Cs229 (by Andrew Ng) – Notes by Xiaofan Wang

Name: XiaoFan Wang

Lecture 2. ┌ Linear Regression.
├ Gradient Descent ($\min J(\theta)$)
└ Normal Equation ($\min J(\theta)$)

一. Supervised Learning.

Giving a training set: $\{(x^{(i)}, y^{(i)}); i=1, \cdots, n\}$

To learn a function h: $x \to y$.
(steering direction)

┌ Regression: y is continuous.
└ Classification: y is discrete.

Training Set.
↓
Learning Algorithm
↓
$x \longrightarrow h \longrightarrow$ predicted y.
(Hypothesis)

二. Linear Regression.

Living Area Size	# Bedrooms	Price
$x_1^{(1)}$ 2104	3	400
$x_1^{(2)}$ 1600	3	330
⋮	⋮	⋮

x: two-dimensional vectors in $\mathbb{R}^2$

x_1: Size, x_2: # bedrooms.

$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

$h(x) = \sum_{j=0}^{2} \theta_j x_j$

parameter $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$ $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$ Always 1; Size; # Bedrooms.

1. Linear Function: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ (θ: weights)

Letting $x_0 = 1$: $h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T x$ ($h_\theta(x) = h(x) \approx y$)

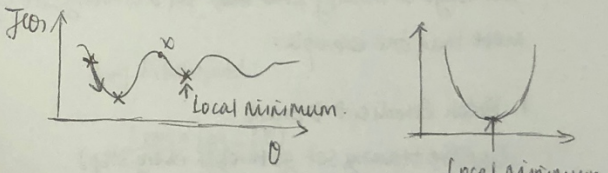
2. cost Function: $J(\theta) = \frac{1}{2} \cdot \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$

LMS Algorithm: choose θ to minimize $J(\theta)$.
(Least Mean Square).

⇒ Change θ to make $J(\theta)$ smaller.
Until hopefully we converge to θ that $\min J(\theta)$.
that's consider the Gradient Descent Algorithm.

三. Gradient Descent Algorithm.

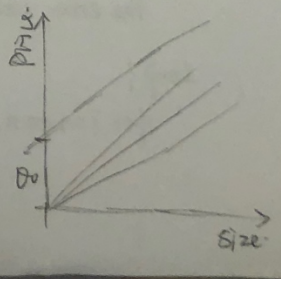
Start with θ. (set $\theta = \vec{0}$)
keep changing θ to reduce $J(\theta)$.



$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta)$ (α: Learning Rate)

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2$$
$$= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y)$$
$$= (h_\theta(x) - y) \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d} \theta_i x_i - y \right)$$
$$= (h_\theta(x) - y) \cdot x_j$$

i: training example #
x: Input (Features)
y: Output (Target Variable)
(x, y): Training Examples.
$(x^{(i)}, y^{(i)})$: i^{th} training Examples.
$x_j^{(i)}$: i runs from 1 to n.
d: # features.
$x^{(i)}$, $\mathbb{R}^{(d+1)}$ dimensional



LMS Rule: $\theta_j := \theta_j + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$

$$\begin{cases} \theta_j := \theta_j - \alpha \cdot (h_\theta(x) - y) x_j \\ \theta_j := \theta_j - \alpha \cdot \underbrace{\sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\frac{\partial J(\theta)}{\partial \theta_j}} \end{cases}$$

Two ways to modify LMS Rule for a training set of more than one example:

1. Batch Gradient Descent.

(entire training set go through every step).

Repeat until convergence {

$\theta_j := \theta_j + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$, (for every j)

}

$\theta_j := \theta_j + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$.

2. Stochastic Gradient Descent.

(update the parameters according to the gradient of the error with respect to that single training example only).

Loop {

for $i=1$ to n, {

$\theta_j := \theta_j + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$, (for every j)

}

}

~~$\theta := \theta + \alpha (y^{(i)} - h_\theta(x^{(i)})) x^{(i)}$.~~

Batch: $\theta_j := \theta_j + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$.

Stochastic: $\theta_j := \theta_j + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$.

Comparison: Batch Gradient Descent has to scan through the entire training set before taking a single step $\Rightarrow$ a costly operation if n is large.

Stochastic Gradient Descent can start making progress right away, and continues to make progress with each example it looks at.

IV. Normal Equation.

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{n1}} & \cdots & \frac{\partial f}{\partial A_{nd}} \end{bmatrix} \qquad \nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \end{bmatrix}$$

$\nabla_\theta J(\theta) = \text{set } \vec{0}$

$J(\theta) = \frac{1}{2} \cdot \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})^2$

since $h_\theta(x^{(i)}) = (x^{(i)})^T \cdot \theta$,

$$X\theta - \vec{y} = \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(n)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(n)}) - y^{(n)} \end{bmatrix}$$

using the fact. $\vec{z}^T \cdot \vec{z} = \sum_i z_i^2$.

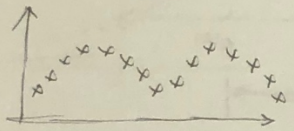
$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \tfrac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y}) \quad \vec{y}^T(X\theta) \\ &= \tfrac{1}{2} \nabla_\theta ((X\theta)^T X\theta - (X\theta)^T\vec{y} - \vec{y}^T(X\theta) + \vec{y}^T\vec{y}) \\ &= \tfrac{1}{2} \nabla_\theta (\theta^T (X^T X)\theta - 2(X^T\vec{y})^T\theta) \\ &= \tfrac{1}{2} (2X^TX\theta - 2X^T\vec{y}) \\ &= X^TX\theta - X^T\vec{y} \quad \text{(set derivatives to 0)} \\ X^TX\theta &= X^T\vec{y} \end{aligned}$$

$\Rightarrow \theta = (X^TX)^{-1} X^T\vec{y}$ (optimal value).

Name: XiaoFan Wang

Lec 3 — Linear Regression Model
- Locally Weighted Regression
- Probabilistic Interpretation
- Logistic Regression
- Newton's Method

一. Locally Weighted Regression.



$$h_\theta(x) = \sum_{j=0}^{d} \theta_j x_j = \theta^T x$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{\#n} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\begin{cases} \theta_0 + \theta_1 x_1 \\ \theta_0 + \theta_1 x_1 + \theta_2 x^2 \\ \theta_0 + \theta_1 x + \theta_2 \sqrt{x} + \theta_3 \log(x) \end{cases}$$

(x_1, x_2, x_3)

"Parametric" Learning Algorithm:

⇒ Fit Fixed set of parameters (θ_i) to data.

"Non-parametric" Learning Algorithm:

⇒ # parameters grow linearly with the size of data.

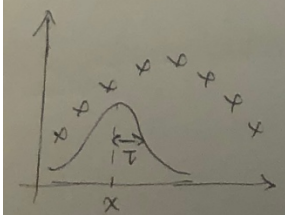
To evaluate h at certain x:

LR: Fit θ to minimize $\frac{1}{2}\sum_i (y^{(i)} - \theta^T x^{(i)})^2$. return $\theta^T x$

Locally Weighted Regression:

Fit θ to minimize $\sum_{i=1}^{n} w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$

$$w^{(i)} = \exp\left(-\frac{\|x^{(i)} - x\|^2}{2\tau^2}\right)$$



If $\|x^{(i)} - x\|$ small, $w^{(i)} \approx 1$

$\|x^{(i)} - x\|$ large, $w^{(i)} \approx 0$

τ: bandwidth.

