Week 9

Density Estimation

Problem Motivation

Anomaly detection example

- Fraud detection:
  - $x^{(i)}$ = features of user i's activities
  - Model $p(x)$ from data.
  - Identify unusual users by checking which have $p(x) < \varepsilon$
- Manufacturing
- Monitoring computers in a data center.
  - $x^{(i)}$ = features of machine i

  x_1 = memory use, x_2 = number of disk accesses/sec,

  x_3 = CPU load, x_4 = CPU load/network traffic.

  ...

Your anomaly detection system flags x as anomalous whenever $p(x) \leq \epsilon$. Suppose your system is flagging too many things as anomalous that are not actually so (similar to supervised learning, these mistakes are called false positives). What should you do?

- Try increasing ϵ.
- Try decreasing ϵ.

Correct

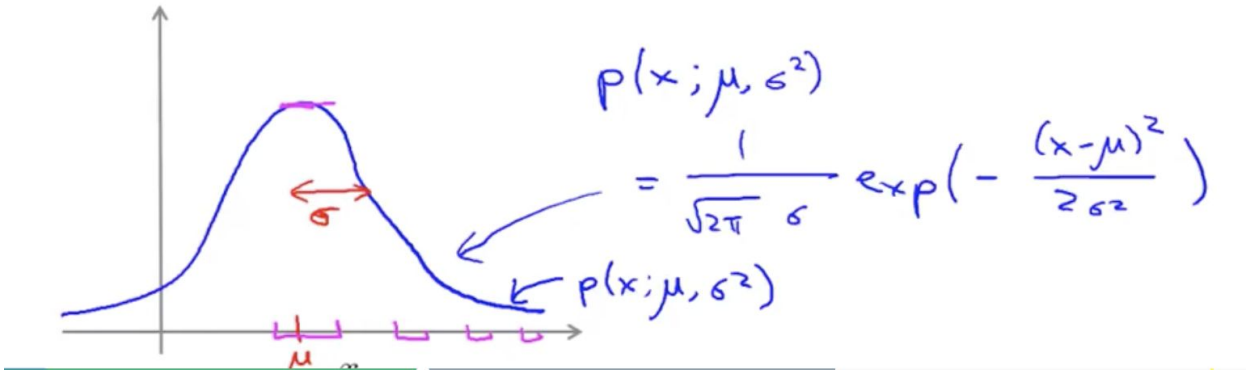
Gaussian Distribution

Gaussian (Normal) distribution

Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ, variance σ^2.

$x \sim \mathcal{N}(\mu, \sigma^2)$ ("distributed as")

σ = standard deviation



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

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Algorithm

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

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

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

   $$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

   Anomaly if $p(x) < \varepsilon$

Building an Anomaly Detection System

Developing and Evaluating an Anomaly Detection System

Aircraft engines motivating example

- 10000 good (normal) engines
- 20 flawed engines (anomalous)

Training set: 6000 good engines ($y = 0$)
CV: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)
Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Algorithm evaluation

- Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$
- On a cross validation/test example x, predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1-score

