AI for Medical Image Classification

- Linear Classifiers -

Hung Hung

Outline

- Preliminary
- Logistic Regression (LR)
- Neural Network (NN)
- Support Vector Machine (SVM)
- Kernel Trick
- Summary

Preliminary

Problem Formulation

- Aim to predict Y by X
  - Response Y
    - 2 classes: $\{0,1\}$ or $\{-1,+1\}$
    - $M+1$ classes: $\{0,1,\ldots,M\}$
  - Covariate $X = \left(X_1, X_2, \ldots, X_p\right)^\top$

- Classification by posterior probability $\pi_j(X) = P(Y = j|X)$
  - Bayes classifier: $\hat{Y} = \mathrm{argmax}_j\ \pi_j(X)$

Q: How to estimate $\pi_j(X)$?

Statistical Inference Procedure

$\pi_j(X) \overset{\text{m}}{=} \pi_j(X;\theta)$

- Random variable $Z = (X, Y) \sim g$
  - Model: $g \overset{\text{m}}{=} f_\theta$ for a known function f_θ with unknown parameter θ
- Aim to estimate θ by the data $\{Z_i\}_{i=1}^{n}$
- Regularized maximum likelihood estimation (MLE)
  - log likelihood function: $\ell(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ln f_\theta(Z_i)$
  - $\max_\theta \ell(\theta) + \lambda J(\theta)$ → $\hat{\theta}$

- λ: penalty
- $J(\theta)$: regularization

Another View of Estimation

- Two distributions
  - Empirical distribution of $\{Z_i\}_{i=1}^n$: $\hat{g} = \frac{1}{n}\sum_{i=1}^n \delta_{Z_i}$
  - Model distribution: f_θ

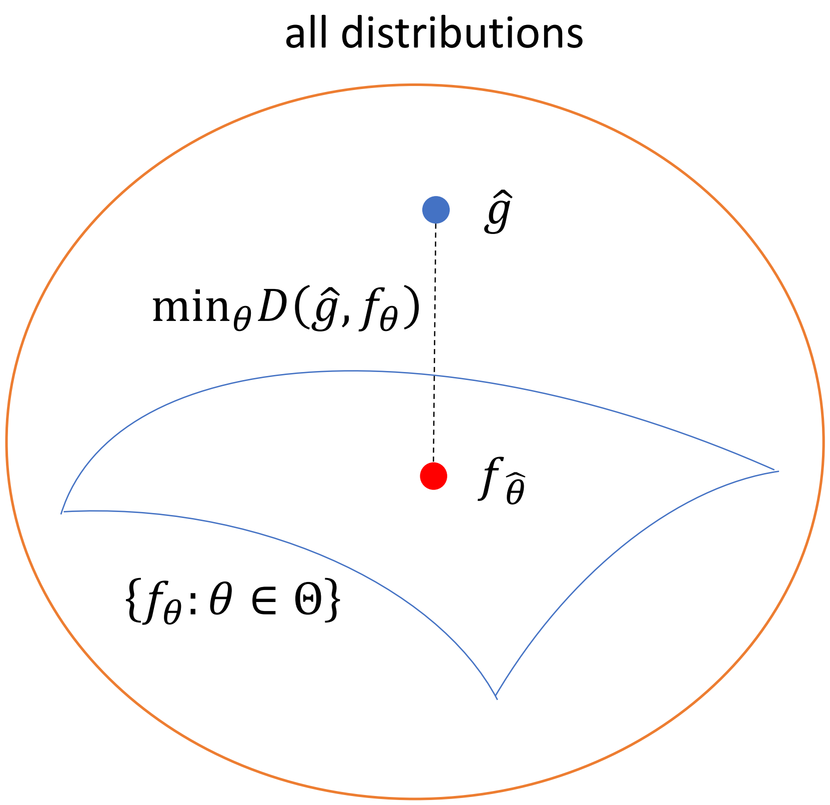
- A geometric interpretation of estimation
  - Find θ so that $\hat{g}$ and f_θ are as close as possible

Q: distance between $\hat{g}$ and f_θ?

- Divergence $D(g, f)$: a measure of distance between g and f
  - $D(g, f) \geq 0$
  - $D(g, f) = 0$ iff $g = f$

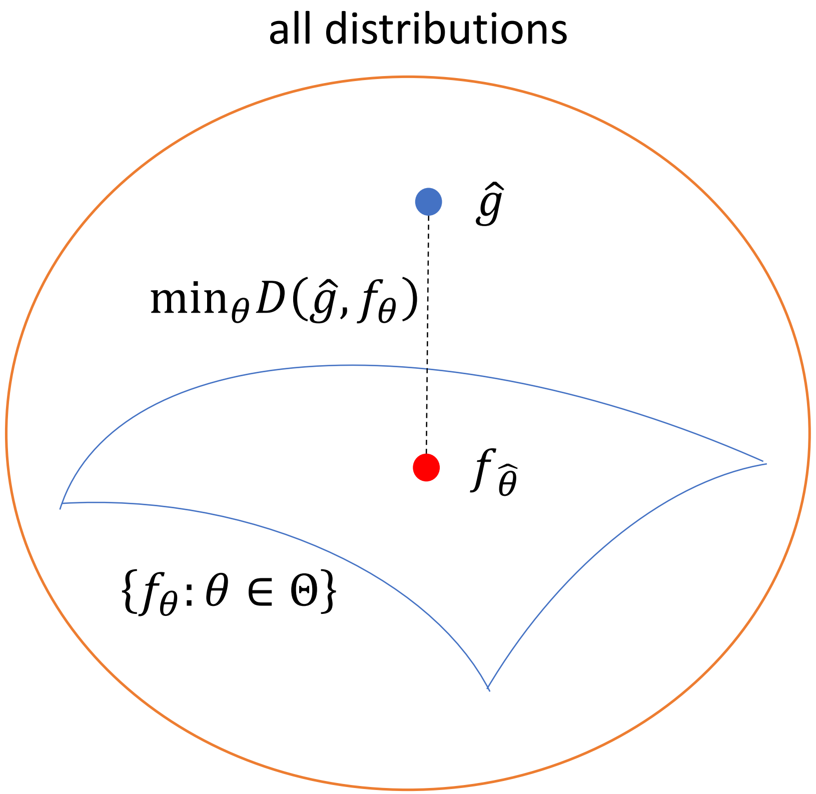
Another View of Estimation

- Minimum divergence estimation
  - $\min_\theta D(\hat{g}, f_\theta) + \lambda J(\theta)$ → $\hat{\theta}$
- Kullback-Leibler (KL) divergence
  - $D_{KL}(g, f) = \int \ln g \cdot g - \int \ln f \cdot g$
  - $D_{KL}(\hat{g}, f_\theta) \propto -\int \ln f_\theta \cdot \hat{g} = -\frac{1}{n}\sum_{i=1}^{n} \ln f_\theta(Z_i)$
  - Minimize D_{KL} = MLE
- D determines the statistical properties of $\hat{\theta}$
  - SVM
  - γ-divergence based robust statistical methods



A Quick Summary for Classification

- Random vector (X, Y)
- Bayes classifier: $\hat{Y} = \text{argmax}_j\, \pi_j(X)$
  - Target: $\pi_j(X) = P(Y = j|X)$
- Model: $\pi_j(X) \stackrel{\text{m}}{=} \pi_j(X; \theta)$ → model distribution f_θ
- Data: $\{(X_i, Y_i)\}_{i=1}^n$ → data distribution $\hat{g}$
- Estimating θ via a proper D
  - $\min_\theta D(\hat{g}, f_\theta) + \lambda J(\theta)$ → $\hat{\theta}$ and, hence, $\pi_j(X; \hat{\theta})$