二. Probabilistic Interpretation

Assume: $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$

$\varepsilon^{(i)} \sim N(0, \sigma^2)$

$$p(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right)$$

$$p(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

"parameterized by" ↑ ; Random variable ↓ ; Mean ↓

$$y^{(i)} | x^{(i)}; \theta \sim N(\theta^T x^{(i)}; \sigma^2)$$

$$L(\theta) = p(\vec{y} | x; \theta)$$

$$= \prod_{i=1}^{n} p(y^{(i)} | x^{(i)}; \theta)$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

Log Likelihood:

$$l(\theta) = \log L(\theta)$$

$$= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\cdot\sigma} \exp\left(\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$$= n \log \frac{1}{\sqrt{2\pi}\cdot\sigma} + \sum_{i=1}^{n} \frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

MLE: Maximum Likelihood Estimation.

Choose θ to maximize $L(\theta)$

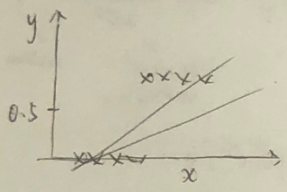
Minimize $\frac{1}{2}\sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 = J(\theta)$

Ⅲ. Logistic Regression

Classification: ① Make assumption $p(y|x;\theta)$

② Compute θ by MLE

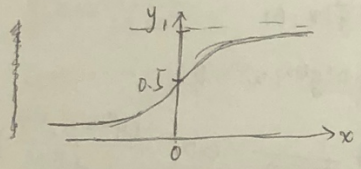
$y \in \{0,1\}$ Binary Classification



Want $h_\theta(x) \in [0,1]$

$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

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



Sigmoid function
= logistic function

Linear Regression: $h_\theta(x) = \theta^T \cdot x$

$p(y=1 \mid x;\theta) = h_\theta(x)$

$p(y=0 \mid x;\theta) = 1 - h_\theta(x)$

$p(y|x;\theta) = h_\theta(x)^y (1-h_\theta(x))^{1-y}$

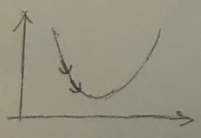
$L(\theta) = p(\vec{y} \mid x;\theta)$

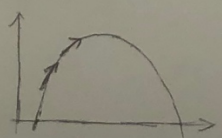
$= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)};\theta)$

$= \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1-h_\theta(x^{(i)}))^{1-y^{(i)}}$

$l(\theta) = \log L(\theta)$

$= \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))$





$\theta_j := \theta_j + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)})) \cdot x_j^{(i)}$

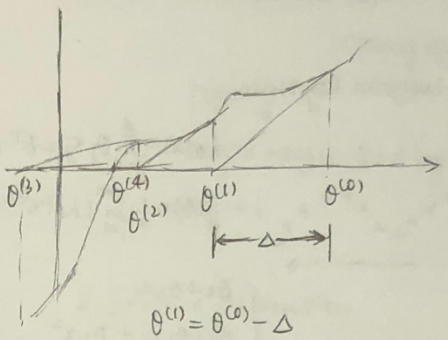
$\frac{\partial l(\theta)}{\partial \theta}$

Ⅳ. Newton's Method

Want: maximize $l(\theta)$

$l'(\theta) = 0$.

Max / Min $\Leftrightarrow$ ~~derivative~~ derivative $= 0$



$\theta^{(1)} = \theta^{(0)} - \Delta$

$f'(\theta^{(0)}) = \frac{f(\theta^{(0)})}{\Delta} = \frac{\text{Height}}{\text{Base}}$ $\left(\Delta = \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}\right)$

$\theta^{(t+1)} = \theta^{(t)} - \frac{f(\theta^t)}{f'(\theta^t)}$ $(f(\theta) = l'(\theta))$

$\theta^{(t+1)} = \theta^{(t)} - \frac{l'(\theta^t)}{l''(\theta^t)}$

Quadratic convergence: $0.1 \to 0.01 \to 0.0001$

$\vec{\theta}$: $\theta^{(t+1)} = \theta^{(t)} + H^{-1} \nabla_\theta l$ $\quad \mathbb{R}^{(d+1)\times(d+1)}$

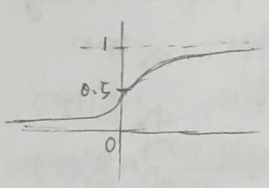
Vector $\mathbb{R}^{d+1}$

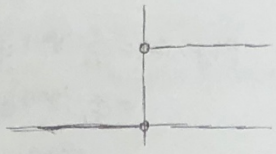
Hessian H: $H_{ij} = \frac{\partial^2 l}{\partial \theta_i \partial \theta_j}$

Name: XiaoFan Wang

Lec 4. Perception
- Exponential Family
- Generalized Linear Models.
- Softmax Regression (Multi-class Classification)

一. Perceptron





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

$h_\theta(x)=\frac{1}{1+e^{-\theta^T x}}$

$g(z)=\begin{cases}1 & z\geq 0\\ 0 & z<0\end{cases}$

$h_\theta(x)=g(\theta^T x)$

Perception Learning Algorithm:

$$\theta_j := \theta_j + \alpha(y^{(i)}-h_\theta(x^{(i)}))\,x_j^{(i)}$$

二. Exponential Family

$$p(y;\eta)=b(y)\exp(\eta^T T(y)-a(\eta))$$

η : natural parameter

$T(y)$: sufficient statistic for Distributions.

$a(\eta)$: Log Partition Function

1. Bernoulli (ϕ) Distribution.

$y\in\{0,1\}$ $\begin{cases}p(y=1;\phi)=\phi\\ p(y=0;\phi)=1-\phi\end{cases}$

$$p(y;\phi)=\phi^y(1-\phi)^{1-y}$$
$$=\exp(y\log\phi+(1-y)\log(1-\phi))$$
$$=\exp\left(\left(\log\left(\frac{\phi}{1-\phi}\right)\right)y+\log(1-\phi)\right)$$

Thus, $\eta=\log\left(\frac{\phi}{1-\phi}\right)$, $\phi=\frac{1}{1+e^{-\eta}}$

(family sigmoid func.)

$T(y)=y$.

$a(\eta)=-\log(1-\phi)=\log(1+e^\eta)$

$b(y)=1$

2. Gaussian Distribution

$$p(y;\mu)=\frac{1}{\sqrt{2\pi}}\exp\left(\frac{-1}{2}(y-\mu)^2\right)$$
$$=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right)\cdot\exp\left(\mu y-\frac{1}{2}\mu^2\right)$$

Thus, $\eta=\mu$

$T(y)=y$.

$a(\eta)=\frac{\mu^2}{2}=\frac{\eta^2}{2}$

$b(y)=\frac{1}{\sqrt{2\pi}}\cdot\exp\left(-\frac{y^2}{2}\right)$

三. Constructing GLM.

Assumption:
1. $y|x;\theta\sim$ Exponential Family (η)
2. predict the expected value: $T(y)=y$
   prediction output: $h(x)=E[y|x]$
3. $\eta=\theta^T\cdot x$
   ↓
   vector valued $\eta_i=\theta_i^T\cdot x$

1. Ordinary Least Squares.

Gaussian $N(\mu, \sigma^2)$

$y|x;\theta \sim N(\mu, \sigma^2)$

Gaussian as an exponential Family Distribution: $\mu = \eta$.

$$h_\theta(x) = E[y|x;\theta]$$
$$= \mu$$
$$= \eta$$
$$= \theta^T x$$

2. Logistic Regression.

Bernoulli Distribution as an Exponential Family:

$$\begin{cases} \phi = \frac{1}{1+e^{-\eta}} \\ y|x;\theta \sim \text{Bernoulli}(\phi) \\ E[y|x;\theta] = \phi \end{cases}$$

$$h_\theta(x) = E[y|x;\theta]$$
$$= \phi$$
$$= \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}$$

3. Softmax Regression. (Multinomial Distribution)

$\phi_1 \cdots \phi_k$ satisfy $\sum_{i=1}^{k} \phi_i = 1$

$\phi_i = p(y=i;\phi)$

$p(y=k;\phi) = 1 - \sum_{i=1}^{k-1} \phi_i$

$\phi_k = 1 - \sum_{i=1}^{k-1} \phi_i$

Define $T(y) \in \mathbb{R}^{k-1}$:

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad T(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad T(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \cdots T(k-1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \quad T(k) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$(T(y))_i$: the ith vector $T(y)$

$1\{2=3\} = 0$

$1\{3 = 5-2\} = 1$

$(T(y))_i = 1\{y=i\}$

$E[(T(y))_i] = p(y=i) = \phi_i$

Multinomial as a ~~family~~ member of Exponential Family:

$$p(y;\phi) = \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \cdots \phi_k^{1\{y=k\}}$$
$$= \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \cdots \phi_k^{1-\sum_{i=1}^{k-1} 1\{y=i\}}$$
$$= \phi_1^{(T(y))_1} \phi_2^{(T(y))_2} \cdots \phi_k^{1-\sum_{i=1}^{k-1}(T(y))_i}$$
$$= \exp((T(y))_1 \log(\phi_1) + (T(y))_2 \log(\phi_2) + \cdots + (1 - \sum_{i=1}^{k-1}(T(y))_i)\log(\phi_k))$$
$$= \exp((T(y))_1 \log(\frac{\phi_1}{\phi_k}) + (T(y))_2 \log(\frac{\phi_2}{\phi_k}) + \cdots + (T(y))_{k-1} \log(\frac{\phi_{k-1}}{\phi_k}) + \log(\phi_k))$$
$$= b(y) \exp(\eta^T T(y) - a(\eta))$$

$$\eta = \begin{bmatrix} \log(\phi_1/\phi_k) \\ \log(\phi_2/\phi_k) \\ \vdots \\ \log(\phi_{k-1}/\phi_k) \end{bmatrix}, \quad a(\eta) = -\log(\phi_k), \quad b(y) = 1$$

$\eta_i = \log \frac{\phi_i}{\phi_k}$, $e^{\eta_i} = \frac{\phi_i}{\phi_k}$, $\phi_k \cdot e^{\eta_i} = \phi_i$

$\phi_k \sum_{i=1}^{k} e^{\eta_i} = \sum_{i=1}^{k} \phi_i = 1$, $\phi_k = \frac{1}{\sum_{i=1}^{k} e^{\eta_i}}$

$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_i}}$, $p(y=i|x;\theta) = \phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_i}} = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_i^T x}}$

log-likelihood:

$$l(\theta) = \sum_{i=1}^{n} \log p(y^{(i)}|x^{(i)};\theta)$$
$$= \sum_{i=1}^{n} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y^{(i)}=l\}}$$

$$h_\theta(x) = E[T(y)|x;\theta]$$
$$= E\left[\begin{bmatrix} 1\{y=1\} \\ \vdots \\ 1\{y=k-1\} \end{bmatrix} \middle| x;\theta \right] = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{\exp(\theta_1^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \\ \vdots \\ \frac{\exp(\theta_{k-1}^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \end{bmatrix}$$

Name: XiaoFan Wang

Lec 5 & Generative Learning Algorithms.
Lec 6

Discriminative Learning Algorithms :

Input x mapping directly to label $\{0,1\}$.

Generative Learning Algorithms:

try to model $p(x|y)$. $p(y)$.

For instance: Dog(0). Elephant (1).

$\begin{cases} p(x|y=0): \text{dogs' features.} \\ p(x|y=1): \text{elephants' features.} \end{cases}$

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

$$p(x) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$$

$$\arg\max_y p(y|x) = \arg\max_y \frac{p(x|y)p(y)}{p(x)}$$

$$= \arg\max_y p(x|y)p(y)$$

一. Gaussian Discriminant Analysis. (GDA)

1. Multivariate Normal Distribution.
(Multivariate Gaussian Distribution)

$\begin{cases} \text{mean vector } \mu \in \mathbb{R}^d \\ \text{covariance matrix } \Sigma \in \mathbb{R}^{d\times d} \quad (\Sigma \geq 0) \\ N(\mu, \Sigma) \end{cases}$

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$E[x] = \int_x x p(x;\mu,\Sigma)dx = \mu$$

$$\text{Cov}(z) = E[(z-E[z])(z-E[z])^T]$$

$$= E[zz^T] - (E[z])(E[z])^T$$

If $x \sim N(\mu,\Sigma)$, then $\text{Cov}(x) = \Sigma$

2. Gaussian Discriminant Analysis Model

$$y \sim \text{Bernoulli}(\phi)$$

$$x|y=0 \sim N(\mu_0, \Sigma_1)$$

$$x|y=1 \sim N(\mu_1, \Sigma_1)$$

$$p(y) = \phi^y(1-\phi)^{1-y}$$

$$p(x|y=0) = \frac{1}{(2\pi)^{d/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma_1^{-1}(x-\mu_0)\right)$$

$$p(x|y=1) = \frac{1}{(2\pi)^{d/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right)$$

Log-likelihood:

$$l(\phi,\mu_0,\mu_1,\Sigma_1) = \log\prod_{i=1}^n p(x^{(i)},y^{(i)};\phi,\mu_0,\mu_1,\Sigma_1)$$

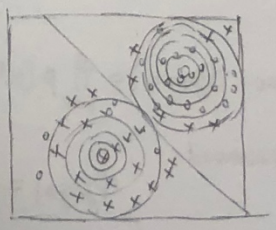
$$= \log\prod_{i=1}^n p(x^{(i)}|y^{(i)};\mu_0,\mu_1,\Sigma_1)p(y^{(i)};\phi)$$

Maximize l: $\phi = \frac{1}{n}\sum_{i=1}^n 1\{y^{(i)}=1\}$

$$\mu_0 = \frac{\sum_{i=1}^n 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)}=0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^n 1\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)}=1\}}$$

$$\Sigma_1 = \frac{1}{n}\cdot\sum_{i=1}^n (x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T$$



3. GDA and Logistic Regression

$$p(y=1 \mid x; \Phi, \Sigma_1, \mu_0, \mu_1) = \frac{1}{1+\exp(-\theta^T x)}$$

If $x|y=0 \sim \text{Poisson}(\lambda_0)$
$x|y=1 \sim \text{poisson}(\lambda_1)$
$\Rightarrow p(y|x)$: logistic

=. Naive Bayes.

$p(x|y)$: x_i is conditionally independent given y.
(Naive Bayes Assumption).

Naive Bayes Classifier :

$$p(x_1 \cdots x_{50000} | y)$$
$$= p(x_1|y)\, p(x_2|y, x_1)\, p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1 \cdots x_{49999})$$
$$= p(x_1|y)\, p(x_2|y)\, p(x_3|y) \cdots p(x_{50000}|y)$$
$$= \prod_{j=1}^{d} p(x_j|y)$$

$$\begin{cases} \Phi_{j|y=1} = p(x_j=1 \mid y=1) \\ \Phi_{j|y=0} = p(x_j=1 \mid y=0) \\ \Phi_y = p(y=1) \end{cases}$$

$$L(\Phi_y, \Phi_{j|y=0}, \Phi_{j|y=1}) = \prod_{i=1}^{n} p(x^{(i)}, y^{(i)})$$

Maximum Likelihood :

$$\begin{cases} \Phi_{j|y=1} = \dfrac{\sum_{i=1}^{n} 1\{x_j^{(i)}=1 \wedge y^{(i)}=1\}}{\sum_{i=1}^{n} 1\{y^{(i)}=1\}} \\ \Phi_{j|y=0} = \dfrac{\sum_{i=1}^{n} 1\{x_j^{(i)}=1 \wedge y^{(i)}=0\}}{\sum_{i=1}^{n} 1\{y^{(i)}=0\}} \\ \Phi_y = \dfrac{\sum_{i=1}^{n} 1\{y^{(i)}=1\}}{n} \end{cases}$$

$$p(y=1|x) = \frac{p(x|y=1)\,p(y=1)}{p(x)}$$
$$= \frac{(\prod_{j=1}^{d} p(x_j|y=1)) \cdot p(y=1)}{(\prod_{j=1}^{d} p(x_j|y=1))p(y=1) + (\prod_{j=1}^{d} p(x_j|y=0))p(y=0)}$$

