L9. Machine Learning (Classification)

Contents

- **Non-Bayesian approaches**
  - Discriminative model
    - ✓ Logistic regression
    - ✓ Neural Network
  - Generative model
    - ✓ Gaussian Discriminative Analysis
    - ✓ Naïve Bayes classification
- **Full Bayesian approach for classification**
  - Bayesian Logistic regression
  - Bayesian Neural Network

Non-Bayesian vs Bayesian

- **Non-Bayesian approaches**

✓ *discriminative* probabilistic classification

$$p(y|x) = f(w^T x)$$

Directly model posterior $p(y|x)$ using parametric form

✓ *Generative* probabilistic classification

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x|y)P(y)}{\sum_{y \in Y} P(x|y)P(y)}$$

Model $P(x|y)$ and $P(y)$ and combined them in Bayes' rule

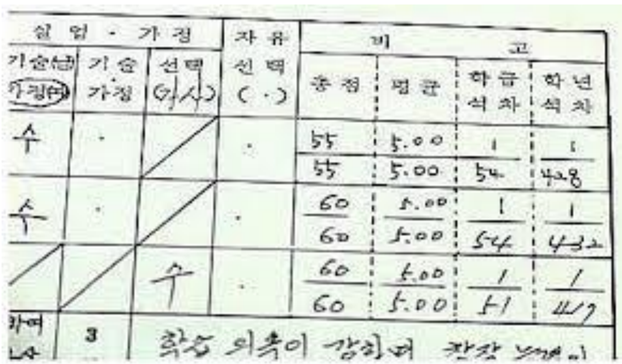
$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}}\, p(y|x)$$

- **Full Bayesian approach for classification**

1. Construct prior $p(w)$
2. Construct likelihood $p(D|w)$, where $D = \{(x_i, y_i)\}_{i=1}^m$
3. Construct posterior $p(w|D) = \frac{p(D|w)p(w)}{p(D)}$
4. Posterior predictive distribution $p(y_*|x_*, D) = \int_w p(y_*|x_*, w)p(w|D)dw$

University admission committee

High school grades



National Exam score

〈2016학년도 대학수학능력시험 성적통지표(예시)〉

수험번호	성 명	생년월일	성별	출신고교 (반 또는 졸업년도)		
12345678	홍 길 동	97.09.05.	남	한국고등학교 (9)		
구 분	국어 영역	수학 영역	영어 영역	사회탐구 영역		제2외국어 /한문 영역
	B형	A형		생활과 윤리	사회 · 문화	일본어 I
표준점수	131	137	141	53	64	69
백 분 위	93	95	97	75	93	95
등 급	2	2	1	4	2	2

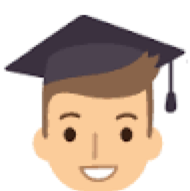
2015. 12. 2.

한국교육과정평가원장

Rejected

Student 1

- Exam: 3/10
- Grades: 4/10



?

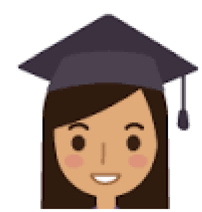
Student 2

- Exam: 7/10
- Grades: 6/10

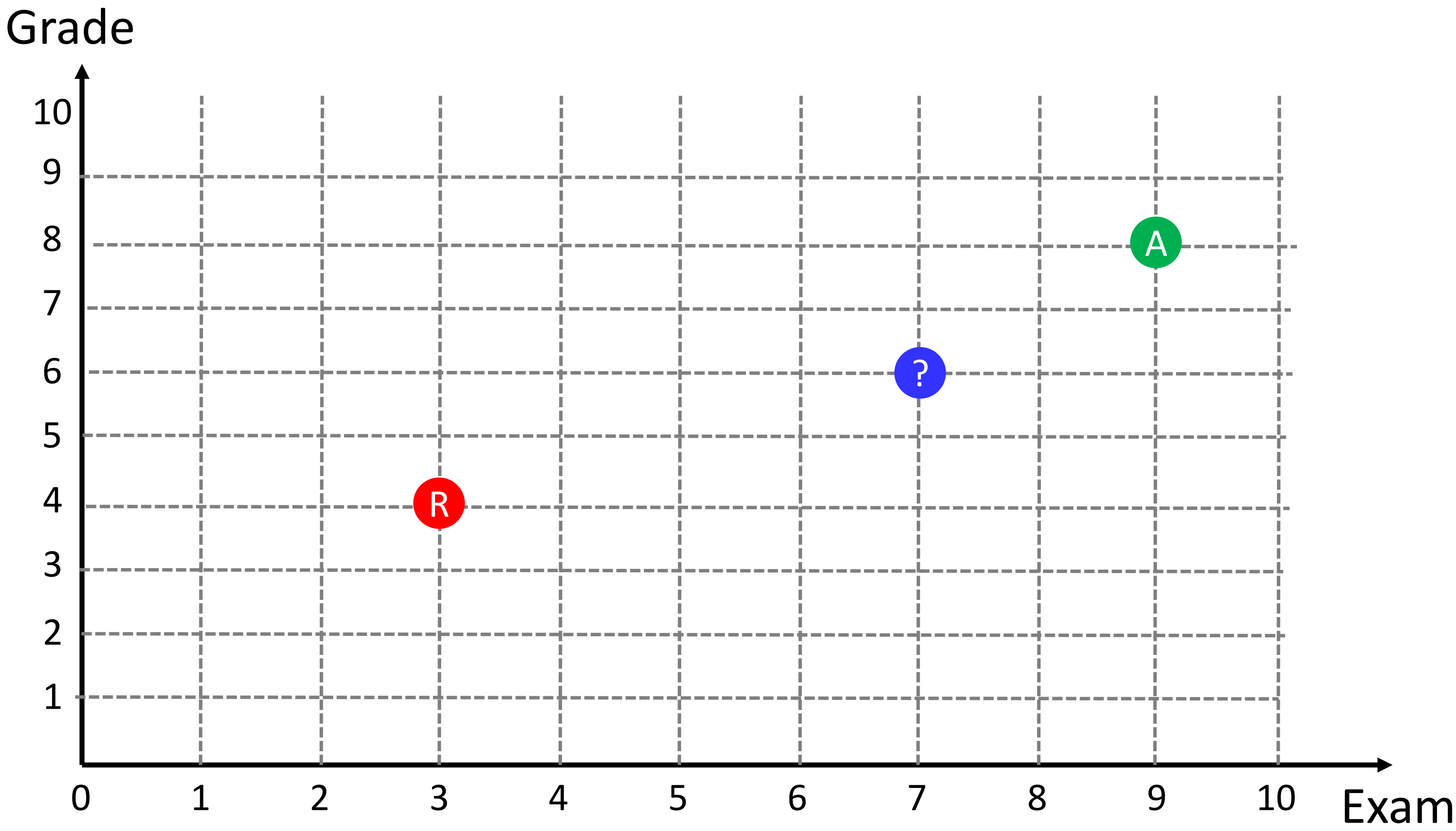
Accepted

Student 3

- Exam: 9/10
- Grades: 8/10

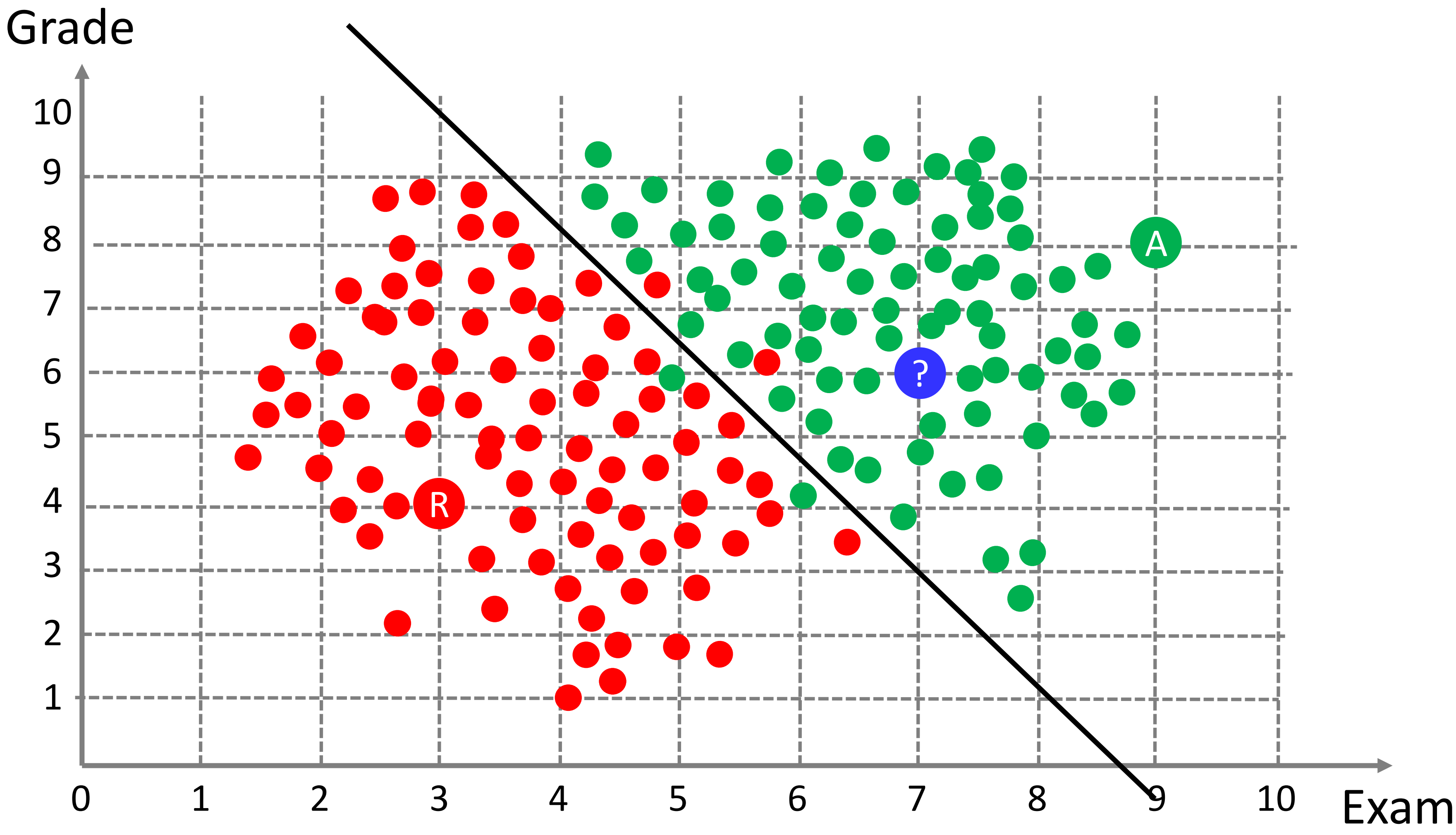


University admission committee



University admission committee

Look at the **historical data** on the admission results



Logistic regression

- Logistic regression is *discriminative* probabilistic linear classification : $p(y|x) = g(w^T x)$

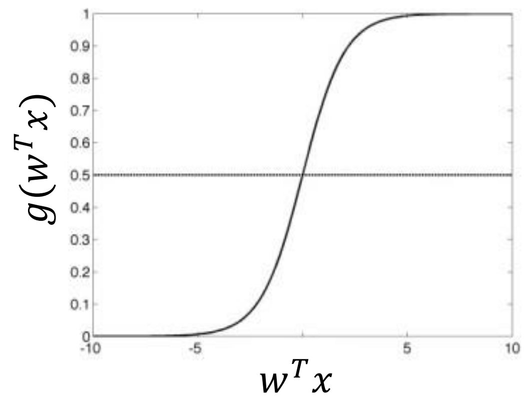
Let's denote p a probability of having $y = 1$

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = w^T x$$

$$\frac{p}{1-p} = \exp(w^T x)$$

$$p = \frac{\exp(w^T x)}{1+\exp(w^T x)} = \frac{1}{1+\exp(-w^T x)} = g(w^T x)$$

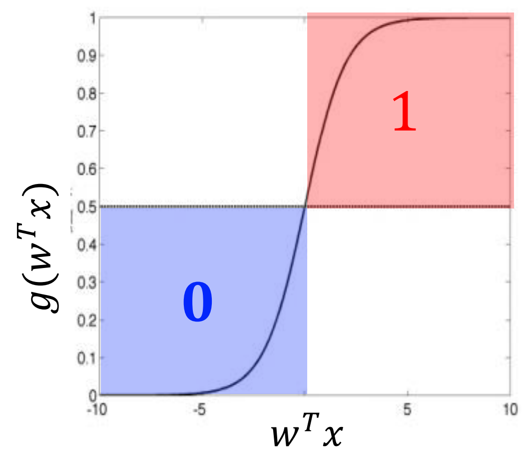
- Larger $w^T x$ →lareger→ $g(w^T x)$ → higher p for $y = 1$
- Smaller $w^T x$ →smaller→ $g(w^T x)$ → lower p for $y = 1$



$$g(z) = \frac{1}{(1+\exp(-w^T x))}$$

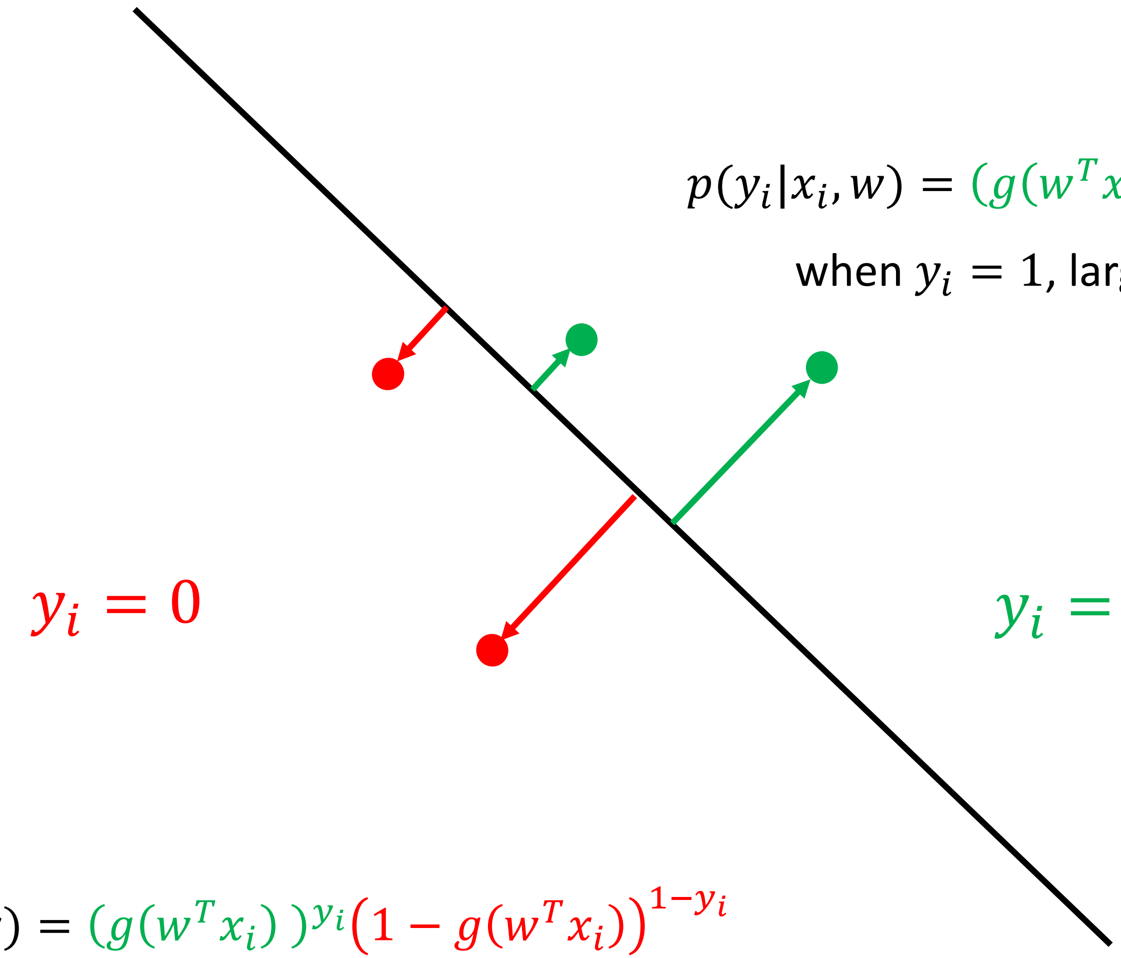
- Classification rule:

$$y = \begin{cases} 0, & \text{if } p(Y=1|x) = g(w^T x) < 0.5 \Leftrightarrow w^T x < 0 \\ 1, & \text{if } p(Y=1|x) = g(w^T x) \geq 0.5 \Leftrightarrow w^T x \geq 0 \end{cases}$$



University admission committee

How to draw **a separating line** ?



$$p(y_i|x_i, w) = \left(g(w^T x_i)\right)^{y_i} \left(1 - g(w^T x_i)\right)^{1-y_i}$$

when $y_i = 1$, larger $g(w^T x_i)$ is better

$$p(y_i|x_i, w) = \left(g(w^T x_i)\right)^{y_i} \left(1 - g(w^T x_i)\right)^{1-y_i}$$

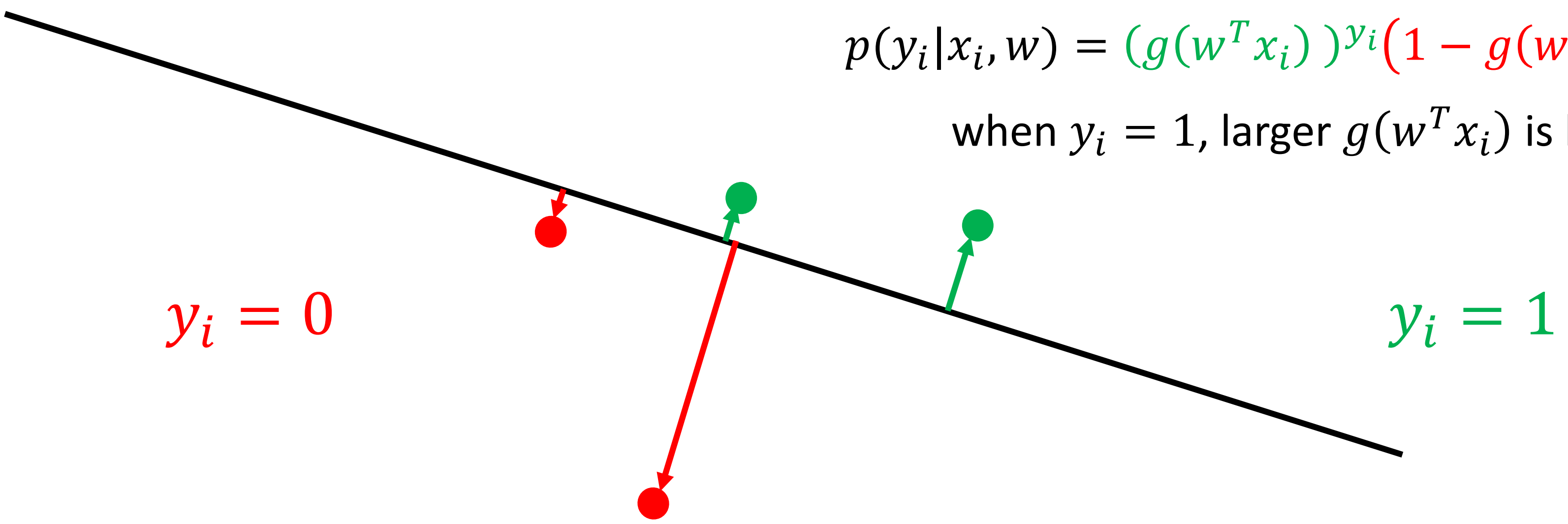
When $y_i = 0$, smaller $g(w^T x_i)$ is better

University admission committee

How to draw **a separating line** ?

$$p(y_i|x_i,w) = (g(w^T x_i))^{y_i}\left(1-g(w^T x_i)\right)^{1-y_i}$$

when $y_i = 1$, larger $g(w^T x_i)$ is better



$$p(y_i|x_i,w) = (g(w^T x_i))^{y_i}\left(1-g(w^T x_i)\right)^{1-y_i}$$

When $y_i = 0$, smaller $g(w^T x_i)$ is better

Logistic regression – objective function

- Likelihood for **a single point** (x_i, y_i) can be specified as

$$p(y_i|x_i,w) = (g(w^T x_i))^{y_i}\left(1 - g(w^T x_i)\right)^{1-y_i}$$

- Likelihood for **whole training data** (X, y) can be specified as

$$p(y|X,w) = \prod_{i}^{m} p(y_i|x_i,w) = \prod_{i=1}^{m} (g(w^T x_i))^{y_i}\left(1 - g(w^T x_i)\right)^{1-y_i}$$

Note that this is similar to the likelihood of Binomial dist.

- **Log**-likelihood

$$L(w) = \log \prod_{i}^{m} p(y_i|x_i,w) = \sum_{i=1}^{m} y_i \log g(w^T x_i) + (1 - y_i) \log\left(1 - g(w^T x_i)\right)$$

Logistic regression – learning (optimization)

- Log-likelihood

$$L(w) = \log \prod_{i}^{m} p(y_i | x_i, w) = \sum_{i=1}^{m} y_i \log g(w^T x_i) + (1 - y_i) \log\left(1 - g(w^T x_i)\right)$$

- We can find the parameters that maximizes the log-likelihood function

$$w^* = \mathrm{argmax}_w \ L(w)$$

- **Gradient ascent** algorithm

Repeat until convergence{

$w_j := w_j + \alpha \frac{\partial}{\partial w_j} L(w)$ (for every j)

}

α : learning rate

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{m} \left(y_i - g(w^T x_i)\right) x_{ij}$$

Logistic regression – learning (optimization)

- **Log**-likelihood

$$L(w) = \log \prod_{i}^{m} p(y_i|x_i, w) = \sum_{i=1}^{m} y_i \log g(w^T x_i) + (1 - y_i) \log\left(1 - g(w^T x_i)\right)$$

- We can find the parameters that maximizes the log-likelihood function

$$w^* = \operatorname{argmax}_w \ L(w)$$

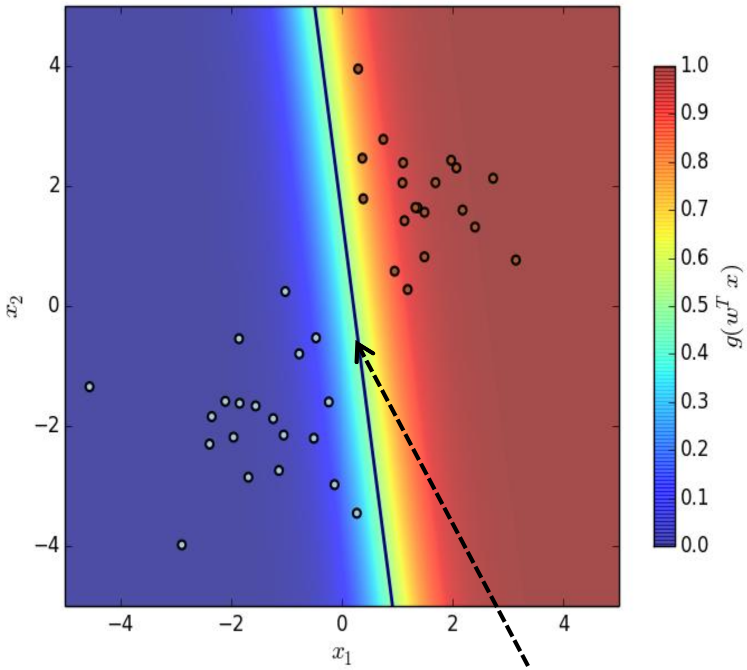
- **Stochastic gradient ascent** algorithm

Repeat until convergence{
    for $i = 1, \dots, m$ {
        $w_j := w_j + \alpha\left(y_i - g(w^T x_i)\right)x_{ij}$ (for every j)
    }
}

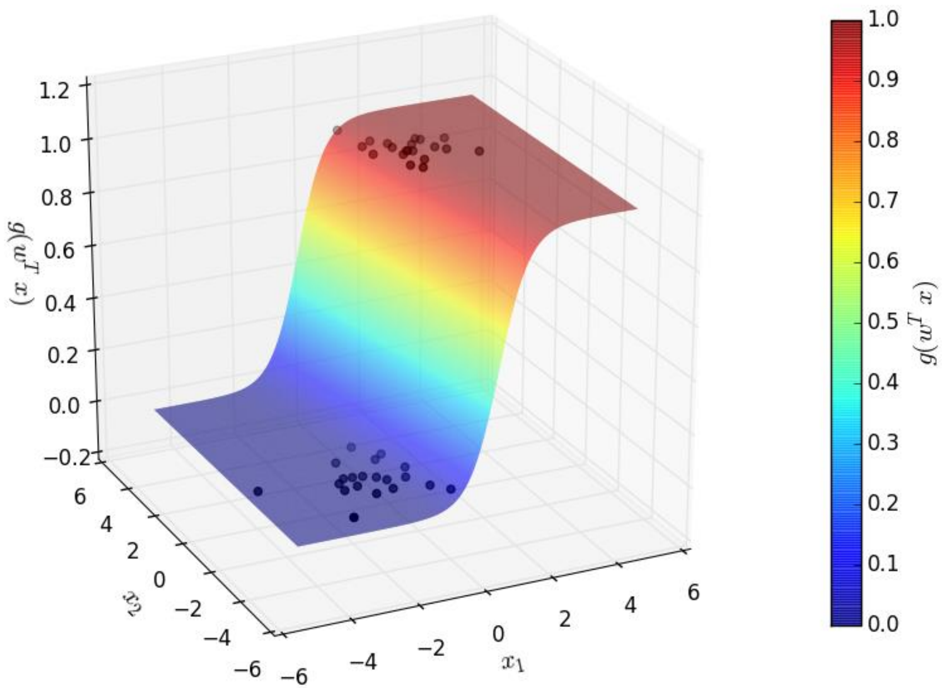
α : learning rate

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{m} \left(y_i - g(w^T x_i)\right)x_{ij} \sim \left(y_i - g(w^T x_i)\right)x_{ij}$$

Logistic regression



Classification line $\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{0}$



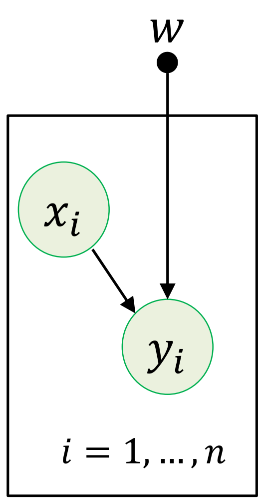
Logistic regression

Jupyter Demo Simulation

Bayesian Logistic Regression

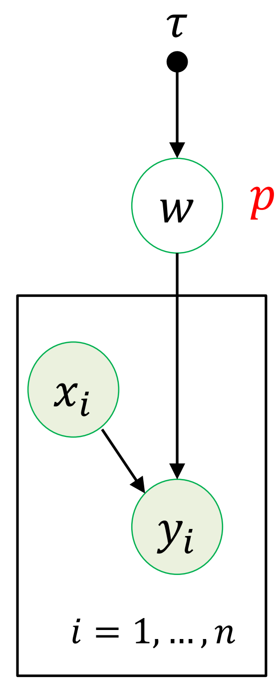
Logistic Regression

Fixed parameter
(to be determined)



Bayesian Logistic Regression

Fixed hyper-parameter



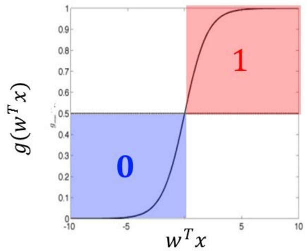
$p(w) = N(w|\mathbf{0}, \tau^2\mathbf{I})$

$$y_i = \begin{cases} 0, \text{if } g(w^T x_i) < 0.5 \Leftrightarrow w^T x_i < 0 \\ 1 \ \text{ if } g(w^T x_i) \geq 0.5 \Leftrightarrow w^T x_i \geq 0 \end{cases}$$

Bayesian Logistic Regression with Gaussian Prior (Ridge Logistic Regression)

- We have a logistic regression model :

$$p(Y=1|x) = g(w^T x) = \frac{1}{(1+\exp(-w^T x))}$$
$$p(Y=0|x) = 1 - g(w^T x)$$



- **Likelihood** can be specified as

$$p(y_i|x_i, w) = (g(w^T x_i))^{y_i} \left(1 - g(w^T x_i)\right)^{1-y_i}$$

for $y = (y_1, \dots, y_m)$

$$p(y|X, w) = \prod_{i}^{m} p(y_i|x_i, w) = \prod_{i=1}^{m} (g(w^T x_i))^{y_i} \left(1 - g(w^T x_i)\right)^{1-y_i}$$

- **Prior** on parameter w can be specified as

$$p(w_j) = N\left(w_j \middle| 0, \tau_i^2\right) = \frac{1}{\sqrt{2\pi\tau_j^2}} \exp\left(-\frac{w_j^2}{2\tau_j^2}\right)$$

for $w = (w_1, \dots, w_n)$

$$p(w) = \prod_{i=1}^{n} N\left(w_j \middle| 0, \tau_i^2\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\tau_j^2}} \exp\left(-\frac{w_j^2}{2\tau_j^2}\right)$$

✓ With the fixed variance τ_j^2 quantifying our belief that w_j is close to 0.

