COMP5211: Machine Learning

Lecture 2

Minhao Cheng

Last lecture

Matrix Derivates

- Before we have $df = f'(x)dx$
- In the vector view:
  - Scalar to vector: $df = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} dx_i = \frac{\partial f}{\partial x}^T dx$ where $\frac{\partial f}{\partial x}$ and dx are $n \times 1$ vector
  - Similarly, scalar to matrix: $df = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{\partial f}{\partial X_{ij}} dX_{ij} = tr(\frac{\partial f}{\partial X}^T dX)$
  - For the derivate, we also have $d(X \pm Y) = dX \pm dY$, $d(XY) = (dX)Y + XdY$, $d(X^T) = (dX)^T$, $dtr(X) = tr(dX)$, $dX^{-1} = -X^{-1}dXX^{-1}$
  - For the trace operation, we also have $a = tr(a)$, , $tr(A \pm B) = tr(A) \pm tr(B)$, $tr(AB) = tr(BA)$, $tr(A^T(B \odot C)) = tr((A \odot B)^T C)$

Last lecture

Matrix Derivates

- Chain rule: f is a function of Y, let Y=AXB, to get $\frac{\partial f}{\partial X}$
  - $df = tr(\frac{\partial f}{\partial Y}^T dY) = tr(\frac{\partial f}{\partial Y}^T AdXB) = tr(B\frac{\partial f}{\partial Y}^T AdX) = tr((A^T \frac{\partial f}{\partial Y} B^T)^T dX)$
    - Since $dY = (dA)XB + A(dX)B + AX(dB) = A(dX)B$ as $dA = 0, dB = 0$
  - So we get $\frac{\partial f}{\partial X} = A^T \frac{\partial f}{\partial Y} B^T$

Last lecture

Matrix calculus

- $f = \|Xw - y\|^2$, solve $\dfrac{\partial f}{\partial w}$, where y is $m \times 1$ vector, X is $m \times n$ matrix, w is $n \times 1$ vector

- $$\begin{aligned} df = d(\|Xw - y\|^2) &= d((Xw - y)^T(Xw - y)) = d((Xw - y)^T)(Xw - y) + (Xw - y)^T d(Xw - y) \\ &= (Xdw)^T(Xw - y) + (Xw - y)^T(Xdw) = 2(Xw - y)^T Xdw \end{aligned}$$

- So $\dfrac{\partial f}{\partial w} = 2X^T(Xw - y)$

Regression

Example

- Classification:
  - Customer record ⟶ Yes/No
- Regression: predicting credit limit
  - Customer record ⟶ dollar amount

Regression

Linear regression

- Classification:
  - Customer record $\longrightarrow$ Yes/No
- Regression: predicting credit limit
  - Customer record $\longrightarrow$ dollar amount
- Linear Regression:
  - $h(x) = \sum_{i=0}^{d} \boxed{w_i} x_i = \boxed{w^T} x$

Linear Regression

The data set

- Training data:
  - $(x1, y1), (x2, y2), \ldots, (x_N, y_N)$
  - $x_n \in \mathbb{R}^d$: feature vector for a sample
  - $y_n \in \mathbb{R}$: observed output (real number)

Linear Regression

The data set

- Training data:
  - $(x1, y1), (x2, y2), \ldots, (x_N, y_N)$
  - $x_n \in \mathbb{R}^d$: feature vector for a sample
  - $y_n \in \mathbb{R}$: observed output (real number)
- Linear regression: find a function $h(x) = w^T x$ to approximate y