Logistic Regression

Linear classifier for binary Y

Logistic Regression (LR)

- Random vector (X, Y)
  - Binary $Y \in \{0,1\}$
  - Covariate $X = \left(X_1, X_2, \dots, X_p\right)^\top$

$I\{\cdot\}$: indicator function

- Bayes classifier: $\hat{Y} = I\{\pi_1(X) > 0.5\}$
  - Target: $\pi_1(X) = P(Y = 1|X)$

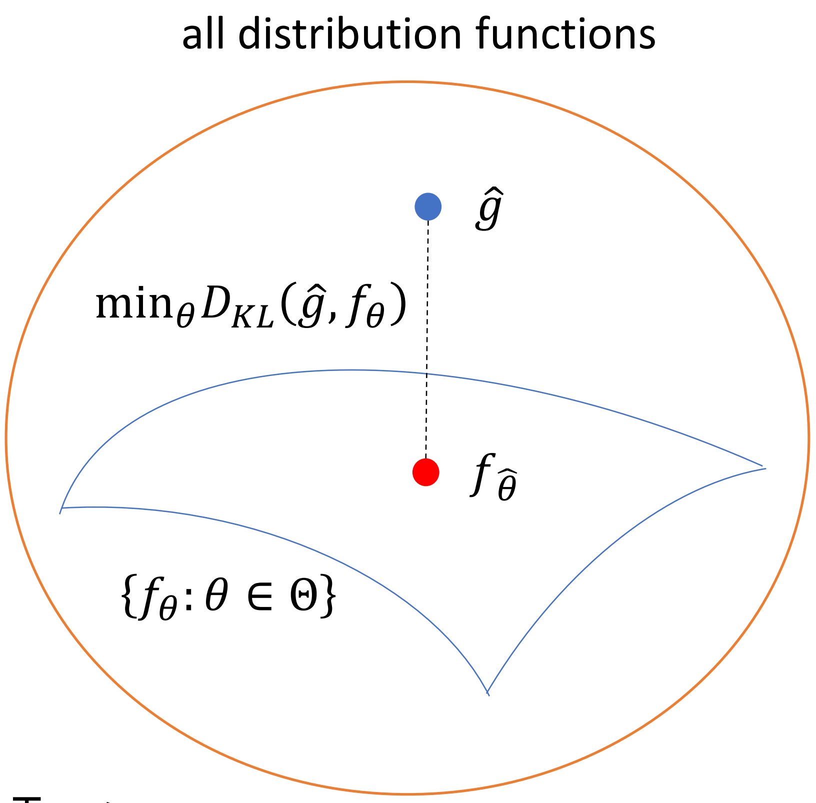
- Model
  - $Y|X \sim$ Bernoulli$\left(\pi_1(X)\right)$
  - $\ln \frac{\pi_1(X)}{1-\pi_1(X)} \overset{\text{m}}{=} \beta_0 + \beta^\top X \Leftrightarrow \pi_1(X) \overset{\text{m}}{=} \frac{\exp(\beta_0+\beta^\top X)}{1+\exp(\beta_0+\beta^\top X)}$

$\theta = (\beta_0, \beta)$

Estimation of LR

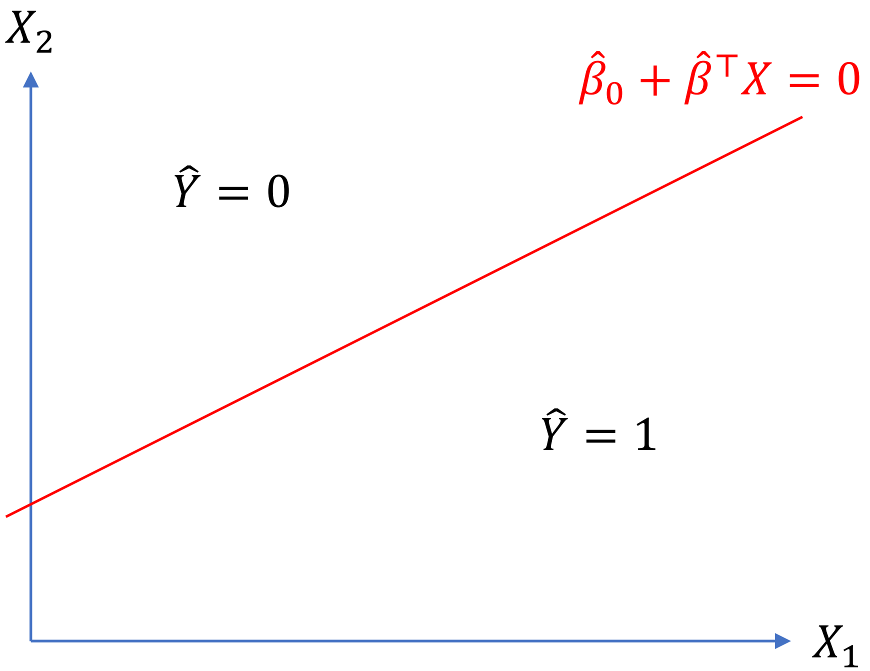
- Two distributions
  - Model: $f_\theta(y|x) = \{\pi_1(x)\}^y \{1 - \pi_1(x)\}^{1-y}$
  - Data: $\{(X_i, Y_i)\}_{i=1}^n \rightarrow \hat{g}$

- Estimation via D_{KL}
  - $\min_\theta D_{KL}(\hat{g}, f_\theta) + \lambda J(\theta) \rightarrow \hat{\theta}$
  - $D_{KL}(\hat{g}, f_\theta) = \frac{1}{n} \sum_i \ln\left\{1 + e^{\beta_0 + \beta^\top X_i}\right\} - Y_i(\beta_0 + \beta^\top X_i)$



Classification of LR

- log-odds ratio: $r(X) = \ln \frac{\pi_1(X)}{1-\pi_1(X)}$
  - $\pi_1(X) > 0.5 \Leftrightarrow r(X) > 0$
- Bayes classifier: $\hat{Y} = I\{r(X) > 0\}$
- Model: $r(X) = \ln \frac{\pi_1(X)}{1-\pi_1(X)} \overset{\mathrm{m}}{=} \beta_0 + \beta^\top X$
- Bayes classifier: $\hat{Y} = I\{\hat{\beta}_0 + \hat{\beta}^\top X > 0\}$
  - Linear classifier with *decision boundary* $\hat{\beta}_0 + \hat{\beta}^\top x = 0$



Multiclass Logistic Regression

Linear classifier for categorical Y

Multiclass Logistic Regression (MLR)