✓ For simple case, $\tau_j^2 = \tau^2$ for $j = 1, \dots, n$

Bayesian Logistic Regression with Gaussian Prior (Ridge Logistic Regression)

- We need to compute **the posterior**:

$$p(w|X,y) = p(y|X,w)p(w)$$
$$= \prod_{i=1}^{m} \left(g(w^T x_i)\right)^{y_i} \left(1 - g(w^T x_i)\right)^{1-y_i} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{w_j^2}{2\tau_j^2}\right)$$

$$\log p(w|X,y) = \sum_{i=1}^{m} y_i \log g(w^T x_i) + (1-y_i) \log\left(1 - g(w^T x_i)\right) + n\log\left(\frac{1}{\sqrt{2\pi\tau^2}}\right) - \sum_{j=1}^{n} \frac{w_j^2}{2\tau_j^2}$$

- The **MAP** estimate of w is then simply

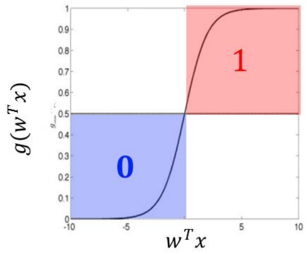
$$\hat{w} = \operatorname*{argmax}_w p(w|X,y)$$
$$= \operatorname*{argmax}_w \log p(w|X,y)$$
$$= \operatorname*{argmax}_w \underbrace{\sum_{i=1}^{m} y_i \log g(w^T x_i) + (1-y_i) \log\left(1 - g(w^T x_i)\right)}_{\text{Data fitness}} \underbrace{- \lambda \|w\|_2^2}_{\text{complexity}}$$

Bayesian Logistic Regression with Laplace Prior (Lasso Logistic Regression)

- We have a logistic regression model :

$$p(Y=1|x) = g(w^T x) = \frac{1}{(1+\exp(-w^T x))}$$

$$p(Y=0|x) = 1 - g(w^T x)$$



- **Likelihood** can be specified as

$$p(y_i|x_i,w) = (g(w^T x_i))^{y_i}\left(1-g(w^T x_i)\right)^{1-y_i}$$

for $y = (y_1, \dots, y_m)$

$$p(y|X,w) = \prod_i^m p(y_i|x_i,w) = \prod_{i=1}^m (g(w^T x_i))^{y_i}\left(1-g(w^T x_i)\right)^{1-y_i}$$

- **Prior** on parameter w can be specified using Laplacian as

$$p(w_j) = \frac{\lambda_j}{2}\exp(-\lambda_j|w_j|)$$

for $w = (w_1, \dots, w_n)$

$$p(w) = \prod_{j=1}^n \frac{\lambda_j}{2}\exp(-\lambda_j|w_j|)$$

✓ With the fixed variance λ_j quantifying our belief that w_j is close to 0.

✓ For simple case, $\lambda_j = \lambda$ for $j = 1, \dots, n$

Bayesian Logistic Regression with Laplace Prior (Lasso Logistic Regression)

- We need to compute **the posterior**:

$$p(w|X,y) = p(y|X,w)p(w)$$
$$= \prod_{i=1}^{m} (g(w^T x_i))^{y_i} (1 - g(w^T x_i))^{1-y_i} \prod_{j=1}^{n} \frac{\lambda_j}{2} \exp(-\lambda_j |w_j|)$$

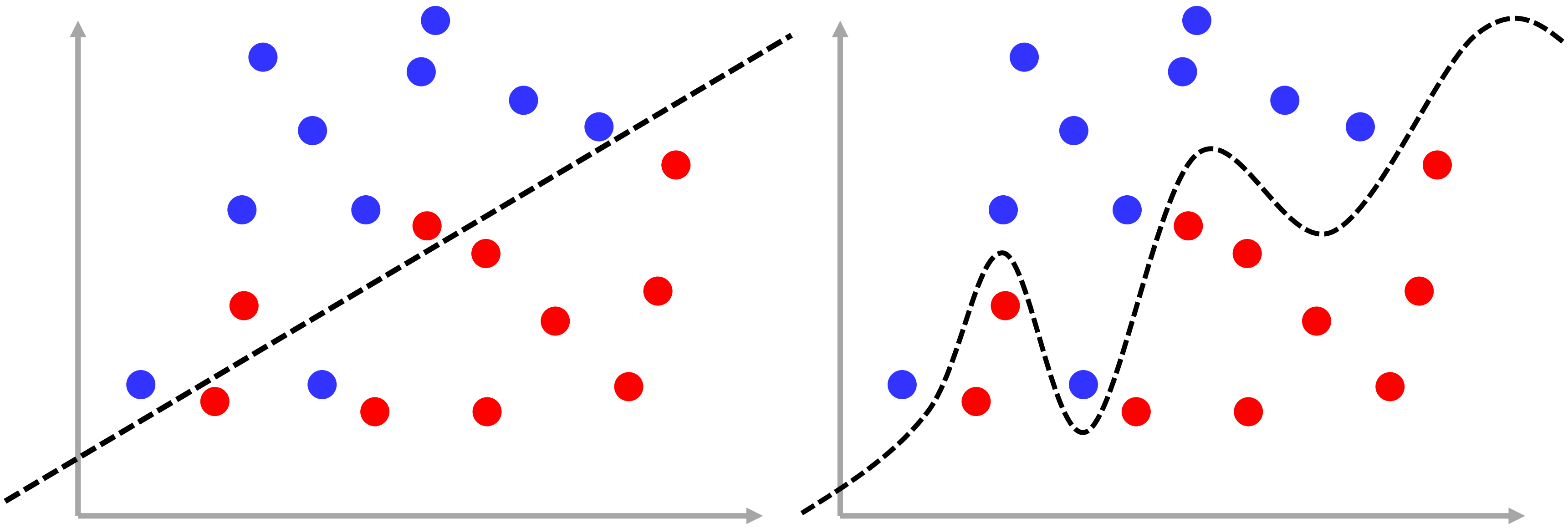
$$\log p(w|X,y) = \sum_{i=1}^{m} y_i \log g(w^T x_i) + (1-y_i) \log(1 - g(w^T x_i)) + n\log\left(\frac{\lambda}{2}\right) - \lambda \sum_{j=1}^{n} |w_j|$$

- The **MAP** estimate of w is then simply

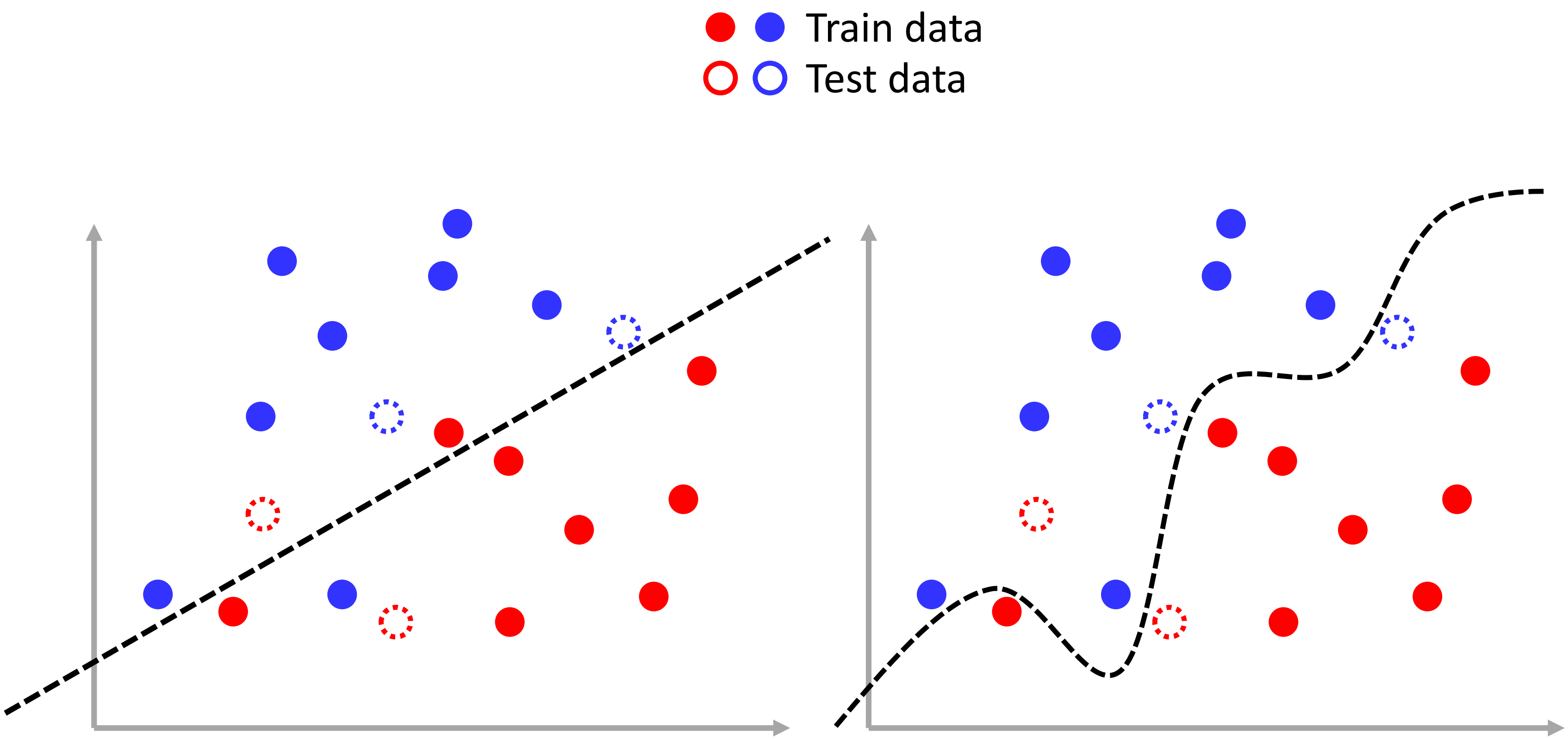
$$\hat{w} = \underset{w}{\operatorname{argmax}}\, p(w|X,y)$$
$$= \underset{w}{\operatorname{argmax}} \log p(w|X,y)$$
$$= \underset{w}{\operatorname{argmax}} \underbrace{\sum_{i=1}^{m} y_i \log g(w^T x_i) + (1-y_i) \log(1 - g(w^T x_i))}_{\text{Data fitness}} \underbrace{- \lambda \sum_{j=1}^{n} |w_j|}_{\text{Complexity (sparsity)}}$$

Model Selection and Evaluation

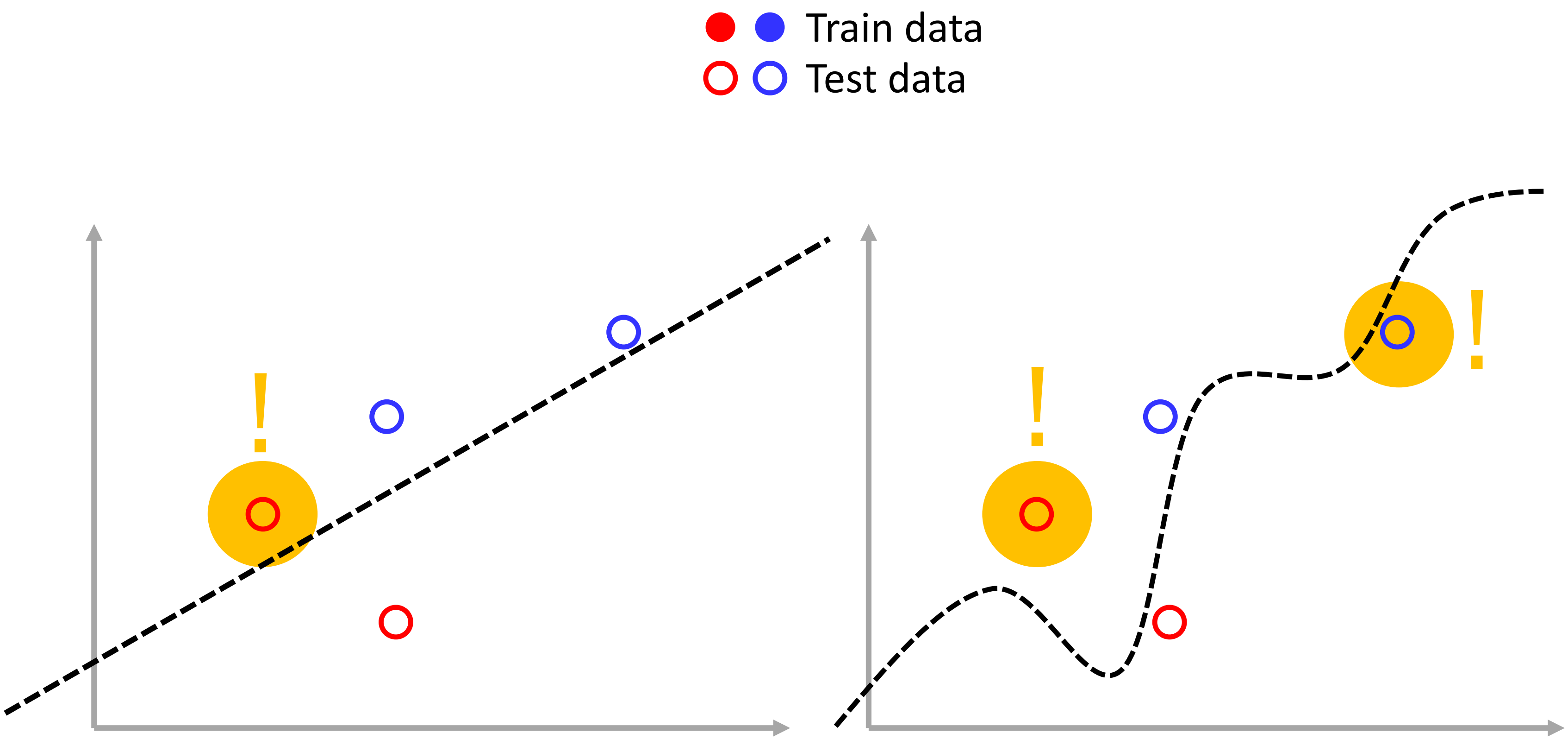
Which model is better



Which model is better



Which model is better

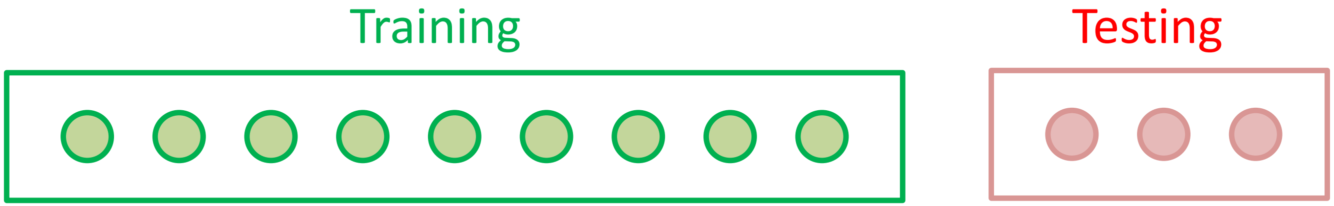


$$\text{Accuracy} = \frac{3}{4} \qquad \text{Accuracy} = \frac{2}{4}$$

Golden rule for machine learning:

Never use test data to train your model!

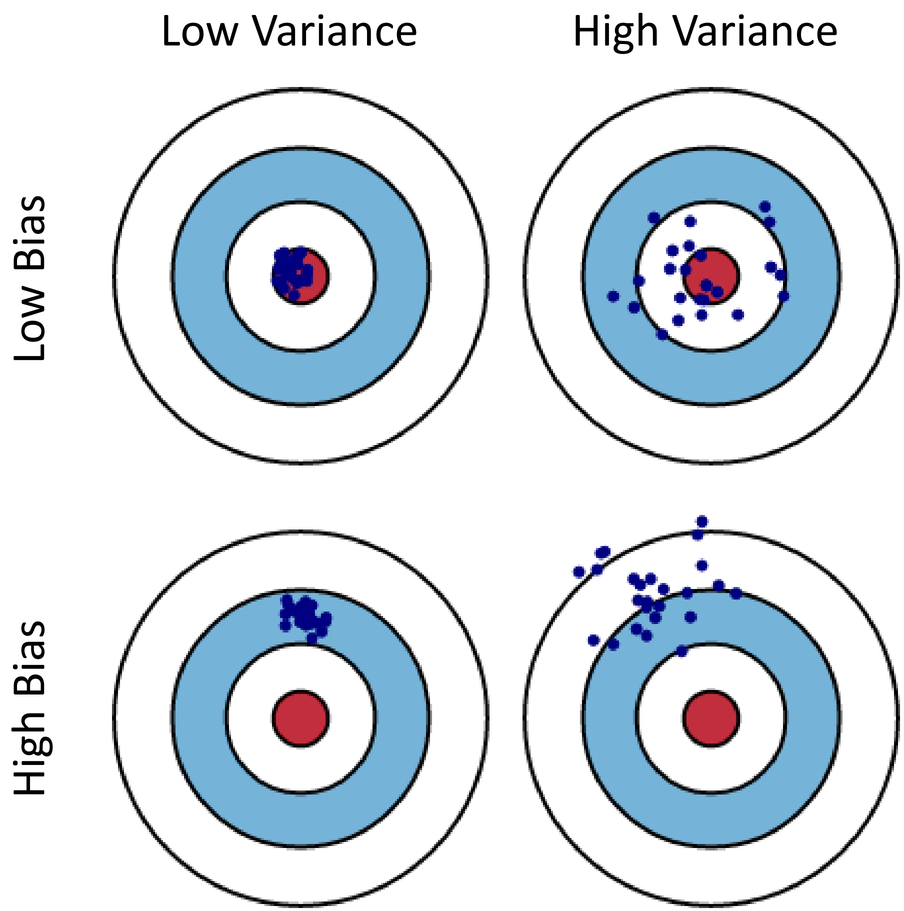
How do we not 'lose' the training data?



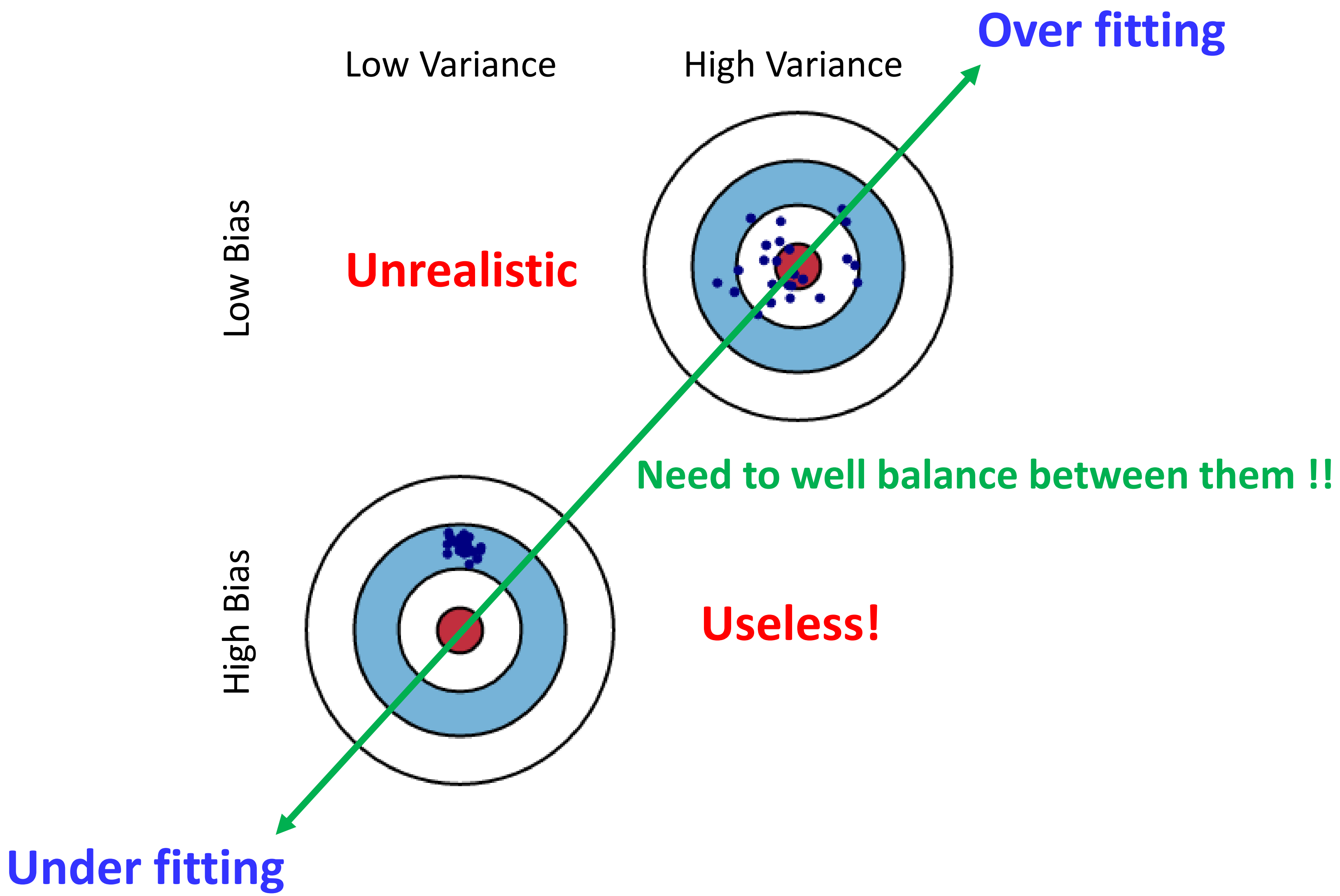
How do we not ‘lose’ the training data?