Can also use cross validation set to choose parameter ε

y=0 have high accuracy, thus put the equal sign in the normal line

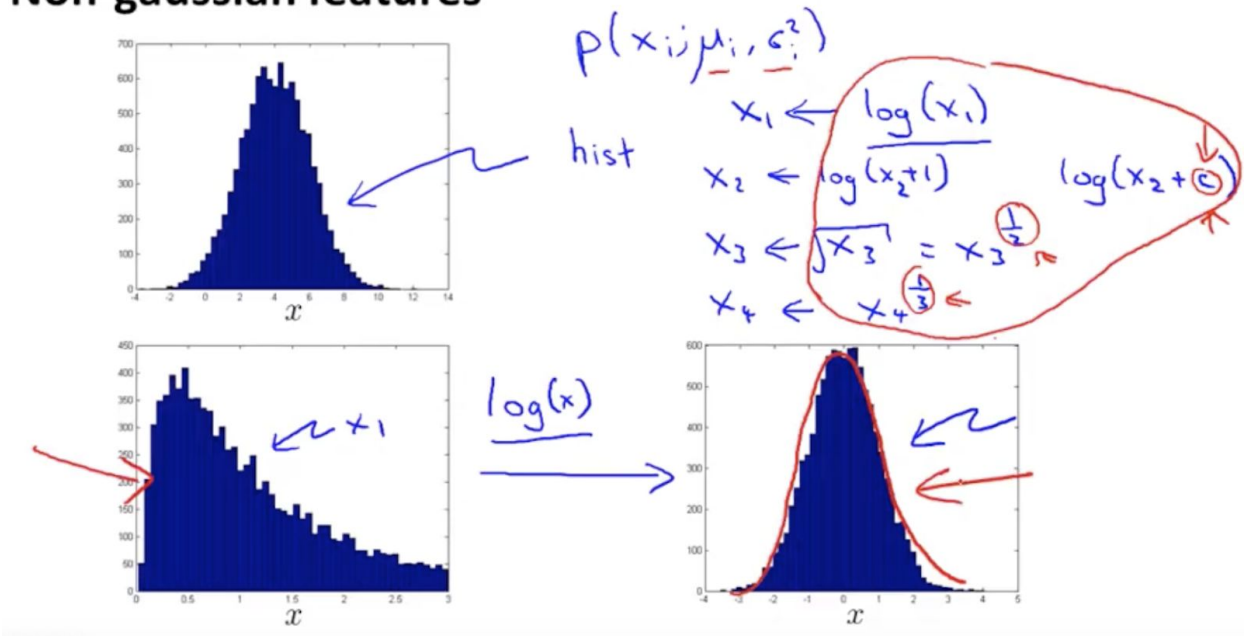
Anomaly Detection vs. Supervised Learning

Anomaly detection	vs.	Supervised learning
Very small number of positive examples ($y = 1$). (0-20 is common). Large number of negative ($y = 0$) examples.		Large number of positive and negative examples.
Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.		Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.

Anomaly detection	vs.	Supervised learning
• Fraud detection		• Email spam classification
• Manufacturing (e.g. aircraft engines)		• Weather prediction (sunny/rainy/etc).
• Monitoring machines in a data center		• Cancer classification

Choosing What Features to Use

Non-gaussian features

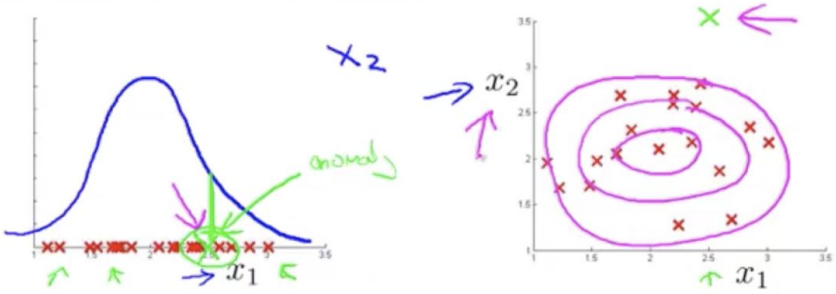


Error analysis for anomaly detection

Want $p(x)$ large for normal examples x.
$p(x)$ small for anomalous examples x.

Most common problem:

$p(x)$ is comparable (say, both large) for normal and anomalous examples



Multivariate Gaussian Distribution

Multivariate Gaussian Distribution

Multivariate Gaussian (Normal) distribution

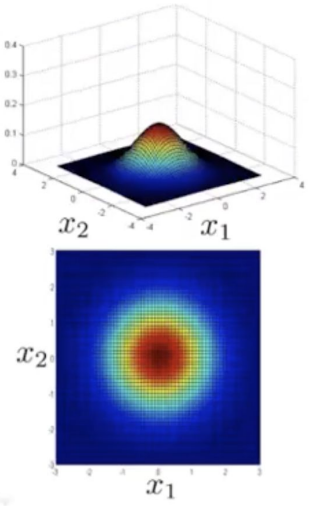
$x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \ldots$, etc. separately.
Model $p(x)$ all in one go.
Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

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

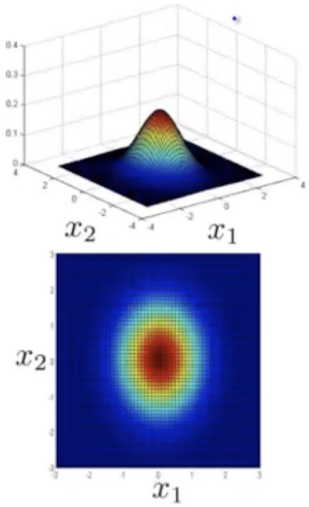
$|\Sigma|$ = determinant of Σ | det (Sigma)