Linear Regression

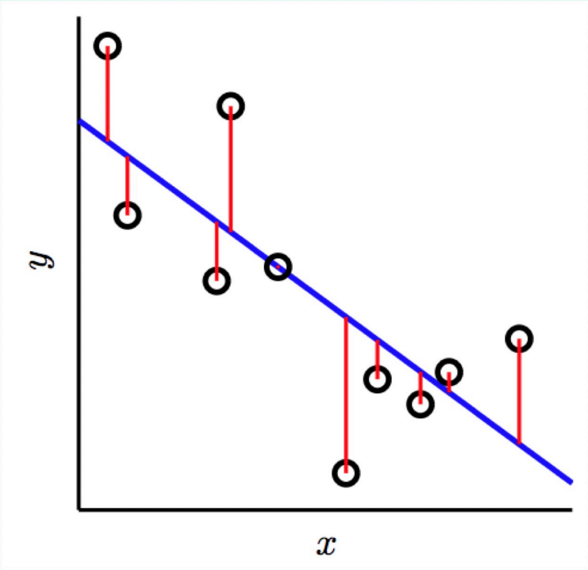
The data set

- Training data:
  - $(x1,y1), (x2,y2), \ldots, (x_N, y_N)$
  - $x_n \in \mathbb{R}^d$: feature vector for a sample
  - $y_n \in \mathbb{R}$: observed output (real number)
- Linear regression: find a function $h(x) = w^T x$ to approximate y
- Measure the error by $(h(x) - y)^2$ (square error)
  - Training error: $E_{\text{train}}(h) = \dfrac{1}{N} \sum_{n=1}^{N} (h(x_n) - y_n)^2$

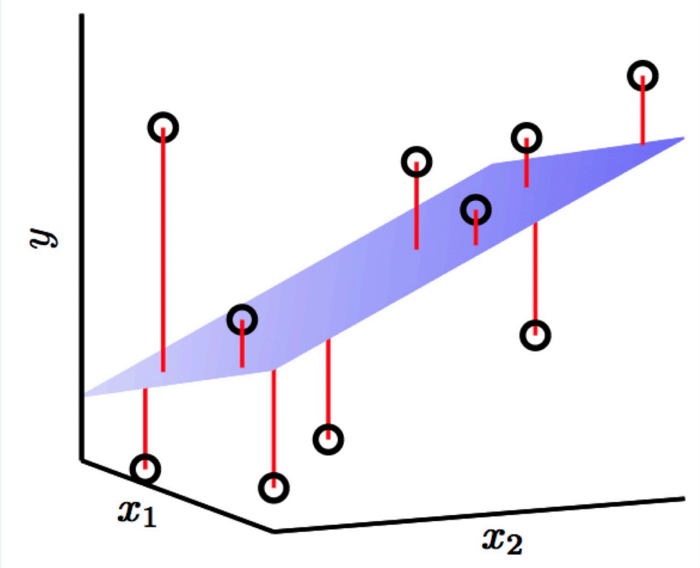
Linear Regression

Illustration

$\mathbf{x} = (x) \in \mathbb{R}$



$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$



Linear Regression

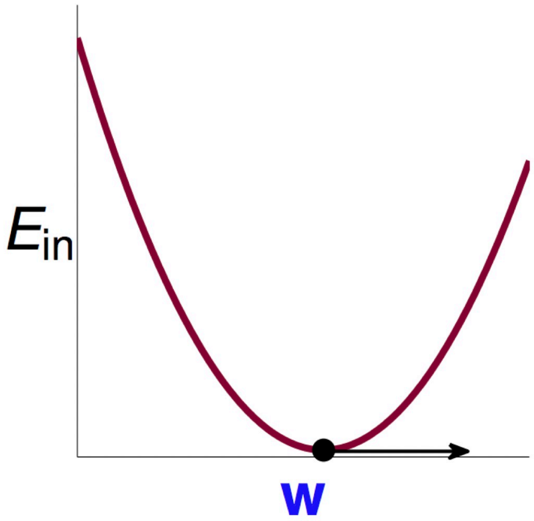
Matrix form

- $$E_{\text{train}}(w) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T w - y_n)^2 = \frac{1}{N}\| \begin{bmatrix} x_1^T w - y_1 \\ x_2^T w - y_2 \\ \vdots \\ x_N^T w - y_N \end{bmatrix} \|^2$$
$$= \frac{1}{N}\| \begin{bmatrix} -x_1^T- \\ -x_2^T- \\ \vdots \\ -x_N^T- \end{bmatrix} w - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \|$$
$$= \frac{1}{N}\| \underbrace{X}_{N\times d} w - \underbrace{y}_{N\times 1} \|^2$$

