Week 1

① Supervised Learning

"right answers" given

② Classification -Discrete valued output

Regression -Continuous valued output

③ Unsupervised Learning - Let the program find structure

clustering

Week 2

① $x_j^{(i)}$: value of feature j in the i^{th} training example

$x_0^{(i)} = 1$ $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

② For gradient descent, feature scaling can make it much faster.

Get every feature into approximately a $-1 \le x_i \le 1$ range

Mean normalization : make features have approximately 0 mean

$$\boxed{x_i \leftarrow \frac{x_i - \mu_i}{s_i}}$$

μ_i : average

s_i : range or standard deviation

③ Normal Equation : $\theta = (X^T X)^{-1} X^T y$

Octave : pinv(X' * X) * X' * y

no need of feature scaling

pro: ① 不用选 α

② 不用 iterate

$$X = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_4 & \cdots \\ & & \vdots & & & \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix}$$

con: slow when n is large (10^4)

因为算 inverse matrix 很慢

If not invertible: ① redundant feature ② too many features

④ Octave

a. 不等: ~=

b. comment : %

c. disp()

d. eye(4) 出来 4x4 idensity matrix

e. size() length()

f. who 显示内存里的变量

whos 更详细的信息.

g. clear ... 删变量

h. save hello.mat v 存 v 进 hello.mat (binary form)

save hello.txt v -ascii (ascii form)

i. A(:) put all elements of A into a single vector

j. sum() floor() ceil() prod()

k. max(A,[],1) per column

max(A,[],2) per row

l. print -dpng 'myplot.png' 把 plot 出来的存成 png

for loop:

```
for i = 1:10,
    v(i) = 2^i;
end;
```

while loop:

```
while i <= 5,
    v(i) = 100;
    i = i+1;
end;
```

define function 在 .m 文件里

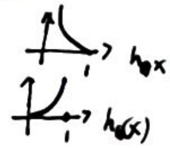
```
function y = squareThisNumber(x)
y = x^2; ...

function [y1, y2] = choosetwo(x)
```

⑤ Logistic Regression Model - Want $0 \le h_\theta(x) \le 1$ It is convex.

$h_\theta(x) = g(\theta^T x)$ where $g(z) = \frac{1}{1+e^{-z}}$ (sigmoid / logistic function)

$h_\theta(x)$ = estimated probability that $y=1$ on input x

cost function: $\mathrm{cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$



$= -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

$$\frac{\partial}{\partial\theta} J(\theta) = \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \cdot \frac{1}{m}$$

$$\theta = \theta - \frac{\alpha}{m} X^T(g(X\theta) - \vec{y})$$

advanced algorithm:

① Conjugate gradient

② BFGS

③ L-BFGS

way to implement advanced algorithms:

```
function [jVal, gradient] = costFunction(theta)
    jVal = [ ... code to compute J(θ)]
    gradient = [ ... code to compute gradient ]
end
```

```
options = optimset('GradObj', 'On', 'MaxIter', '100');
initialTheta = zeros(2,1)
[optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta, options)
```

bool: 是否 converge　　pointer

☆ 当有多种分类时: multiclass classification: one-vs-all

(6) Overfitting Problem

Solution: i. - 手选 which features to keep

- Model selection algorithm

ii. Regularization

- keep all features but reduce magnitude of parameters
- 当有很多features而它们全有用的时候用

$$\min \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2$$

gradient descent $\Rightarrow$

$$\theta_0 := \theta_0-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j(1-\alpha\frac{\lambda}{m})-\frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}$$

normal equation $\Rightarrow$

$$\theta=(X^TX+\lambda A)^{-1}X^Ty \qquad A=\begin{bmatrix}0&0&0&\\0&1&0&0\\0&0&1&0\\0&0&0&\ddots\end{bmatrix}$$

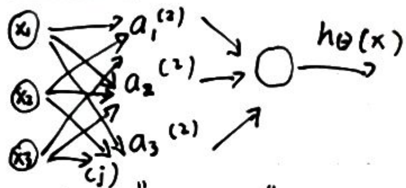
For logistic regression

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)}))+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]+\frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

$$\frac{\partial}{\partial\theta_j}J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}+\boxed{\frac{\lambda}{m}\theta_j}$$

当j=0，蓝框项为0

Week 4



(1) Neural Network

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\theta^{(j)}$ will be of dimension $s_{j+1}\times(s_j+1)$

$a_i^{(j)}$ "activation" of unit i in layer j

$\theta^{(j)}$ matrix of weights controlling function mapping from layer j to $j+1$

Week 5

① Neural Network (Classification)

L: total # of layers in network

s_l: # of layers in layer l

Cost Function: $h_\Theta(x) \in R^k$ $\quad (h_\Theta(x))_i$ = ith output

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left[y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_{(l+1)}} \sum_{j=1}^{s_l} (\Theta_{ji}^{(l)})^2$$

- The double sum simply adds up the logistic regression costs calculated for each cell in the output layer.
- The triple sum adds up squares of all the individual Θs in the entire network.
- The i in the triple sum does not refer to training example i.