K-Fold Cross Validation

Under fitting and Over fitting : The Bias and Variance Trade off

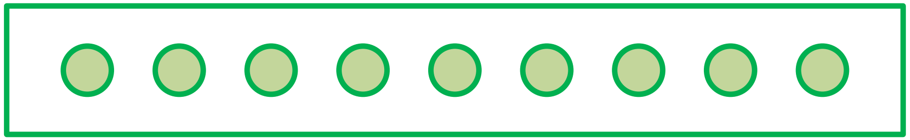


Under fitting and Over fitting : The Bias and Variance Trade off



Model selection and training

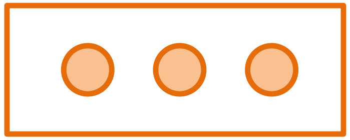
Training set



Training the model

- Fit the model parameters

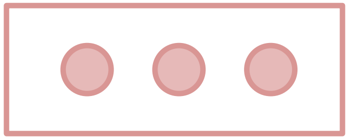
Validation set



Make decision about the model

- Select hyper parameters
  - Degree
  - Features,
  - Structures…

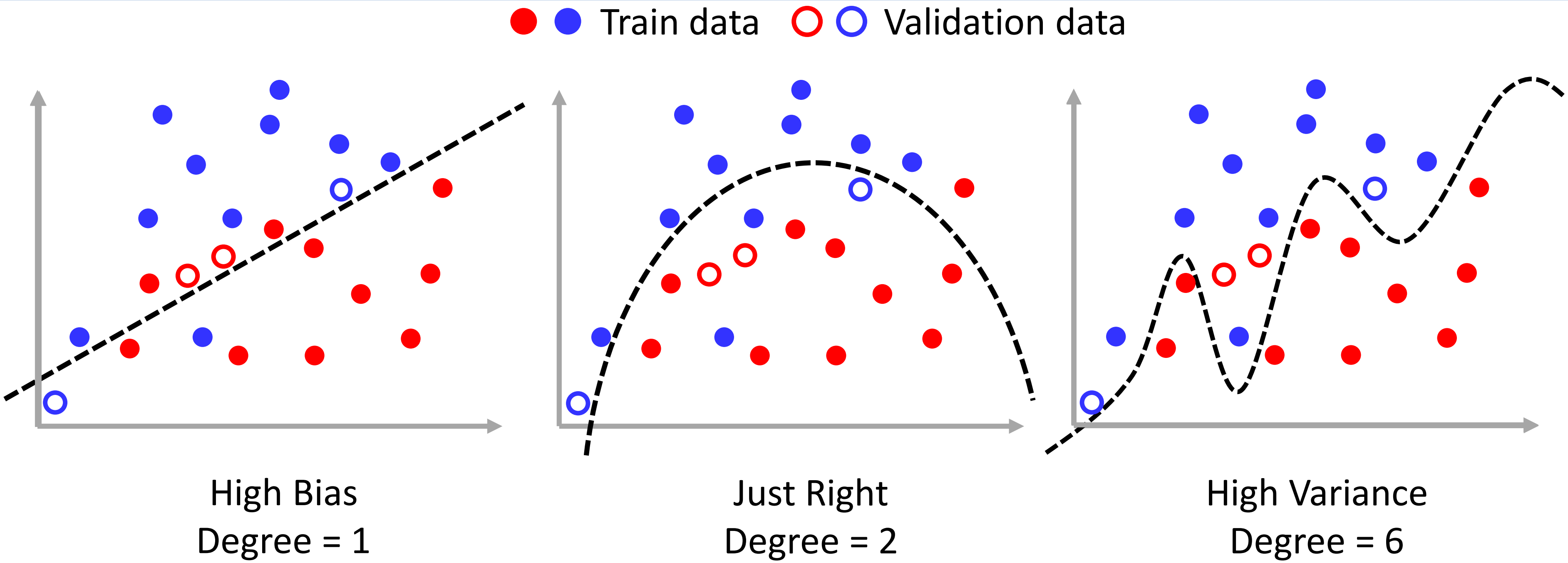
Test set



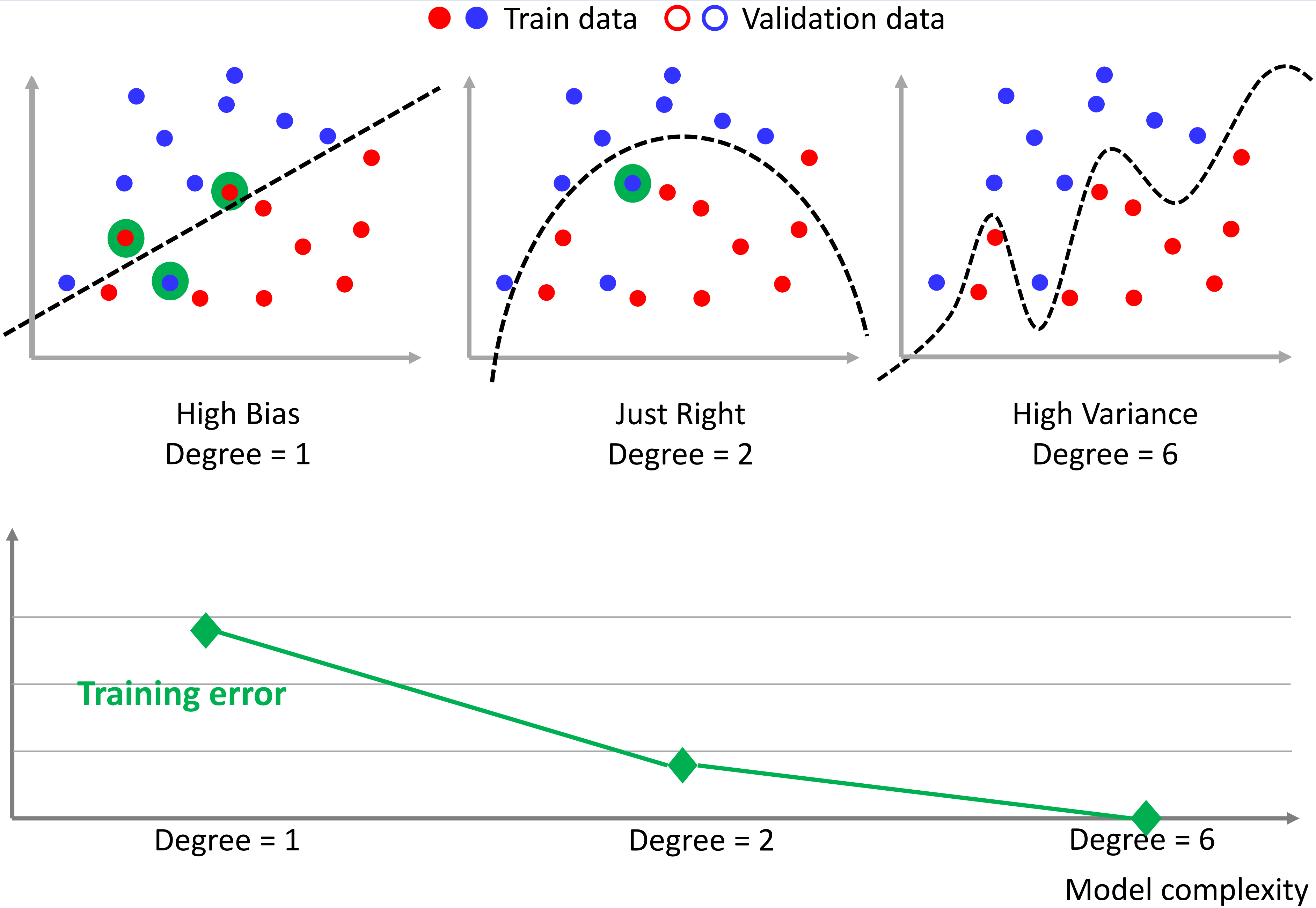
Final testing

- Never make decision based on test set
- its just for evaluation!

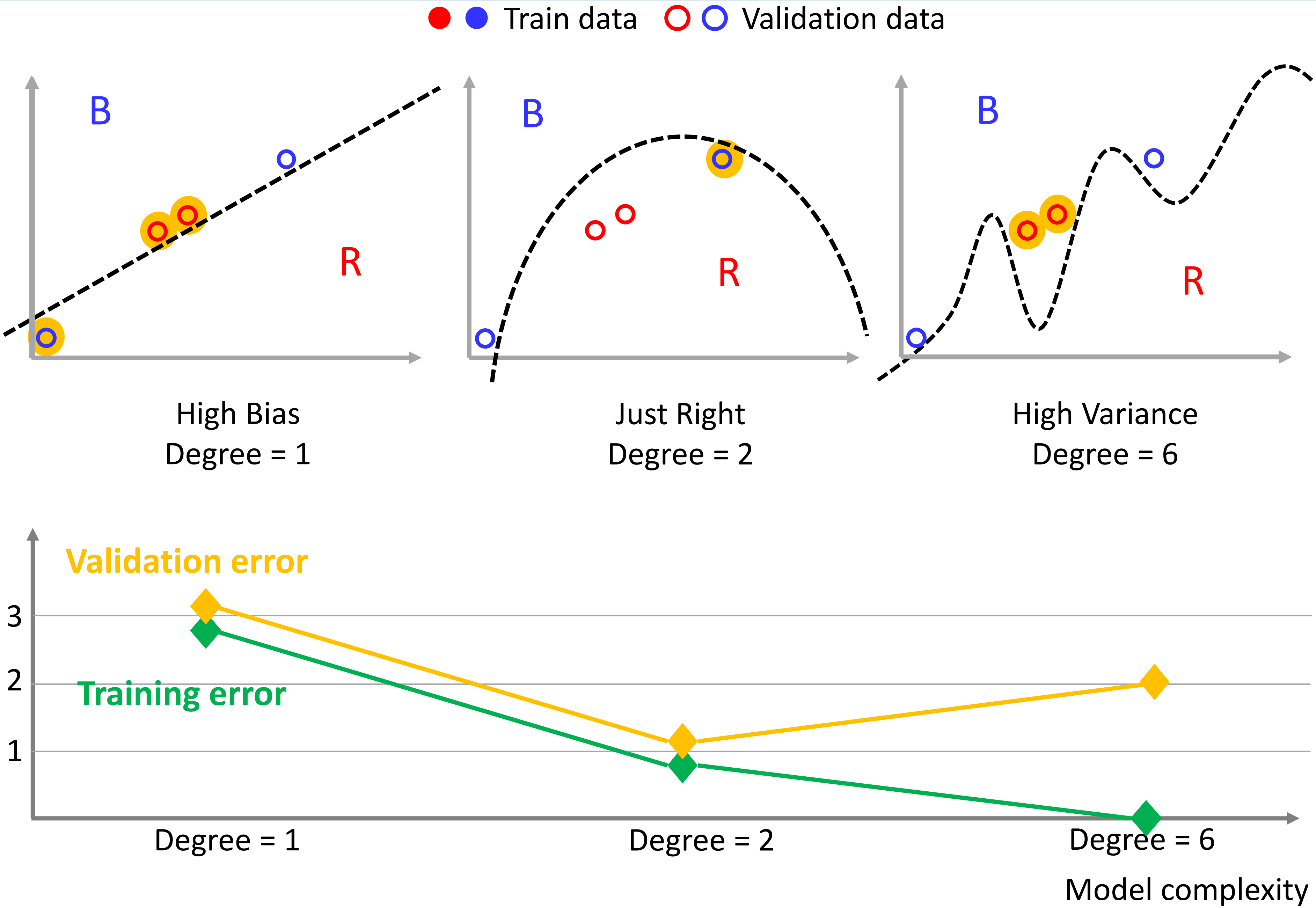
Model Complexity Graph



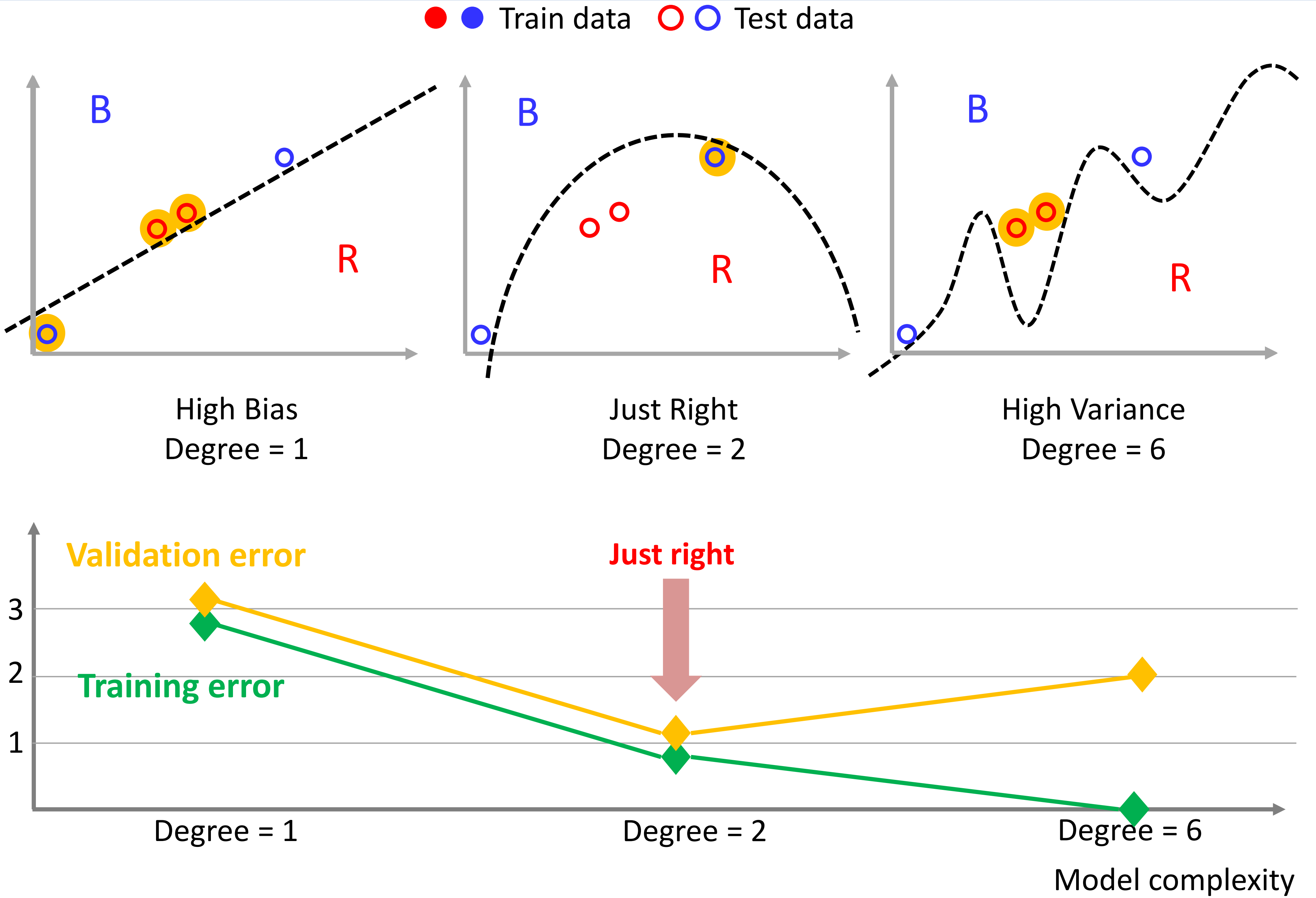
Model Complexity Graph



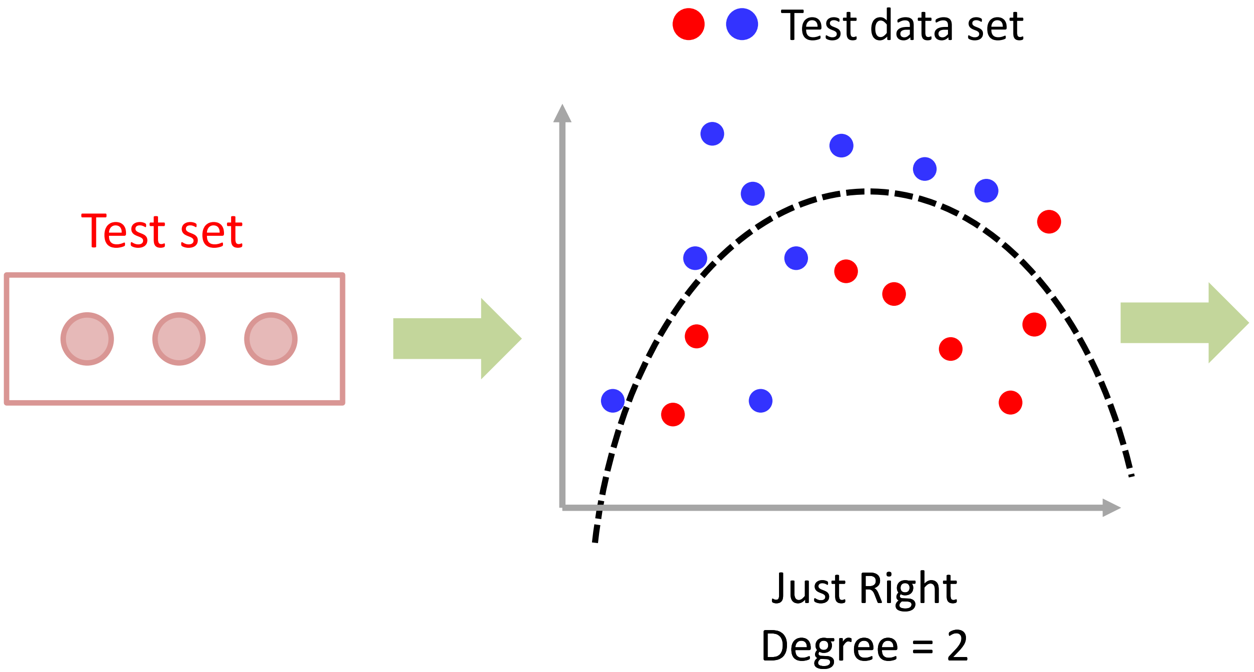
Model Complexity Graph



Model Complexity Graph



Model Complexity Graph



Evaluate performance:

- accuracy,
- Precision
- Recall
- etc.

Bayesian Logistic Regression with Gaussian Prior (Ridge Logistic Regression)

Jupyter Demo Simulation

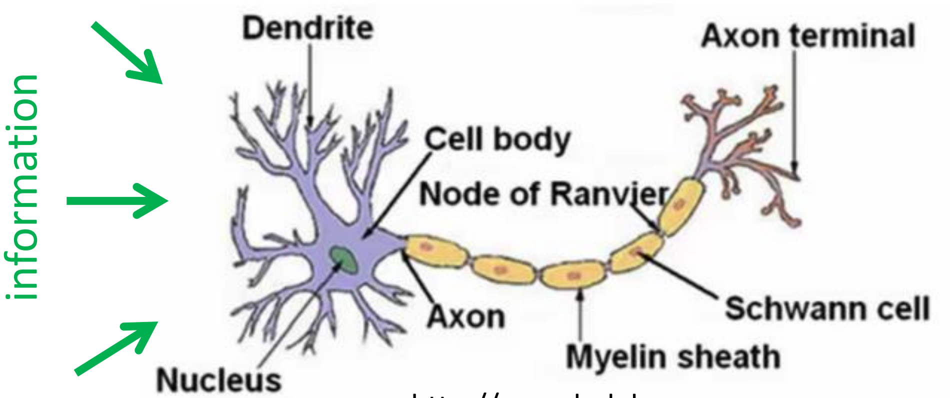
Fully Bayesian Logistic Regression with MCMC algorithm

Jupyter Demo Simulation (working on)

Neural Network

Concept

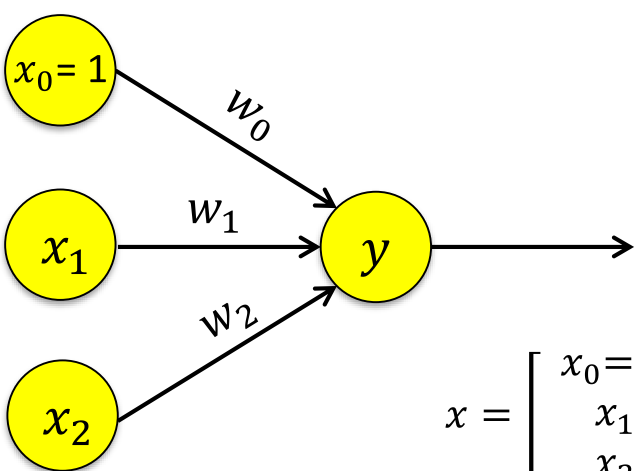
Neuron



<http://www.holehouse.org>

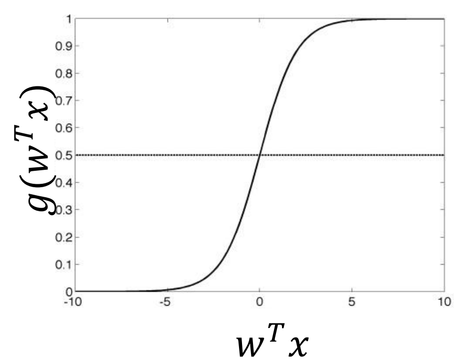
Dendrite: receive signal from multiple neurons
Cell body: Process signal
Axon: Send signal to other neuron

Logistic regression mimics the functionality of a single neuron



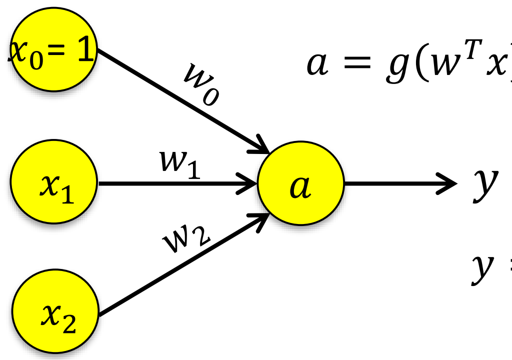
$$y = g(w^T x) = \frac{1}{(1 + \exp(-w^T x))}$$

$$x = \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \end{bmatrix}, \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$



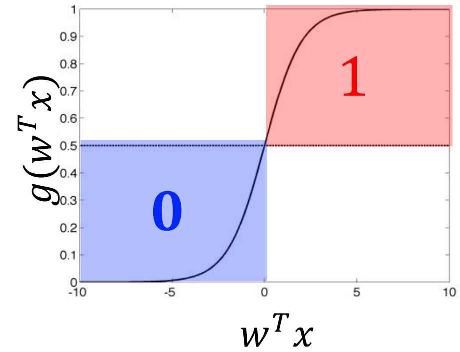
Concept

Classification with logistic regression

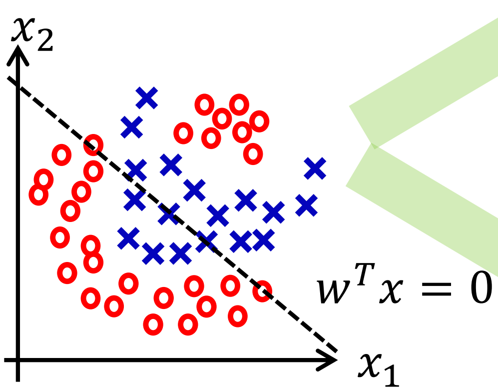


$$a = g(w^T x) = \frac{1}{(1 + \exp(-w^T x))}$$

$$y = h_w(x) = \begin{cases} 1, \text{if } a \geq 0.5 \Rightarrow w^T x \geq 0 \\ 0, \text{if } a < 0.5 \Rightarrow w^T x < 0 \end{cases}$$

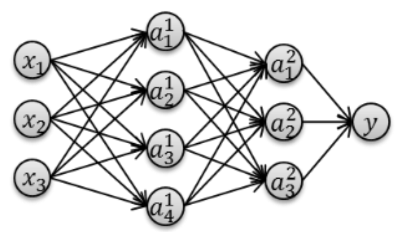


How to obtain non-linear decision boundaries?

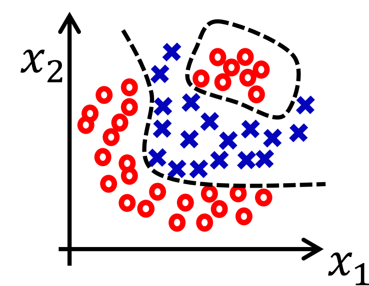


Linear decision boundary by single logistic regression

Use multilayer neural networks



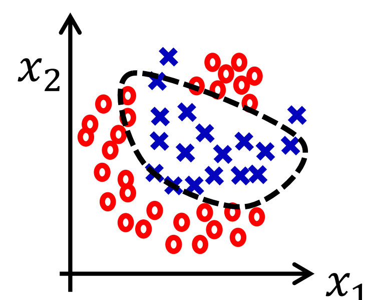
Any continuous function can be approximated well with a growing number of hidden unis.



Use non-linear feature mapping

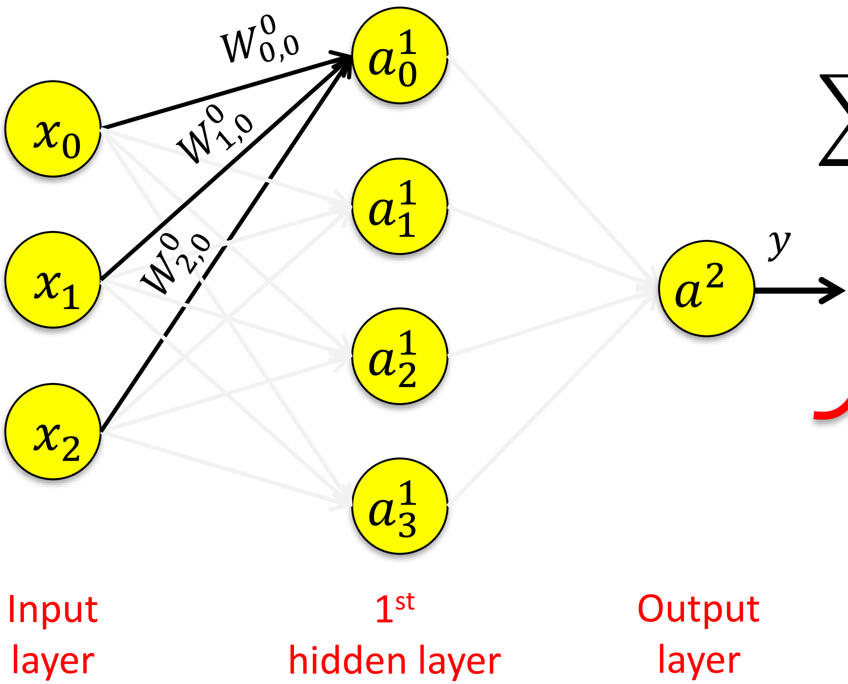
$$\phi : \chi \mapsto \mathcal{H}$$

e.g., $\phi(x) = (x_1, x_2, \sqrt{2} x_1 x_2, x_1^2, x_2^2)$



Use kernel method (simplify the computation)

Concept



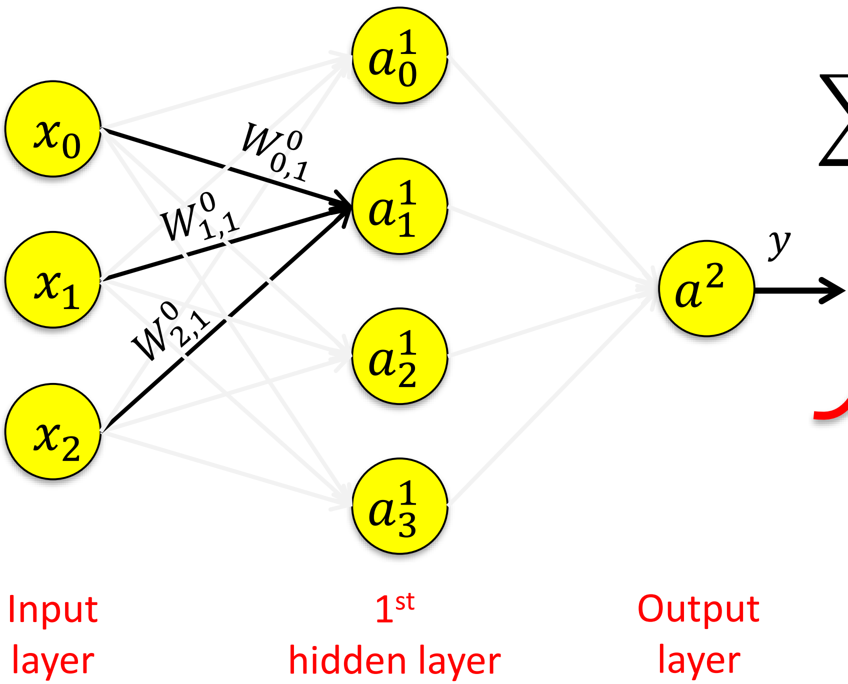
$\sum$ Weighted sum of the previous node values:

$$z^1_0 = W^0_{0,0}x_0 + W^0_{1,0}x_1 + W^0_{2,0}x_2$$

Logistic transform:

$$a^1_0 = g(z^1_0) = g\left(W^0_{0,0}x_0 + W^0_{1,0}x_1 + W^0_{2,0}x_2\right)$$

Concept

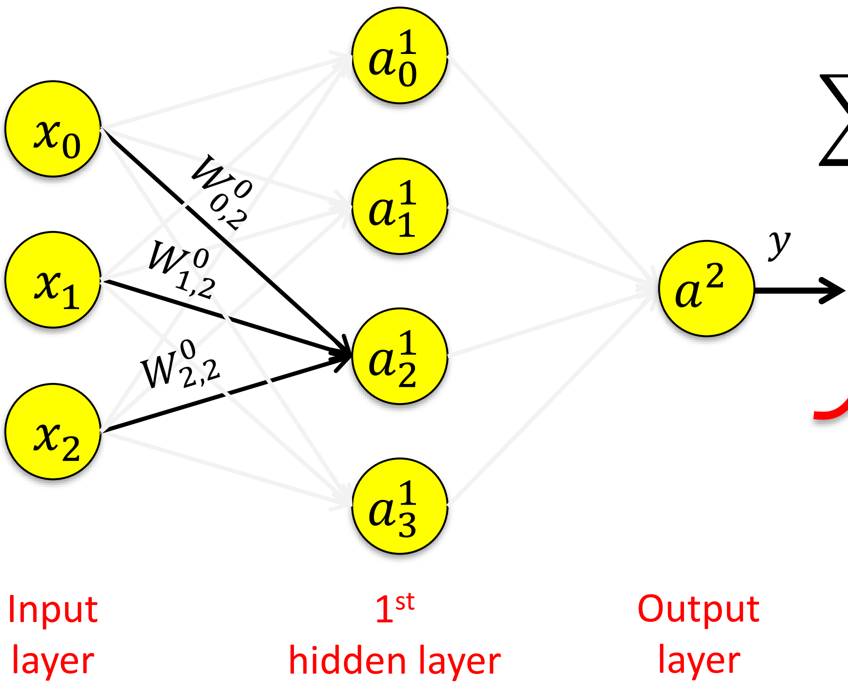


$\sum$ Weighted sum of the previous node values:

$$z_1^1 = W_{0,1}^0 x_0 + W_{1,1}^0 x_1 + W_{2,1}^0 x_2$$

Logistic transform:

$$a_1^1 = g(z_1^1) = g\left(W_{0,1}^0 x_0 + W_{1,1}^0 x_1 + W_{2,1}^0 x_2\right)$$