Linear Regression

Minimize E_{train}

- $min_w f(w) = \|Xw - y\|^2$
  - E_{train}: continuous, differentiable, convex
  - Necessary condition of optimal w:
  - $$\nabla f(w^*) = \begin{bmatrix} \frac{\partial f}{\partial w_0}(w^*) \\ \vdots \\ \frac{\partial f}{\partial w_d}(w^*) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$



Linear Regression

Minimizing f

- $$f(w) = \|Xw - y\|^2 = w^T X^T X w - 2w^T X^T y + y^T y$$
  $$\nabla f(w) = 2(X^T X w - X^T y)$$
  $$\nabla f(w^*) = 0 \Rightarrow \underbrace{X^T X w^* = X^T y}_{\text{normal equation}}$$

Linear Regression

Minimizing f

$$f(w) = \|Xw - y\|^2 = w^T X^T X w - 2w^T X^T y + y^T y$$

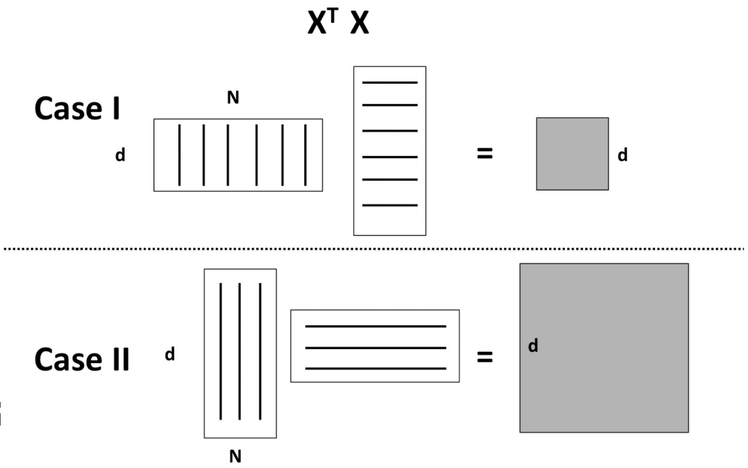
$$\nabla f(w) = 2(X^T X w - X^T y)$$

- $$\nabla f(w^*) = 0 \Rightarrow \underbrace{X^T X w^* = X^T y}_{\text{normal equation}}$$

- $\Rightarrow w^* = (X^T X)^{-1} X^T y$ How?

Linear Regression
Solutions

- Case I: $X^T X$ is invertible $\Rightarrow$ Unique solution
  - Often when $N > d$
  - Yes, $w* = (X^T X)^{-1} X^T y$
- Case II: $X^T X$ is non-invertible $\Rightarrow$ Many solutions
  - Often when $d > N$



Linear Regression

Linear System Solver

- A “linear system”:
  - Find the minimum 2-norm solution of $\min_{w} ||Xw - y||$

Linear Regression

Linear System Solver

- A "linear system":
  - Find the minimum 2-norm solution of $\min_w \|Xw - y\|$
- Let $X = U\Sigma V^T$ be the SVD of X:
  - $U^T U = I$
  - $V^T V = I$
  - $\Sigma =$ diag$[\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0]$, $(\sigma_1, \ldots, \sigma_r > 0)$
  - Solution: $w^+ = X^+ y$
    - where $X^+ = V\Sigma^+ U^T$, $\Sigma^+ =$ diag$[1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_r, 0, \ldots, 0]$

Linear Regression

Linear System Solver

- A “linear system”:
    - Find the minimum 2-norm solution of $\min_w \|Xw - y\|$
- Let $X = U\Sigma V^T$ be the SVD of X:
    - $U^T U = I$
    - $V^T V = I$
    - $\Sigma =$ diag$[\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0]$, $(\sigma_1, \ldots, \sigma_r > 0)$
    - Solution: $w^+ = X^+ y$
        - where $X^+ = V\Sigma^+ U^T$, $\Sigma^+ =$ diag$[1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_r, 0, \ldots, 0]$
        - X^+: pseudo-inverse of X
        - Why?