1. Laplace Smoothing.

$$\Phi_{35000|y=1} = \frac{\sum_{i=1}^{n} 1\{x_{35000}^{(i)}=1 \wedge y^{(i)}=1\}}{\sum_{i=1}^{n} 1\{y^{(i)}=1\}} = 0.$$

$$\Phi_{35000|y=0} = \frac{\sum_{i=1}^{n} 1\{x_{35000}^{(i)}=1 \wedge y^{(i)}=0\}}{\sum_{i=1}^{n} 1\{y^{(i)}=0\}} = 0.$$

Maximize Likelihood : $\Phi_j = \dfrac{\sum_{i=1}^{n} 1\{z^{(i)}=j\}}{n}$

Φ_j might end up as zero $\Rightarrow$ problem

Laplace Smoothing : $\Phi_j = \dfrac{1+\sum_{i=1}^{n} 1\{z^{(i)}=j\}}{k+n}$.

$$\Phi_{j|y=1} = \frac{1+\sum_{i=1}^{n} 1\{x_j^{(i)}=1 \wedge y^{(i)}=1\}}{2+\sum_{i=1}^{n} 1\{y^{(i)}=1\}}$$

$$\Phi_{j|y=0} = \frac{1+\sum_{i=1}^{n} 1\{x_j^{(i)}=1 \wedge y^{(i)}=0\}}{2+\sum_{i=1}^{n} 1\{y^{(i)}=0\}}$$

2. Event Models for Text Classification

$$p(y) \prod_{j=1}^{d} p(x_j|y)$$

likelihood : $L(\Phi_y, \Phi_{k|y=0}, \Phi_{k|y=1})$

$$= \prod_{i=1}^{n} p(x^{(i)}, y^{(i)})$$
$$= \prod_{i=1}^{n} \left(\prod_{j=1}^{d_i} p(x_j^{(i)}|y; \Phi_{k|y=0}, \Phi_{k|y=1})\right) \cdot p(y^{(i)}; \Phi_y)$$

Maximizing Likelihood:

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{d_i} 1\{x_j^{(i)}=k \wedge y^{(i)}=1\}}{\sum_{i=1}^{n} 1\{y^{(i)}=1\}\, d_i}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{d_i} 1\{x_j^{(i)}=k \wedge y^{(i)}=0\}}{\sum_{i=1}^{n} 1\{y^{(i)}=0\}\, d_i}$$

$$\phi_y = \frac{\sum_{i=1}^{n} 1\{y^{(i)}=1\}}{n}$$

If applying Laplace smoothing:

$$\phi_{k|y=1} = \frac{1+\sum_{i=1}^{n}\sum_{j=1}^{d_i} 1\{x_j^{(i)}=k \wedge y^{(i)}=1\}}{|V|+\sum_{i=1}^{n} 1\{y^{(i)}=1\}\, d_i}$$

$$\phi_{k|y=0} = \frac{1+\sum_{i=1}^{n}\sum_{j=1}^{d_i} 1\{x_j^{(i)}=k \wedge y^{(i)}=0\}}{|V|+\sum_{i=1}^{n} 1\{y^{(i)}=0\}\, d_i}$$

While not necessarily the very best classification algorithm, the Naive Bayes classifier often work superisingly well, It is often also a very good "first thing to try" given its simplicity and ease of implementation.

Name: Xiaofan Wang.

Lec 7. Kernel Methods & SVM

-. Kernel Methods

1. feature maps

$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$

let function $\phi: \mathbb{R} \to \mathbb{R}^4$ defined as:

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4$$

let $\theta \in \mathbb{R}^4$: $\theta_0, \theta_1, \theta_2, \theta_3$.

$\theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0 = \theta^T \cdot \phi(x)$

ϕ: feature map (map the attributes to the features)

2. LMS with features

Batch Gradient Descent: $\theta := \theta + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)})) x^{(i)}$

$= \theta + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)}) x^{(i)}$

Let $\phi: \mathbb{R}^4 \Rightarrow \mathbb{R}^p$, map x to the feature $\phi(x)$ in $\mathbb{R}^p$

$\theta := \theta + \alpha \cdot \sum_{i=1}^{n} (y^{(i)} - \theta^T \cdot \phi(x^{(i)})) \phi(x^{(i)})$

Stochastic Gradient Descent: $\theta := \theta + \alpha \cdot (y^{(i)} - \theta^T \phi(x^{(i)})) \phi(x^{(i)})$

3. LMS with the kernel trick.

$\phi(x)$ is high-dimensional (dimensional $\vec{\theta}$ by coefficients $\beta_1 \cdots \beta_n$)

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ \vdots \end{bmatrix}$$

At initialization: $\theta = 0 = \sum_{i=1}^{n} 0 \cdot \phi(x^{(i)})$

At some point: $\theta = \sum_{i=1}^{n} \beta_i \phi(x^{(i)})$

$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T \cdot \phi(x^{(i)})) \phi(x^{(i)})$

$= \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{n} (y^{(i)} - \theta^T \cdot \phi(x^{(i)})) \phi(x^{(i)})$

$= \sum_{i=1}^{n} (\beta_i + \alpha (y^{(i)} - \theta^T \cdot \phi(x^{(i)}))) \cdot \phi(x^{(i)})$

New β_i

$\beta_i := \beta_i + \alpha (y^{(i)} - \theta^T \cdot \phi(x^{(i)}))$

Replace θ by $\theta = \sum_{j=1}^{n} \beta_j \phi(x^{(j)})$:

$\forall i \in \{1 \cdots n\}$, $\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)})\right)$

$\phi(x^{(j)})^T \phi(x^{(i)}) = \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$

all pairs: $O(p)$.

compute $\phi(x^{(i)})$:

$\langle \phi(x), \phi(z) \rangle = 1 + \sum_{i=1}^{d} x_i z_i + \sum_{i,j \in \{1 \cdots d\}} x_i x_j z_i z_j + \sum x_i x_j x_k z_i z_j z_k$

$= 1 + \sum_{i=1}^{d} x_i z_i + \left(\sum_{i=1}^{d} x_i z_i\right)^2 + \left(\sum_{i=1}^{d} x_i z_i\right)^3$

$= 1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3$

Kernel: $K(x, z) \triangleq \langle \phi(x), \phi(z) \rangle$

Final Algorithm:

compute all the values $K(x^{(i)}, x^{(j)}) \triangleq \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$ using equation for $i, j \in \{1 \cdots n\}$.

Set $\beta := 0$.

Loop: $\forall i \in \{1 \cdots n\}$

$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)})\right)$

let K be the $n \times n$ matrix with $K_{ij} = K(x^{(i)}, x^{(j)})$

$\beta := \beta + \alpha (\vec{y} - K\beta)$

compute prediction:

$\theta^T \cdot \phi(x) = \sum_{i=1}^{n} \beta_i \phi(x^{(i)})^T \phi(x)$

$= \sum_{i=1}^{n} \beta_i K(x^{(i)}, x)$

4. properties of kernels

① $K(x,z)=\phi(x)^T\phi(z)$

Suppose $x, z \in \mathbb{R}^d$, $K(x,z)=(x^Tz)^2$

$$K(x,z)=\left(\sum_{i=1}^{d} x_i z_i\right)\left(\sum_{j=1}^{d} x_j z_j\right)$$

$$=\sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j z_i z_j$$

$$=\sum_{i,j=1}^{d} (x_i x_j)(z_i z_j)$$

Thus, $K(x,z)=\langle \phi(x), \phi(z)\rangle$

feature mapping:

$$\phi(x)=\begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

High-dimension.

require $O(d^2)$ time

$K(x,z)$ take only $O(d)$ time.

