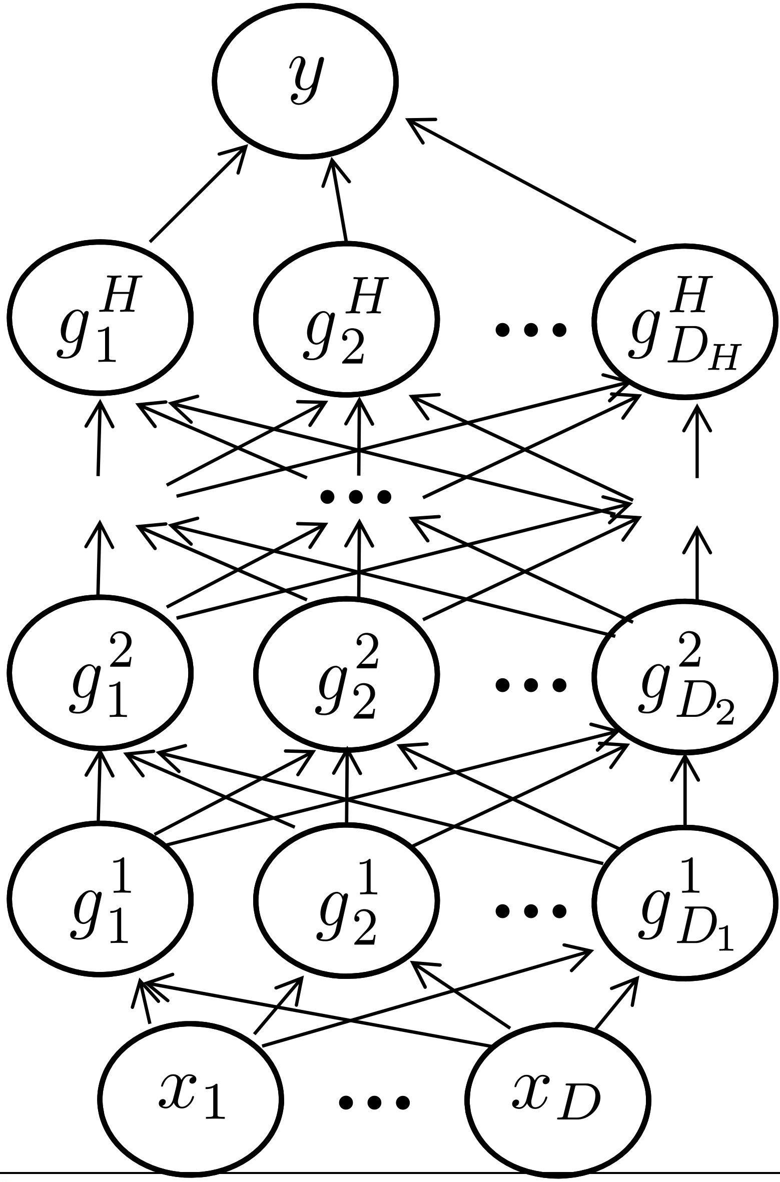


$$y = \sigma\left(\sum_{i=1}^{D_H} w^{out,i} g_i^H\right)$$

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



Artificial Neural Networks



$$y = \sigma(\mathbf{w}^{out\top}\mathbf{g}^H(\mathbf{x}))$$

$$\mathbf{g}^{h+1}(\mathbf{x}) = \sigma(W^{h+1\top}\mathbf{g}^h(\mathbf{x}))$$

$$W^h = \begin{pmatrix} w_1^{h,1} & \cdots & w_{D_{h-1}}^{h,1} \\ w_1^{h,2} & \cdots & w_{D_{h-1}}^{h,2} \\ & \vdots & \\ w_1^{h,D_h} & \cdots & w_{D_{h-1}}^{h,D_h} \end{pmatrix}$$

$$\mathbf{g}^1(\mathbf{x}) = \sigma(W^{1\top}\mathbf{x})$$

Optimization in ANN

$$\widehat{y}(\mathbf{x}) \leftarrow f(\mathbf{g}^{H}(\mathbf{g}^{H-1}\cdots(\mathbf{g}^{1}(\mathbf{x}))))$$
$$= \sigma(W^{H\top}\sigma(W^{H-1\top}\ldots\sigma(W^{1\top}\mathbf{x})))$$