Linear Regression

Linear System Solver (Cont*)

- $\|Xw - y\|^2 = \|U\Sigma V^T w - y\|^2 = \|\Sigma V^T w - U^T y\|^2$ since ?
- Let $z = V^T w$,
  - The solution of $\min_w \|\Sigma V^T w - U^T y\|^2$ is equivalent to find the solution of
    - $\min_z \|\Sigma z - U^T y\|^2$
      - Since Σ is diagonal, so the solution is $z^+ = \Sigma^+ U^T y$
      - So $w^+ = Vz^+ = V\Sigma^+ U^T y$

Computation complexity

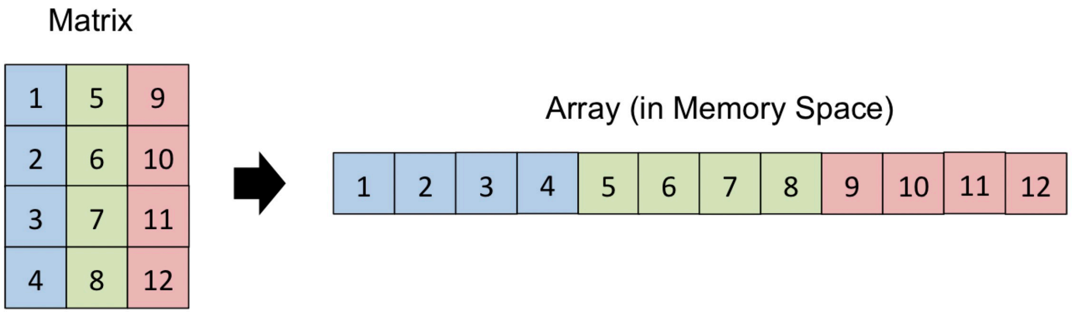
Dense vector and sparse vector

- If $x, y \in \mathbb{R}^m$ are dense:
  - $x + y, x - y, x^T y$: $O(m)$ operations
- If $x, y \in \mathbb{R}^m$, x is dense and y is sparse:
  - $x + y, x - y, x^T y$: $O(\text{nnz}(y))$ operations
- If $x, y \in \mathbb{R}^m$ and both of them are sparse:
  - $x + y, x - y, x^T y$: $O(\text{nnz}(y) + nnz(x))$ operations

Computation complexity

Dense matrix

- Let $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{m\times n},\ s \in \mathbb{R}$:
  - $A + B, sA, A^T : O(mn)$ operations
- If $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^{n\times 1}$
  - $Ab : O(mn)$ operations
- If $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times n}$,
  - $AB : O(n^3)$operations; theoretical best: $O(n^{2.xxx})$
- $A \in \mathbb{R}^{m\times k}, B \in \mathbb{R}^{k\times n}$,
  - $AB : O(mnk)$ operations;



Computation complexity

Sparse matrix format

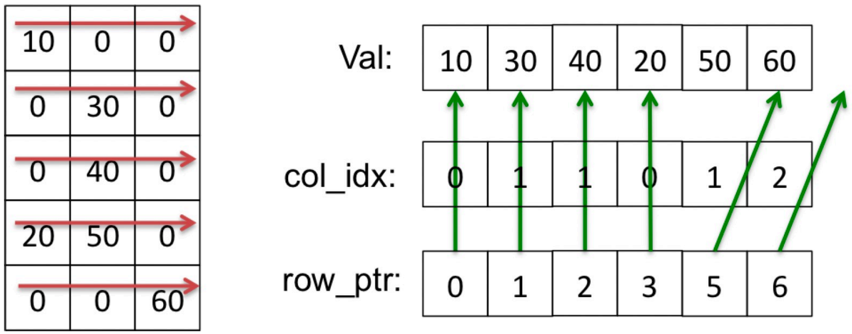
- Widely-used format: Compressed Sparse Column (CSC), Compressed Sparse Row (CSR), …