② $K(x,z)=(x^Tz+c)^2$

$$=\sum_{i,j=1}^{d} (x_i x_j)(z_i z_j) + \sum_{i=1}^{d} (\sqrt{2c}\, x_i)(\sqrt{2c}\, z_i) + c^2$$

feature mapping:

$$\phi(x)=\begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}\,x_1 \\ \sqrt{2c}\,x_2 \\ \sqrt{2c}\,x_3 \\ c \end{bmatrix}$$

$K(x,z)=(x^Tz+c)^k$ to a feature mapping $\binom{d+k}{k}$ feature space.

Despite working in $O(d^k)$-dimensional space,

$K(x,z)$ takes only $O(d)$ time.

③ Gaussian kernel (kernels as similarity metrics)

$$K(x,z)=\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

$K(x,z)=\phi(x)^T\phi(z)$

④ Necessary conditions for valid kernels.

Kernel Matrix: $K_{ij}=K(x^{(i)}, x^{(j)})$

$$=\phi(x^{(i)})^T\phi(x^{(j)})$$

$$=\phi(x^{(j)})^T\phi(x^{(i)})$$

$$=K(x^{(j)}, x^{(i)})=K_{ji}$$

$$z^TKz=\sum_i\sum_j z_i K_{ij} z_j$$

$$=\sum_i\sum_j z_i \phi(x^{(i)})^T\phi(x^{(j)}) z_j$$

$$=\sum_i\sum_j z_i \sum_k \phi_k(x^{(i)})\phi_k(x^{(j)}) z_j$$

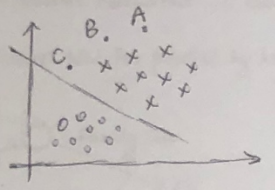
$$=\sum_k\sum_i\sum_j z_i \phi_k(x^{(i)})\phi_k(x^{(j)}) z_j$$

$$=\sum_k\left(\sum_i z_i \phi_k(x^{(i)})\right)^2 \geq 0$$

$\sum_{i,j} a_i a_j = (\sum_i a_i)^2$ for $a_i = z_i \phi_k(x^{(i)})$

二. Support Vector Machines

1. Margins: Intuition



Training Example: Positive & Negative.

Separate hyperplane: A Decision Boundary ~~($\theta^T x$)~~

(by the equation: $\theta^T x = 0$)

2. Notation

Class label: $y \in \{-1, 1\}$

Classifier: $h_{w,b}(x) = g(w^T x + b)$ → intercept b (separate from the other parameters)

$$\begin{cases} g(z) = 1 & \text{if } z \geq 0 \\ g(z) = -1 & \text{otherwise} \end{cases}$$

3. Functional and Geometric Margins

Given a training example $(x^{(i)}, y^{(i)})$,

Define the functional margin of (w, b):

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

If $y^{(i)} = 1$, then for the functional margin to be large,
we need $w^T x^{(i)} + b$ to be a large positive number.

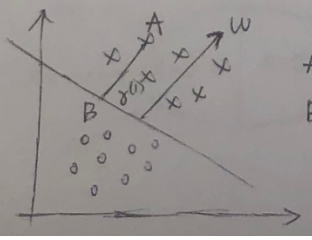
If $y^{(i)} = -1$, then for the functional margin to be large,
we need $w^T x^{(i)} + b$ to be a large negative number.

If $y^{(i)}(w^T x^{(i)} + b) > 0$, then prediction is correct.

Hence, a large functional margin represents a confident and a correct prediction.

Given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1 \cdots n\}$,

$$\hat{\gamma} = \min_{i=1 \cdots n} \hat{\gamma}^{(i)}$$



A: $x^{(i)}$.

B: $x^{(i)} - \gamma^{(i)} \cdot \frac{w}{\|w\|}$ (lies on decision boundary) $\rightarrow w^T x + b = 0$

$$w^T\left(x^{(i)} - \gamma^{(i)} \cdot \frac{w}{\|w\|}\right) + b = 0.$$

$$\gamma^{(i)} = \frac{w^T x^{(i)} + b}{\|w\|} = \left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}$$

$$\gamma^{(i)} = y^{(i)}\left(\left(\frac{w}{\|w\|}\right)^T x^{(i)} + \frac{b}{\|w\|}\right)$$

$$\gamma = \min_{i=1 \cdots n} \gamma^{(i)}$$

4. Optimal Margin Classifier

$\max_{\gamma,w,b} \gamma$

s.t. $y^{(i)}(w^T x^{(i)}+b) \geq \gamma$. $(i=1\cdots n)$

If $\|w\|=1$, $\gamma=\frac{\hat{\gamma}}{\|w\|}$

$\max_{\gamma,w,b} \frac{\hat{\gamma}}{\|w\|}$

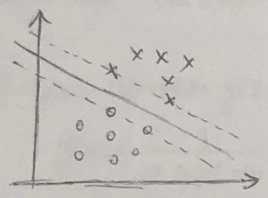
s.t. $y^{(i)}(w^T x^{(i)}+b) \geq \hat{\gamma}$. $(i=1\cdots n)$

If $\hat{\gamma}=1$, $\gamma=\frac{\hat{\gamma}}{\|w\|} \sim \frac{1}{\|w\|} \sim \|w\|^2$

$\min_{w,b} \frac{1}{2}\|w\|^2$

s.t. $y^{(i)}(w^T x^{(i)}+b) \geq 1$. $(i=1\cdots n)$

constraint: $\Rightarrow g_i(w) = -y^{(i)}(w^T x^{(i)}+b)+1 \leq 0$



Lagrangian for optimization:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i[y^{(i)}(w^T x^{(i)}+b)-1]$$

$$\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{n}\alpha_i y^{(i)} x^{(i)} = 0$$

$$w = \sum_{i=1}^{n}\alpha_i y^{(i)} x^{(i)}$$

Derivative to b: $\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{n}\alpha_i y^{(i)} = 0$

$$L(w,b,\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\cdot\sum_{i,j=1}^{n} y^{(i)}y^{(j)}\alpha_i\alpha_j (x^{(i)})^T x^{(j)} - b\cdot\sum_{i=1}^{n}\alpha_i y^{(i)}$$

$$L(w,b,\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\cdot\sum_{i,j=1}^{n} y^{(i)}y^{(j)}\alpha_i\alpha_j (x^{(i)})^T x^{(j)} \quad (\alpha_i \geq 0)$$

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\cdot\sum_{i,j=1}^{n} y^{(i)}y^{(j)}\alpha_i\alpha_j \langle x^{(i)}, x^{(j)}\rangle$$

s.t. $\alpha_i \geq 0$. $(i=1\cdots n)$

$$\sum_{i=1}^{n}\alpha_i y^{(i)} = 0.$$

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T}x^{(i)} + \min_{i:y^{(i)}=1} w^{*T}x^{(i)}}{2}$$

$$w^T\cdot x + b = \left(\sum_{i=1}^{n}\alpha_i y^{(i)} x^{(i)}\right)^T x + b$$

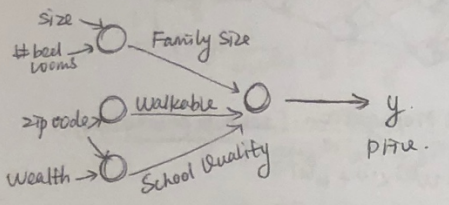
$$= \sum_{i=1}^{n}\alpha_i y^{(i)}\langle x^{(i)}, x\rangle + b.$$

Name: Xiaofan Wang

Lec 8. Lec 9 Neural Networks

-. Neural Networks (end-to-end learning)

Small NN for predicting housing prices:



- Input Layer: Four Features x_1, x_2, x_3, x_4
- Hidden Layer: Three Internal Neurons (Black Box)
- Output Layer: One Output Neuron.