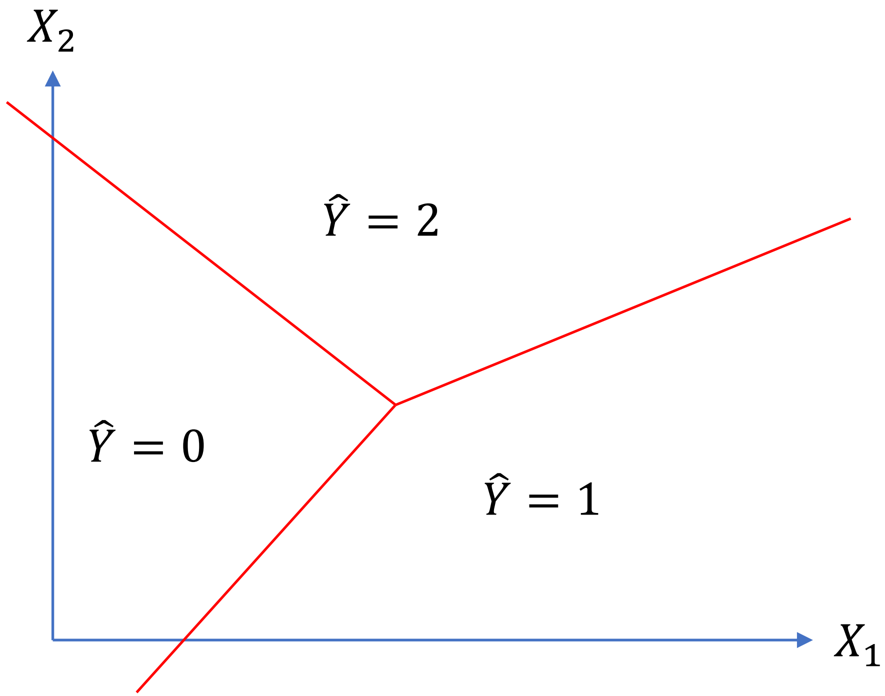
- Random vector (X, Y)
  - Response $Y \in \{0,1,\dots,M\}$
  - Covariate $X = \left(X_1, X_2, \dots, X_p\right)^\top$
- Bayes classifier: $\hat{Y} = \mathrm{argmax}_{0 \le j \le M}\ \pi_j(X)$
  - Target: $\pi_j(X) = P(Y = j|X)$
- Model
  - $Y|X \sim \text{Multinomial}\big(\pi(X)\big)$ with $\pi(X) = \big(\pi_0(X), \dots, \pi_M(X)\big)$
  - $\ln \frac{\pi_j(X)}{\pi_0(X)} \stackrel{\text{m}}{=} \beta_{j0} + \beta_j^\top X \;\Leftrightarrow\; \pi_j(X) \stackrel{\text{m}}{=} \frac{\exp\left(\beta_{0j} + \beta_j^\top X\right)}{1 + \sum_{j=1}^{M} \exp(\beta_0 + \beta^\top X)},\ 1 \le j \le M$

$\theta = \{\beta_{0j}, \beta_j : 1 \le j \le M\}$

Multiclass Logistic Regression (MLR)

- Two distributions
  - Model: $f_\theta(y|x) = \prod_{j=0}^{M}\{\pi_j(X)\}^{I(y=j)}$
  - Data: $\{(X_i, Y_i)\}_{i=1}^{n} \rightarrow \hat{g}$

- Estimation via D_{KL}
  - $\min_\theta D_{KL}(\hat{g}, f_\theta) + \lambda J(\theta) \rightarrow \hat{\theta}$



- The case of $M = 2$
  - $\hat{Y} = 0$ if $\beta_{10} + \beta_1^\top X < 0$ and $\beta_{20} + \beta_2^\top X < 0$
  - $\hat{Y} = 1$ if $\beta_{10} + \beta_1^\top X > 0$ and $\beta_{10} + \beta_1^\top X > \beta_{20} + \beta_2^\top X$
  - $\hat{Y} = 2$ if $\beta_{20} + \beta_2^\top X > 0$ and $\beta_{10} + \beta_1^\top X < \beta_{20} + \beta_2^\top X$

$$\ln \frac{\pi_j(X)}{\pi_0(X)} \stackrel{\mathrm{m}}{=} \beta_{j0} + \beta_j^\top X$$

Neural Network

Non-linear extension of MLR

Neural Network (NN)

- Random vector (X, Y)
  - Response $Y \in \{0,1,\dots,M\}$
  - Covariate $X = (X_1, X_2, \dots, X_p)^\top$

MLR uses linear transformations of X to model $\pi_j(X) \stackrel{\mathrm{m}}{=} \frac{\exp(\beta_{0j}+\beta_j^\top X)}{1+\sum_{l=1}^{M}\exp(\beta_{0l}+\beta_l^\top X)}$

- Bayes classifier: $\hat{Y} = \mathrm{argmax}_{0\le j\le M}\, \pi_j(X)$
  - Target: $\pi_j(X) = P(Y = j|X)$

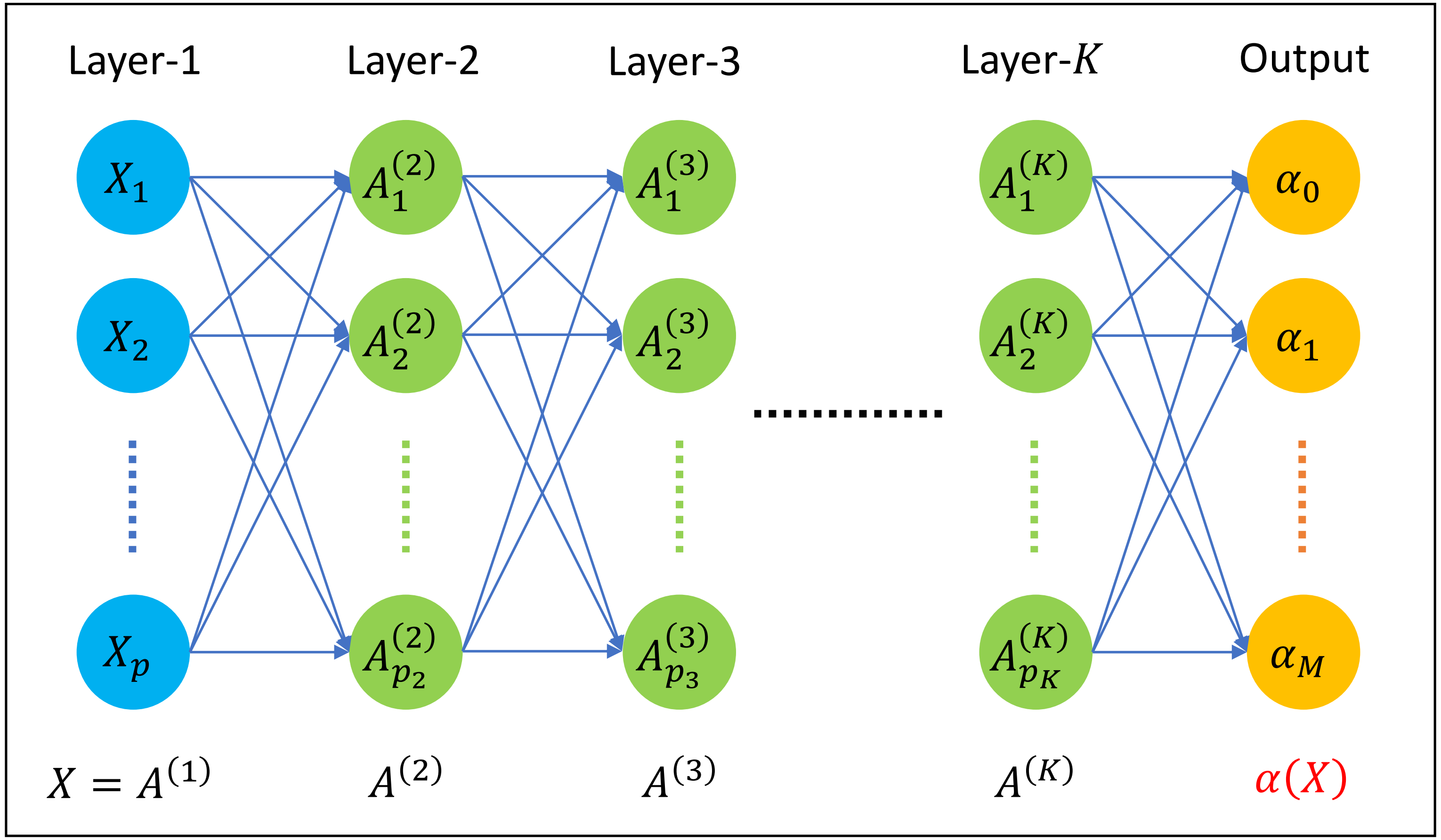
- Model
  - $Y|X \sim \text{Multinomial}(\pi(X))$ with $\pi(X) = (\pi_0(X), \dots, \pi_M(X))$
  - NN uses non-linear transformations $\alpha_j(X)$ to model $\pi_j(X) \stackrel{\mathrm{m}}{=} \frac{\exp(\alpha_j(X))}{\sum_{l=0}^{M}\exp(\alpha_l(X))}$