- CSR: three arrays for storing an $m \times n$ matrix with nnz nonzero

  - val (nnz real numbers): the values of each nonzero elements

  - row_ind (nnz integers): the column indices corresponding to the values

  - col_ptr ($m + 1$ integers): the list of value indexes where each column starts



Computation complexity

Sparse matrix operations

- Let $A \in \mathbb{R}^{m \times n}$ (sparse), $B \in \mathbb{R}^{m \times n}$ (sparse or dense), $s \in \mathbb{R}$:
  - $A + B, sA, A^T : O(nnz)$ operations
- If $A \in \mathbb{R}^{m \times n}$(sparse), $b \in \mathbb{R}^{n \times 1}$
  - $Ab : O(nnz)$ operations
- If $A \in \mathbb{R}^{n \times k}$ (sparse), $B \in \mathbb{R}^{k \times n}$ (dense),
  - $AB : O(nnz(A)n)$ operations (use sparse BLAS)
- $A \in \mathbb{R}^{n \times k}$ (sparse), $B \in \mathbb{R}^{k \times n}$ (sparse),
  - $AB : O(nnz(A)nnz(B)/k)$ in average
  - $AB : O(nnz(A)n)$ worst case
  - The resulting matrix will be much denser

Linear Regression

Computational complexity

- Computational cost for computing $(X^TX)^{-1}X^Ty$:
    - Computing X^TX: $O(d^2N)$ time
    - Computing matrix inversion: $O(d^3)$ time
    - Overall complexity: $O(d^2N + d^3)$
- What if $d, N \approx$ millions?
    - (Use iterative algorithms, next class)

Logistic Regression

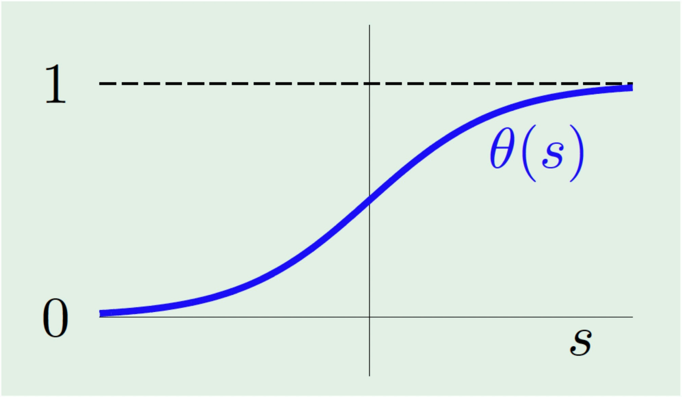
Binary Classification

- Input: training data $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ and corresponding outputs $y_1, y_2, \ldots, y_n \in \{+1, -1\}$
- Training: compute a function f such that $\text{sign}(f(x_i)) \approx y_i$ for all i
- Prediction: given a testing sample $\tilde{x}$, predict the output as $\text{sign}(f(x_i))$

Logistic Regression

Binary Classification

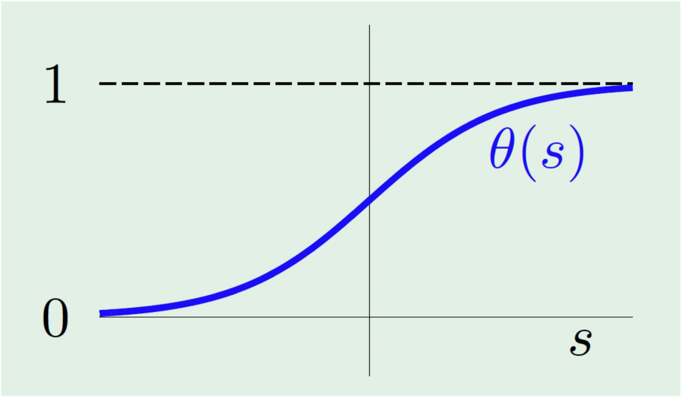
- Assume linear scoring function: $s = f(x) = w^T x$
- Logistic hypothesis:
  - $P(y = 1 \mid x) = \theta(w^T x)$,
  - Where $\theta(s) = \dfrac{e^s}{1 + e^s} = \dfrac{1}{1 + e^{-s}}$
    - It is called sigmoid function