If logistic regression $g(z)$ as a single neuron:

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

Break $g(x)$ into two computations:

$z = w^T x + b$.

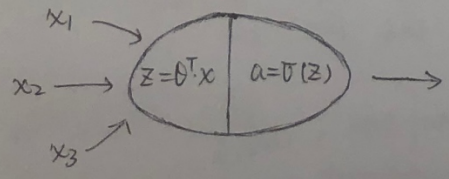
$a = \sigma(z)$. $\left(\sigma(z) = \frac{1}{1+e^{-z}}\right)$

$a = g(z)$ → activation function:

- $g(z) = \frac{1}{1+e^{-z}}$ (sigmoid)
- $g(z) = \max(z, 0)$ (ReLU)
- $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ (tanh)

In general, $g(z)$ is non-linear function.

For instance, logistic regression as a single neuron:



In the 1st layer:

1st Hidden Unit: $z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$, $a_1^{[1]} = g(z_1^{[1]})$

2nd Hidden Unit: $z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$, $a_2^{[1]} = g(z_2^{[1]})$

3rd Hidden Unit: $z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}$, $a_3^{[1]} = g(z_3^{[1]})$

1st Layer Activations: $a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$

2nd Layer Activation: $a_1^{[2]} = g(z_1^{[2]})$ → final Output prediction.

=. Vectorization

1. Vectorizing the Output Computation.

$$\begin{bmatrix} z_1^{[1]} \\ \vdots \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} \cdots w_1^{[1]T} \cdots \\ \vdots \\ \cdots w_4^{[1]T} \cdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ \vdots \\ b_4^{[1]} \end{bmatrix}$$

$z^{[1]} \in \mathbb{R}^{4\times 1}$ $\quad w^{[1]} \in \mathbb{R}^{4\times 3}$ $\quad x \in \mathbb{R}^{3\times 1}$ $\quad b^{[1]} \in \mathbb{R}^{4\times 1}$

$z^{[1]} = w^{[1]} x + b^{[1]}$. $\quad a^{[1]} = g(z^{[1]})$

Define $g(z)$ as: $g(z) = \frac{1}{1+e^{-z}}$ $(z \in \mathbb{R})$

Matlab pseudocode: $g(z) = 1./(1+\exp(-z))$ where $z \in \mathbb{R}^d$

$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$, $\quad a^{[2]} = g(z^{[2]})$

$(1\times 1) \quad (1\times 4)\ (4\times 1)\ (1\times 1) \qquad (1\times 1) \quad (1\times 1)$

$z^{[2]} = w^{[2]} a^{[1]}$

$= w^{[2]} g(z^{[1]})$ By definition

$= w^{[2]} z^{[1]}$ since $g(z) = z$

$= w^{[2]} w^{[1]} x$

$= \tilde{w} x$ where $\tilde{w} = w^{[2]} w^{[1]}$

2. Vectorization Over Training Examples

Activations for each examples : $\begin{cases} z^{[1](1)} = w^{[1]} x^{(1)} + b^{[1]} \\ z^{[1](2)} = w^{[1]} x^{(2)} + b^{[1]} \\ z^{[1](3)} = w^{[1]} x^{(3)} + b^{[1]} \end{cases}$

$$\vec{X} = \begin{bmatrix} \vdots & \vdots & \vdots \\ x^{(1)} & x^{(2)} & x^{(3)} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$Z^{[1]} = \begin{bmatrix} \vdots & \vdots & \vdots \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} \\ \vdots & \vdots & \vdots \end{bmatrix} = \underbrace{W^{[1]} X}_{4\times 3} + \underbrace{b^{[1]}}_{4\times 1}$$

$$\underbrace{\tilde{b}^{[1]}}_{4\times 3} = \begin{bmatrix} \vdots & \vdots & \vdots \\ b^{[1]} & b^{[1]} & b^{[1]} \\ \vdots & \vdots & \vdots \end{bmatrix} \qquad Z^{[1]} = W^{[1]} X + \tilde{b}^{[1]}$$

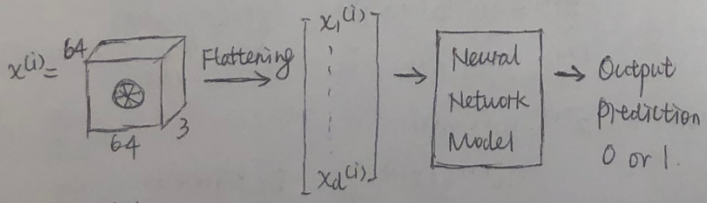
① Initialize the parameters $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$, to small random numbers. For each example, we compute the output "probability" from the sigmoid function $a^{[2](i)}$.

② Using the logistic Regression log likelihood :

$$\sum_{i=1}^{m} \left[y^{(i)} \log a^{[2](i)} + (1-y^{(i)}) \log (1 - a^{[2](i)}) \right]$$

③ Maximize the function using Gradient Descent. The Maximization procedure corresponds to training the Neural Network.

3. Backpropagation.



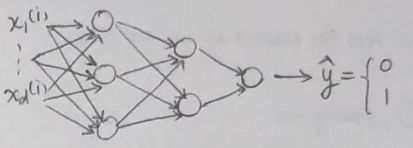
NN model :

(i) Network Architecture : how many layers. neurons. how layers are connected.

(ii) parameters : initialization. optimization. Analyzing these parameters.

1. Parameter Initialization.

Fully-connected Layer : All inputs are connected to (1st Hidden Layer). all neurons in the next layer.



① Forward propagation. (compute parameters) :

$$z^{[1]} = w^{[1]} x^{(i)} + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$
$$z^{[2]} = w^{[2]} \cdot a^{[1]} + b^{[2]}$$
$$a^{[2]} = g(z^{[2]})$$
$$z^{[3]} = w^{[3]} \cdot a^{[2]} + b^{[3]}$$
$$\hat{y}^{(i)} = a^{[3]} = g(z^{[3]})$$

② Update parameters. (compute Loss L) :

$$L(\hat{y}, y) = -\left[(1-y) \log (1-\hat{y}) + y \log \hat{y} \right]$$

$$w^{[1]} = w^{[1]} - \alpha \cdot \frac{\partial L}{\partial w^{[1]}} \quad \text{(learning rate} \times \text{Gradient)}$$

$$b^{[1]} = b^{[1]} - \alpha \cdot \frac{\partial L}{\partial b^{[1]}}$$

$$a^{[1]} = g(z^{[1]}) = g(w^{[1]} x^{(i)} + b^{[1]}).$$

Each Neuron will receive the exact same gradient update value.

Initialization : $W^{[1]} \sim N\left(0, \sqrt{\frac{2}{n^{[1]} + n^{[1-1]}}}\right)$

↓ the Number of neuron in layer l.

variance $\sigma(in) \approx \sigma(out)$.

2. optimization (using Stochastic Gradient Descent)

$w^{[1]}$ on the loss is more complex than that of $w^{[2]}$.