Neural Network (NN)

- Two distributions
  - Model: $f_\theta(y|x) = \prod_{j=0}^{M} \{\pi_j(X)\}^{I(y=j)}$
  - Data: $\{(X_i, Y_i)\}_{i=1}^{n} \rightarrow \hat{g}$
- Estimation via D_{KL}
  - $\min_\theta D_{KL}(\hat{g}, f_\theta) + \lambda J(\theta) \rightarrow \hat{\theta}$
- NN and MLR differs in the ways of modeling $\pi_j(X)$

Non-Linear Transformation $\alpha(X)$

NN with K layers



intercept: $A_1^{(k)} = 1 \ \forall\ k$

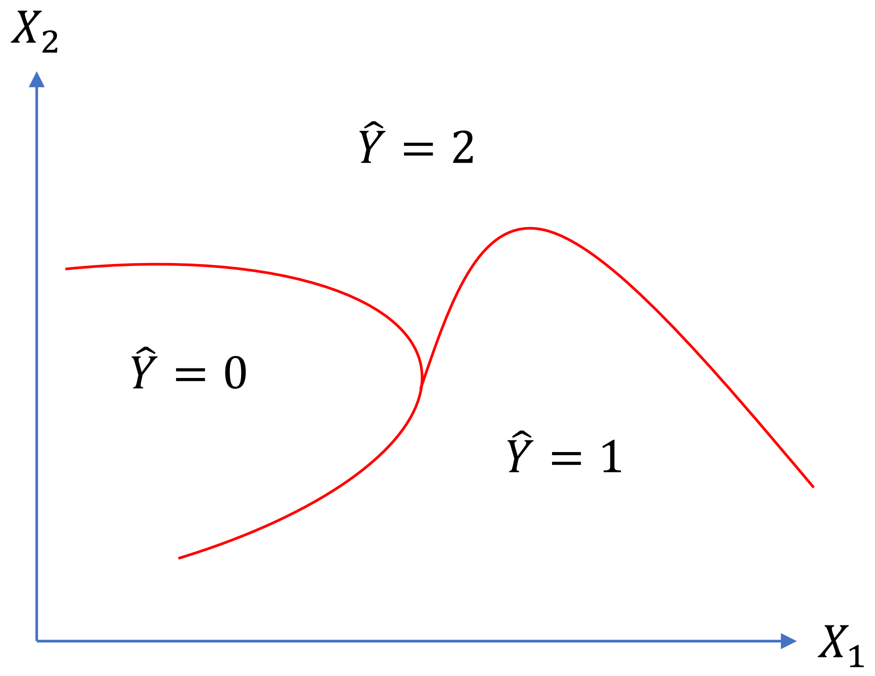
- Input: Layer-1
  - $X = A^{(1)} \in R^p$
- Hidden: Layer-2 to Layer-K
  - $A^{(k+1)} = h^{(k+1)}\left(W^{(k)} A^{(k)}\right)$
  - $W^{(k)}$: $p_{k+1} \times p_k$ matrix
  - $h^{(k+1)}$: non-linear function
- Output: $\alpha(X) = W^{(K)} A^{(K)} \in R^{M+1}$
- Parameter: $\theta = \left(W^{(1)}, \ldots, W^{(K)}\right)$

Neural Network (NN)

- NN is a non-linear extension of MLR
  - The decision boundary is non-linear
  - NN = MLR if $h^{(k)}$'s are linear functions

- Specification of NN
  - Number of layers K
  - Number of nodes p_k
  - The choice of $h^{(k)}$
  - Regularization function $J(\theta)$ and its penalty λ

- NN is usually overparameterized
  - Regularization is necessary!



Choices of $h^{(k)}$:

- tanh: $h(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- rectified linear: $h(x) = z_+$
- leaky rectified linear: $h(z) = z_+ - az_-$

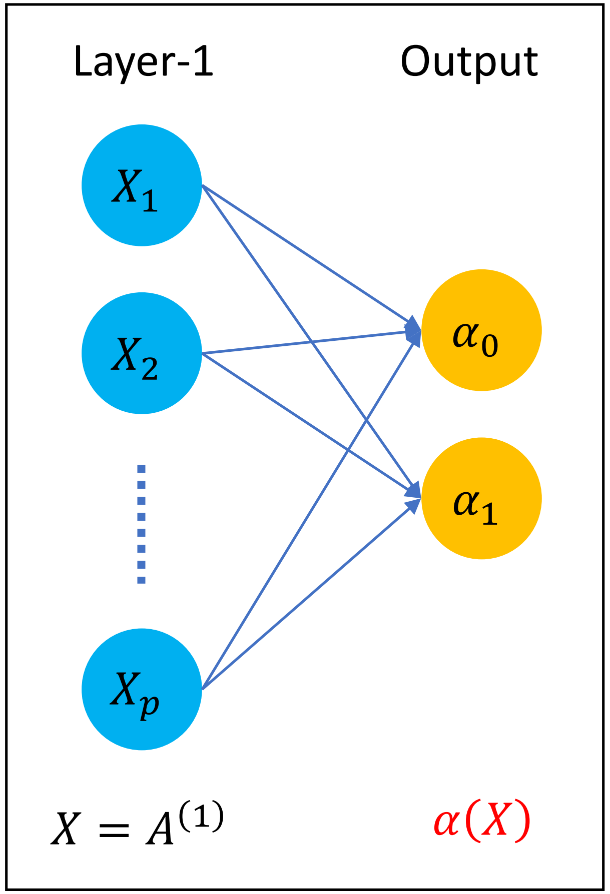
LR as a Special Case of NN

- Specification of NN
  - $K = 1$
  - $p_{K+1} = 2 \rightarrow W^{(1)} = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}_{2\times p}$

- $\alpha(X) = W^{(1)}A^{(1)} = \begin{bmatrix} W_1 X \\ W_2 X \end{bmatrix}$
  - $\pi_1(X) = \dfrac{\exp(W_2 X)}{\exp(W_1 X) + \exp(W_2 X)} = \dfrac{\exp(\beta^\top X)}{1+\exp(\beta^\top X)}$
  - $\pi_0(X) = 1 - \pi_1(X)$