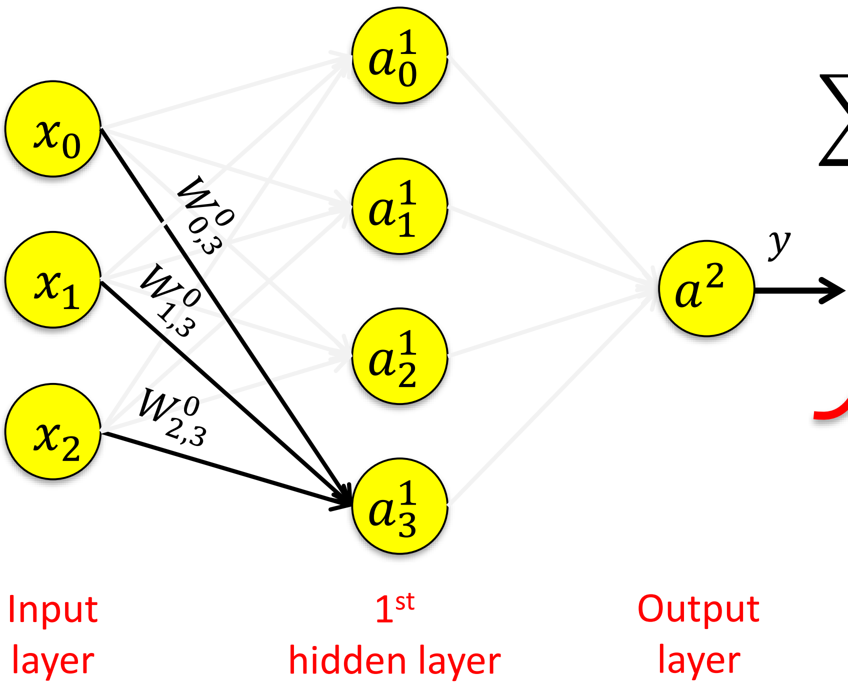
$\sum$ Weighted sum of the previous node values:

$$z_2^1 = W_{0,2}^0 x_0 + W_{1,2}^0 x_1 + W_{2,2}^0 x_2$$

Logistic transform:

$$a_2^1 = g(z_2^1) = g\left(W_{0,2}^0 x_0 + W_{1,2}^0 x_1 + W_{2,2}^0 x_2\right)$$

Concept



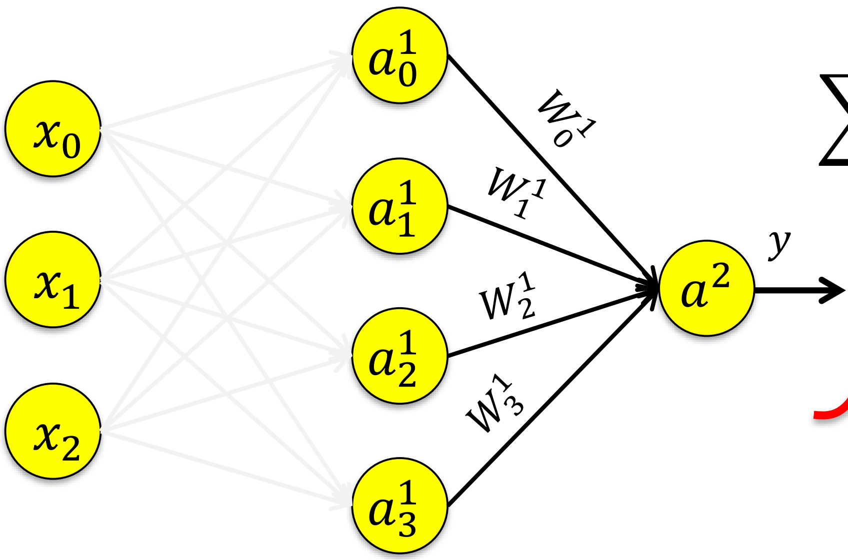
$\sum$ Weighted sum of the previous node values:

$$z^1_3 = W^0_{0,3}x_0 + W^0_{1,3}x_1 + W^0_{2,3}x_2$$

Logistic transform:

$$a^1_3 = g(z^1_3) = g\left(W^0_{0,3}x_0 + W^0_{1,3}x_1 + W^0_{2,3}x_2\right)$$

Concept



Input layer

1st hidden layer

Output layer

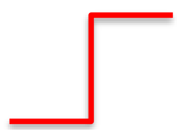
$\sum$ Weighted sum of the previous node values:

$$z^2 = W_0^1 a_0^1 + W_1^1 a_1^1 + W_2^1 a_2^1 + W_3^1 a_3^1$$

Logistic transform:

$$a^2 = g(z^2)$$

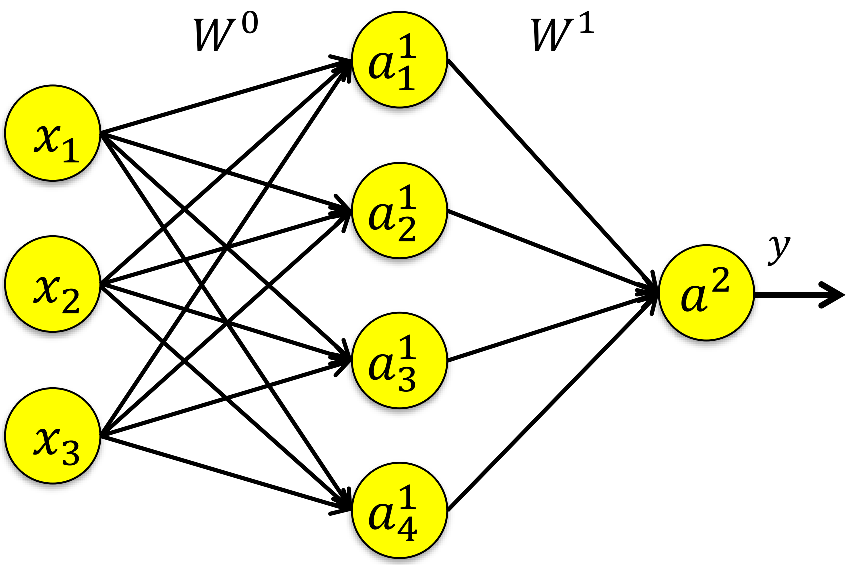
$$= g(W_0^1 a_0^1 + W_1^1 a_1^1 + W_2^1 a_2^1 + W_3^1 a_3^1)$$



Output node:

$$y = \begin{cases} 1, \text{if } a^2 \geq 0.5 \\ 0, \text{if } a^2 < 0.5 \end{cases}$$

Concept



Input layer

1st hidden layer

Output layer

$\sum$

In every layer, two computations, weighted sum and the evaluations of sigmoid functions, are conducted to find the values at the next layer

Input layer → 1st hidden layer

Linear combination

$$\begin{bmatrix} z_0^1 \\ z_1^1 \\ z_2^1 \\ z_3^1 \end{bmatrix} = \begin{bmatrix} W_{0,0}^0 W_{1,0}^0 W_{2,0}^0 \\ W_{0,1}^0 W_{1,1}^0 W_{2,1}^0 \\ W_{0,2}^0 W_{1,2}^0 W_{2,2}^0 \\ W_{0,3}^0 W_{1,3}^0 W_{2.3}^0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$$z^1 = W^1 x$$

Sigmoid

$$\begin{bmatrix} a_0^1 \\ a_1^1 \\ a_2^1 \\ a_3^1 \end{bmatrix} = g\left(\begin{bmatrix} z_0^1 \\ z_1^1 \\ z_2^1 \\ z_3^1 \end{bmatrix}\right)$$

$$a^1 = g(z^1)$$

1st hidden layer → output layer

Linear combination

$$[z^2] = [W_0^1 W_1^1 W_2^1 W_3^1] \begin{bmatrix} a_0^1 \\ a_1^1 \\ a_2^1 \\ a_3^1 \end{bmatrix}$$

$$z^2 = W^2 a^1$$

Sigmoid

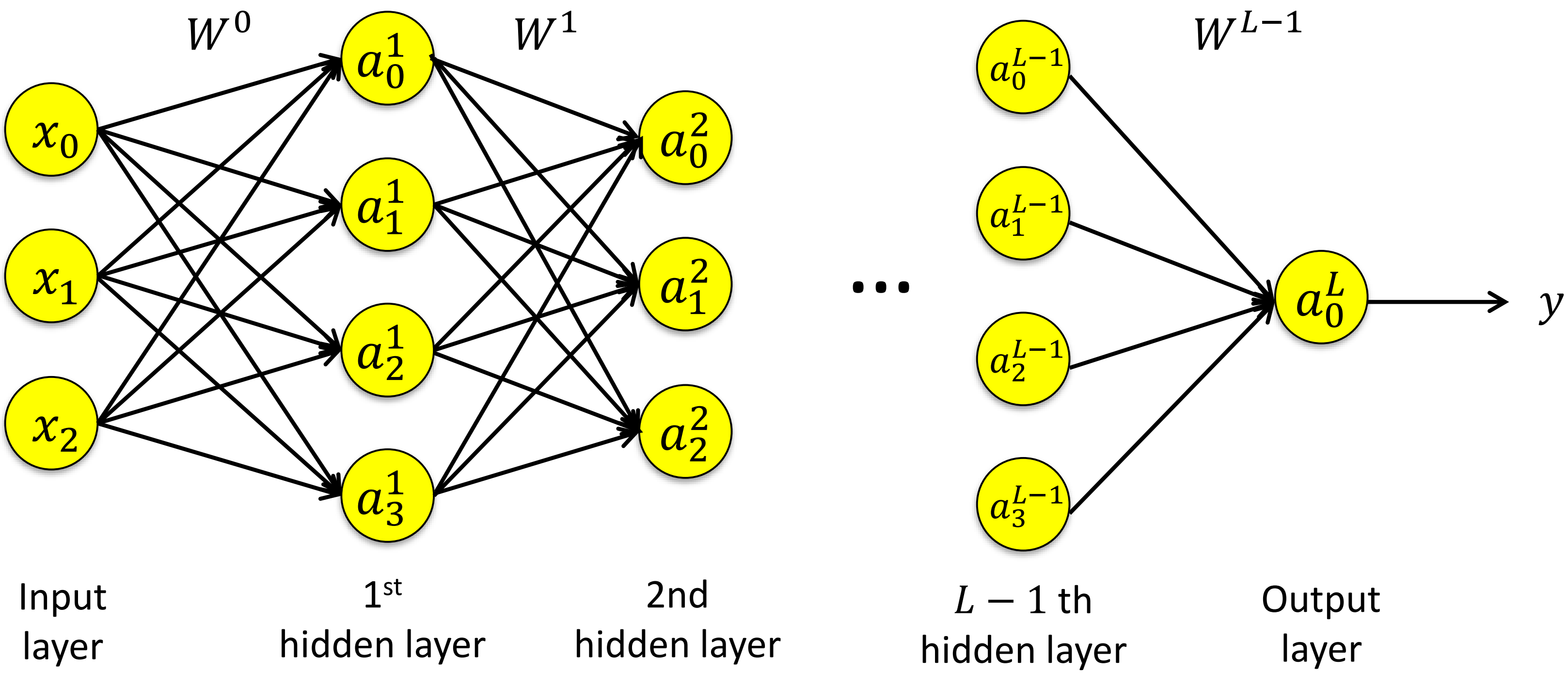
$$[\, a^2] = g([z^2])$$

$$a^2 = g(z^2)$$

Output layer

$$y = h_W(x) = \begin{cases} 1 \text{ if } a^2 \geq 0.5 \\ 1 \text{ if } a^2 < 0.5 \end{cases}$$

Prediction

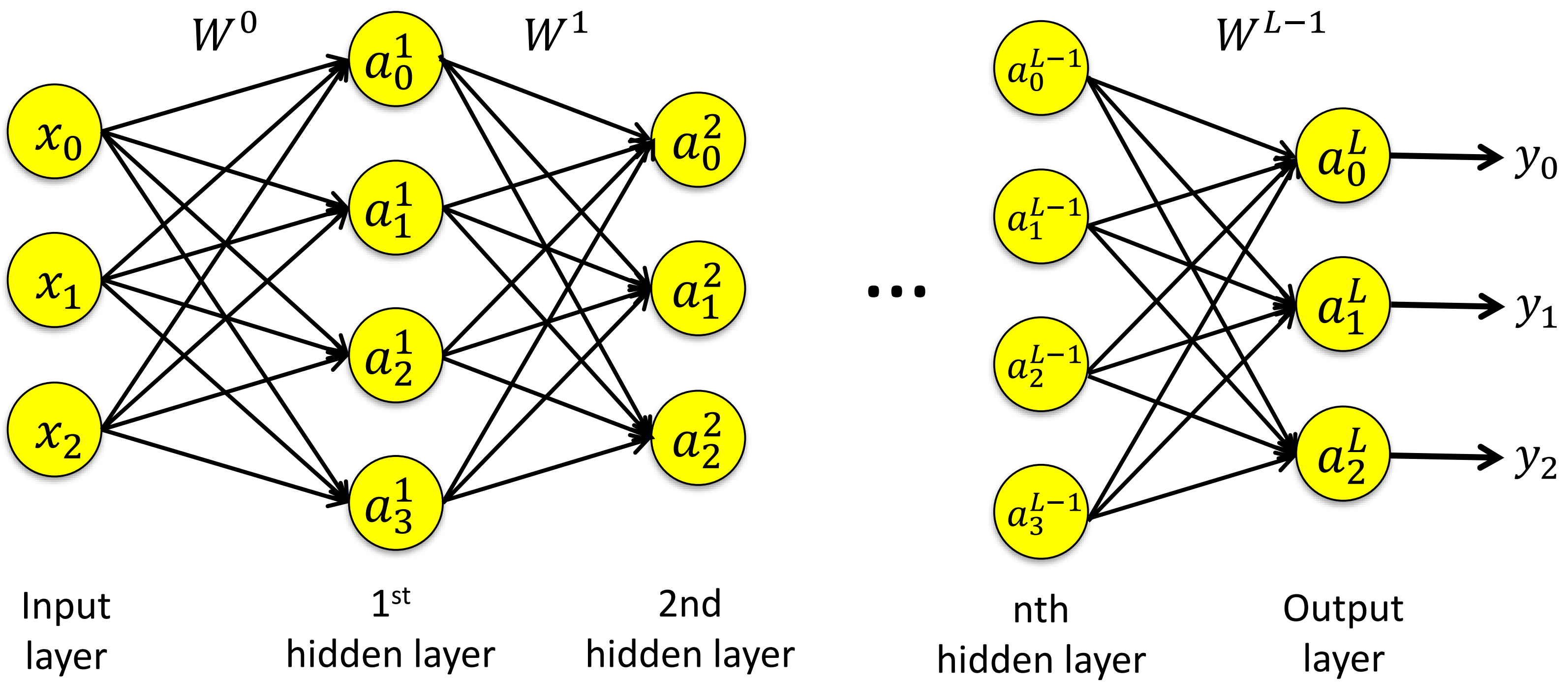


Prediction (Forward propagation)

Input layer	$z^1 = W^1 x, a^1 = g(z^1)$
Hidden layer	$\begin{cases} \text{for } l = 1, \dots, L \\ \quad z^l = W^{l-1} a^{l-1}, a^l = g(z^l) \end{cases}$
Output layer	$y = h_W(x) = \begin{cases} 1 \text{ if } a_0^L \geq 0.5 \\ 1 \text{ if } a_0^L < 0.5 \end{cases}$

- By adding more hidden layers, more complex features can be constructed.
- Prediction is conducted by sequence of matrix multiplication and the evaluations of logistic function

Multi-labels Classification



$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = h_W(x) = \text{one of} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- Multiple binary classifications are executed for a certain class and the rest class. (one vs. all method)

Cost Functions For Neural Network

- **Log-likelihood of a logistic regression**

$$\sum_{i=1}^{m} y_i \log g(w^T x_i) + (1 - y_i) \log\left(1 - g(w^T x_i)\right) - \lambda \sum_{i=1}^{n} w_i^2$$

Penalizing parameters

$$g(w^T x) = \frac{1}{(1 + \exp(-w^T x))}$$

- **Log-likelihood of a Neural Network**

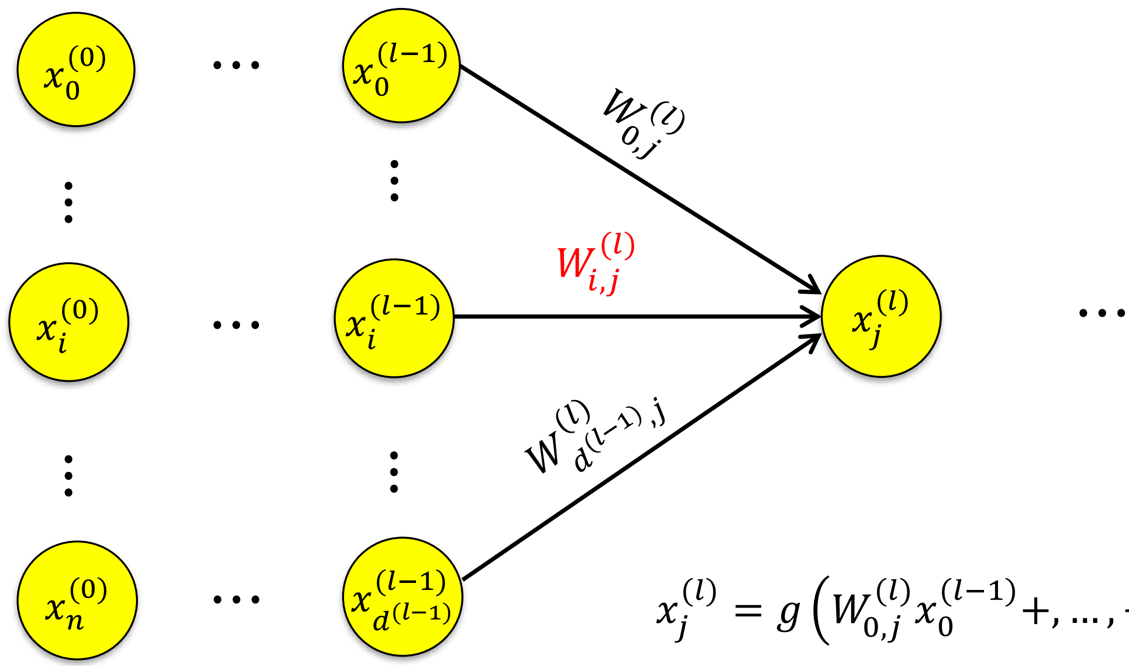
$$\sum_{i=1}^{m} y_i \log G_W(x_i) + (1 - y_i)\left(1 - \log\left(G_W(x_i)\right)\right) - \lambda \sum_{l}^{L} \sum_{i} \sum_{j} \left(W_{i,j}^{l}\right)^2$$

Penalizing parameters

$G_W(x) = g(W_3^T g(W_2^T g(W_1^T x)))$ is a nested function of sigmoid functions

→ **Training is difficult!!**

Forward Propagation



$$x_j^{(l)} = g\left(W_{0,j}^{(l)} x_0^{(l-1)} +, \dots, +W_{i,j}^{(l)} x_i^{(l-1)} +, \dots +, W_{d^{(l-1)},j}^{(l)} x_{d^{(l-1)}}^{(l-1)}\right)$$

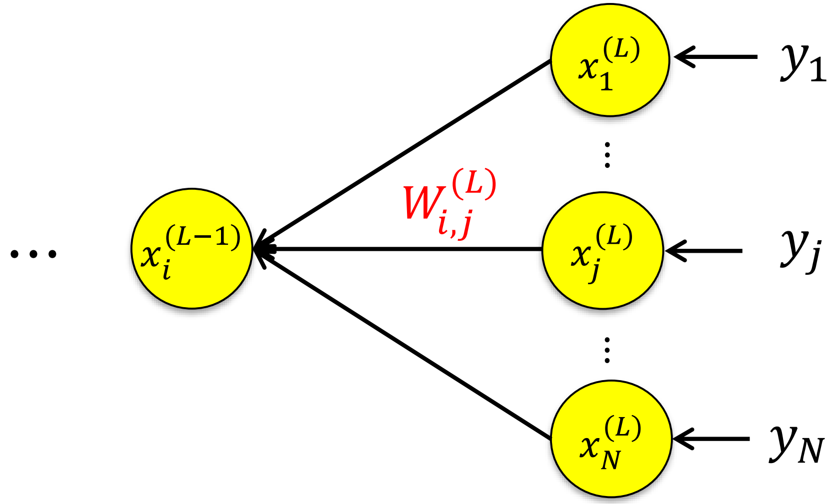
$$= g\left(\sum_{k=0}^{d^{(l-1)}} W_{k,j}^{(l)} x_k^{(l-1)}\right)$$

$$= g\left(\left(W_{\cdot,j}^{(l)}\right)^T \boldsymbol{x}^{(l-1)}\right)$$

$$= h_{W_{\cdot,j}^{(l)}}\left(\boldsymbol{x}^{(l-1)}\right)$$

The value for each node is determined by logistic regression for a single layer

Backward Propagation



$$E = \frac{1}{2}\sum_{k=1}^{N}\left(x_k^{(L)} - y_k\right)^2$$

At output node

$$E = \frac{1}{2}\sum_{k=1}^{N}\left(x_k^{(L)} - y_k\right)^2$$

$$\frac{\partial E}{\partial W_{i,j}^{(L)}} = \left(x_j^{(L)} - y_j\right)\frac{\partial}{\partial W_{i,j}^{(L)}} x_j^{(L)}$$

$$= e_j^{(L)} \frac{\partial}{\partial W_{i,j}^{(L)}} g\left(z_j^{(L)}\right)$$

$e_j^{(L)} = x_j^{(L)} - y_j$

Define the summed variable for simplicity

$z_j^{(L)} = \sum_{k=0}^{d^{(L-1)}} W_{k,j}^{(L)} x_k^{(L-1)}$, then $x_j^{(L)} = g\left(z_j^{(L)}\right)$

$$= e_j^{(L)} g'\left(z_j^{(L)}\right)\frac{\partial}{\partial W_{i,j}^{(L)}} z_j^{(L)}$$

$$= e_j^{(L)} g'\left(z_j^{(L)}\right) x_i^{(L-1)}$$

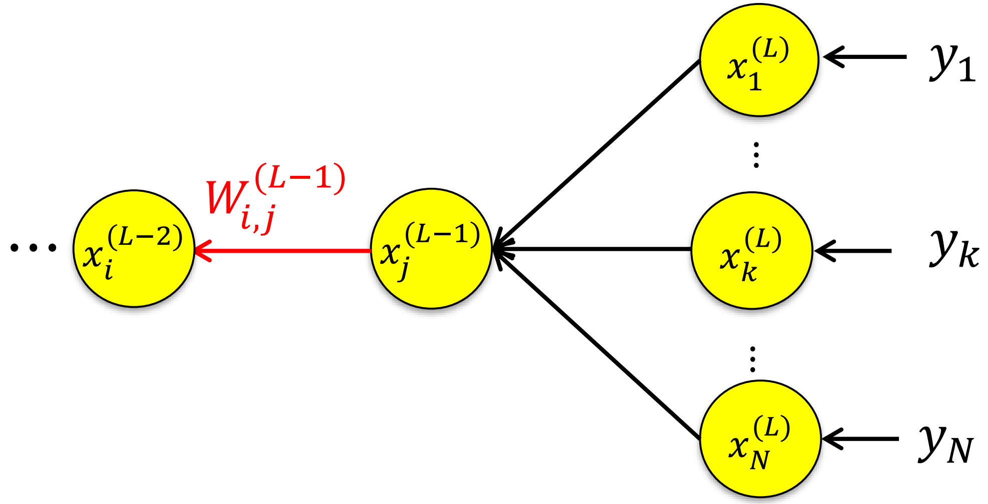
$$= x_i^{(L-1)} \delta_j^{(L)}$$

where $\delta_j^{(L)} = e_j^{(L)} g'\left(z_j^{(L)}\right)$

Backward Propagation

At hidden layer

$$E = \frac{1}{2}\sum_{k=1}^{N}\left(x_k^{(L)} - y_k\right)^2$$



$$\frac{\partial E}{\partial W_{i,j}^{(L-1)}} = \sum_{k=1}^{N}\left(x_k^{(L)} - y_k\right)\frac{\partial}{\partial W_{i,j}^{(L-1)}} x_k^{(L)}$$

$$= \sum_{k=1}^{N} e_k^{(L)} \frac{\partial}{\partial W_{i,j}^{(L-1)}} g\left(z_k^{(L)}\right)$$

$$= \sum_{k=1}^{N} e_k^{(L)} g'\left(z_k^{(L)}\right)\frac{\partial}{\partial W_{i,j}^{(L-1)}} z_k^{(L)}$$

$e_k^{(L)} = x_k^{(L)} - y_k$

$E = \frac{1}{2}\sum_{k=1}^{N}\left(x_k^{(L)} - y_k\right)^2$

$$= \sum_{k=1}^{N} e_k^{(L)} g'\left(z_k^{(L)}\right)\frac{\partial}{\partial W_{i,j}^{(L-1)}} W_{j,k}^{(L)} x_j^{(L-1)}$$

$z_k^{(L)} = \sum_{r=0}^{d^{(L-1)}} W_{r,k}^{(L)} x_r^{(L-1)}$

$$= \sum_{k=1}^{N} e_k^{(L)} g'\left(z_k^{(L)}\right)\frac{\partial W_{j,k}^{(L)} x_j^{(L-1)}}{\partial x_j^{(L-1)}}\frac{\partial x_j^{(L-1)}}{\partial W_{i,j}^{(L-1)}}$$

$$= \sum_{k=1}^{N} e_k^{(L)} g'\left(z_k^{(L)}\right) W_{j,k}^{(L)} g'\left(z_j^{(L-1)}\right) x_i^{(L-2)}$$

$\frac{\partial x_j^{(L-1)}}{\partial W_{i,j}^{(L-1)}} = g'\left(z_j^{(L-1)}\right) x_i^{(L-2)}$ from previous slide

$$= x_i^{(L-2)} g'\left(z_j^{(L-1)}\right)\sum_{k=1}^{N} e_k^{(L)} g'\left(z_k^{(L)}\right) W_{j,k}^{(L)}$$

$$= x_i^{(L-2)}\ \delta_j^{(L-1)}$$

where $\delta_j^{(L-1)} = g'\left(z_j^{(L-1)}\right)\sum_{k=1}^{N} e_k^{(L)} g'\left(z_k^{(L)}\right) W_{j,k}^{(L)} = g'\left(z_j^{(L-1)}\right)\sum_{k=1}^{N} \delta_j^{(L)} W_{j,k}^{(L)}$