$w^{[2]}$ is "closer" to the output $\hat{y}$.

$$\frac{\partial \mathcal{L}}{\partial w^{[2]}} = \frac{-\partial}{\partial w^{[2]}}\left((1-y)\log(1-\hat{y}) + y\log\hat{y}\right)$$

$$= -(1-y)\frac{\partial}{\partial w^{[2]}}\log\left(1-g(w^{[2]}a^{[1]}+b^{[2]})\right) - y\frac{\partial}{\partial w^{[2]}}\log\left(g(w^{[2]}a^{[1]}+b^{[2]})\right)$$

$$= -(1-y)\frac{1}{1-g(w^{[2]}a^{[1]}+b^{[2]})}(-1)\,g'(w^{[2]}a^{[1]}+b^{[2]})\,a^{[1]T} - y\cdot\frac{1}{g(w^{[2]}a^{[1]}+b^{[2]})}\cdot g'(w^{[2]}a^{[1]}+b^{[2]})a^{[1]T}$$

$$= (1-y)\,\sigma(w^{[2]}a^{[1]}+b^{[2]})\,a^{[1]T} - y(1-\sigma(w^{[2]}a^{[1]}+b^{[2]}))\,a^{[1]T}$$

$$= (1-y)\,a^{[2]}a^{[1]T} - y(1-a^{[2]})\,a^{[1]T}$$

$$= (a^{[2]}-y)\cdot a^{[1]T}$$

The derivative sigmoid function: $g' = \sigma' = \sigma(1-\sigma)$

$$a^{[2]} = \sigma(w^{[2]}a^{[1]}+b^{[2]})$$

$\mathcal{L}$ depend on $\hat{y} = a^{[2]}$

$$\frac{\partial \mathcal{L}}{\partial w^{[1]}} = \frac{\partial \mathcal{L}}{\partial a^{[2]}}\cdot\frac{\partial a^{[2]}}{\partial w^{[1]}} = \frac{\partial \mathcal{L}}{\partial a^{[2]}\cdot\partial a^{[2]}\cdot\partial z^{[2]}}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{[2]}}\cdot\frac{\partial a^{[2]}}{\partial z^{[2]}}\cdot\frac{\partial z^{[2]}}{\partial w^{[1]}}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{[2]}}\cdot\frac{\partial a^{[2]}}{\partial z^{[2]}}\cdot\frac{\partial z^{[2]}}{\partial a^{[1]}}\cdot\frac{\partial a^{[1]}}{\partial w^{[1]}}$$

$$= \underbrace{\frac{\partial \mathcal{L}}{\partial a^{[2]}}\cdot\frac{\partial a^{[2]}}{\partial z^{[2]}}}_{\frac{\partial \mathcal{L}}{\partial z^{[2]}} = (a^{[2]}-y)}\cdot\underbrace{\frac{\partial z^{[2]}}{\partial a^{[1]}}}_{w^{[2]}}\cdot\underbrace{\frac{\partial a^{[1]}}{\partial z^{[1]}}}_{g'(z^{[1]})}\cdot\underbrace{\frac{\partial z^{[1]}}{\partial w^{[1]}}}_{a^{[0]}} = (a^{[2]}-y)\cdot w^{[2]}\cdot g'(z^{[1]})\cdot a^{[0]}$$

$$\frac{\partial \mathcal{L}}{\partial w^{[1]}} = (a^{[2]}-y)\,w^{[2]}\,g'(z^{[1]})\,a^{[0]} = w^{[2]T}\circ g'(z^{[1]})(a^{[2]}-y)\,a^{[0]T}$$

2×3	1×1	1×2	2×1	3×1	2×1	2×1	1×1	1×3

$$w^{[1]} = w^{[1]} - \alpha\cdot\frac{\partial J}{\partial w^{[1]}}, \quad \text{cost function } J = \frac{1}{n}\cdot\sum_{i=1}^{n}\mathcal{L}^{(i)}$$

(mini-batch GD) $\quad J_{mb} = \frac{1}{B}\cdot\sum_{i=1}^{B}\mathcal{L}^{(i)}$

(mini-batch stochastic GD)

Another Method: Momentum

$$\begin{cases} V_{dw^{[1]}} = \beta\cdot V_{dw^{[1]}} + (1-\beta)\cdot\dfrac{\partial J}{\partial w^{[1]}} \\ w^{[1]} = w^{[1]} - \alpha\cdot V_{dw^{[1]}} \end{cases}$$

3. Analyzing the Parameters

Training Set : 96% Accuracy.

Testing Set : 64% Accuracy.

Solution :
- Collecting more data
- Employing regularization.
- Making the model shallower

① L2 Regularization.

$$J_{L2} = J + \frac{\lambda}{2}\|w\|^2$$

$$= J + \frac{\lambda}{2}\sum_{i,j}|w_{ij}|^2$$

$$= J + \frac{\lambda}{2}w^T \cdot w.$$

Update rule with L2 Regularization :

$$w = w - \alpha \cdot \frac{\partial J}{\partial w} - \alpha \frac{\lambda}{2} \cdot \frac{\partial w^T w}{\partial w}$$

$$= (1 - \alpha\lambda) w - \alpha \cdot \frac{\partial J}{\partial w}$$

This penalization increases the cost J, which encourages individual parameters to be small in magnitude, which is a way to reduce overfitting.

② Parameter Sharing.

A ball may appear in any region of image. and not always the center. The logistic regression will likely predict no ball.

Solution : Convolutional Neural Networks.

Suppose θ not a vector but a Matrix

$$a = \sum_{i=1}^{4}\sum_{j=1}^{4} \theta_{ij} x_{ij}$$

the parameter θ have "seen" all pixels of the image.

Name: Xiaofan Wang

Lec 10. Bias-Variance & Error Analysis

一. Bias-Variance Tradeoff

$E_{(x,y)\sim \text{test set}} |\hat{f}(x) - y|^2$

this MSE is too high:
(Mean Squared Error)

- Overfitting: the model is too closely to training set
  doesn't generalize well to other examples
- Underfitting: the model didn't gather enough information from training set. doesn't capture the link between the feature x and target y.
- The data is simply noisy: Neither overfitting nor underfitting. The high MSE is due to the amount of noise in the dataset.

$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim E(\varepsilon_i) = 0. \ Var(\varepsilon_i) = \sigma^2.$

$\hat{f}(x_j)$ is random, $\Rightarrow$ Bias $E(\hat{f}(x) - f(x))$

compute MSE on the test set:

$$\text{Test MSE} = E(y - \hat{f}(x))^2)$$
$$= E((\varepsilon + f(x) - \hat{f}(x))^2)$$
$$= E(\varepsilon^2) + E((f(x) - \hat{f}(x))^2)$$
$$= \sigma^2 + (E(f(x) - \hat{f}(x)))^2 + Var(f(x) - \hat{f}(x))$$
$$= \sigma^2 + (\text{Bias}\hat{f}(x))^2 + Var(\hat{f}(x))$$

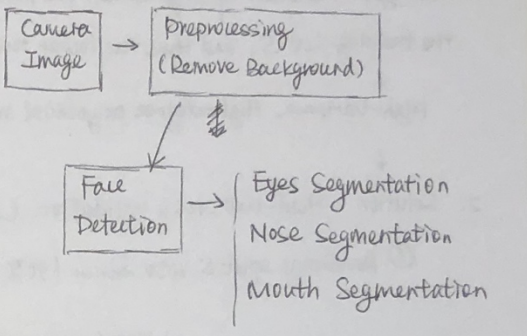
- High Bias: Underfitting
- High Variance: Overfitting.
- Large σ^2: Noisy Data

Analyzing the performance of Machine Learning Algorithm:

- reduce Bias without increasing the variance.
- reduce Variance without increasing the Bias.

→ Bias-Variance Tradeoff.

二. Error Analysis



Name: Xiaofan Wang

Lec 11 | Regularization
Model Selection.

一. Cross Validation

1. Using empirical risk minimization for model selection:

① Train each model M_i on S, to get some hypothesis h_i.

② Pick the hypothesis with the smallest training error.

The higher the order of polynomial, the better it will fit the training set S, and thus the lower the training error.

↓

High-Variance, High-degree polynomial model

↓

2. Solution: Hold-out cross validation (Algorithm).

① Randomly split S into S_{train} (70%) and S_{cv} (30%)

↓

Hold-out cross validation set.