Logistic Regression

Binary Classification

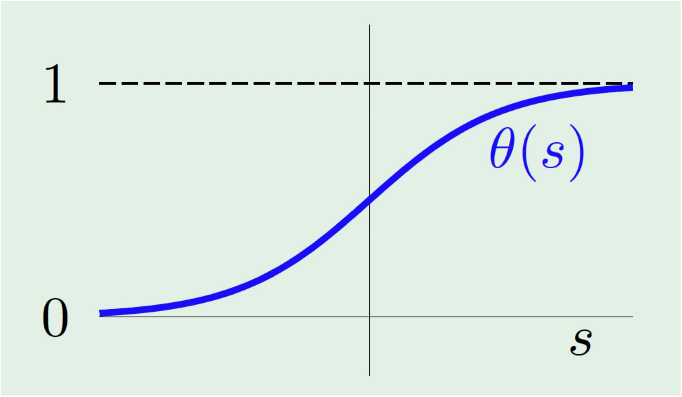
- Assume linear scoring function: $s = f(x) = w^T x$
- Logistic hypothesis:
  - $P(y = 1 \mid x) = \theta(w^T x)$,
  - Where $\theta(s) = \dfrac{e^s}{1 + e^s} = \dfrac{1}{1 + e^{-s}}$
- How about $P(y = -1 \mid x)$?



Logistic Regression

Binary Classification

- Assume linear scoring function: $s = f(x) = w^T x$
- Logistic hypothesis:
  - $P(y = 1 \,|\, x) = \theta(w^T x)$,
  - Where $\theta(s) = \dfrac{e^s}{1 + e^s} = \dfrac{1}{1 + e^{-s}}$



- How about $P(y = -1 \,|\, x)$?
  - $P(y = -1 \,|\, x) = 1 - \dfrac{1}{1 + e^{-w^T x}} = \dfrac{1}{1 + e^{w^T x}} = \theta(-w^T x)$

Logistic Regression

Binary Classification

$s = f(x) = w^T x$

2 1
0 3

$[\quad] \begin{bmatrix} 2 \\ 1 \\ 6 \\ 3 \end{bmatrix}$

- Assume linear scoring function: $s = f(x) = w^T x$
- Logistic hypothesis:
  - $P(y = 1 \mid x) = \theta(w^T x)$,
  - Where $\theta(s) = \dfrac{e^s}{1 + e^s} = \dfrac{1}{1 + e^{-s}}$
- How about $P(y = -1 \mid x)$?
  - $P(y = -1 \mid x) = 1 - \dfrac{1}{1 + e^{-w^T x}} = \dfrac{1}{1 + e^{w^T x}} = \theta(-w^T x)$
- Therefore, $P(y \mid x) = \theta(y w^T x)$

$y = \pm 1$

Logistic Regression

Maximizing the likelihood

- Likelihood of $\mathscr{D} = (x_1, y_1), \ldots, (x_N, y_N)$:
- $\prod_{n=1}^{N} P(y_n \mid x_n) = \prod_{n=1}^{N} \theta(y_n w^T x_n)$

[[784], [784] ...]

[[0, 0, 0], [0, 0, 0]]

size 2, 3,

[2, 2]

[784 784 784 784 ... 784]

64

Logistic Regression

$\left[\ 784\ \ 784\ \ 784\ \cdots\ 784\ \right]$

$64.$

Maximizing the likelihood

- Likelihood of $\mathcal{D} = (x_1, y_1), \ldots, (x_N, y_N)$:
- $$\prod_{n=1}^{N} P(y_n \mid x_n) = \prod_{n=1}^{N} \theta(y_n w^T x_n)$$

$$\theta(s) = \frac{1}{1+e^{-s}}$$

$$\theta(y_n w^T x_n) = \frac{1}{1+e^{-y_n w^T x_n}}$$

- Find w to maximize the likelihood!
- $$\max_w \prod_{n=1}^{N} \theta(y_n w^T x_n)$$
  $$\leftrightharpoons \max_w log(\prod_{n=1}^{N} \theta(y_n w^T x_n))$$
  $$\leftrightharpoons \min_w - \sum_{n=1}^{N} log(\theta(y_n w^T x_n)) \quad \min - \sum_{n=1}^{N} \frac{1}{1+e^{-\cancel{x}}}$$
  $$\leftrightharpoons \min_w \sum_{n=1}^{N} log(1 + e^{-y_n w^T x_n})$$