$\beta^\top \triangleq W_2 - W_1$

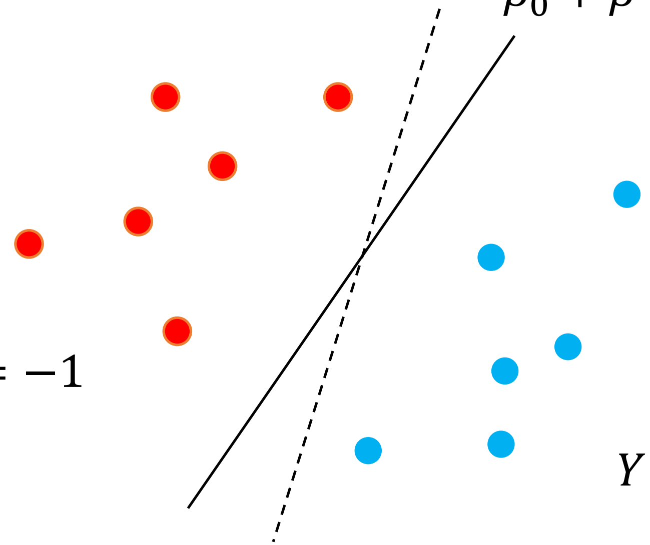
NN with 1 layer



Support Vector Machine

Linear classifier for binary Y

Support Vector Machine (SVM)



- Random vector (X, Y)
  - Binary $Y \in \{-1,1\}$
  - Covariate $X = \left(X_1, X_2, \dots, X_p\right)^\top$

- Construct $\beta_0 + \beta^\top x = 0$ that separates two groups perfectly
  - Prediction rule: $\hat{Y} = \mathrm{sign}(\beta_0 + \beta^\top x)$

$Y \in \{0,1\} \rightarrow \hat{Y} = I\{\beta_0 + \beta^\top X > 0\}$

- Non-identifiability for the target

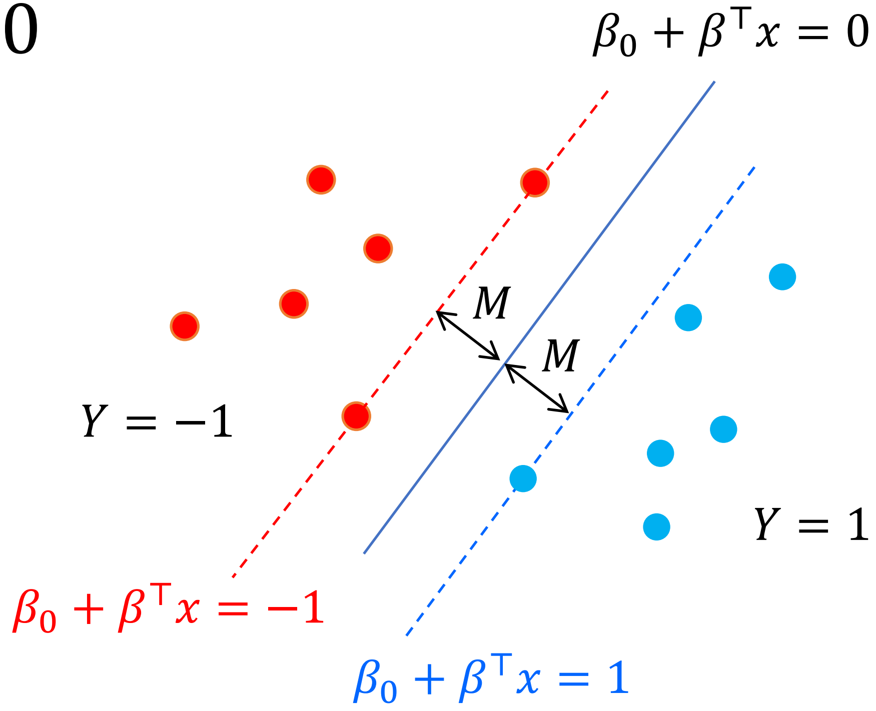
- Find the separating line with "maximum margin"

Margin of $\beta_0 + \beta^\top x = 0$

- Shifting distance without changing the classification result

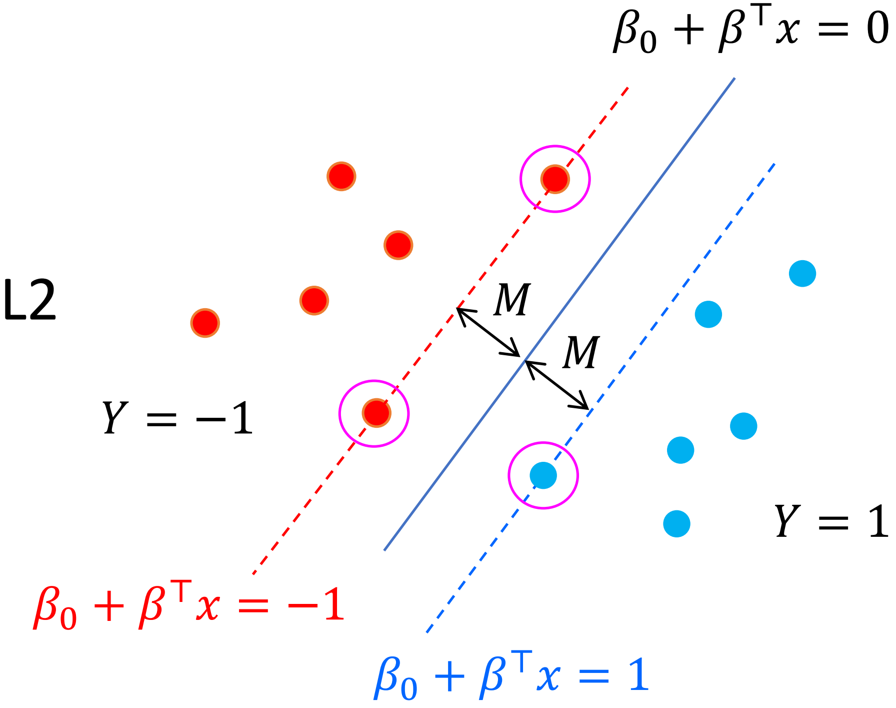
- Two separating lines parallel to $\beta_0 + \beta^\top x = 0$
  - L1: $\beta_0 + \beta^\top x = 1$
  - L2: $\beta_0 + \beta^\top x = -1$

- Margin of $\beta_0 + \beta^\top x = 0$
  - Distance between L1 and L2: $2M = \frac{2}{||\beta||}$



Support Vector Machine (SVM)

- Aim: separating line $\beta_0 + \beta^\top x = 0$ with maximum margin $2M = \frac{2}{||\beta||}$
  - Able to tolerate more violation in future application



- $\max_{\beta_0,\beta} 2M$ s.t. $\frac{1}{||\beta||} Y_i(\beta_0 + \beta^\top X_i) \geq M \; \forall \, i$
  - $\frac{1}{||\beta||} Y_i(\beta_0 + \beta^\top X_i)$ → distance from X_i to L1 or L2