Forward Backward Propagation algorithm

Forward propagation

$$x_j^{(l)} = g\left(\sum_{k=0}^{d^{(l-1)}} W_{k,j}^{(l)} x_k^{(l-1)}\right)$$

$$g(z) = \frac{1}{(1+\exp(-z))}$$

Backward propagation

$$\delta_j^{(l)} = \begin{cases} e_j^{(l)} g'\left(z_j^{(l)}\right) & \text{If } l = L \text{ (output layer)} \\ g'\left(z_j^{(l)}\right) \sum_{k=1}^{N} \delta_j^{(l+1)} W_{j,k}^{(l+1)} & \text{If } l < L \text{ (hidden layer)} \end{cases}$$

$$g'\left(z_j^{(l)}\right) = x_j^{(l)}\left(1 - x_j^{(l)}\right)$$

Forward Backward Propagation algorithm

1. Initialize all weights $W_{i,j}^{(l)}$ at random
2. For $t = 0, 1, 2, ..$ do
3. Pick a single data point in $\boldsymbol{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}); i = 1, \dots, m\}$
4. **Forward propagation** : compute all $x_j^{(l)}$
5. **Backward propagation** : compute all $\delta_j^{(l)}$
6. Update the weights $W_{i,j}^{(l)} \leftarrow W_{i,j}^{(l)} - \alpha x_i^{(l-1)} \delta_j^{(l)}$
7. Iterate until $W_{i,j}^{(l)}$ converges
8. Return the final weights $W_{i,j}^{(l)}$

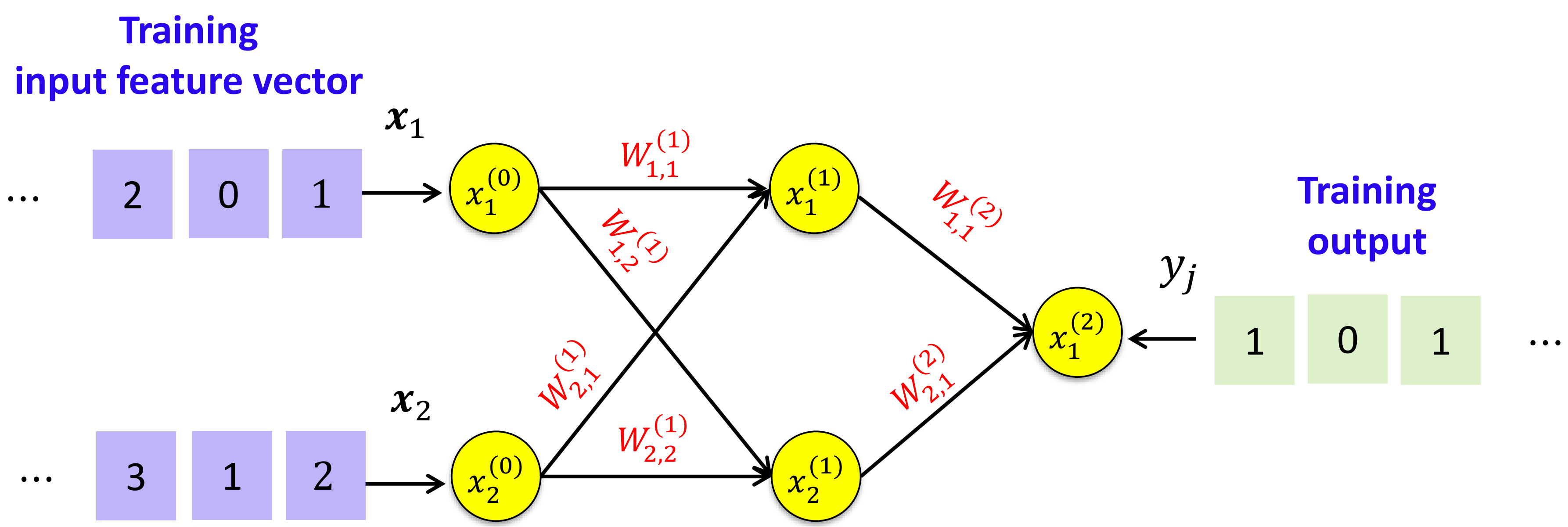
Applying gradient decent

$$W_{i,j}^{(l)} = W_{i,j}^{(l)} - \alpha \frac{\partial E}{\partial W_{i,j}^{(l)}}$$

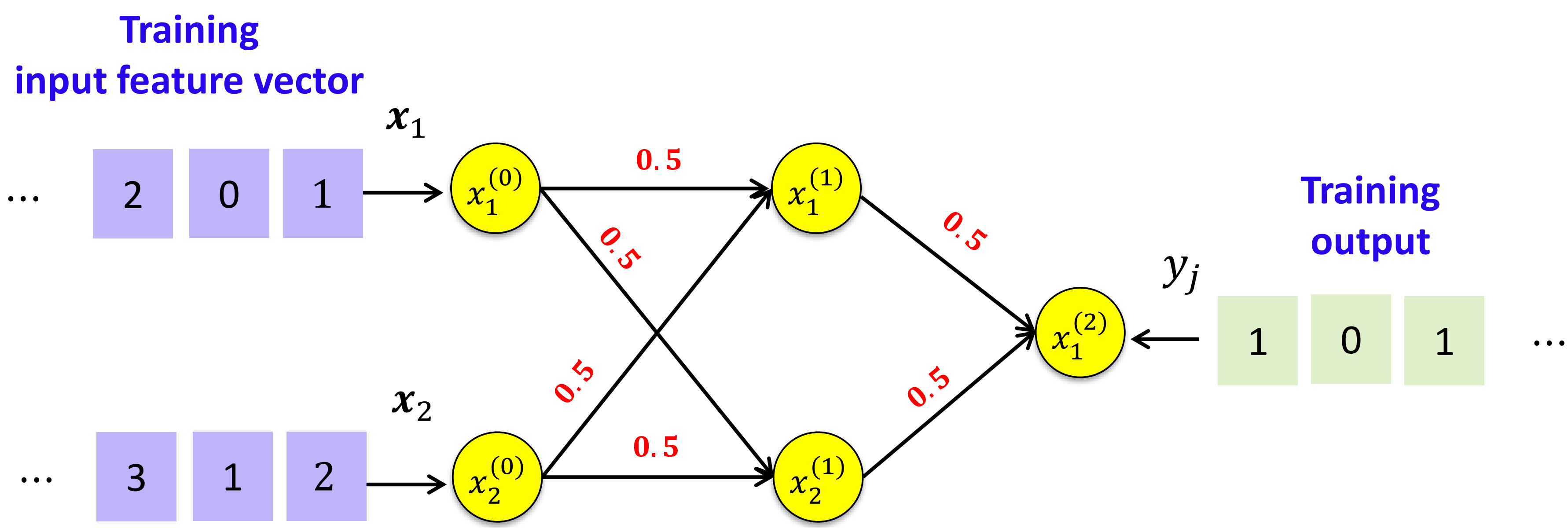
$$\frac{\partial E}{\partial W_{i,j}^{(l)}} = x_i^{(l-1)} \delta_j^{(l)}$$

α is learning rate

Illustrations

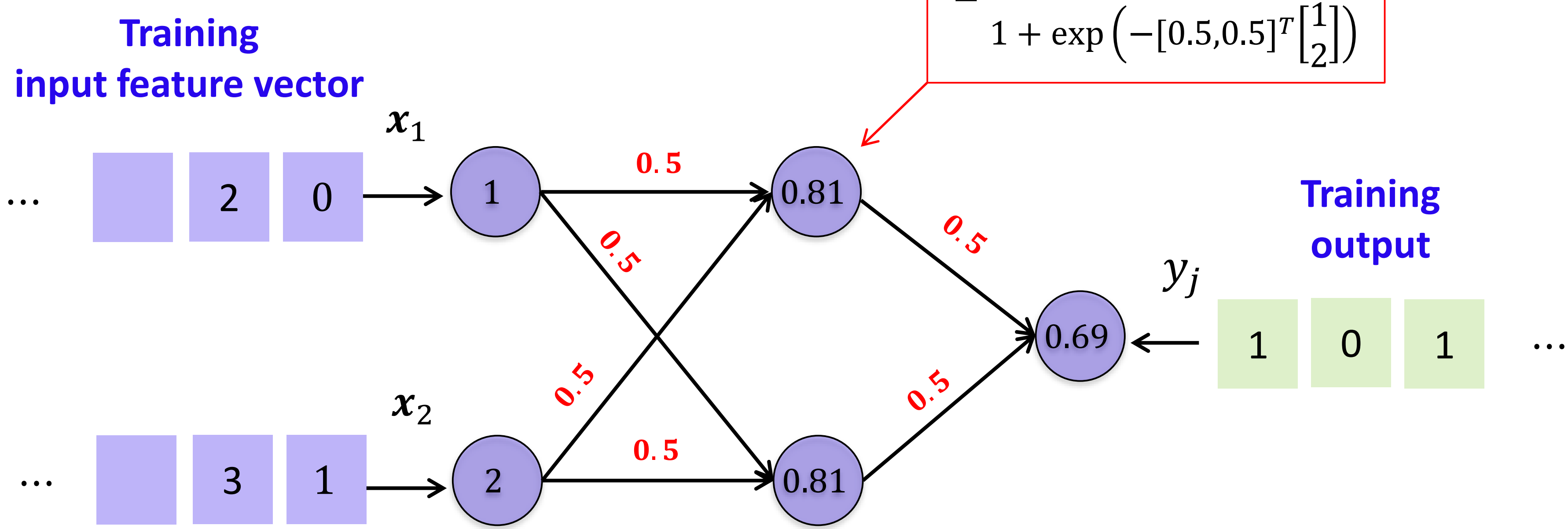


Initialize weight parameters $w_{i,j}^{(L-1)}$ (iteration = 0)



Forward Propagation (iteration = 1)

$$x_1^{(1)} = g\left(\sum_{k=1}^{2} W_{k,1}^{(1)} x_k^{(0)}\right) = \frac{1}{1 + \exp\left(-[0.5,0.5]^T \begin{bmatrix}1\\2\end{bmatrix}\right)}$$

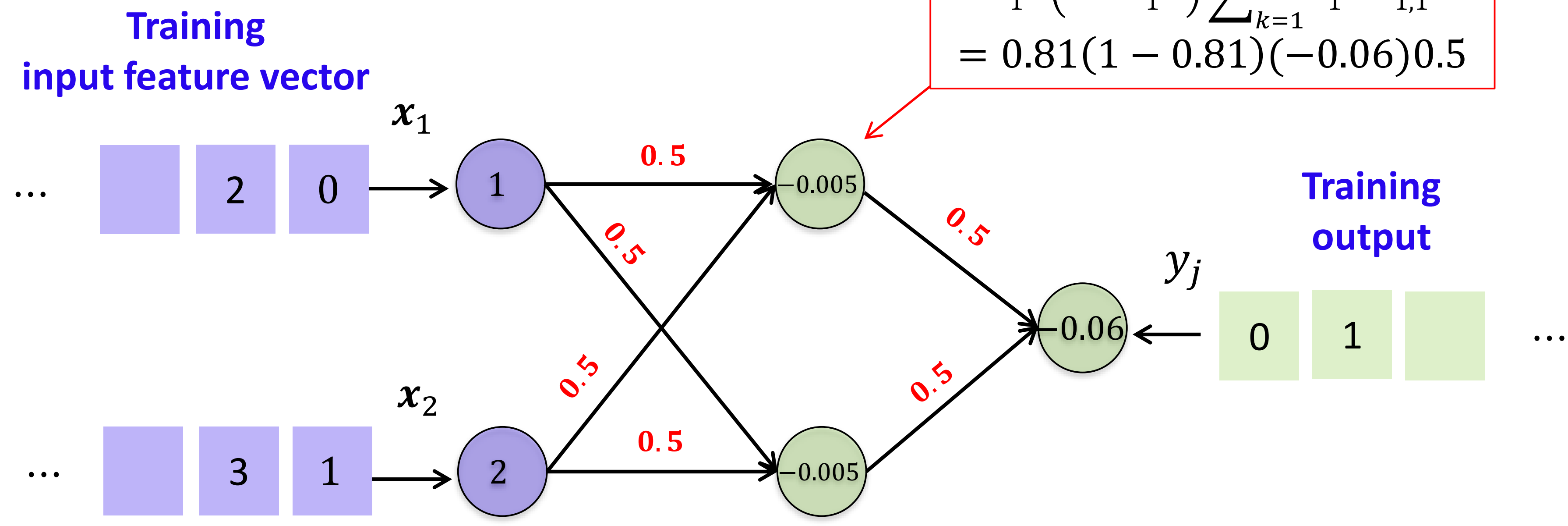


Forward propagation

$$x_j^{(l)} = g\left(\sum_{k=0}^{d^{(l-1)}} W_{k,j}^{(l)} x_k^{(l-1)}\right)$$

Backward Propagation (iteration = 1)

$$x_1^{(1)}\left(1 - x_1^{(1)}\right)\sum_{k=1}^{1} \delta_1^{(2)} W_{1,1}^{(2)}$$
$$= 0.81(1 - 0.81)(-0.06)0.5$$

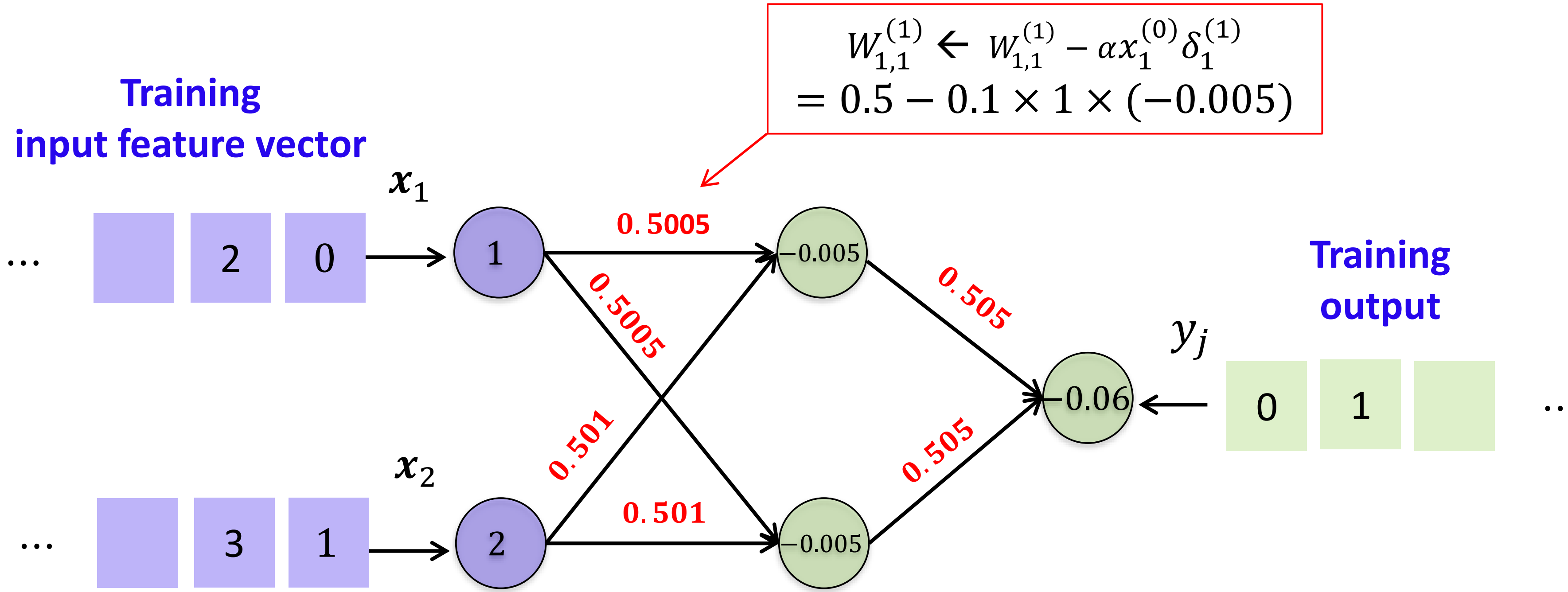


Backward propagation

$$\delta_j^{(l)} = \begin{cases} e_j^{(l)} g'\left(z_j^{(l)}\right) \\ g'\left(z_j^{(l)}\right) \sum_{k=1}^{N} \delta_j^{(l+1)} W_{j,k}^{(l+1)} \end{cases}$$

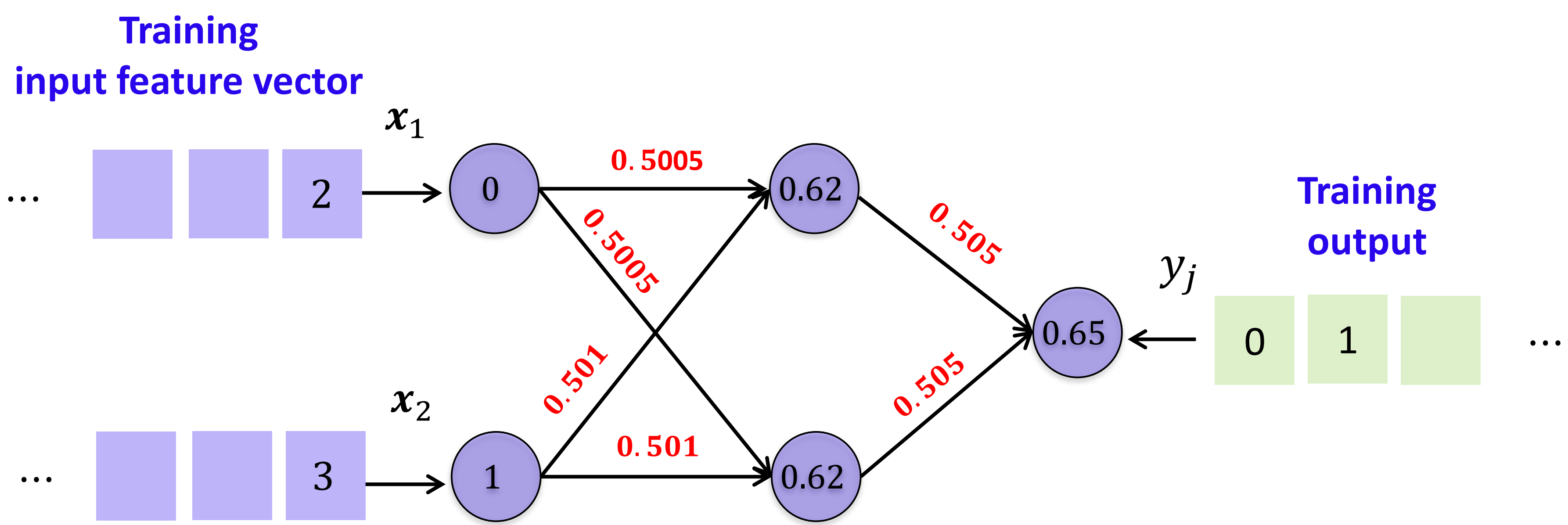
$$g'\left(z_j^{(l)}\right) = x_j^{(l)}\left(1 - x_j^{(l)}\right)$$

Parameters updating (iteration = 1)



Update the weights $W_{i,j}^{(l)} \leftarrow W_{i,j}^{(l)} - \alpha x_i^{(l-1)} \delta_j^{(l)}$

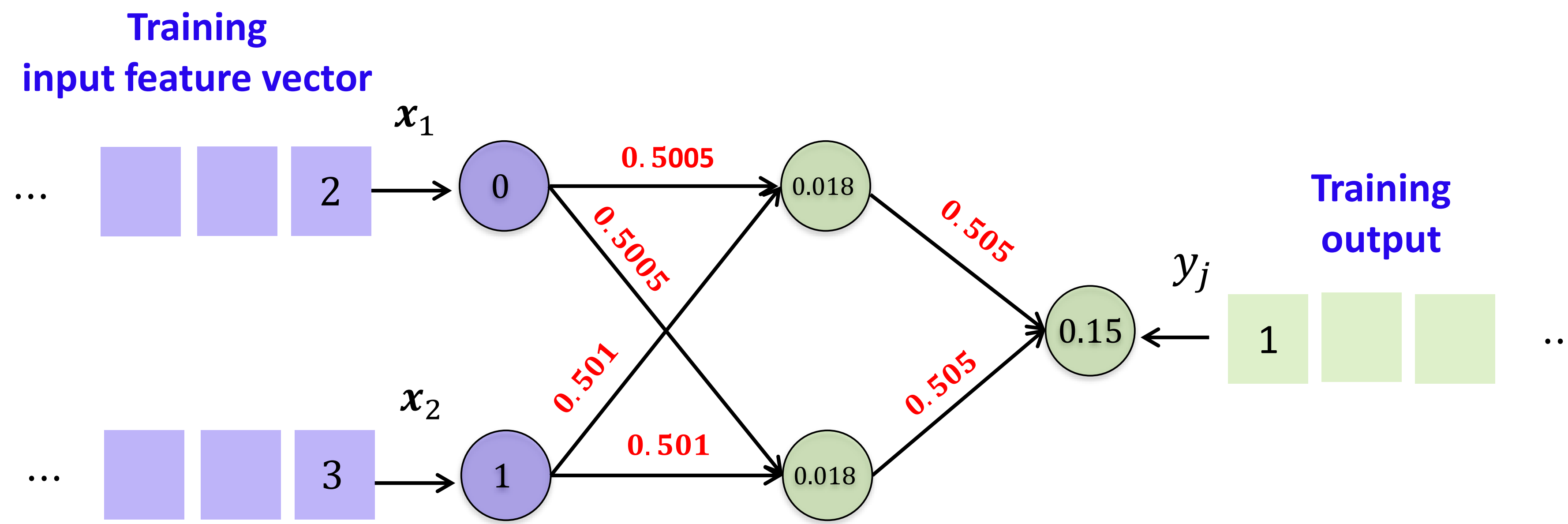
Forward Propagation (iteration = 2)



Forward propagation

$$x_j^{(l)} = g\left(\sum_{k=0}^{d^{(l-1)}} W_{k,j}^{(l)} x_k^{(l-1)}\right)$$

Backward Propagation (iteration = 2)

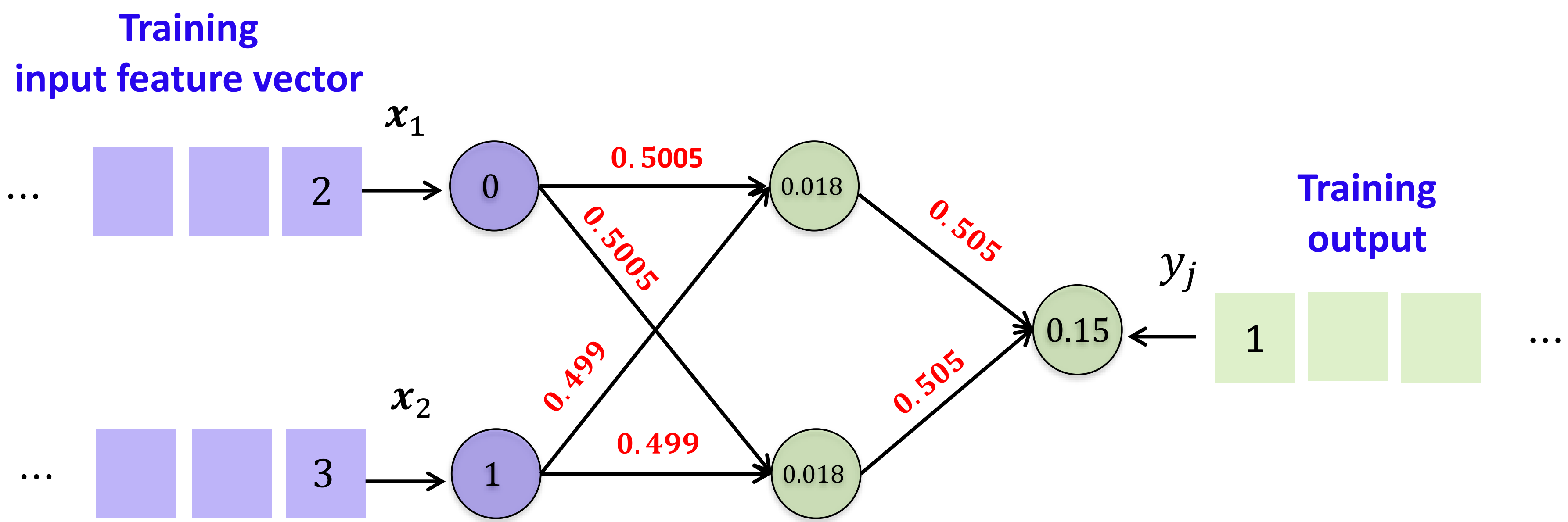


Backward propagation

$$\delta_j^{(l)} = \begin{cases} e_j^{(l)} g'\left(z_j^{(l)}\right) \\ g'\left(z_j^{(l)}\right) \sum_{k=1}^{N} \delta_j^{(l+1)} W_{j,k}^{(l+1)} \end{cases}$$

$$g'\left(z_j^{(l)}\right) = x_j^{(l)}\left(1 - x_j^{(l)}\right)$$

Backward Propagation (iteration = 2)



Update the weights $W_{i,j}^{(l)} \leftarrow W_{i,j}^{(l)} - \alpha x_i^{(l-1)} \delta_j^{(l)}$

Generative model

Generative model

1. Define Class prior $p(y)$ and likelihood $P(x|y)$

2. Learn the parameters of the models, $P(y)$ and $P(x|y)$

3. Express posterior distribution on class y given the input vector x

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x|y)P(y)}{\sum_{y \in Y} P(x|y)P(y)}$$

4. Prediction step: any new input feature vector x_{new} can be classified according to the maximum a posteriori detection principle (MAP)

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y|x_{new}) = \underset{y}{\operatorname{argmax}} \frac{P(x_{new}|y)P(y)}{\sum_y P(x_{new}|y)P(y)}$$

$$= \underset{y}{\operatorname{argmax}}\, P(x_{new}|y)P(y)$$

Generative model

1. Define prior and likelihood

$$p(x|y = \text{dog}), p(y = \text{dog})$$
$$p(x|y = \text{cat}),\ p(y = \text{cat})$$

2. Learn the parameters for the models

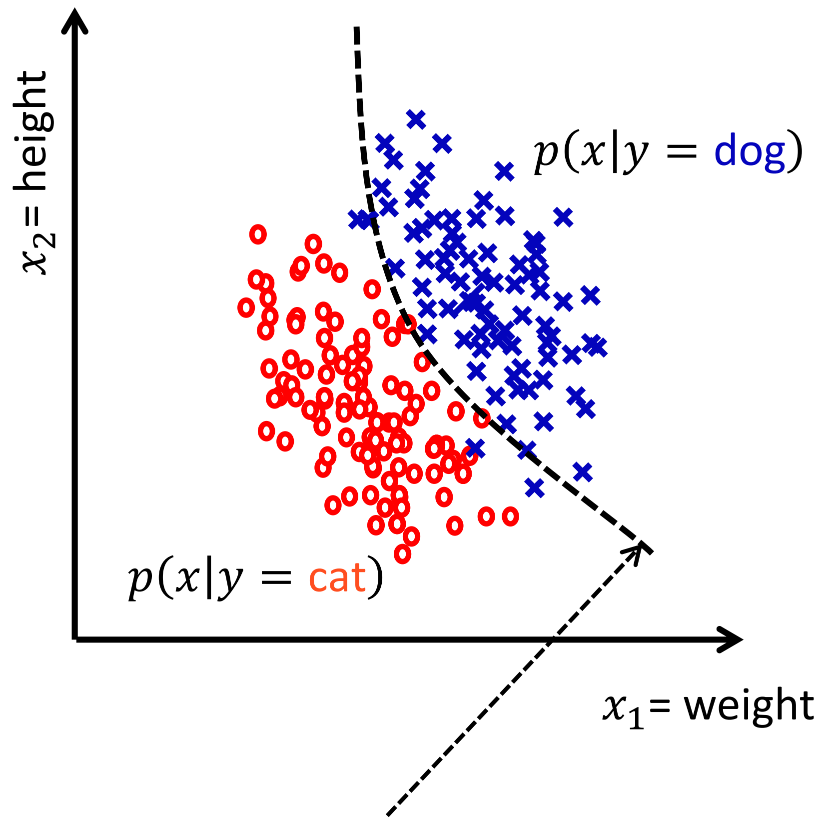
3. Construct the posterior distribution on class

$$P(y = \text{dog}|x) = \frac{p(x|y = \text{dog})p(y = \text{dog})}{\sum_{y\in\{doc,cat\}} p(x|y)p(y)}$$

$$P(y = \text{cat}|x) = \frac{p(x|y = \text{cat})p(y = \text{cat})}{\sum_{y\in\{doc,cat\}} p(x|y)p(y)}$$