In order to get $\frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta)$ to minimize $J(\Theta)$, we use Backpropagation algorithm

With training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})\}$

a. Set $\Delta_{ij}^{(l)} = 0$ (for all i, j, l)

b. For $i = 1$ to m

  i. set $a^{(1)} = x^{(i)}$

  ii. perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots L$

  iii. Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

  Use BP ← iv. Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

  v. $\Delta^{(l)} += \delta^{(l+1)} (a^{(l)})^T$

  $\Delta_{ij}^{(l)} += a_j^{(l)} \delta_i^{(l+1)}$

c. $D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

BP:

$\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) .* a^{(l)} .* (1-a^{(l)})$

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

为了用 fminunc，传 theta 前要用 theta(:) unroll，cost function 传出来的 gradient 也要 unroll. 重组可以用 reshape 从 vector 变回 matrix

We can use gradient checking to verify we are getting right gradient.

$$\frac{\partial}{\partial\Theta_j}J(\Theta) \approx \frac{J(\Theta_1, \dots, \Theta_j+\epsilon, \dots \Theta_n) - J(\Theta_1 \dots \Theta_n)}{2\epsilon}$$

神经网络的 Θ 需要 random initialization

theta = rand(m, n) * (2* INIT_EPSILON) - INIT_EPSILON

Week 6

① Evaluating a Hypothesis

fixing We adjust the algorithm by:

high variance • getting more training example

high variance • Trying smaller sets of features

bias • Trying additional features

• Increasing or decreasing λ
variance bias

a. Learn Θ using training set

b. compute test error

Linear Regression:

$$J_{test}(\Theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\Theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)^2$$

Classification:

$$\text{Test error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\Theta(x_{test}^{(i)}), y_{test}^{(i)})$$

$$err(h_\Theta(x), y) = \begin{cases} 1 & \text{if } h_\Theta(x) \geq 0.5 \text{ and } y=0 \text{ or } h_\Theta(x) < 0.5 \text{ and } y=1 \\ 0 & \text{otherwise} \end{cases}$$

Bias (underfit)

High $J_{train}(\Theta)$ and $J_{cv}(\Theta)$

$J_{cv}(\Theta) \approx J_{train}(\Theta)$

Variance (overfit)

High $J_{cv}(\Theta)$ Low $J_{train}(\Theta)$

$J_{cv}(\Theta) \gg J_{train}(\Theta)$

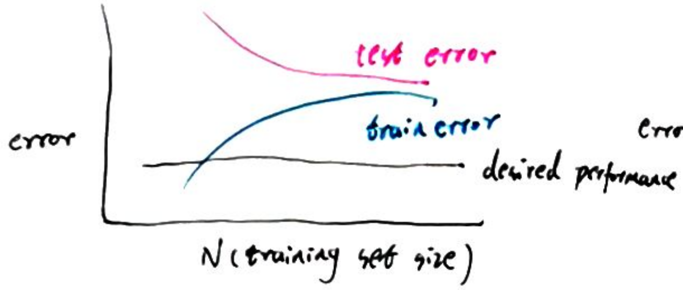
Choosing λ

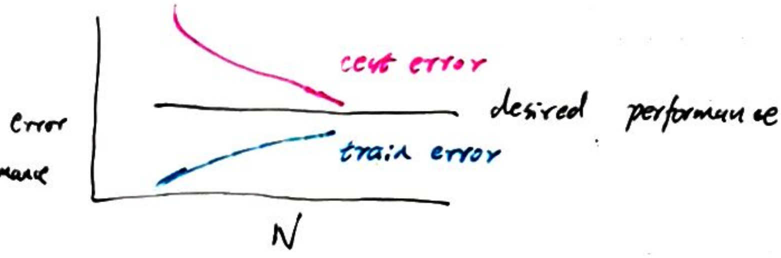
用不同 λ 和 degree 和 variant 组合学 Θ

学的时候用 regularized cost

然后算 $J_{cv}(\Theta)$，选最低 $J_{cv}(\Theta)$

$\lambda = \{0.01, 0.02, \dots 10.24\}$





High bias

Feeding more data does not help

High variance

Feeding more data helps

$$\text{Precision} = \frac{\text{True Positives}}{\#\text{predicted as positive}} \qquad \text{Recall} = \frac{\text{True Positives}}{\#\text{actual positives}}$$