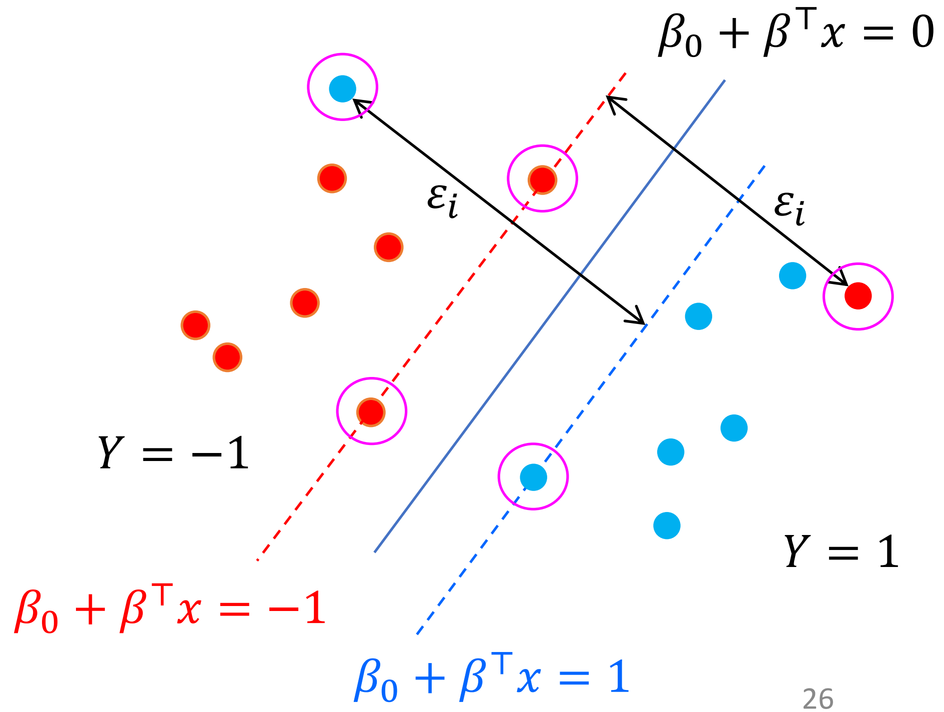
- $\min_{\beta_0,\beta} ||\beta||^2$ s.t. $Y_i(\beta_0 + \beta^\top X_i) \geq 1 \; \forall \, i$
  - $\hat{\beta} = \sum_{i \in S} \alpha_i X_i$ for some α_i with *support set* S

SVM with Soft Margin

- Non-separable case → allow for violation

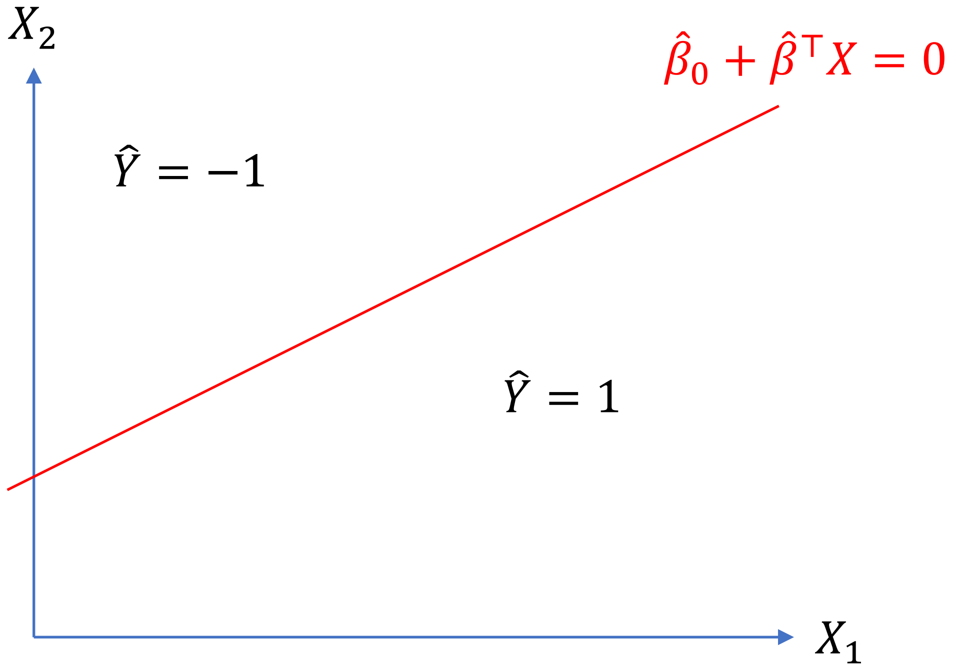
- $\min_{\beta_0,\beta} ||\beta||^2$ s.t. $Y_i(\beta_0 + \beta^\top X_i) \geq 1 - \varepsilon_i, \varepsilon_i > 0, \sum_i \varepsilon_i \leq B$
  - B: maximum total amount of violation
  - $\hat{\beta} = \sum_{i \in S} \alpha_i X_i$ with the *support set* S

- B is a tuning parameter
  - Larger B → larger support set S
  - The role of regularization



SVM with Soft Margin

- Prediction rule: $\hat{Y} = \text{sign}\left(\hat{\beta}_0 + \hat{\beta}^\top X\right)$
- Decision boundary: $\hat{\beta}_0 + \hat{\beta}^\top = 0$
- Linear classifier
- Q: Connection to Bayes classifier?



Statistical View of SVM

- $\min_{\beta_0,\beta} ||\beta||^2$ s.t. $\sum_i [1 - Y_i(\beta_0 + \beta^\top X_i)]_+ \leq B$
- $\min_{\beta_0,\beta} ||\beta||^2 + C \sum_i [1 - Y_i(\beta_0 + \beta^\top X_i)]_+$ with Lagrange multiplier C
- $\min_{\beta_0,\beta} \frac{1}{n} \sum_i [1 - Y_i(\beta_0 + \beta^\top X_i)]_+ + \frac{1}{2}\lambda ||\beta||^2$ with $\lambda = \frac{2}{nC}$
- $\min_{\beta_0,\beta} \int L(y, \beta_0 + \beta^\top x)\hat{g} + \frac{1}{2}\lambda ||\beta||^2$
  - $L(y,z) = [1 - yz]_+$ → *hinge loss*
  - Empirical distribution $\hat{g}$ of data

$\approx \min_\theta D(\hat{g}, f_\theta) + \lambda J(\theta)$

- Divergence ≈ Loss
- $D(\hat{g}, f_\theta) \propto \int L(y, \beta_0 + \beta^\top x)\hat{g}$

Statistical View of SVM

- Population loss function of SVM
  - $E[1 - Yh(X)]_+$ with $h(x) = \beta + \beta^\top x$

$\pi_1(X) = P(Y = 1|X)$

- Population target of SVM
  - $E\{[1 - Yh(X)]_+|X\} = \pi_1(X)[1 - h(X)]_+ + \{1 - \pi_1(X)\}[1 + h(X)]_+$
  - Minimized at $h^*(X) = \begin{cases} +1, \text{if } \pi_1(X) > 0.5 \\ -1, \text{if } \pi_1(X) < 0.5 \end{cases}$ → the Bayes classifier!
  - $h^*(X) = \text{sign}\big(r(X)\big)$ with log-odds ratio $r(X) = \ln \frac{\pi_1(X)}{1-\pi_1(X)}$

LR Revisit with $Y \in \{-1,1\}$

- LR criterion: $\min_\theta D_{KL}(\hat{g}, f_\theta) + \lambda J(\theta)$
  - $Y \in \{0,1\}$: $D_{KL}(\hat{g}, f_\theta) = \frac{1}{n}\sum_i \ln\left\{1 + e^{\beta_0 + \beta^\top X_i}\right\} - Y_i(\beta_0 + \beta^\top X_i) \rightarrow \hat{Y} = I\{\beta_0 + \beta^\top > 0\}$
  - $Y \in \{-1,1\}$: $D_{KL}(\hat{g}, f_\theta) = \frac{1}{n}\sum_i \ln\left\{1 + e^{-Y_i(\beta_0 + \beta^\top X_i)}\right\} \rightarrow \hat{Y} = \text{sign}(\beta_0 + \beta^\top X)$