4. Classify animal based on MAP estimation:

$$\hat{y} = \underset{y\in Y}{\operatorname{argmax}} P(y|x^{new})$$



Decision boundary

$$p(y = \text{dog}|x) = p(y = \text{cat}|x)$$

The shape of a decision boundary changes depending on the assumptions on the model (ex., linear, quadratic, …)

Multivariate Gaussian Distribution

Univariate Gaussian

$$N(x;\mu,\sigma)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Mean $\mu = E[X]$

variance $\sigma^2 = \mathrm{var(X)} = \mathrm{E}[(X-E[X])^2]$

Multivariate Gaussian

$$N(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})=\frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

Mean vector

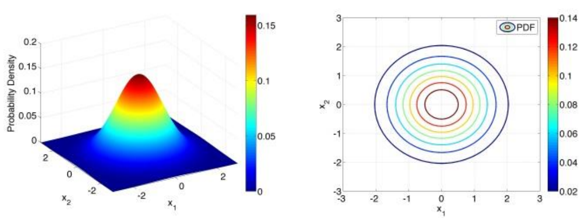
$$\boldsymbol{\mu}=\begin{bmatrix}E[X_1]\\ \vdots \\ E[X_n]\end{bmatrix}$$

Covariance matrix

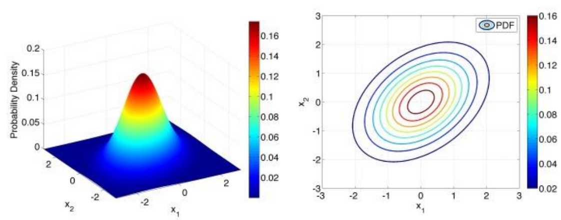
$$\boldsymbol{\Sigma}=\begin{bmatrix}Cov[X_1,X_1] & \cdots & Cov[X_1,X_n]\\ \vdots & \ddots & \vdots \\ Cov[X_n,X_1] & \cdots & Cov[X_n,X_n]\end{bmatrix}$$

$$Cov[X,Z]=E[(X-E[X])(Z-E[Z])]$$

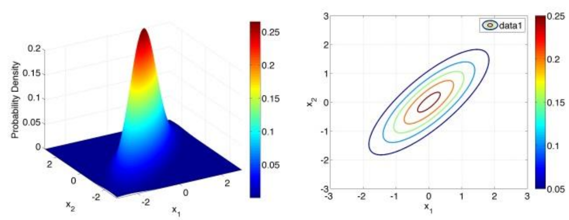
$\boldsymbol{\mu}=\begin{bmatrix}0\\0\end{bmatrix}$ $\boldsymbol{\Sigma}=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$



$\boldsymbol{\mu}=\begin{bmatrix}0\\0\end{bmatrix}$ $\boldsymbol{\Sigma}=\begin{bmatrix}1 & 0.4\\0.4 & 1\end{bmatrix}$

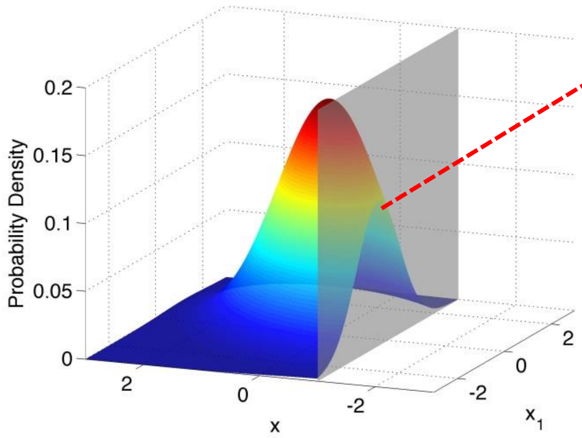


$\boldsymbol{\mu}=\begin{bmatrix}0\\0\end{bmatrix}$ $\boldsymbol{\Sigma}=\begin{bmatrix}1 & 0.8\\0.8 & 1\end{bmatrix}$



Multivariate Gaussian Distribution

Conditionalization→ Gaussian



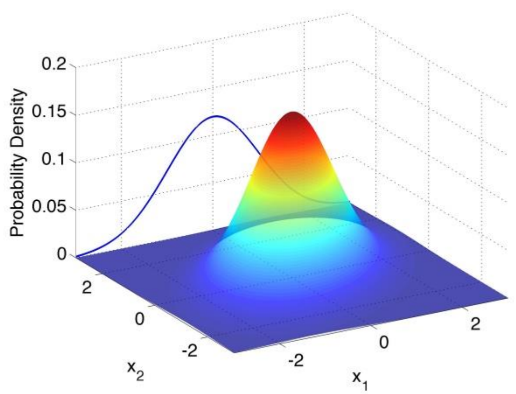
$$P(x_1|x_2) = \frac{P(x_1,x_2)}{P(x_2)}$$

(Graph does not show normalization)

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$$

$$X_1|\{X_2 = x_2\} \sim N\left(\Sigma_{21}\Sigma_{11}^{-1}(x_2 - \mu_1) + \mu_{2,}\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)$$

Marginalization→ Gaussian



$$P(X_1) = \int_{x_2=-\infty}^{x_2=\infty} P(X_1,X_2 = x_2)\, dx_2$$

(Graph does not show normalization)

Concept

1. The class **prior** is represented as multinomial distribution

$$p(y=j)=\phi_j,\ \left(\sum_{j=1}^{N}\phi_j=1\right)$$

2. The distribution of input feature $\boldsymbol{x}$ conditional on the ouput class y is modeled as multivariate Gaussian distribution

$$p(\boldsymbol{x}|y=j)=N\left(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\right)=\frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_j|}}\exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)^T\boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)\right)$$

$\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j$: the mean vector and covariance matrix for the jth class

Parameter learning for GDA

- Using the training data $\boldsymbol{D} = \{(x_i, y_i); i = 1, \dots, m\}$, the parameter sets for GDA are:

$\boldsymbol{\phi} = \{\phi_1, \dots, \phi_N\}$: set of priors
$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N\}$: set of mean vectors
$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_N\}$: set of covariance matrices

- The parameters are found as ones maximizing the log-likelihood of data

$$\log p(D|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi}) = \log \prod_{i=1}^{m} p(x_i|y_i, \boldsymbol{\mu_i}, \boldsymbol{\Sigma}_\mathrm{i}) P(y_i|\boldsymbol{\phi_i})$$
$$= \sum_{i=1}^{m} \log \boldsymbol{p}(x_i|y_i, \boldsymbol{\mu_i}, \boldsymbol{\Sigma}_\mathrm{i}) P(y_i|\boldsymbol{\phi_i})$$

The log-likelihood function is **concave function** in terms of the parameters
→ the optimum parameters are analytically derived as

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} 1\{y_i = j\}$$

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{m} 1\{y_i = j\}\, x_i}{\sum_{i=1}^{m} 1\{y_i = j\}}$$

$$\boldsymbol{\Sigma}_j = \frac{1}{\sum_{i=1}^{m} 1\{y_i = j\}} \sum_{i=1}^{m} 1\{y_i = j\} \left(x_i - \boldsymbol{\mu}_{y^{(i)}}\right) \left(x_i - \boldsymbol{\mu}_{y^{(i)}}\right)^T$$

Indication function

$$1\{y_i = j\} = \begin{cases} 1, \text{if} \quad y_i = j \\ 0, \text{otherwise} \end{cases}$$

Class Prediction

- The probability of class $y = j$ given the new input x^{new} can be computed

$$P(y = j\,|x^{new}) \sim P(x^{new}|y = j)P(y = j) \qquad p(y|x^{new}) = \frac{P(x^{new}|y)P(y)}{P(x^{new})}$$

$$= \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_j|}} \exp\left(-\frac{1}{2}\left(x^{new} - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1}\left(x^{new} - \boldsymbol{\mu}_j\right)\right)\phi_j$$

- The class can be selected using MAP estimation:

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}}\, p(y|x^{new})$$

- The boundary surface between two neighboring classes i and j $\left(\text{i.e.}, P(y = i|\boldsymbol{x}) = P(y = j|\boldsymbol{x})\right)$

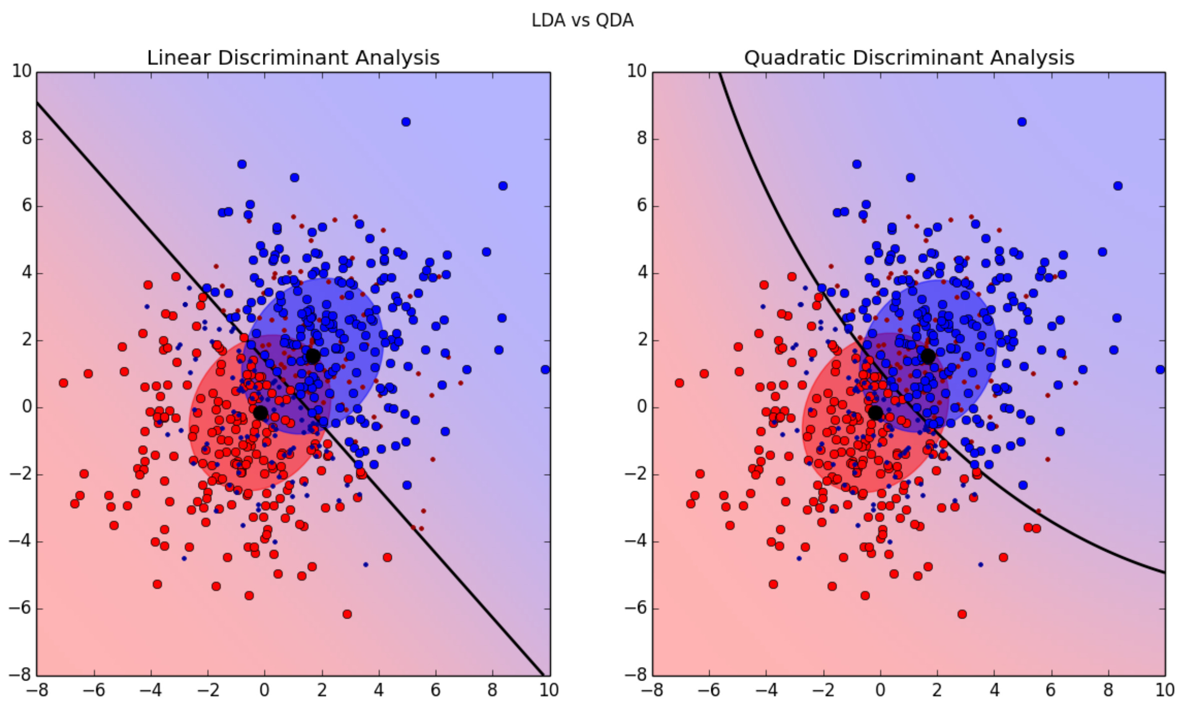
$$\frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)\right)\phi_i = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_j|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)\right)\phi_j$$

$$\rightarrow \boldsymbol{x}^T\left(\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_j^{-1}\right)\boldsymbol{x} - 2\left(\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\right)\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j + \log\frac{\phi_j|\boldsymbol{\Sigma}_i|}{\phi_i|\boldsymbol{\Sigma}_j|} = 0$$

Example (binary-classes)

The boundary surface between two neighboring classes i and j $\left(\text{i.e.}, P(y=i|\boldsymbol{x}) = P(y=j|\boldsymbol{x})\right)$

$$\boldsymbol{x}^T\left(\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_j^{-1}\right)\boldsymbol{x} - 2\left(\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\right)\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j + \log\frac{\phi_j|\boldsymbol{\Sigma}_i|}{\phi_i|\boldsymbol{\Sigma}_j|} = 0$$



Linear discriminant analysis

$\Sigma_i = \Sigma$ for all i

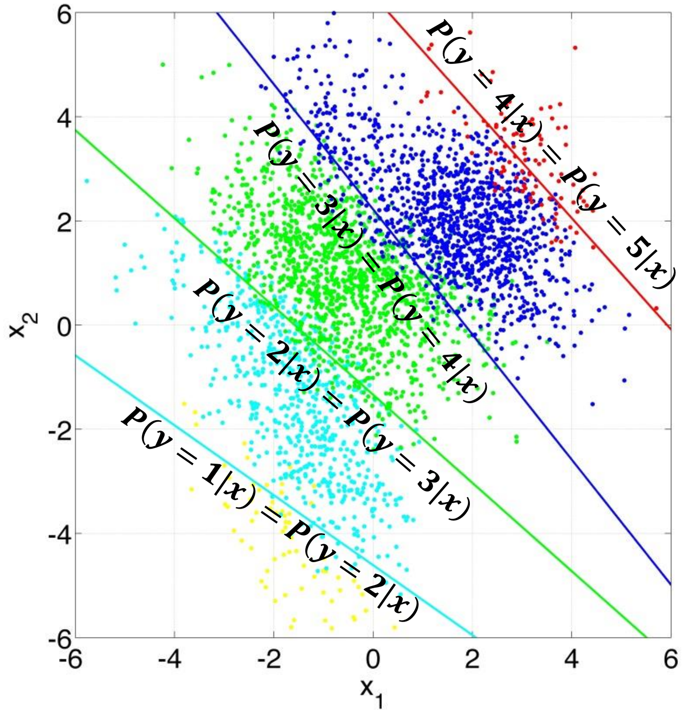
Quadratic discriminant analysis

Σ_i for each i

Example (multi-classes)

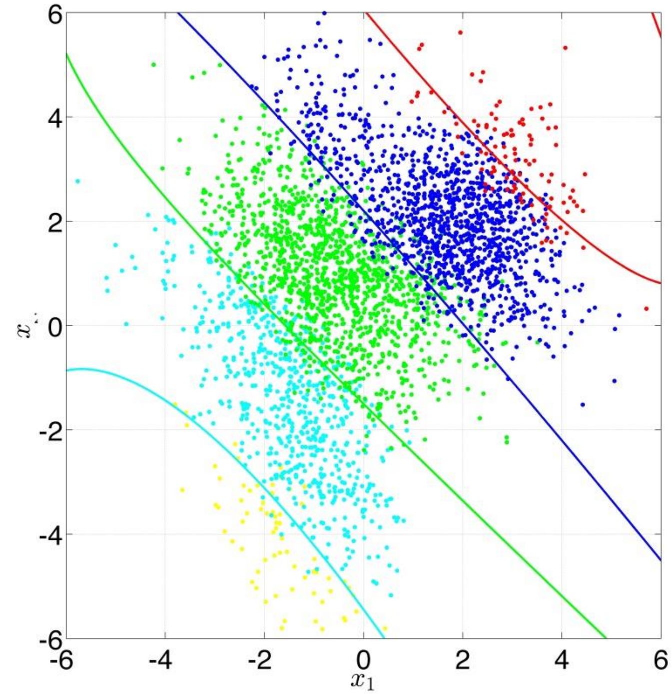
The boundary between the two neighboring classes i and j by setting $P(y=i|\boldsymbol{x}) = P(y=j|\boldsymbol{x})$, which yields

$$\boldsymbol{x}^T\left(\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_j^{-1}\right)\boldsymbol{x} - 2\left(\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\right)\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j + \log\frac{|\boldsymbol{\Sigma}_i|}{|\boldsymbol{\Sigma}_j|} = 0$$



Linear discriminant analysis

$\Sigma_i = \Sigma$ for all i



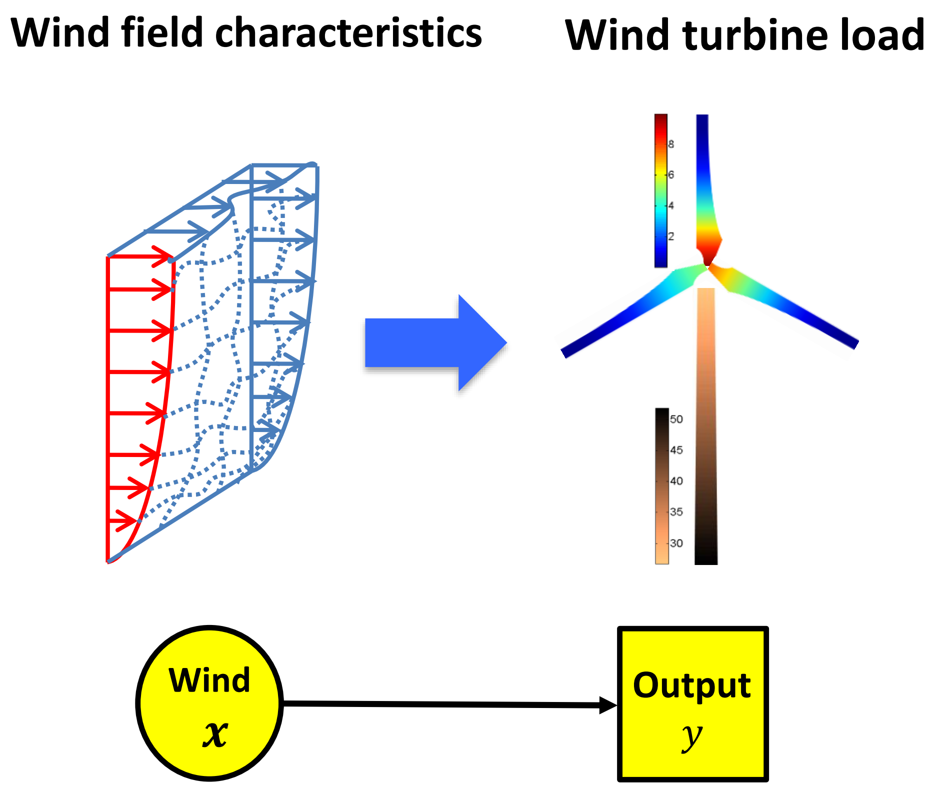
Quadratic discriminant analysis

Σ_i for each i

Gaussian Discriminant Analysis

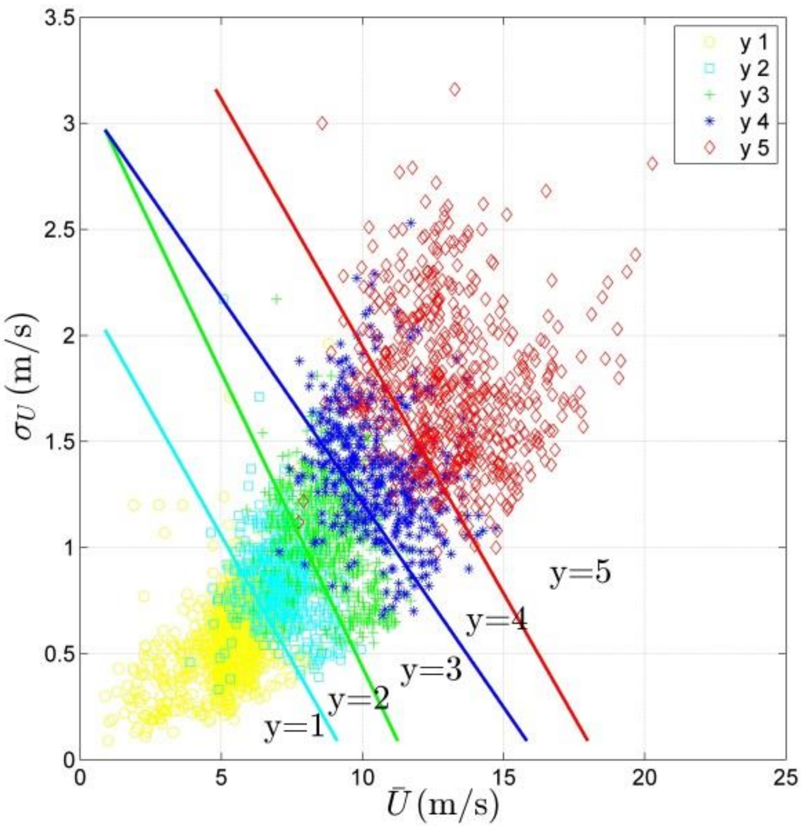
Application to wind turbine monitoring data

Study how wind field characteristics affect wind turbine response class.



→ **Construct the posterior probability mass function for the response class** $p(y|x)$

Application to wind turbine monitoring data



Classification boundaries between the ith and the jth class is determined as:

$$p(y = i|\boldsymbol{x}) = p(y = j|\boldsymbol{x})$$

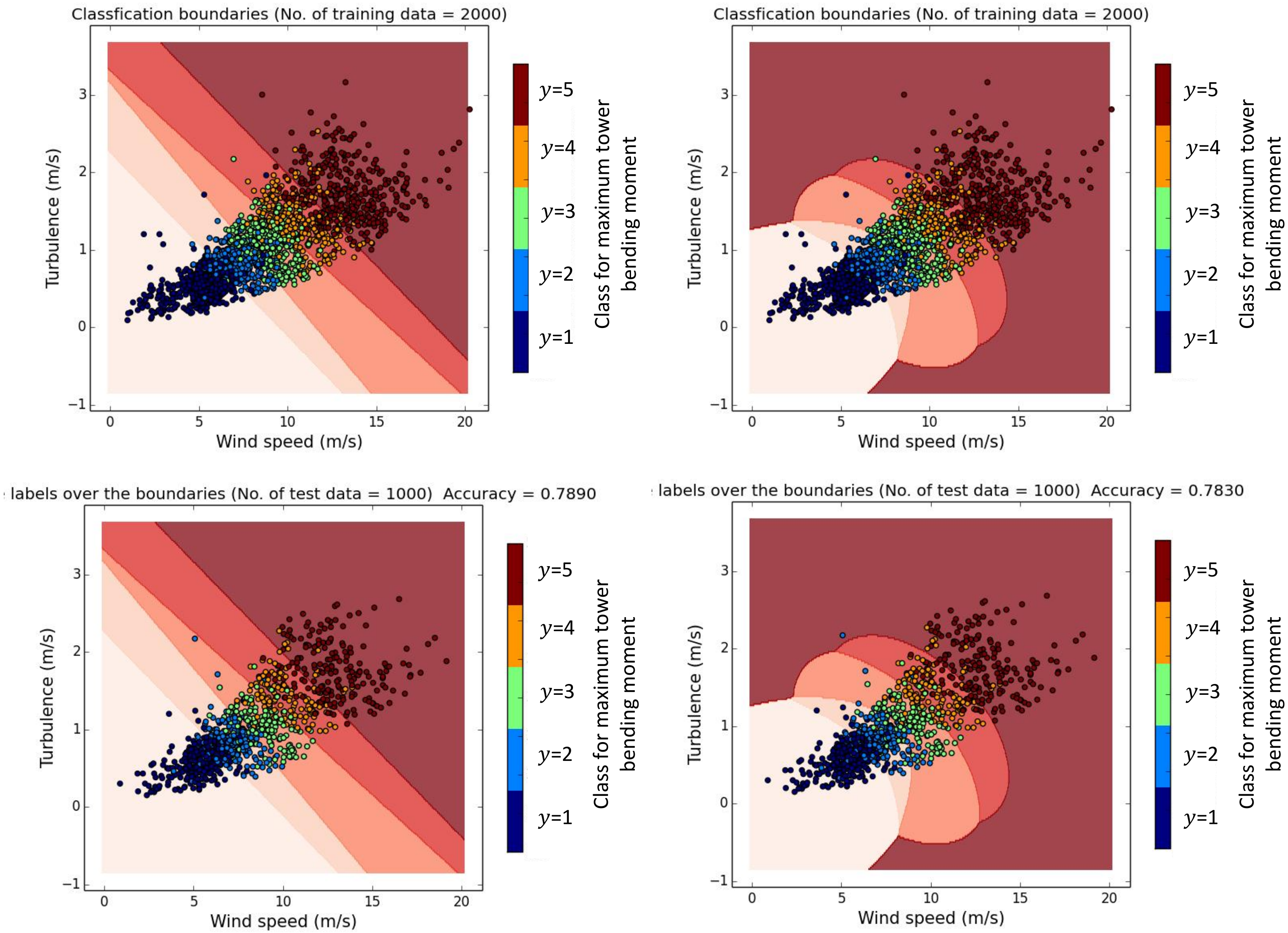
- Higher wind speed and higher turbulence tend to cause higher blade bending moment.
- Accuracy for the classification is approximately 80 %.
- Including more input features can increases the accuracy ration.