② Train each model M_i on S_{train} only, to get h_i

③ Select and Output the hypothesis h_i that had the smallest error $\hat{\varepsilon}_{cv}(h_i)$ on the hold-out cross validation set.

3. K-fold cross Validation. (Hold Out less Data Each time).

① Randomly Split S into k disjoint subsets of $\frac{m}{k}$ training examples each. ($S_1 \cdots S_k$).

② For each model M_i,

For $j=1 \cdots k$,

Train the Model M_i on $S_1 \cup \cdots \cup S_{j-1} \cup S_{j+1} \cup \cdots S_k$ (train on all the data except S_j) to get some hypothesis h_{ij}

Test the hypothesis h_{ij} on S_j, to get $\hat{\varepsilon}_{S_j}(h_{ij})$

③ pick the model M_i with the lowest estimated generation error, and retain that model on the entire training set S.

4. Leave-one-out cross Validation.

Repeatly train on all but one of the training examples in S, and test on that held-out example.

The resulting $m=k$ errors are then ~~taken~~ averaged together to obtain our estimate of the generalization error of a model.

二. Feature Selection.

1. Algorithm: Forward Search. (Wrapper).

① Initialize $F = \emptyset$

② Repeat {

(a) For $i=1 \cdots n$ if $i \notin F$, let $F_i = F \cup \{i\}$.
Use cross Validation to evaluate features F_i

(b) Set F to be the best feature subset found on step (a)

}.

③ Select and Output the best feature subset that was evaluated during the entire search procedure.

2. Algorithm: Backward Search (Filter)

Pick k features with the largest scores $S(i)$

Define $S(i)$ to be the correlation between $x_i \sim y_i$

$$MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y_i \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)\,p(y)}$$

↓

$S(i)$: Mutual Information.

KL divergence:

$$MI(x_i, y) = KL(p(x_i, y) \,||\, p(x_i)\,p(y))$$

3. Bayesian Statistics and Regularization

Parameter Fitting using Maximum Likelihood (ML):

$$\theta_{ML} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)} | x^{(i)}; \theta)$$

Bayesian View : θ random variable

prior distribution $p(\theta)$.

Given a training set $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$

Compute the posterior Distribution on the parameters:

$$p(\theta|S) = \frac{p(S|\theta) \cdot p(\theta)}{p(S)}$$

$$= \frac{(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)) p(\theta)}{\int_{\theta} (\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta) p(\theta)) d\theta}$$

Bayesian Logistic Regression:

$$p(y^{(i)} | x^{(i)}, \theta) = h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{(1-y^{(i)})}$$

$$h_\theta(x^{(i)}) = \frac{1}{1 + \exp(-\theta^T x^{(i)})^3}$$

Compute the posterior Distribution on θ:

$$p(y|x, S) = \int_{\theta} p(y|x, \theta) \cdot p(\theta|S) \, d\theta.$$

$$E[y|x, S] = \int_{y} y \, p(y|x, S) \, dy.$$

The MAP (maximum a posteriori):

$$\theta_{MAP} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)} | x^{(i)}, \theta) p(\theta)$$

prior $p(\theta)$: $\theta \sim N(0, \tau^2 I)$

Bayesian Logistic Regression turns out to be an effective algorithm for text classification, even though in text classification we usually have $n >> m$.

Name: Xiaofan Wang

Lec 12. K-means Clustering Algorithm.

(Unsupervised Learning problem).

Given a training set $\{x^{(1)}, \dots x^{(n)}\}$.

Group the data into a few "clusters"

$x^{(i)} \in \mathbb{R}^d$, no labels $y^{(i)}$.

The K-means Clustering Algorithm:

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

   For every i, set $c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|^2$

   For each j, set $\mu_j := \frac{\sum_{i=1}^n 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^n 1\{c^{(i)}=j\}}$

   }

Inner loop repeatedly carries out two steps:

(i) Assigning each training example $x^{(i)}$ to the ~~closest~~ closest cluster centroid μ_j.

(ii) Moving each cluster centroid μ_j to the mean of the points assigned to it.

Define Distortion Function:

$$J(c, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

K-means is exactly coordinate descent on J.

The inner loop of K-means repeatedly min~J with c, fixed μ. Then minimize J with μ, fixed c

$\Rightarrow$ J must monotonically decrease, the value of J must converge.

Distortion Function J: non-convex function.

coordinate descent on J is not guaranteed to converge to the global minimum.

(K-means ~~are~~ ~ susceptible to local optima)

Very often K-means will work fine and come up with very good clusterings ~

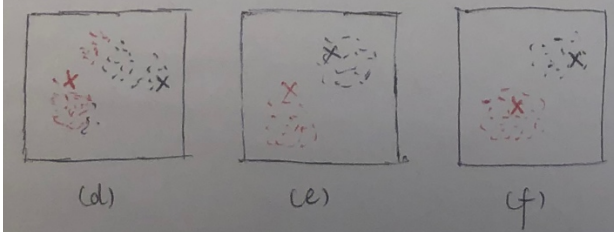
(To avoid getting stuck in bad local optima, running K-means many times.)

Then, out of all the different clusterings found, pick the one that gives the lowest distortion $J(c, \mu)$

$\Rightarrow$ (a) Original dataset

(b) Random initial cluster centroids.

(c-f) running two iterations of K-means.



Name: Xiaofan Wang

Lec 13. Mixtures of Gaussians and EM algorithm.

一. Joint Distribution: $p(x^{(i)}, z^{(i)}) = p(x^{(i)} | z^{(i)}) p(z^{(i)})$

$z^{(i)} \sim$ Multinomial (ϕ)

$x^{(i)} | z^{(i)} = j \sim N(\mu_j, \Sigma_j)$

⇓

$$l(\phi, \mu, \Sigma) = \sum_{i=1}^{n} \log p(x^{(i)}; \phi, \mu, \Sigma)$$

$$= \sum_{i=1}^{n} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)} | z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi)$$

$$l(\phi, \mu, \Sigma) = \sum_{i=1}^{n} \log p(x^{(i)} | z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi)$$

Maximizing this with respect to ϕ, μ, Σ, gives the parameters:

$$\phi_j = \frac{1}{n} \cdot \sum_{i=1}^{n} 1\{z^{(i)} = j\}.$$

$$\mu_j = \frac{\sum_{i=1}^{n} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{n} 1\{z^{(i)} = j\}}.$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{n} 1\{z^{(i)} = j\}}.$$

二. EM Algorithm. (iterative algorithm. ~ two main steps)

1. In the E-step: try to guess the value of the $z^{(i)}$'s.
2. In the M-step: update the parameters of our models

Repeat until convergence: {

(E-step) For each i, j, set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

$$\phi_j := \frac{1}{n} \sum_{i=1}^{n} w_j^{(i)}$$

$$\mu_j := \frac{\sum_{i=1}^{n} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{n} w_j^{(i)}}$$

$$\Sigma_j := \frac{\sum_{i=1}^{n} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{n} w_j^{(i)}}$$

}

In the E-step, ~~calcu~~ calculate the posterior probability of our parameters the $z^{(i)}$'s.

Using Bayes Rule:

$$p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi)}{\sum_{l=1}^{k} p(x^{(i)} | z^{(i)} = l; \mu, \Sigma) p(z^{(i)} = l; \phi)}$$

In the M-step. $z^{(i)}$: indicating from which Gaussian each datapoint had come, $\sim w_j^{(i)}$.

The EM Algorithm is also reminiscent of the K-means clustering algorithm, except that instead of the "hard" cluster assignments $c^{(i)}$, we instead have the "soft" assignments $w_j^{(i)}$. Similar to K-means, it is also susceptible to local optima, so reintializing at several different initial parameters may be a good idea.