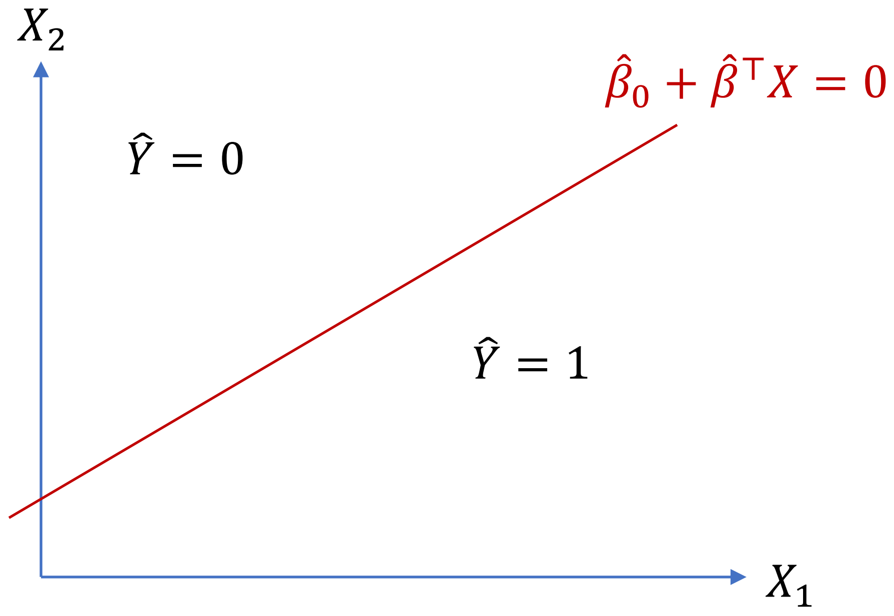
- $\min_{\beta_0, \beta} \int L(y, \beta_0 + \beta^\top x)\hat{g} + \frac{1}{2}\lambda||\beta||^2$
  - $L(y, s) = \ln(1 + e^{-ys})$: the logistic loss for $y \in \{-1,1\}$
  - Empirical distribution $\hat{g}$

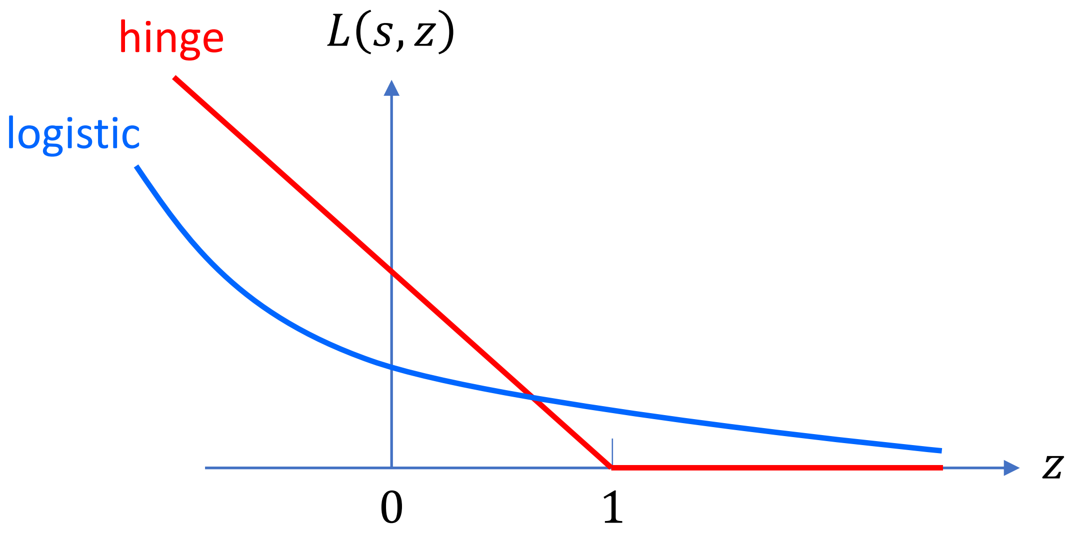
$J(\theta) = \frac{1}{2}||\beta||^2$

- Population target of LR
  - $E\left\{\ln\left(1 + e^{-Yh(X)}\right) | X\right\} = \pi_1(X) \ln\left(1 + e^{-h(X)}\right) + \{1 - \pi_1(X)\} \ln\left(1 + e^{h(X)}\right)$
  - Minimized at $h^*(X) = r(X)$

SVM vs LR

- (Linear) Bayes classifier: $\hat{Y} = \text{sign}\left(\hat{\beta}_0 + \hat{\beta}^\top X\right)$
- $\min_{\beta_0,\beta} \int L(y, \beta_0 + \beta^\top x)\hat{g} + \frac{1}{2}\lambda||\beta||^2$ with different loss
  - SVM: $L(y,z) = [1 - yz]_+$
  - LR: $L(y,s) = \ln(1 + e^{-ys})$
- Different targets regarding $r(X) = \ln\frac{\pi_1(X)}{1-\pi_1(X)}$
  - SVM: $\text{sign}\left(r(X)\right)$
  - LR: $r(X)$
- $\text{sign}\left(r(X)\right)$ suffices for classification
  - Complexity: $r(X) > \text{sign}\left(r(X)\right)$
  - Robustness in classification: SVM > LR
  - Precision of interpretation: LR > SVM





Kernel Trick

Non-linear extension of SVM and LR

Basic Idea

- An extension of linear model to its non-linear version

- $\int L(y, \beta_0 + \beta^\top x)\hat{g} + \frac{1}{2}\lambda||\beta||^2$
  - Linear model
  - L2 norm regularization: $||\beta||^2$

- Stationary equation: $\int \frac{\partial L(y, \beta_0 + \beta^\top x)}{\partial(\beta^\top X)} x \hat{g} + \lambda \beta = 0$
  - $\hat{\beta} = \sum_{i=1}^{n} \alpha_i X_i$ for some α_i

- Prediction score: $\hat{\beta}^\top x = \sum_{i=1}^{n} \alpha_i \left(X_i^\top x\right) = \sum_{i=1}^{n} \alpha_i \langle x, X_i \rangle$
  - Depends only on inner products $\langle x, X_i \rangle, i = 1, \ldots, n$

Kernel Trick

- Non-linear extension → non-linear transformation of X
  - NN → $\alpha(X)$
  - interaction terms X_1X_2, polynomial terms X_1^2 ...
- Kernel Trick → high-dimensional transformation $X \rightarrow \phi(X) \in R^{\infty}$
  - The exact form of ϕ is NOT important
  - Only need to calculate $\langle \phi(x), \phi(z) \rangle$
- Kernel function $K(x, z)$ → $\langle \phi(x), \phi(z) \rangle \triangleq K(x, z)$
  - E.g., $K(x, z) = \exp\left(-\gamma \left|\left|x - z\right|\right|^2\right)$
  - ϕ is implicitly determined by K

Non-linear Extension

- Choose a kernel $K(x,z) = \langle \phi(x), \phi(z) \rangle$
  - $\phi(x)$ is implicitly determined