Multivariate Gaussian (Normal) examples

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

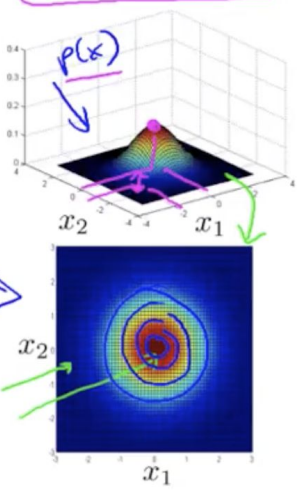


$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$

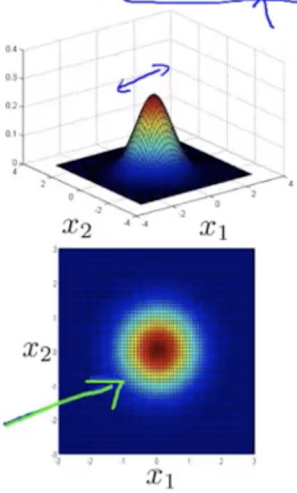


Multivariate Gaussian (Normal) examples

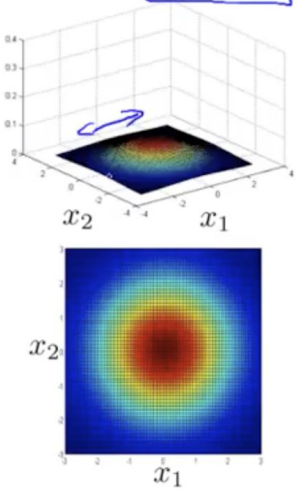
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

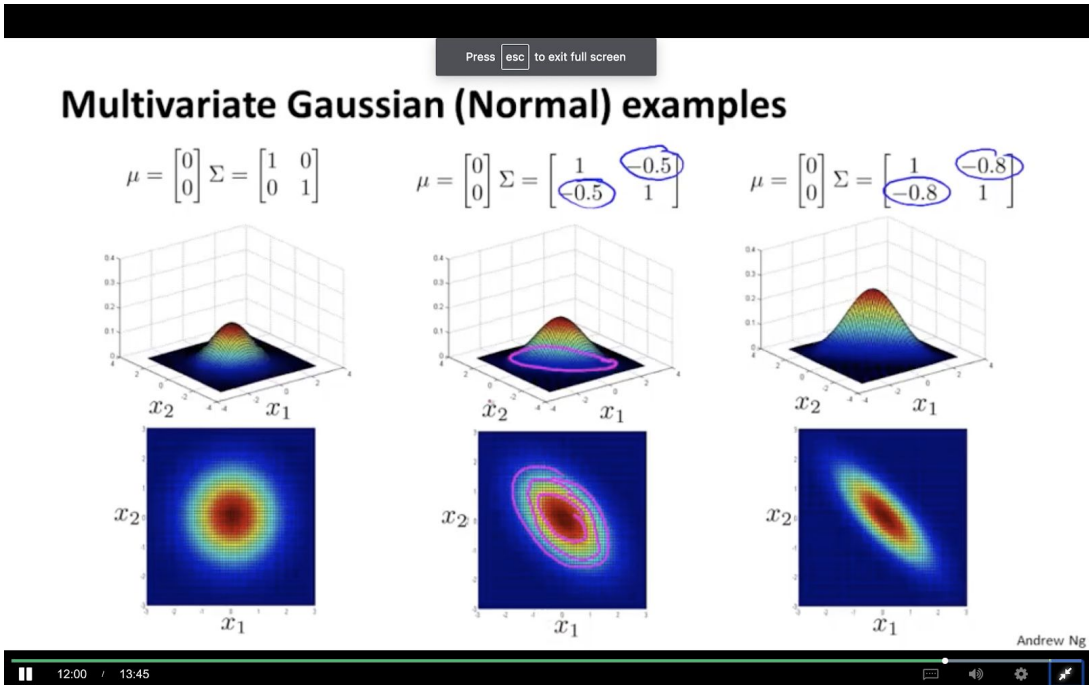


$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$



$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$





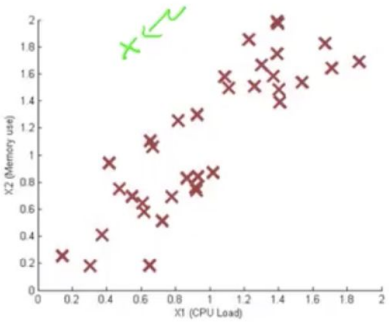
Anomaly Detection using the Multivariate Gaussian Distribution