- Empirical risk

$$L = \frac{1}{2}\sum_{i=1}^{N}||y_i - \widehat{y}(\mathbf{x}_i)||^2$$

  – Derivative of the empirical risk

$$\frac{dL}{dW^h} = \sum_{i=1}^{N}(y_i - \widehat{y}_i)\left(-\frac{\partial \widehat{y}_i}{\partial W^h}\right) \in \mathbb{R}^{D_h \times D_h}$$

$$\widehat{y}_i = \widehat{y}(\mathbf{x}_i)$$

Chain Rule

$$\frac{dL}{dW^h} = \sum_{i=1}^{N} (y_i - \widehat{y}_i) \left(-\frac{\partial \widehat{y}_i}{\partial W^h} \right)$$

$$\frac{\partial \widehat{y}}{\partial W^h} = \frac{\partial \widehat{y}}{\partial \mathbf{g}^H} \frac{\partial \mathbf{g}^H}{\partial \mathbf{g}^{H-1}} \cdots \frac{\partial \mathbf{g}^h}{\partial W^h}$$

$$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$$

$$\frac{\partial \widehat{y}}{\partial \mathbf{g}^H} = \frac{\partial \sigma(\mathbf{w}^{out\top} \mathbf{g}^H)}{\partial \mathbf{g}^H} = \sigma(\mathbf{w}^{out\top} \mathbf{g}^H)(1 - \sigma(\mathbf{w}^{out\top} \mathbf{g}^H))\mathbf{w}^{out}$$

$$\frac{\partial \mathbf{g}_i^{h+1}}{\partial \mathbf{g}^h} = \frac{\partial \sigma([W^{h+1\top} \mathbf{g}^h]_i)}{\partial \mathbf{g}^h} = \sigma([W^{h+1\top} \mathbf{g}^h]_i)(1 - \sigma([W^{h+1\top} \mathbf{g}^h]_i))W_{:i}^{h+1}$$

$$\frac{\partial \mathbf{g}_i^h}{\partial W_{:i}^h} = \frac{\partial \sigma([W^{h\top} \mathbf{g}^{h-1}]_i)}{\partial W_{:i}^h} = \sigma([W^{h\top} \mathbf{g}^{h-1}]_i)(1 - \sigma([W^{h\top} \mathbf{g}^{h-1}]_i))\mathbf{g}^{h-1}$$

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

$$\frac{\partial \mathbf{g}_i^h}{\partial W_{:j}^h} = 0, \qquad i \neq j$$

Update Rule

$$
\begin{aligned}
W^h &\leftarrow W^h - \gamma \frac{dL}{dW^h} \\
&= W^h - \gamma \sum_{i=1}^{N} (y_i - \widehat{y}_i) \left(-\frac{\partial \widehat{y}_i}{\partial W^h} \right) \\
&= W^h + \gamma \sum_{i=1}^{N} (y_i - \widehat{y}_i) \frac{\partial \widehat{y}_i}{\partial \mathbf{g}^H} \frac{\partial \mathbf{g}^H}{\partial \mathbf{g}^{H-1}} \cdots \frac{\partial \mathbf{g}^h}{\partial W^h}
\end{aligned}
$$

Delta Rule (Reuse Derivative)

$$\delta_i^h \equiv (y_i - \widehat{y}_i)\frac{\partial \widehat{y}_i}{\partial \mathbf{g}^H}\frac{\partial \mathbf{g}^H}{\partial \mathbf{g}^{H-1}}\cdots\frac{\partial \mathbf{g}^{h+1}}{\partial \mathbf{g}^h}$$

$$\delta_i^h = \delta_i^{h+1}\frac{\partial \mathbf{g}^{h+1}}{\partial \mathbf{g}^h}$$