- Transformed data: $\{(\phi(X_i), Y_i)\}_{i=1}^{n}$

- Bayes classifier: $\mathrm{sign}\left(\beta_0 + \beta^\top \phi(x)\right)$

- Estimation via the loss L
  - $\min_{\beta_0,\beta} \int L\left(y, \beta_0 + \beta^\top \phi(x)\right) \hat{g} + \frac{1}{2}\lambda ||\beta||^2$
  - $\hat{\beta} = \sum_{i=1}^{n} \alpha_i \phi(X_i)$ for some α_i

Non-linear Extension

- $\hat{\beta} = \sum_{i=1}^{n} \alpha_i \phi(X_i)$ and $K(x, z) = \langle \phi(x), \phi(z) \rangle$
  - Bayes classifier: $\hat{\beta}^\top \phi(x) = \sum_{i=1}^{n} \alpha_i K(x, X_i)$
  - Regularization: $||\hat{\beta}||^2 = \sum_{ij} \alpha_i \alpha_j K\left(X_i, X_j\right) = \alpha^\top K \alpha$

$K = \left[K_{ij}\right]_{n \times n}$ with $K_{ij} = K\left(X_i, X_j\right)$

- Criterion: $\min_{\alpha_0, \alpha} \int L\left(y, \alpha_0 + \sum_{i=1}^{n} \alpha_i K(x, X_i)\right) \hat{g} + \frac{1}{2} \lambda \alpha^\top K \alpha$
  - hinge loss $L(y, z) = [1 - yz]_+$ → Kernel SVM
  - logistic loss $L(y, s) = \ln(1 + e^{-ys})$ → Kernel LR

- (Non-linear) Bayes classifier: $\mathrm{sign}\left(\hat{\alpha}_0 + \sum_{i=1}^{n} \hat{\alpha}_i K(x, X_i)\right)$

- Extra tuning parameters for $K(x, z)$
  - γ of $K(x, z) = \exp\left(-\gamma ||x - z||^2\right)$

Another View of Kernel Trick

$\min_{\beta_0,\beta} \int L(y, \beta_0 + \beta^\top x)\hat{g} + \frac{1}{2}\lambda||\beta||^2$

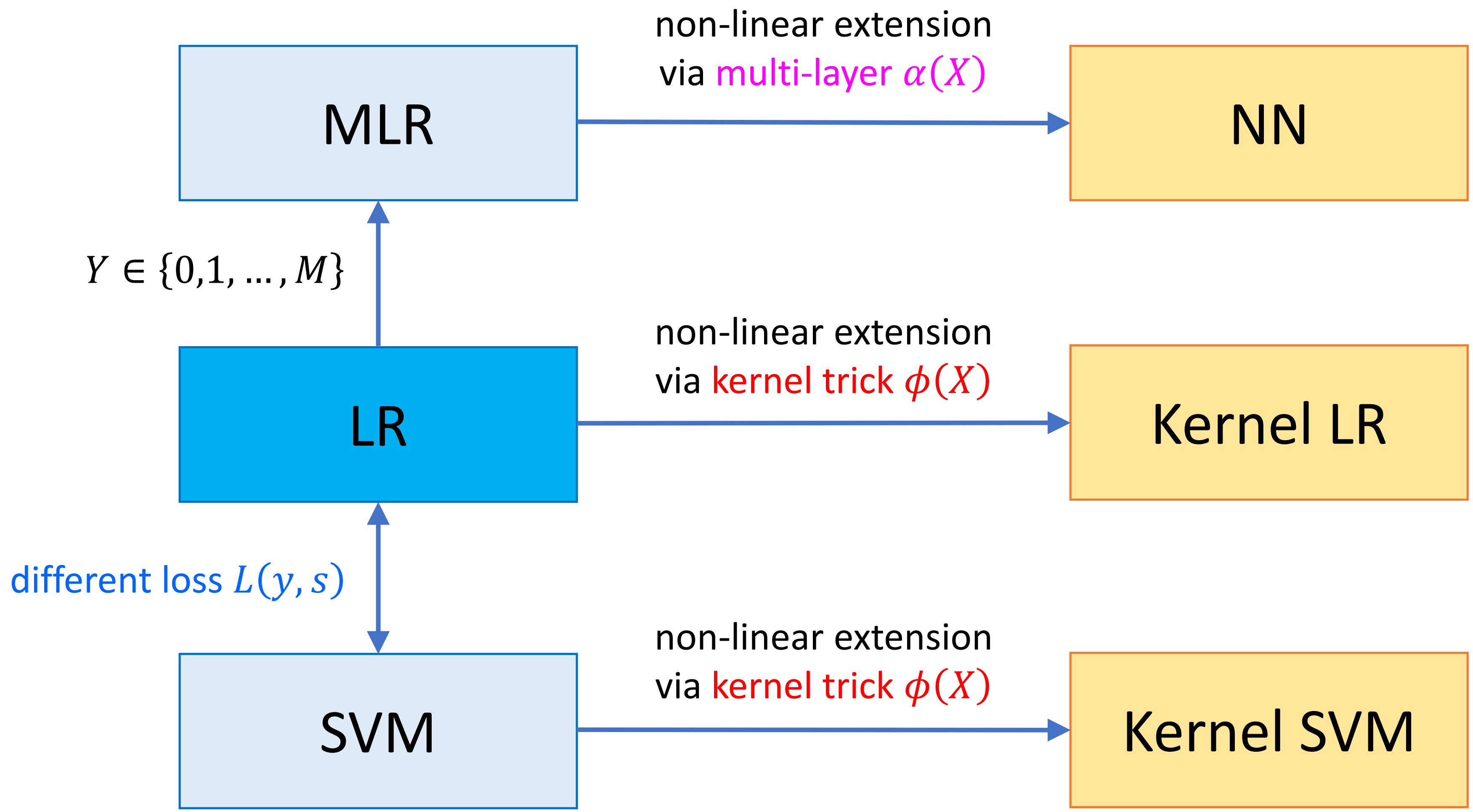
- $\min_{\alpha_0,\alpha} \int L(y, \alpha_0 + \sum_{i=1}^n \alpha_i K(x, X_i))\hat{g} + \frac{1}{2}\lambda\alpha^\top K\alpha$

- Fitting by kernel data $\{(\hat{\phi}(X_i), Y_i)\}_{i=1}^n$ with regularization $\alpha^\top K\alpha$
  - $x \to \hat{\phi}(x) = [K(x, X_1), \dots, K(x, X_n)] \in R^n$, an approximation of $\phi(x) \in R^\infty$

- $\min_{\alpha_0,\alpha} \int L\left(y, \alpha_0 + \alpha^\top \hat{\phi}(x)\right)\hat{g} + \frac{1}{2}\lambda\alpha^\top K\alpha$ → $(\hat{\alpha}_0, \hat{\alpha})$
  - Bayes classifier: $\mathrm{sign}\left(\hat{\alpha}_0 + \hat{\alpha}^\top \hat{\phi}(x)\right)$

Existing codes suffice to implement Kernel Trick!

Summary


MLR
NN
LR
Kernel LR
SVM
Kernel SVM
non-linear extension via multi-layer $\alpha(X)$
non-linear extension via kernel trick $\phi(X)$
non-linear extension via kernel trick $\phi(X)$
$Y \in \{0,1,\ldots,M\}$
different loss $L(y,s)$