Application to wind turbine monitoring data

Linear discriminant analysis | Quadratic discriminant analysis

Training

Testing

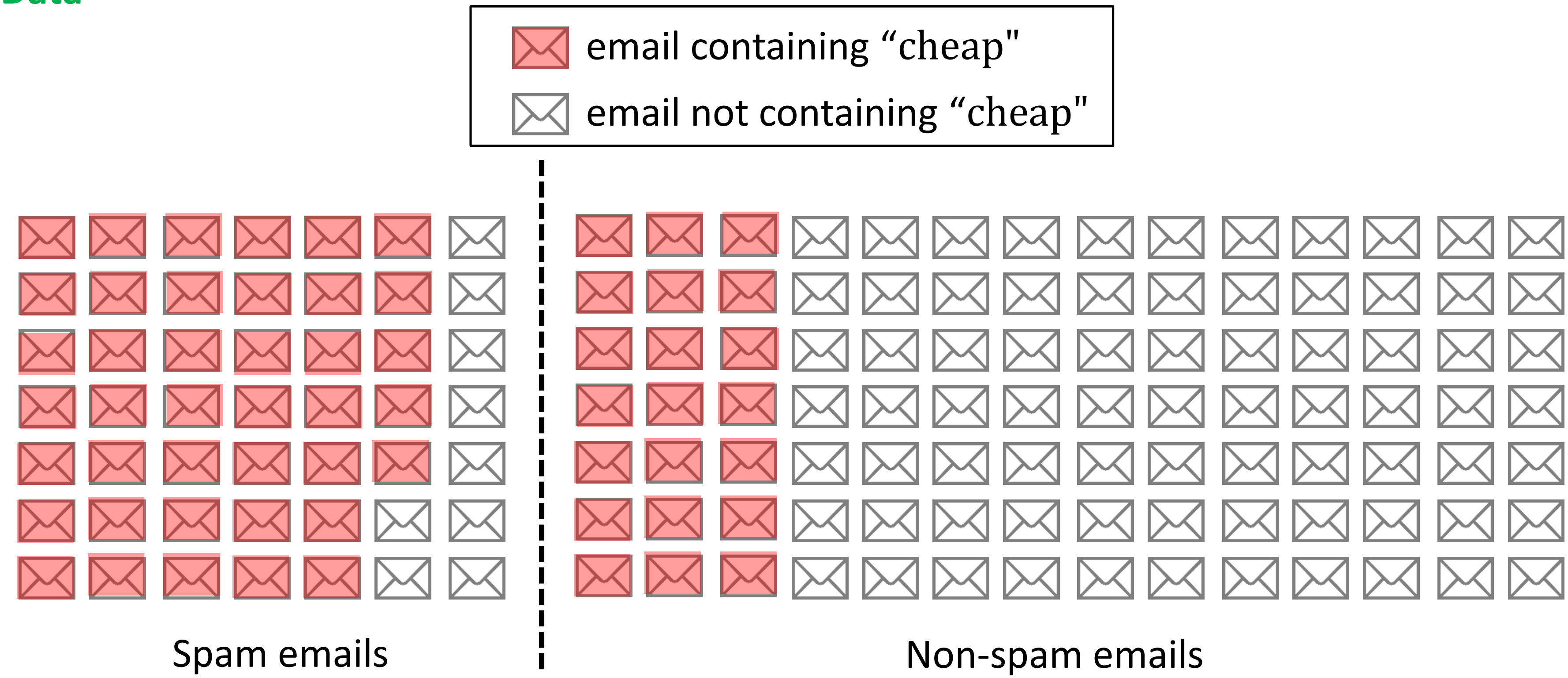


Detecting Spam e-mails

- Input: $x =$ email message
- Output $y \in$ {Spam, non-spam}
- Objective: Obtain a classifier f

$$x \longrightarrow \boxed{f} \longrightarrow y$$

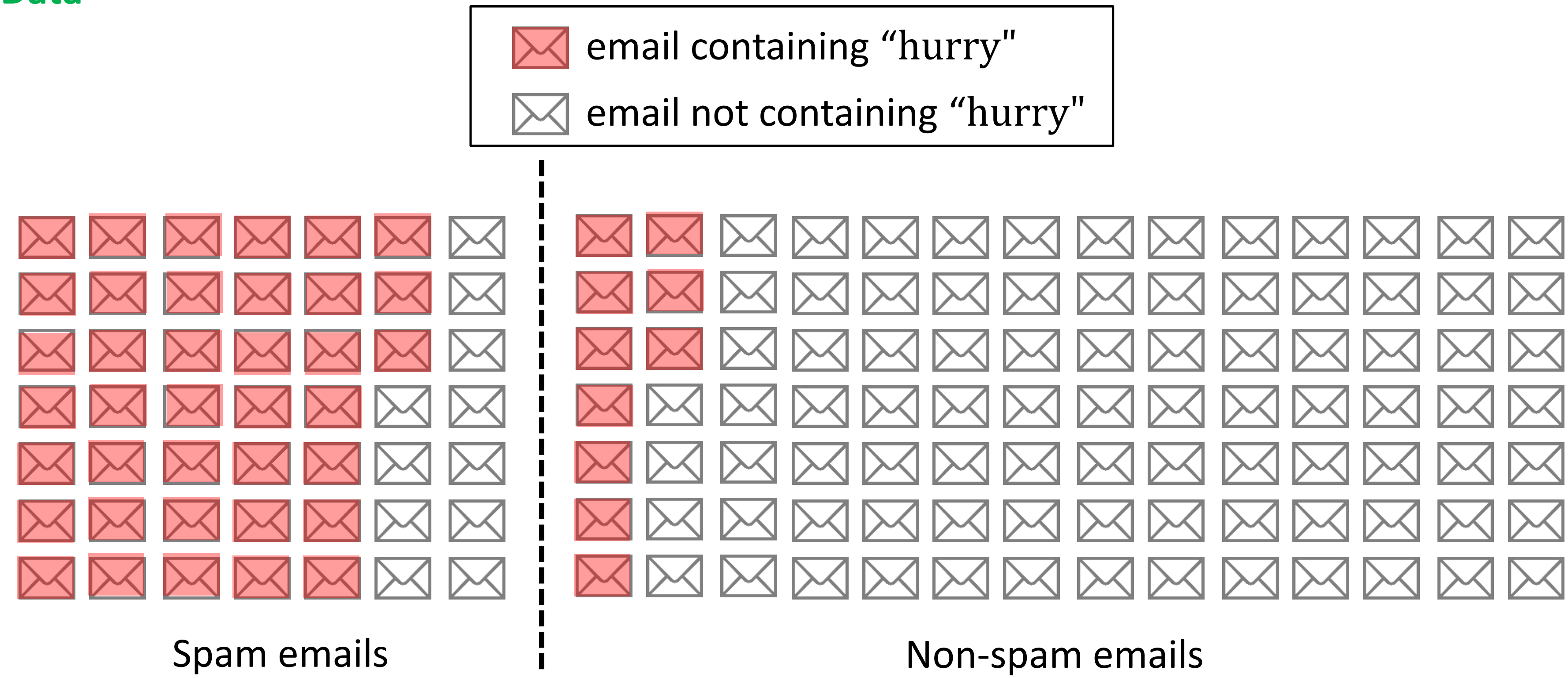
Data



$$P(\text{cheap}|y=\text{spam})=\frac{40}{49}\qquad P(\text{cheap}|y=\text{nonspam})=\frac{21}{98}$$

Naïve Bayes Classification

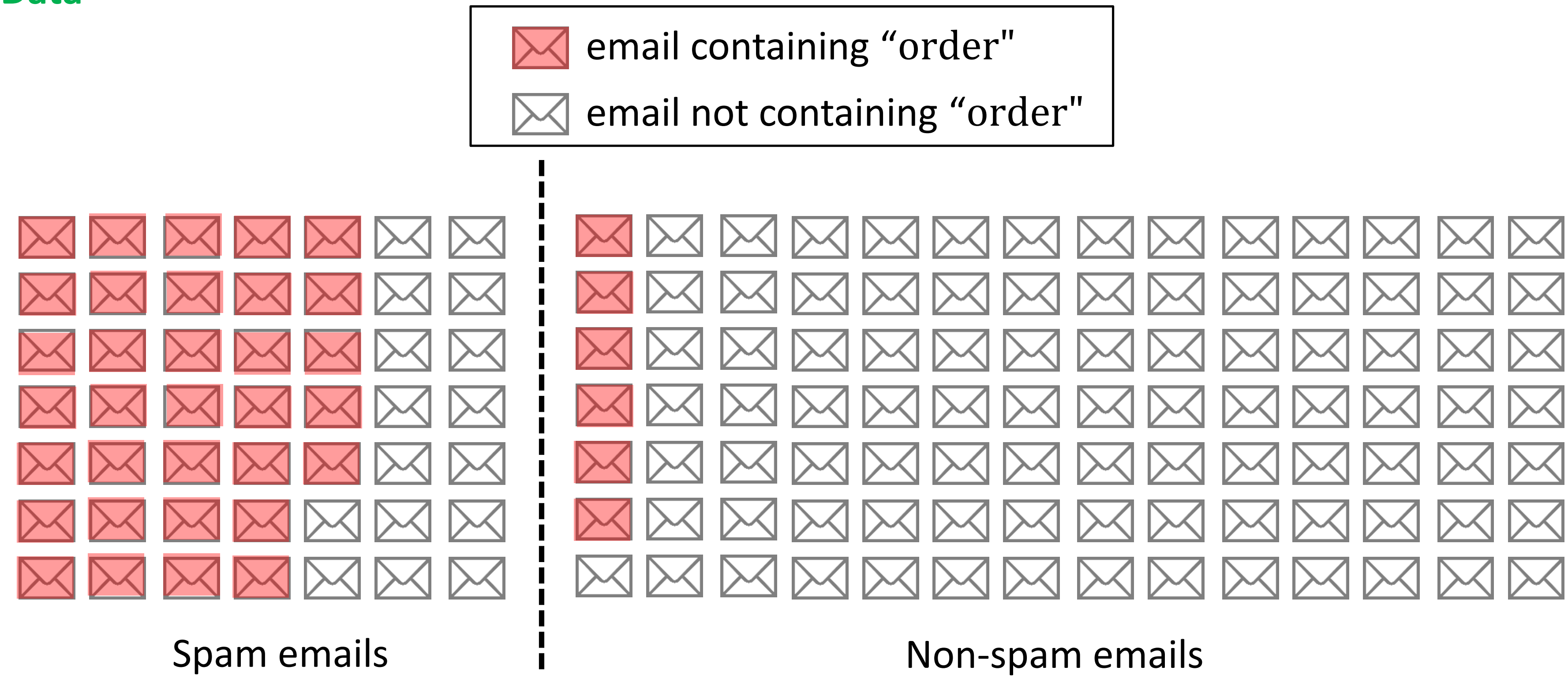
Data



$$P(\text{"hurry"}|y = \text{spam}) = \frac{38}{49} \qquad P(\text{"hurry"}|y = \text{nonspam}) = \frac{10}{98}$$

Naïve Bayes Classification

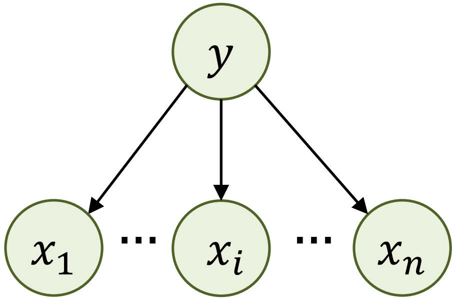
Data



$$P(\text{"order"} | y = \text{spam}) = \frac{33}{49} \qquad P(\text{"order"} | y = \text{nonspam}) = \frac{6}{98}$$

Naïve Bayes Classification

- Now it's a time to build a model for spam classifier

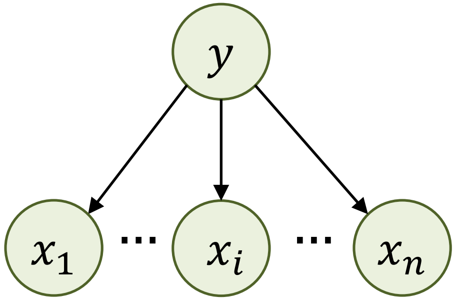


- Input $x = \{x_1, x_2, \dots, x_i, \dots, x_n\}$ $\quad x_i = \begin{cases} 1 & \text{if } i\text{th word is in the email} \\ 0 & \text{otherwise} \end{cases}$
- Output $y = \begin{cases} 1 & \text{if Spam} \\ 0 & \text{otherwise} \end{cases}$
- $p(y)$ is a prior on class
- Naïve Bayes Model assumes x_i (attributes) are conditionally independent given y model. Thus, the likelihood is

$$P(x|y) = \prod_{i=1}^{m} P(x_i|y)$$

Naïve Bayes Classification

- Now it's a time to build a model for spam classifier



- Posterior : $P(y|x) = \dfrac{P(x|y)P(y)}{P(x)} = \dfrac{P(x|y)P(y)}{\sum_{y \in Y} P(x|y)P(y)}$

- Class prediction : $\hat{y} = \underset{y}{\operatorname{argmax}} P(y|x) = \underset{y}{\operatorname{argmax}} \dfrac{P(x|y)P(y)}{\sum_y P(x|y)P(y)}$

$$= \underset{y}{\operatorname{argmax}} \prod_{i=1}^{m} P(x_i|y)\, P(y)$$

$$= \underset{y}{\operatorname{argmax}} \prod_{i=1}^{m} P(x_i|y)\, P(y)$$

Naïve Bayes Classification

- Training the model

$$D = (x_1, y_1), \dots, (x_i, y_i), \dots, (x_m, y_m) \qquad\qquad x_i = (x_{i1}, \dots, x_{in})$$

- Parameterization

$$\phi_{j|y=1} = p(x_j = 1|y = 1) \qquad 1 - \phi_{j|y=1} = p(x_j = 0|y = 1)$$

$$\phi_{j|y=0} = p(x_j = 1|y = 0) \qquad 1 - \phi_{j|y=0} = p(x_j = 0|y = 0)$$

$$\phi_y = p(y = 1) \qquad 1 - \phi_y = p(y = 0)$$

- Cost function = posterior

$$L\left(\phi_{j|y=1}, \phi_{j|y=0}, \phi_y\right) = \prod_{i=1}^{m} P(y_i, x_i|\boldsymbol{\phi}) = \prod_{i=1}^{m} P(x_i|y_i, \boldsymbol{\phi})p(y_i|\boldsymbol{\phi}) \prod_{i=1}^{m} \prod_{j=1}^{n} P\left(x_{ij}\middle|y_i, \boldsymbol{\phi}\right)p(y_i|\boldsymbol{\phi})$$

- Maximizing log likelihood with respect to the parameters leads

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_{ij} = 1 \cap y_i = 1\}}{\sum_{i=1}^{m} 1\{y_i = 1\}} \qquad \phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_{ij} = 1 \cap y_i = 0\}}{\sum_{i=1}^{m} 1\{y_i = 0\}} \qquad \phi_{y=0} = \frac{\sum_{i=1}^{m} 1\{y_i = 1\}}{m}$$

- Example

$$P(\text{cheap}|y = \text{spam}) = \frac{40}{49} \qquad P(\text{cheap}|y = \text{nonspam}) = \frac{21}{98}$$

$$P(\text{"hurry"}|y = \text{spam}) = \frac{38}{49} \qquad P(\text{"hurry"}|y = \text{nonspam}) = \frac{10}{98}$$

$$P(\text{"order"}|y = \text{spam}) = \frac{33}{49} \qquad P(\text{"order"}|y = \text{nonspam}) = \frac{6}{98}$$

x =(if cheap, if hurry, if order)

$p(y = \text{spam}) = p(y = \text{non} - \text{spam}) = 0.5$

- The new email has been arrived with $x = (1, 1, 0)$

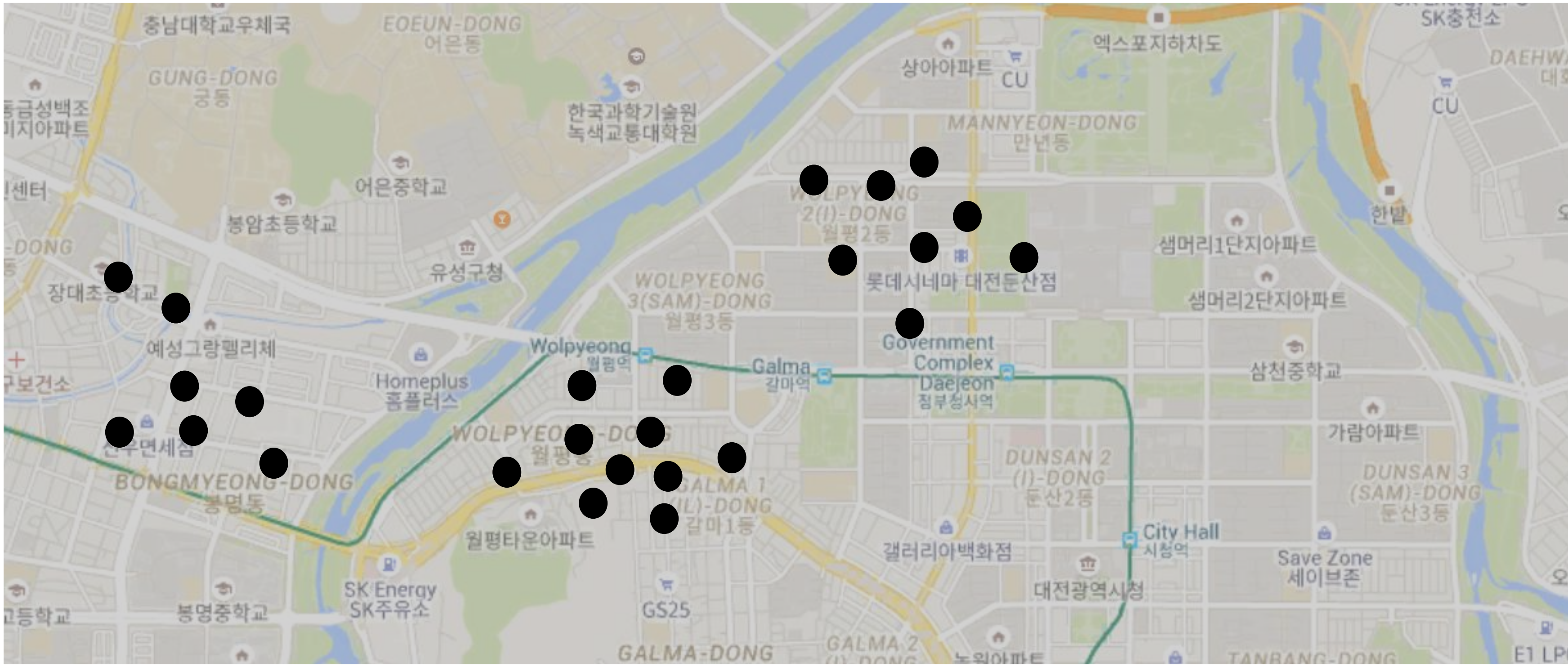
$$p\{y = 1|x = (1,1,0)\} \propto P\{x = (1,1,0)\,|y = 1\}P(y = 1) = \frac{40}{49}\frac{38}{49}\left(1 - \frac{33}{49}\right) = 0.206$$

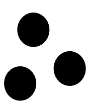
$$p\{y = 0|x = (1,1,0)\} \propto P\{x = (1,1,0)\,|y = 0\}P(y = 0) = \frac{21}{49}\frac{10}{49}\left(1 - \frac{6}{49}\right) = 0.077$$

$$p\{y = 0|x = (1,1,0)\} = \frac{0.206}{0.206 + 0.077} = 0.730, \qquad p\{y = 0|x = (1,1,0)\} = 0.270$$

Unsupervised Learning

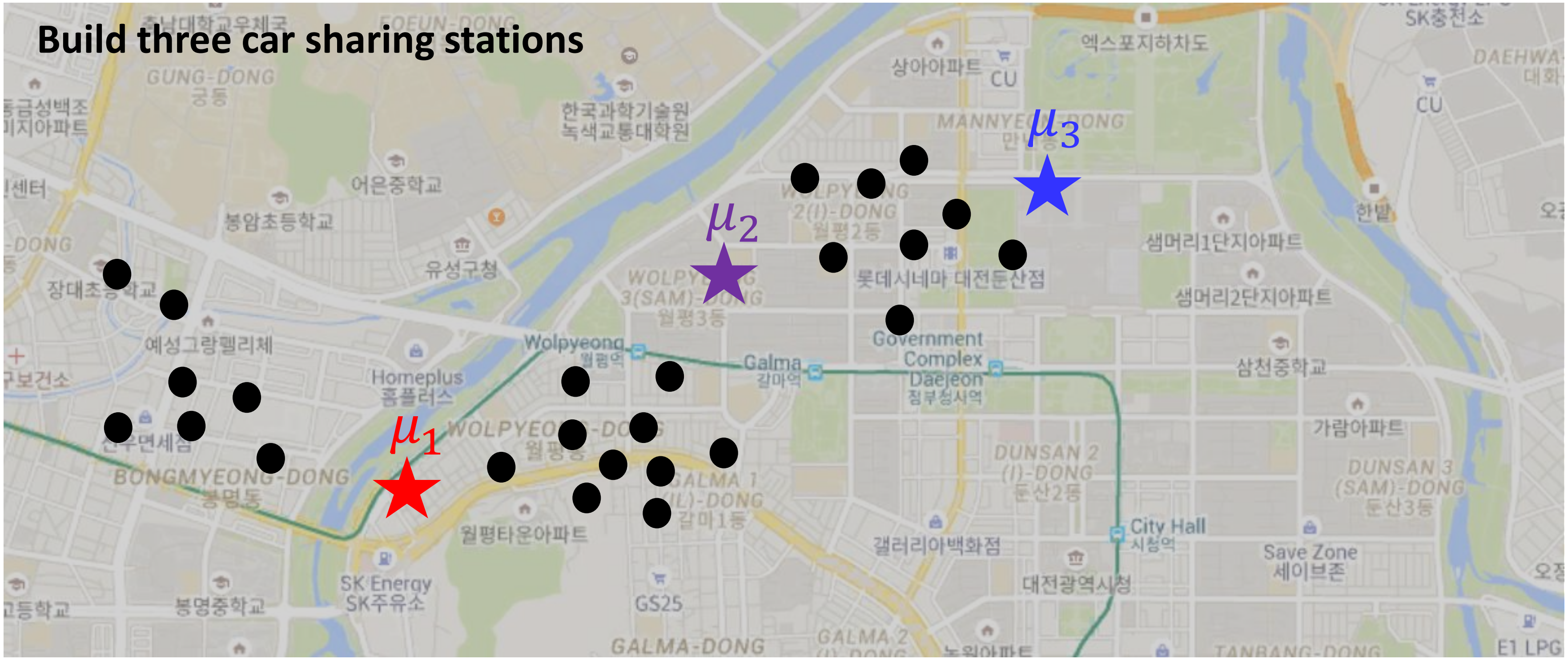
K-means algorithm



 Potential demands for car sharing service

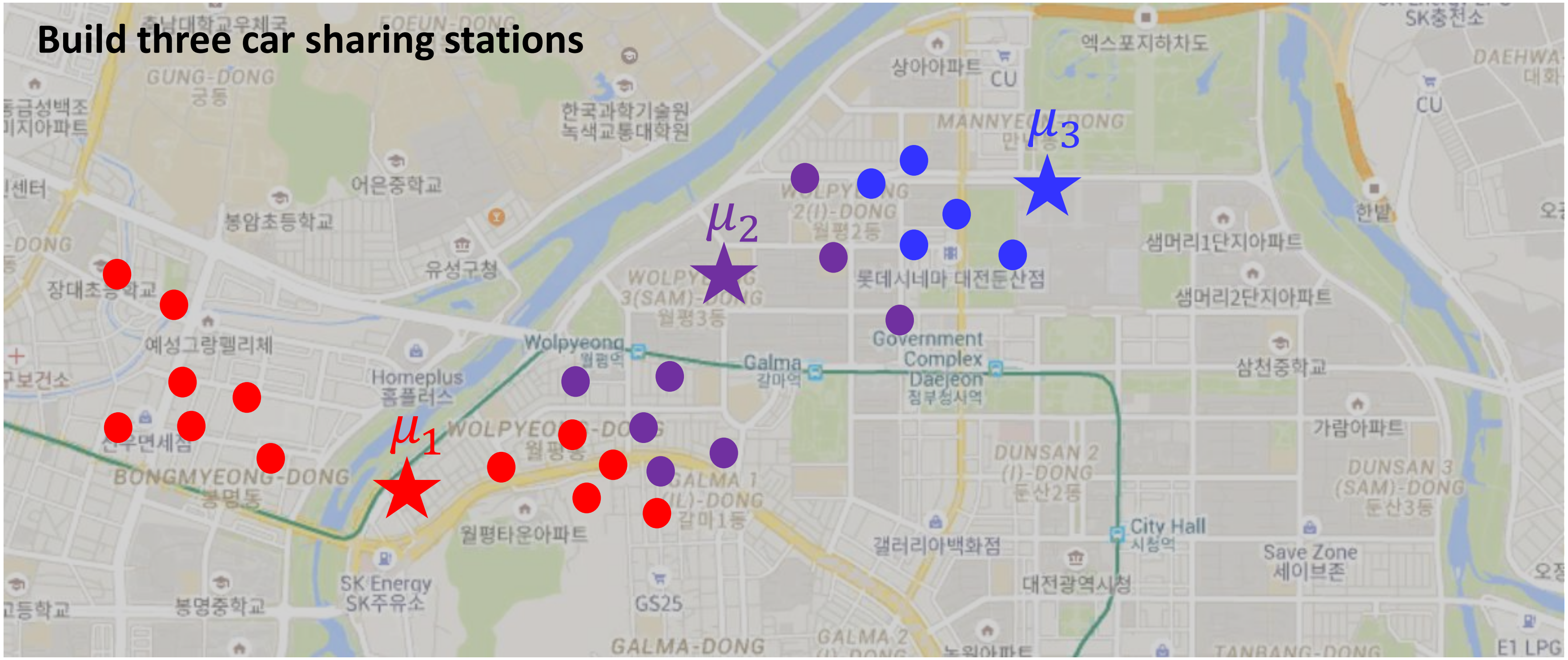
Build three car sharing stations

K-means algorithm



1. Initialize cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$ randomly

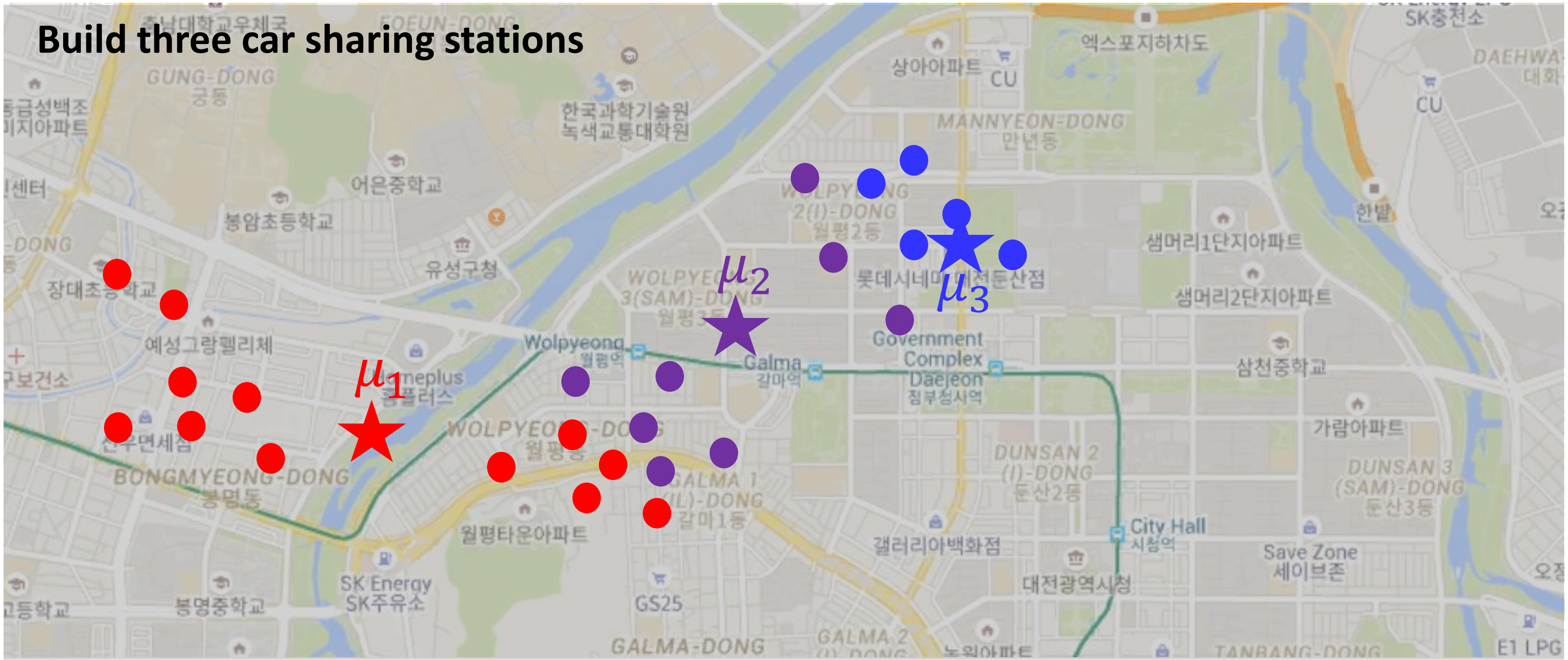
K-means algorithm



1. Initialize cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$ randomly
2. Repeat until convergence: {

    For every i, set

$$c^{(i)} := \underset{j}{\operatorname{argmin}} \left\| x^{(i)} - \mu_j \right\|^2$$