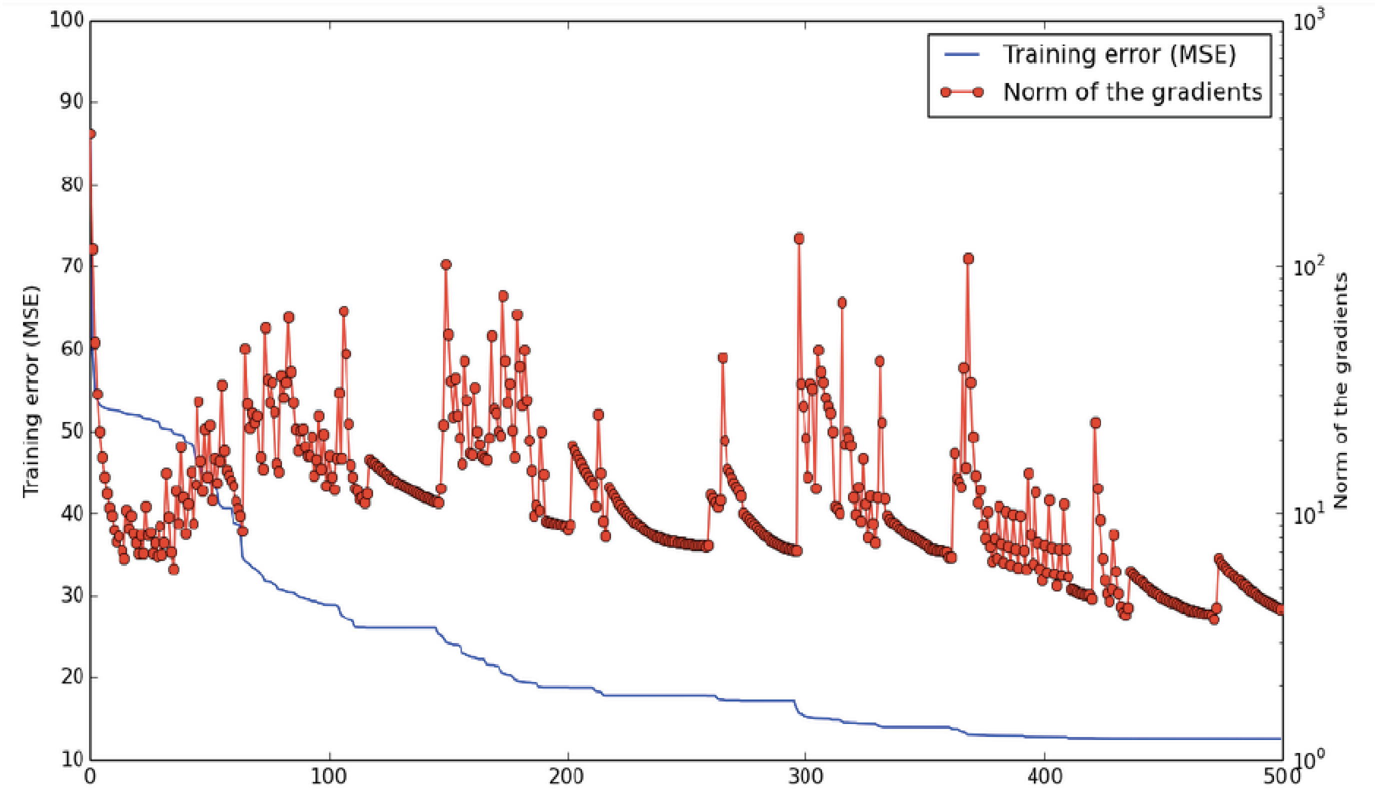
Backpropagation of error

- Update of weights of the level *h* using backpropagated error

$$W^h \leftarrow W^h + \gamma \sum_{i=1}^{N} \delta_i^h \frac{\partial \mathbf{g}^h}{\partial W^h}$$