Positive: in presense of rare class that we want to detect

$$F_1\ \text{Score} = 2\frac{PR}{P+R}$$

Week 7 SVM (convex)

① Kernels

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 f_1 + \cdots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

这个 f 是 kernel

都要符合 Mercer's Theorem

polynomial, string, chi-square, histogram intersection

Gaussian kernel: Normalize first

分子是 $l^{(1)}$ 到 x 的距离平方

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

如果 $x \approx l^{(1)}$: $f_1 \approx 1$

如果 x is far from $l^{(1)}$: $f_1 \approx 0$

σ^2 越小，对距离要求更严格 lower bias, higher variance

什么时候用 Logistic Regression or SVM。

① 当 n is large (relative to m): 用 Linear Regression or SVM without kernel

② 当 n is small, m is intermediate: SVM with Gaussian Kernel

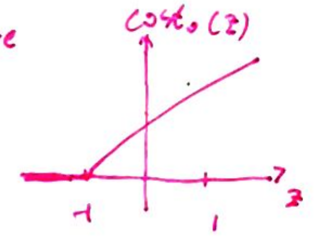
③ 当 n is small, m is large: add more features

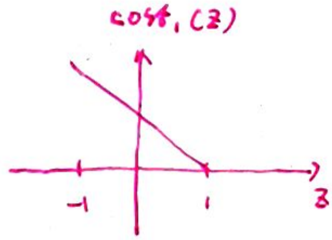
then use logistic regression or SVM without kernel

The SVM solves

$$\min_\theta \quad C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T f^{(i)}) + (1-y^{(i)}) cost_0(\theta^T f) \right] + \sum_{j=1}^{n} \theta_j^2 \cdot \frac{1}{2}$$

where





classification problem

provide maximum margin

如果 C = 大，得出的答案跟 logistic regression 一样，C 通常很大

☆ SVM VS Logistic Regression

① Logistic Regression is more sensitive to outlier.

② LR 给 probability，SVM 直接给 0 或 1 的结果

结论

先用 LR，如果发现不是 linearly seperable 用 SVM with non-linear kernel

SVM software package : liblinear, libsvm ⋯ 都要符合 Mercer's Theorem

Need to specify : ① Choice of parameter c

② Choice of kernel

很多 packages 已有内置 multi-class classification，没有的话用 one-VS-all

Week 8 Unsupervised Machine Learning + PCA

K-Means Method

① $c^{(i)}$ = index of cluster $(1, 2, \cdots K)$ to which example $x^{(i)}$ is assigned

μ_K = cluster centroid k $(\mu_k \in R^n)$

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

② K-Means Objective

$$\boxed{\min_{\substack{c^{(1)}\cdots c^{(m)}\\ \mu_1\cdots\mu_K}} J(c^{(1)}, \cdots c^{(m)}, \mu_1, \cdots \mu_K) = \min \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2}$$

③ K-Means Algorithm

a. 随机 initialize K cluster centroids $\mu_1 \cdots \mu_K \in R^n$

在1到m随机选K个数然后以那些数在X里选点做μ

有机会stuck在local minimum，要 多用几次不同随机起点。找最低J

b. Repeat {

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for k = 1 to K

μ_k := mean of points assigned to cluster k

}

④ 选K的方法（怎么知道选多少个cluster）

Elbow Method：一个个去试，找J转折点。

不过有时候没转折点。

⑤ Random Initialization

```
randIdx = randperm(size(X,1));
centroids = X(randIdx(1:K), :);
```

Principal Component Analysis

用途：

1. Compression：从 n-dimensions 压成 k-dimensions
   ① Reduce memory / disk needed to store data
   ② Speed up learning algorithm
   当真的慢或者真没内存再用PCA，不是首选的优化方法
2. Visualization：$k=2$ or 3

不应该使用的情况：To prevent overfitting，应该用 regularization

算法 Algorithm

① 预处理：mean normalization / feature scaling

② 计算 "covariance matrix"

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$$

vectorized：
Sigma = (1/m) * X' * X

③ 计算 "eigenvectors" of matrix Σ

[U, S, V] = svd(Sigma);

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \quad S = \begin{bmatrix} S_{11} & & & 0 \\ & S_{22} & & \\ & & \ddots & \\ 0 & & & S_{nn} \end{bmatrix}$$

④ 得出新坐标 z

压缩 dimension：U = U(:, 1:k)

$z = U^T x$

vectorized：
Z = X * U

Reconstruction：$x^{(i)}_{approx} = U_{reduced} \cdot z^{(i)}$ X_rec = Z * U_reduced'