1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$ randomly
2. Repeat until convergence: {

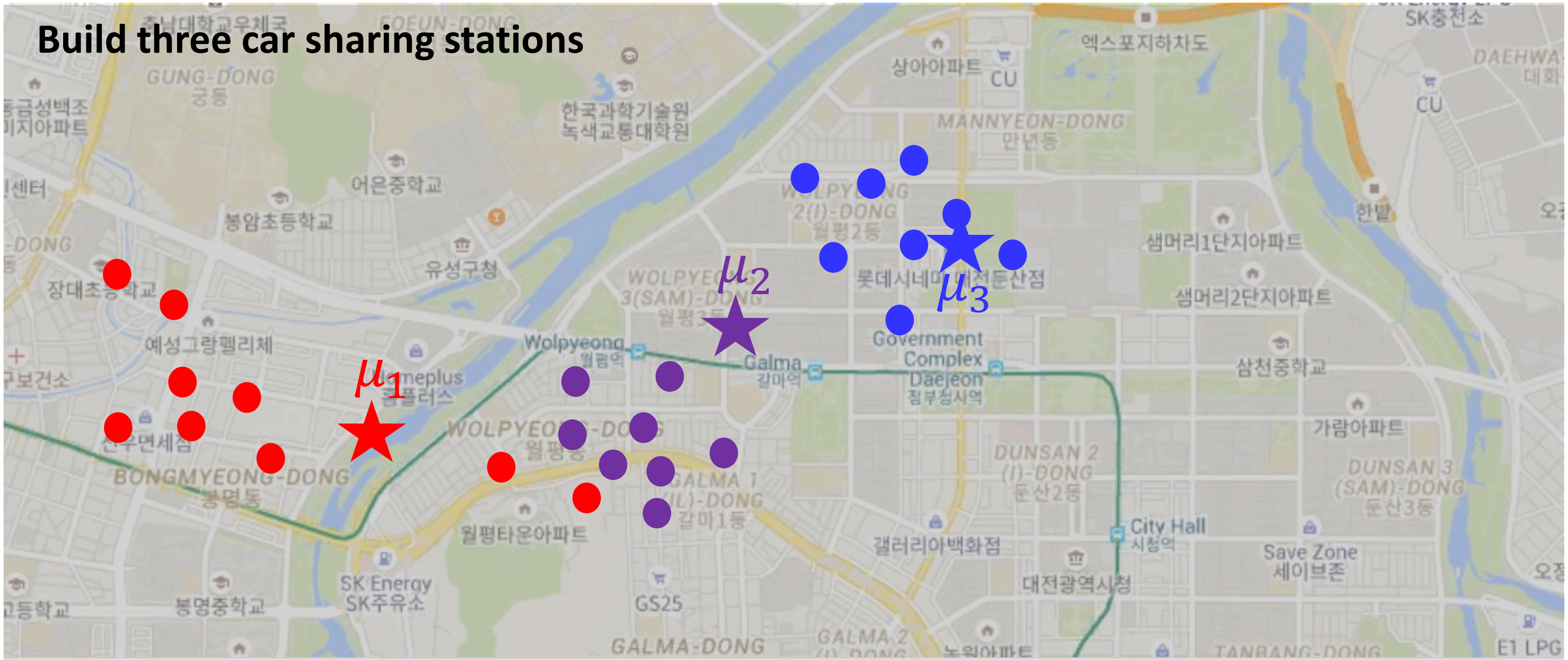
For every i, set

$$c^{(i)} := \underset{j}{\operatorname{argmin}} \left\| x^{(i)} - \mu_j \right\|^2$$

For every j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(j)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}



1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$ randomly
2. Repeat until convergence: {

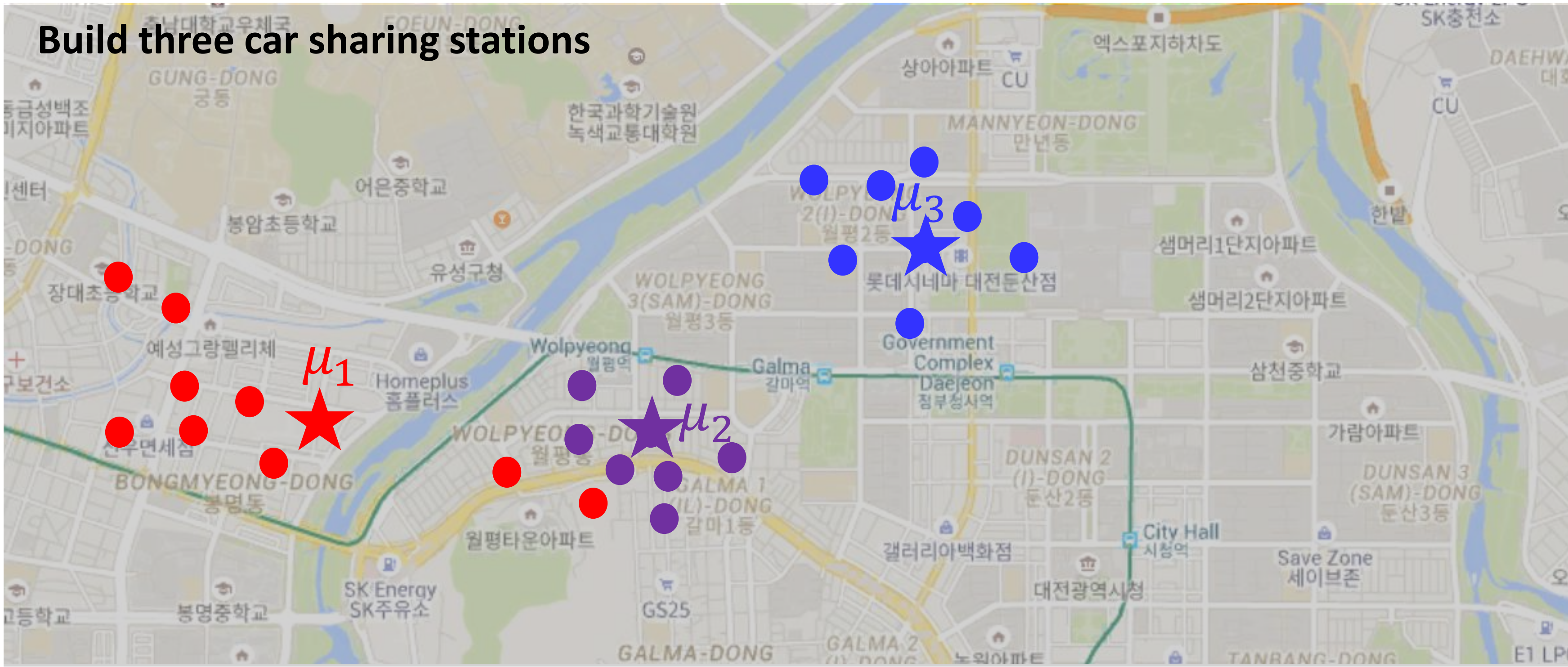
For every i, set

$$c^{(i)} := \underset{j}{\operatorname{argmin}} \left\| x^{(i)} - \mu_j \right\|^2$$

For every j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(j)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}



1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$ randomly
2. Repeat until convergence: {

For every i, set

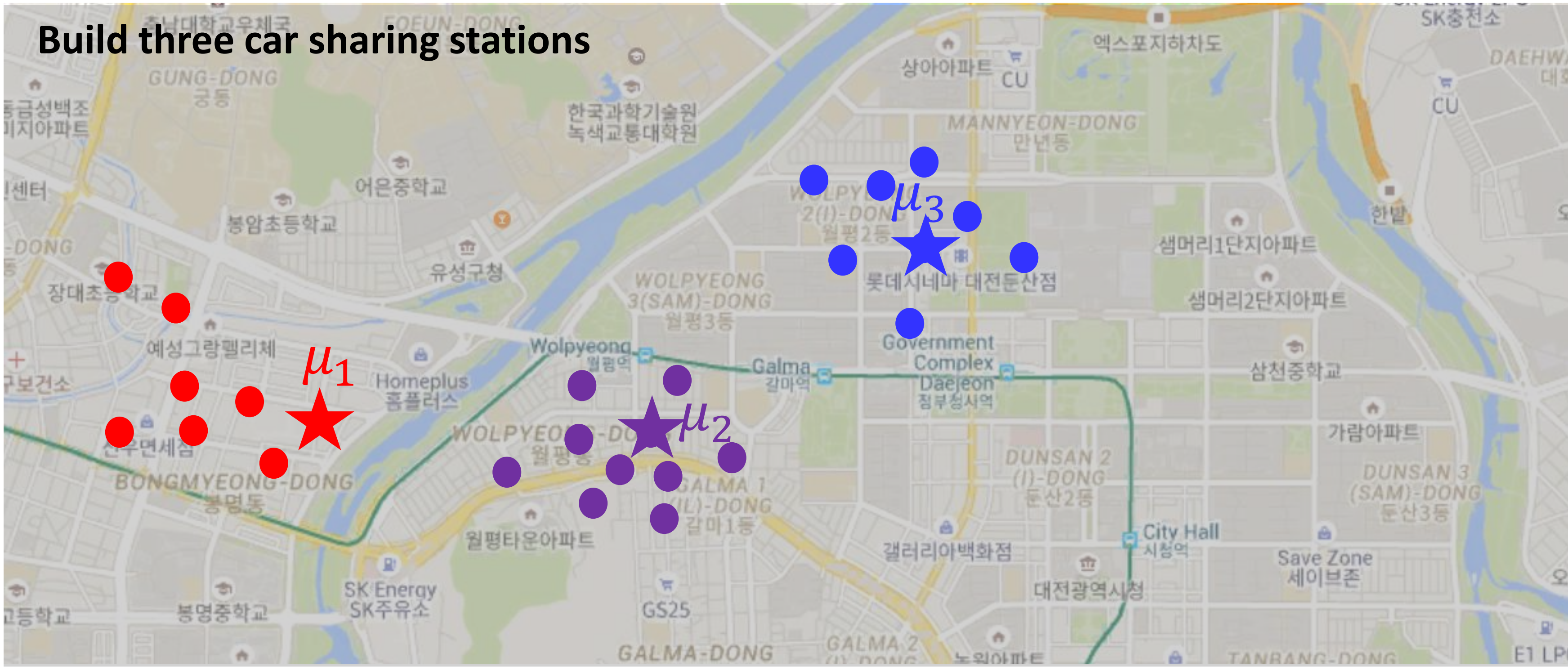
$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

For every j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(j)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}

K-means algorithm



1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$ randomly
2. Repeat until convergence: {

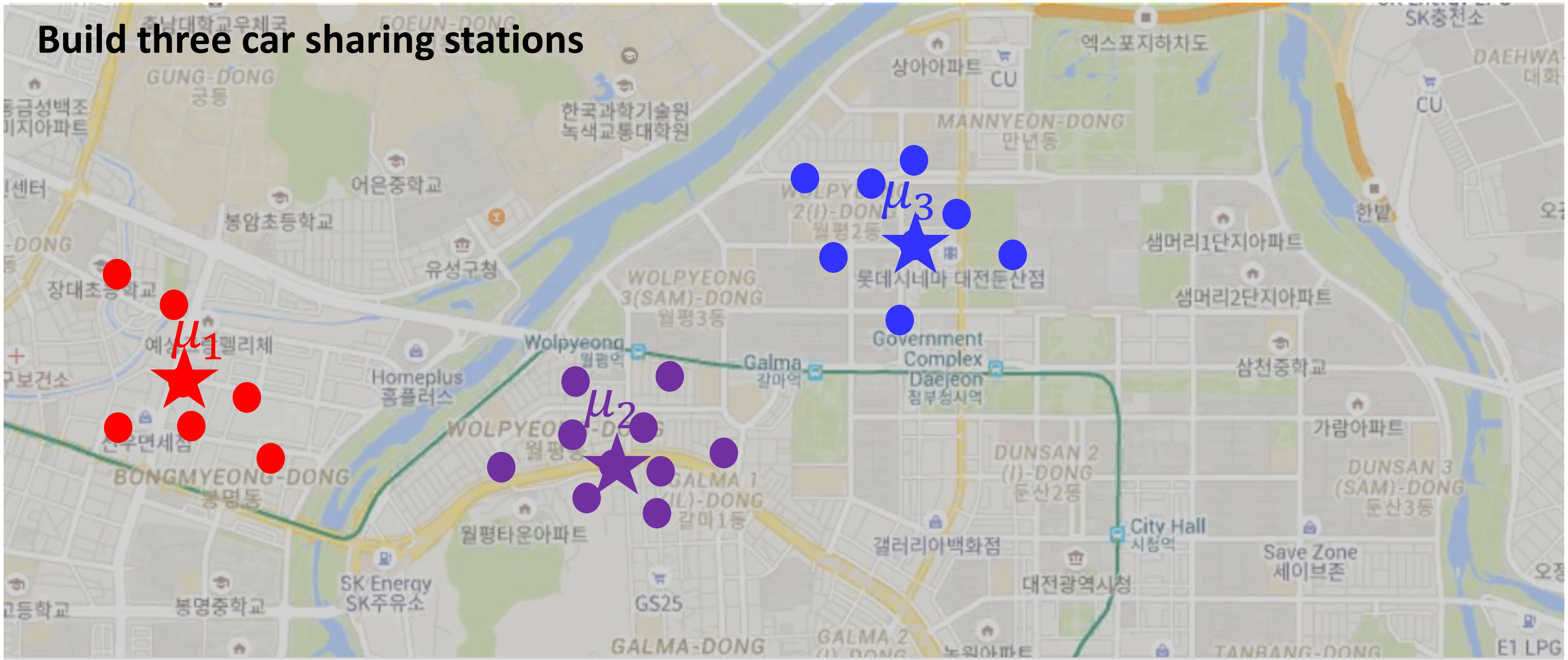
For every i, set

$$c^{(i)} := \underset{j}{\operatorname{argmin}} \left\| x^{(i)} - \mu_j \right\|^2$$

For every j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(j)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}



1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$ randomly
2. Repeat until convergence: {

For every i, set

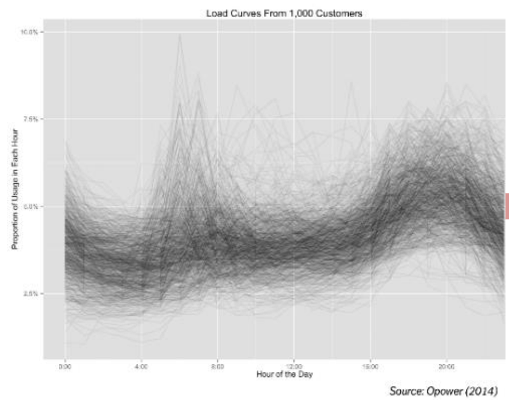
$$c^{(i)} := \underset{j}{\operatorname{argmin}} \left\| x^{(i)} - \mu_j \right\|^2$$

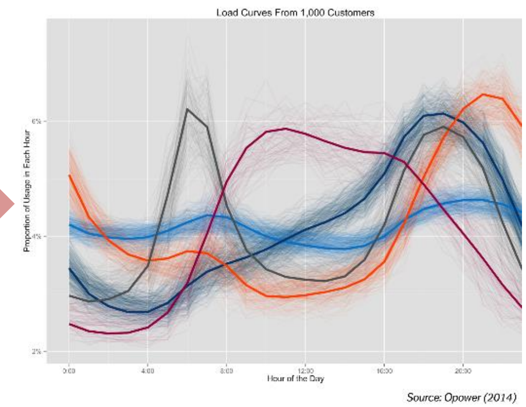
For every j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(j)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}

K-means algorithm : applications

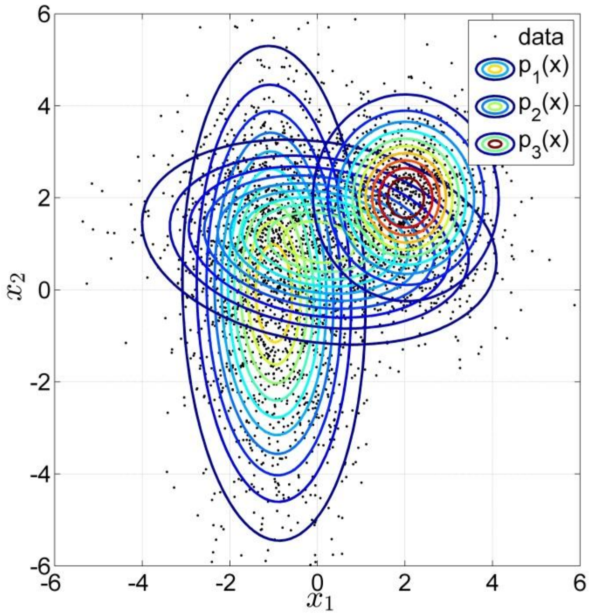




에너지 사용량 데이터 취득　　　에너지 사용 패턴 클러스터링

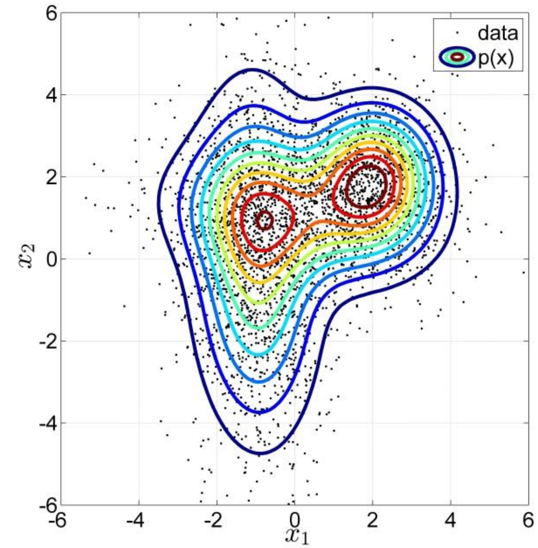
Gaussian Mixture Models: a density estimation method

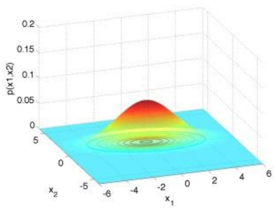
Modeling a probability density as a combination of K Gaussian components



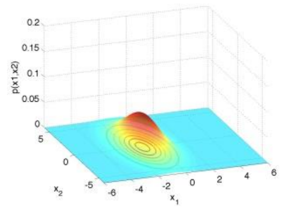
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p_k(\boldsymbol{x})\varphi_k$$

Weighted sum of Gaussian PDFs

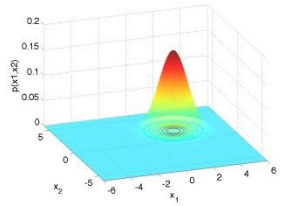




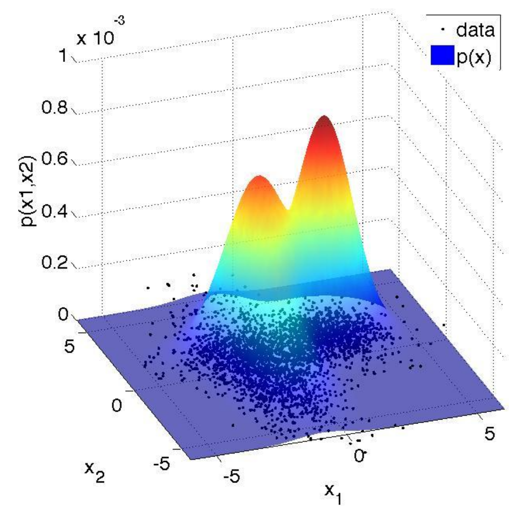
$p_1(\boldsymbol{x})$



$p_2(\boldsymbol{x})$



$p_3(\boldsymbol{x})$



Gaussian Mixture Models

- A probability density for input $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, is modeled as a weighed sum of K Gaussian distribution

$$p(\boldsymbol{x}; \boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} p_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\varphi_k$$

- the kth component density is of a form of Gaussian

$$p_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = N(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}_k|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\right)$$

  - φ_k: (mixture) weight for kth Gaussian component ($\sum_{k=1}^{K} \varphi_k = 1$)
  - $\boldsymbol{\mu}_k$: mean vector for the kth Gaussian component
  - $\boldsymbol{\Sigma}_k$: covariance matrix for the kth PDF

- The parameters, $\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$, for GMM

K: number of GPDFs
$\boldsymbol{\varphi} = \{\varphi_1, \dots, \varphi_K\}$: set of weights
$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$: set of mean vectors
$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K\}$: set of covariance matrices

are found as ones maximizing the log-likelihood of data

$$l(\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{K} p(\boldsymbol{x}^{(i)} | z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}; \boldsymbol{\varphi})$$

- Due to the latent variable $z^{(i)}$ representing the Gaussian PDF from which the data $x^{(i)}$ is drawn, the log likelihood is not explicitly defined → Difficult to optimize the GMM parameters $\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.

The parameters, $\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$, for GMM

K: number of GPDFs
$\boldsymbol{\varphi} = \{\varphi_1, \dots, \varphi_K\}$: set of weights
$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$: set of mean vectors
$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K\}$: set of covariance matrices

are found as ones maximizing the log-likelihood of data

$$l(\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{K} p(\boldsymbol{x}^{(i)} | z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}; \boldsymbol{\varphi})$$

Due to the latent variable $z^{(i)}$ representing the Gaussian PDF from which the data $x^{(i)}$ is drawn, the log likelihood is not explicitly defined → Difficult to optimize the GMM parameters $\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.

$$l(\boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\varphi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{K} p\left(\boldsymbol{x}^{(i)} \middle| z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) p\left(z^{(i)}; \boldsymbol{\varphi}\right)$$

$$= \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{K} Q_i\left(z^{(i)}\right) \frac{p(\boldsymbol{x}^{(i)}|z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}; \boldsymbol{\varphi})}{Q_i(z^{(i)})}$$

$$\geq \sum_{i=1}^{m} \sum_{z^{(i)}=1}^{K} Q_i\left(z^{(i)}\right) \log \frac{p(\boldsymbol{x}^{(i)}|z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}; \boldsymbol{\varphi})}{Q_i(z^{(i)})}$$

Jensen's inequality:
$f(E[\mathrm{X}]) \geq E[f(X)]$ if f is concave

Expected Maximization (EM) algorithm (Ref: Dempster, et.al., 1977)

Repeat until convergence {

E-Step: for each i, set

$$Q_i\left(z^{(i)}\right) = p(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\varphi})$$

Estimated parameters in the previous step

(soft estimation of $z^{(i)}$)

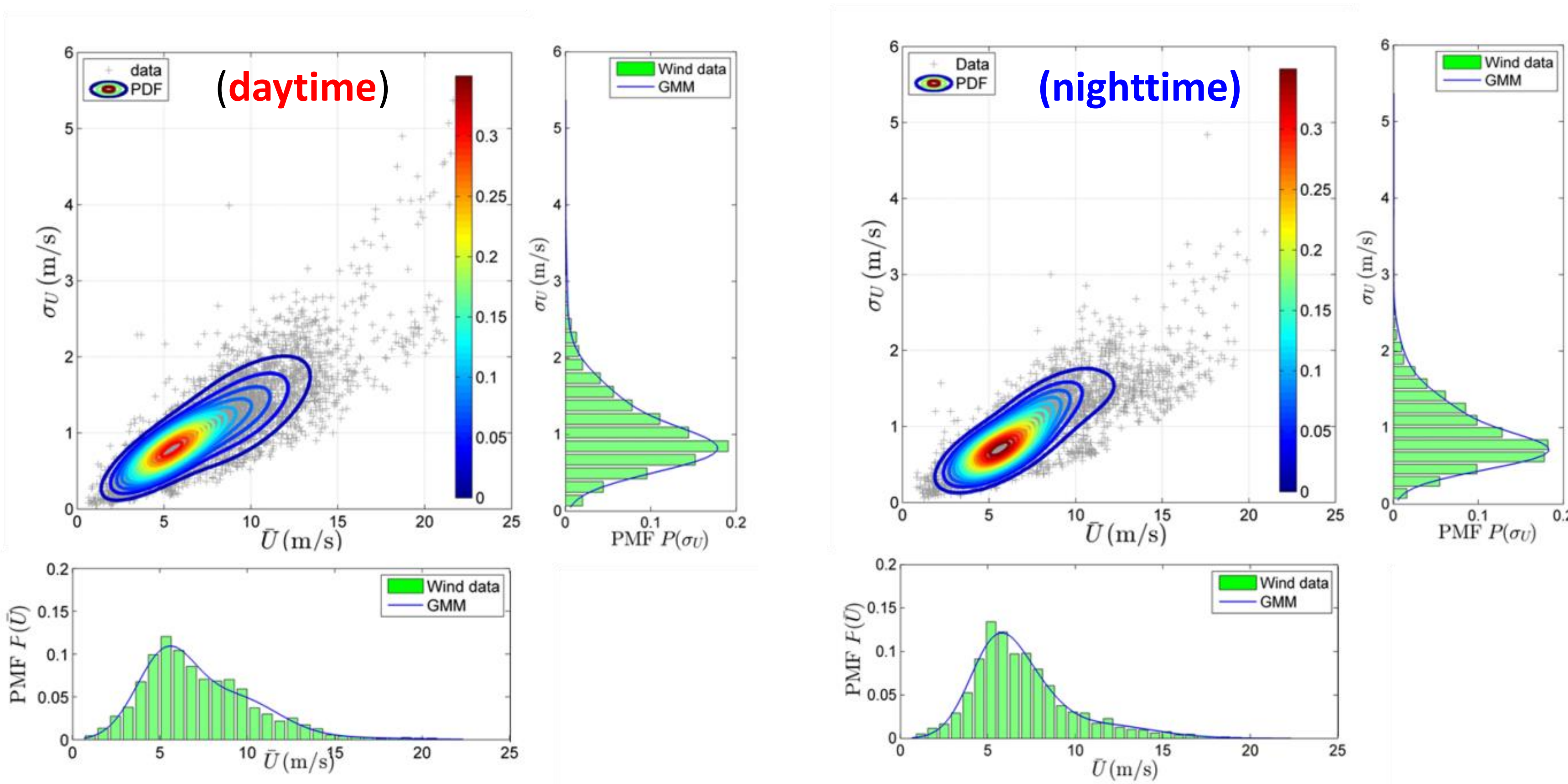
M-Step: maximize the following the log-likelihood with respect to $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\varphi}$

$$\sum_{i=1}^{m} \sum_{j=1}^{K} Q_i\left(z^{(i)} = j\right) \log \frac{p(\boldsymbol{x}^{(i)}|z^{(i)}=j; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}=j; \boldsymbol{\varphi})}{Q_i\left(z^{(i)}=j\right)}$$

}

Concave function→ can be easily maximized

Application to wind monitoring data



$p(\boldsymbol{x}|t =$ **daytime**$)$ $\qquad$ $P(\boldsymbol{x}|t =$ **nighttime**$)$

- The wind field characteristics are represented 2-dimensional PDF.
- The differences between the daytime and nighttime wind fields can be studied by comparing the two PDFs.