Local Minima in ANN

- Saddle point analysis

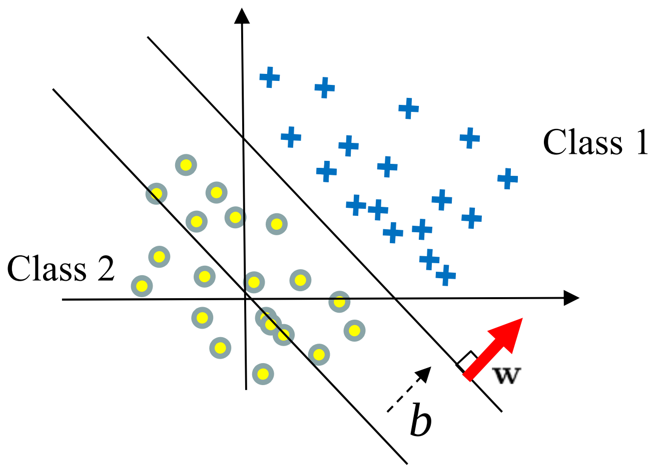


Razvan Pascanu et al. "On the saddle point problem for non-convex optimization." arXiv preprint arXiv:1405.4604 (2014)
Anna Choromanska et al. "The loss surfaces of multilayer networks." arXiv preprint arXiv:1412.0233 (2014)

Bias in ANN

- Linear classifier

$$\mathbf{w}^\top \mathbf{x} - b \lessgtr 0 \quad \rightarrow \quad \mathbf{w}'^\top \mathbf{x}' \lessgtr 0$$



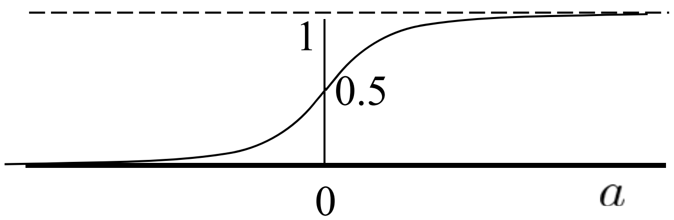
$$\mathbf{w}' = \begin{pmatrix} w_1 \\ \vdots \\ w_D \\ -b \end{pmatrix} \qquad \mathbf{x}' = \begin{pmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{pmatrix}$$

Implement a bias of the affine equation in a linear equation.

Different Activation Functions

- Sigmoid

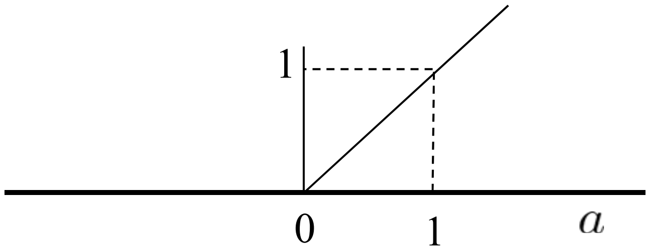
$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



$$\frac{d\sigma(a)}{da} = \sigma(1-\sigma)$$

- Rectified Linear Unit (ReLU)

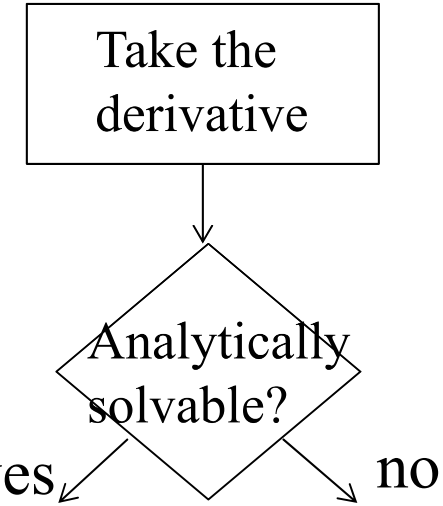
$$\sigma(a) = \max\{0, a\}$$



$$\frac{d\sigma(a)}{da} = \begin{cases} 1 & a > 0 \\ 0 & a \leq 0 \end{cases}$$

Determine the Problem I Am Solving!!

"Take the derivative, first"



Ridge regression

- Closed form solution
- Eigenvector problem

Fisher discriminant analysis

Support vector machines

- Convex problem
- Non-convex problem

Logistic regression
Artificial neural network

ANY QUESTIONS?