Anomaly detection with the multivariate Gaussian

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

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

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



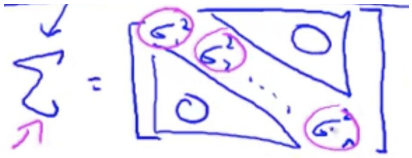
2. Given a new example x, compute

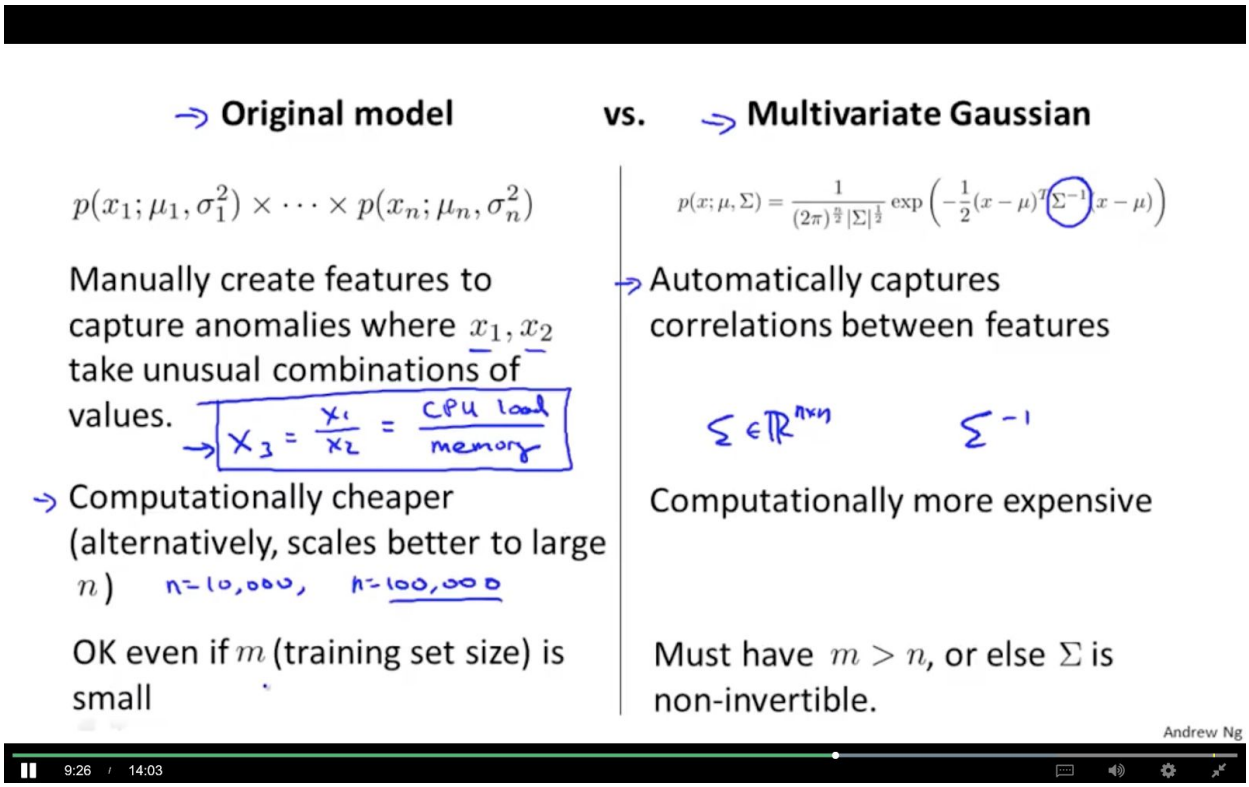
$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Flag an anomaly if $p(x) < \varepsilon$

Andrew Ng

2:37 / 14:03





The original model $p(x_1;\mu_1,\sigma_1^2) \times \cdots \times p(x_n;\mu_n,\sigma_n^2)$ corresponds to a multivariate Gaussian where the contours of $p(x;\mu,\Sigma)$ are axis-aligned.

Quiz

1. If you do not have any labeled data (or if all your data has label y =0), then it is still possible to learn p(x), but it may be harder to evaluate the system or choose a good value of ε
2. Anomaly detection only models the negative examples, whereas an SVM learns to discriminate between positive and negative examples, so the SVM will perform better when you have many positive and negative examples.