怎么选 k？（压缩到什么 dimension）

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \le 0.01$$

=> 99% of variance is retained

用svd：for given k：

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \le 0.01$$

Week 9 Anomaly Detection + Recommender Systems

①

Anomaly Detection Algorithm

1. Choose feature x_i that we think might be indicative of anomalous examples

2. Calculate parameters $\mu_1 \cdots \mu_n$, $\sigma_1^2 \cdots \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

m个 sample n个 feature

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

3. Given new example x. compute $p(x)$

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

$$= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right)$$ (Normal Distribution)

Anomaly if $p(x) < \varepsilon$

Training set 全是正常的，用来学习 μ_j 和 σ_j^2

Validation set 混合正常+不正常，用来调整 ε

Test set 混合正常+不正常，用 F1 score 评价表现

什么时候用 Anomaly Detection	or Supervised Learning
① 很少 positive example ② 很多 negative example ③ 异常的种类千奇百怪 很可能跟已收集到的异常不同	有许多 positive 和 negative 的例子可供学习
例子 ① Fraud Detection ② Manufacturing ③ Monitoring machines at a data center	① Email spam detection ② weather prediction ③ Cancer classification

注意：要把 non-Gaussian feature 转换为 Gaussian

可考虑的方法：$\log(x_i + c)$

x_i ^ c

Original Model vs. Multivariate Gaussian

Original Model	Multivariate Gaussian
① 要自己创造 feature 来考虑 x_i, x_j 之间的关系在内	① 自动考虑 feature 间的关系
② 资源要求小，适合多 feature	② 资源要求大
	③ 一定要 $m > n$

Anomaly Detection with multivariate Gaussian Algorithm

1. Fit model $p(x)$ by setting

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

2. Given a new example x, compute

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

flag anomaly if $p(x) < \varepsilon$

作业中问题：

① (A .* R).^2 不加括号会失败

② logical array 改里面数字要全定义为 float 或 inf 8

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values — to break symmetry
2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm)

$$J = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$$x_k^{(i)} := x_k^{(i)} - \alpha \underbrace{\left(\sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right)}_{= \frac{\partial}{\partial x_k^{(i)}} J}$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)}_{= \frac{\partial}{\partial \theta_k^{(j)}} J}$$

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)

$y^{(i,j)}$ = rating by user j on movie i

$\theta^{(j)}$: parameter vector for user j

$x^{(i)}$: feature vector for movie i

For user j, movie i, predicted rating : $(\theta^{(j)})^T (x^{(i)})$

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ \vdots \\ -(x^{(n_m)})^T- \end{bmatrix} \qquad \Theta = \begin{bmatrix} -(\theta^{(1)})^T- \\ \vdots \\ -(\theta^{(n_u)})^T- \end{bmatrix}$$

vectorized: $Y = X \Theta^T$

算cost和gradient的MATLAB vectorized 代码

```
estimated_error = ((X * Theta' - Y) .* R);
J = 0.5 * sum((estimated_error .^2), 'all');
J = J + lambda/2 *(sum(Theta .^2, 'all') + sum(X .^2, 'all'));

for i = 1:num_movies
    for k = 1:num_features
        X_grad(i,k) = sum(estimated_error(i,:)' .* Theta(:,k));
    end
    X_grad(i,:) = X_grad(i,:) + lambda * X(i,:);
end

for i = 1:num_users
    for k = 1:num_features
        Theta_grad(j,k) = sum(estimated_error(:,j) .* X(:,k));
    end
    Theta_grad(j,:) = Theta_grad(j,:) + lambda * Theta(j,:);
end
```

Week 10

① Stochastic Gradient Descent Algorithm

每次(步)根据一个sample优化全局参数 比batch快

1. Randomly shuffle (reorder) training examples

2. Repeat {

for $i := 1, \cdots, m$ {

$\theta_j := \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ (for every $j = 0, \cdots, n$)

}

}

② Mini-batch gradient descent

1. shuffle

say $b = 10$, $m = 1000$

2. Repeat {

for $i = 1, 11, 21, 31, \cdots 991$ {

$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$ (for every $j = 0, \cdots, n$)

}

}

①和②都可以动态变化学习率α，越往后越小来找更好的参数

③ Online learning

适合continuous data flow

来一个优化一次然后舍弃

④ Map Reduce and Data parallelism

只要算法是training set的某种和就可分拆平行计算

Week 11

① Pipeline：流程.

机器学习中经常把一件工作分拆成几个工序

比如 Photo OCR： Image → Text Detection → Character Segmentation → Character Recognition

② Artificial Data Synthesis

可以通过自己对原始数据集的加工来扩大训练集

比如：清晰语音加背景噪音

图像的变形

③ Ceiling Analysis

用来分析 pipeline 中哪些工序更需要改进

人工标记出输出那个工序的完美结果，然后观察整体算法性能提升幅度