Logistic Regression

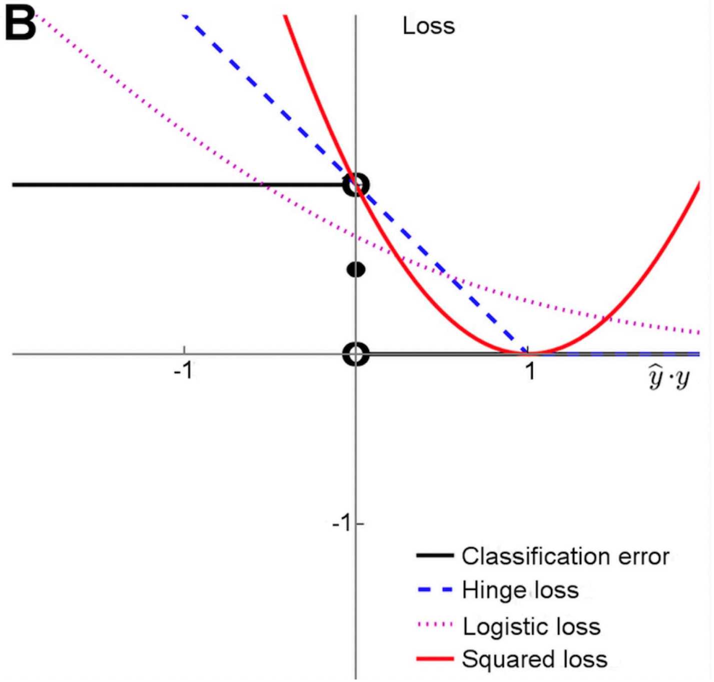
Empirical Risk Minimization (linear)

- Linear classification/regression:

- $$min_w \frac{1}{N} \sum_{n=1}^{N} \text{loss}(\underbrace{w^T x_n}_{\hat{y}_n\text{:the predicted score}}, y_n)$$

  - Linear regression:
    $\text{loss}(h(x_n), y_n) = (w^T x_n - y_n)^2$

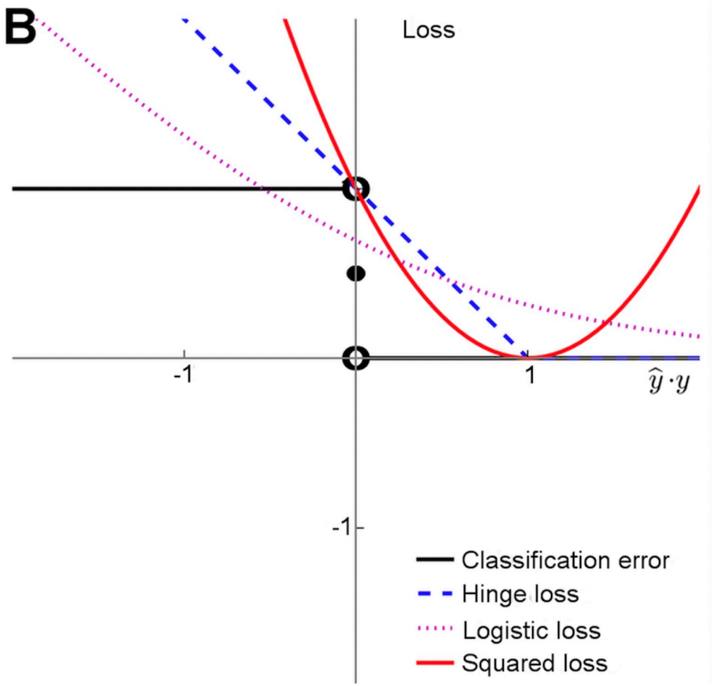
  - Logistic regression:
    $\text{loss}(h(x_n), y_n) = \log(1 + e^{-y_n w^T x_n})$



Support Vector Machines

Hinge loss

- Replace the logistic loss by hinge loss:
- $$\min_{w} \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n w^T x_n)$$



Logistic Regression

Empirical Risk Minimization (general)

- Assume $f_W(x)$ is the decision function to be learned
  - (W is the parameters of the function)
- General empirical risk minimization
  - $min_{W} \frac{1}{N} \sum_{n=1}^{N} \text{loss}(f_{W}(x_n), y_n)$
- Example: Neural network ($f_W(\cdot)$ is the network)