Predicting Movie Ratings

Problem Formulation

Recommender System

n_u = no. users

n_m = no. movies

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

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j) = 1$)

Content Based Recommendations

Content-based recommender systems

$n_u = 4$, $n_m = 5$

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
Love at last 1	5	5	0	0	0.9	0
Romance forever 2	5	?	?	0	1.0	0.01
Cute puppies of love 3	? 4.95	4	0	?	0.99	0
Nonstop car chases 4	0	0	5	4	0.1	1.0
Swords vs. karate 5	0	0	5	?	0	0.9

$x_0 = 1$, $x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$, $n = 2$

- For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

$\theta^{(j)} \in \mathbb{R}^{n+1}$

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$ $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ $(\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$

Andrew Ng

5:41 / 14:31

Problem formulation

- $r(i,j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating by user j on movie i (if defined)

- $\theta^{(j)}$ = parameter vector for user j
- $x^{(i)}$ = feature vector for movie i
- For user j, movie i, predicted rating: $(\theta^{(j)})^T (x^{(i)})$

$\theta^{(j)} \in \mathbb{R}^{n+1}$

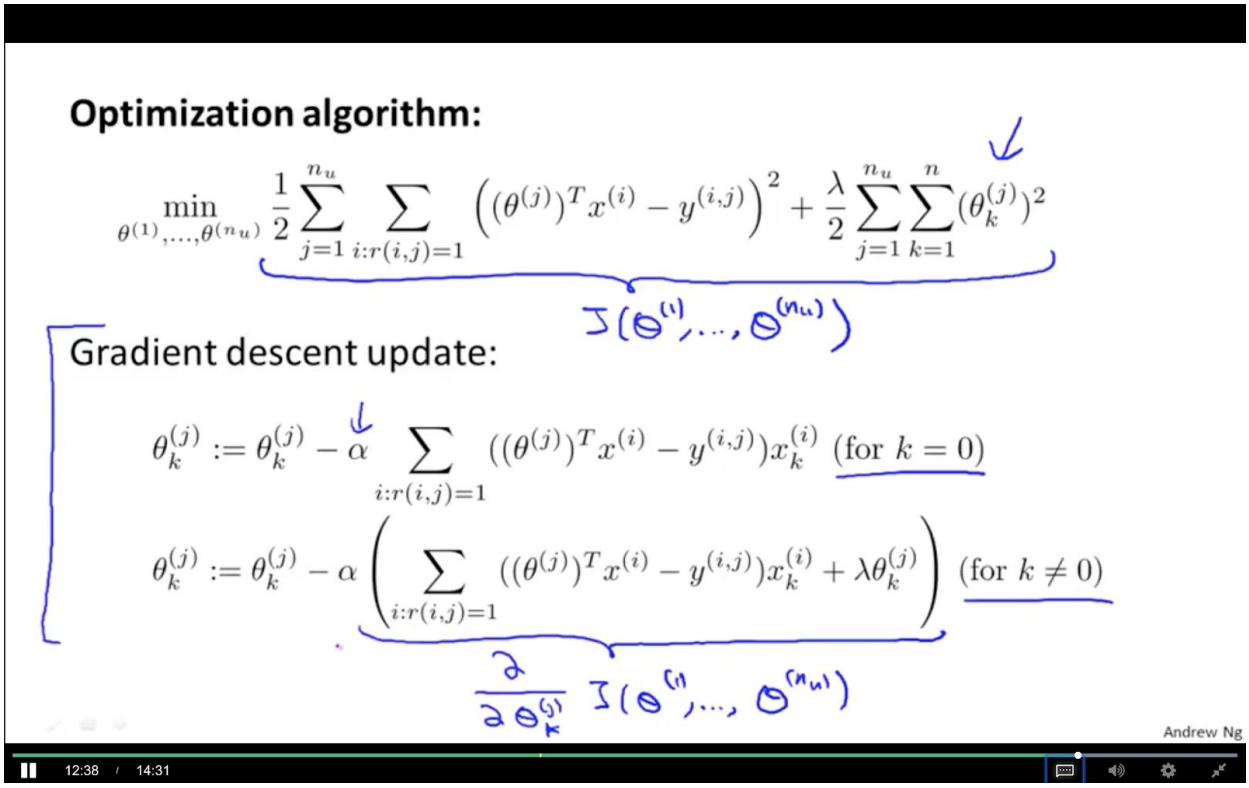
- $m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$:

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

Andrew Ng

9:32 / 14:31



Collaborative Filtering

Collaborative Filtering

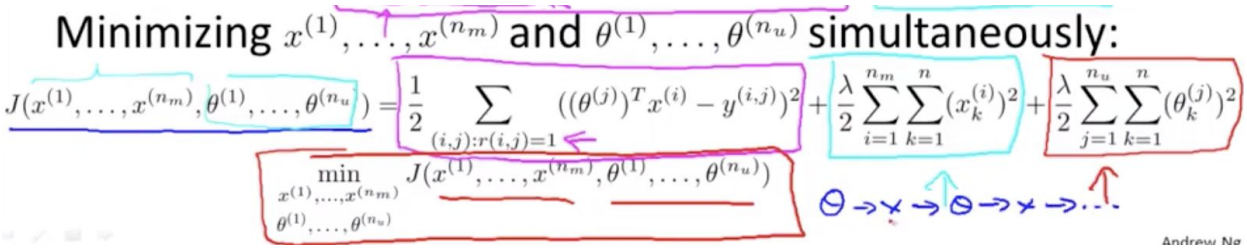
Recommender System

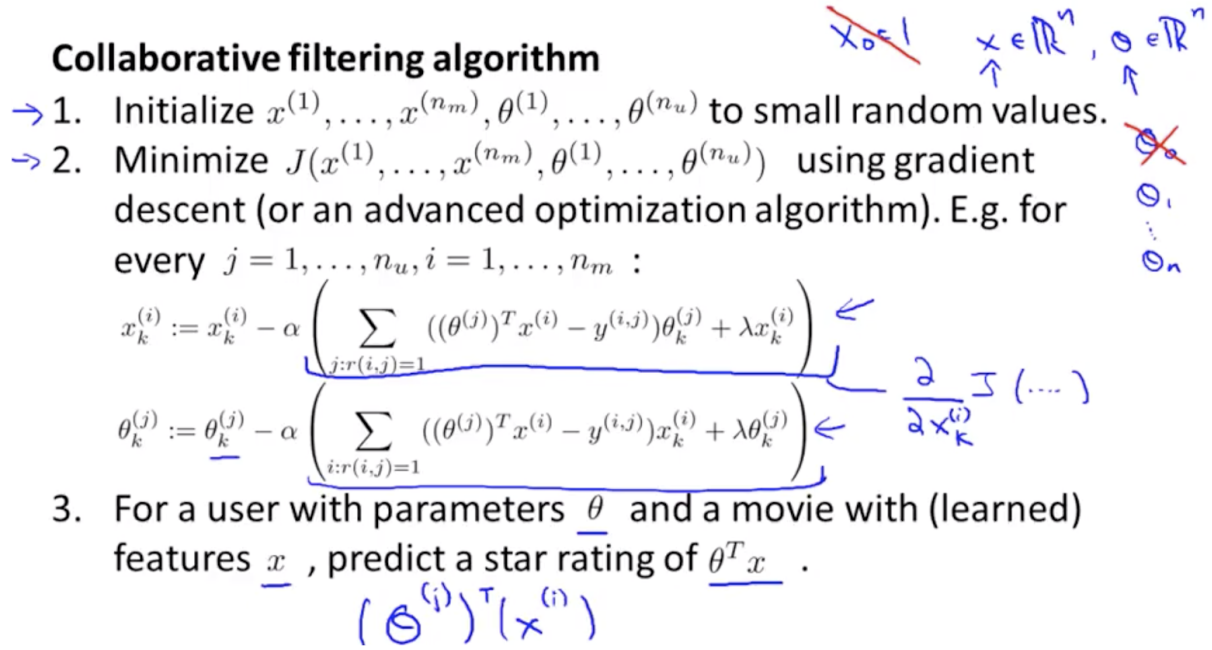
Collaborative filtering

Given $x^{(1)},\ldots,x^{(n_m)}$ (and movie ratings),
can estimate $\theta^{(1)},\ldots,\theta^{(n_u)}$

Given $\theta^{(1)},\ldots,\theta^{(n_u)}$,
can estimate $x^{(1)},\ldots,x^{(n_m)}$

Collaborative Filtering Algorithm





In the algorithm we described, we initialized $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values. Why is this?

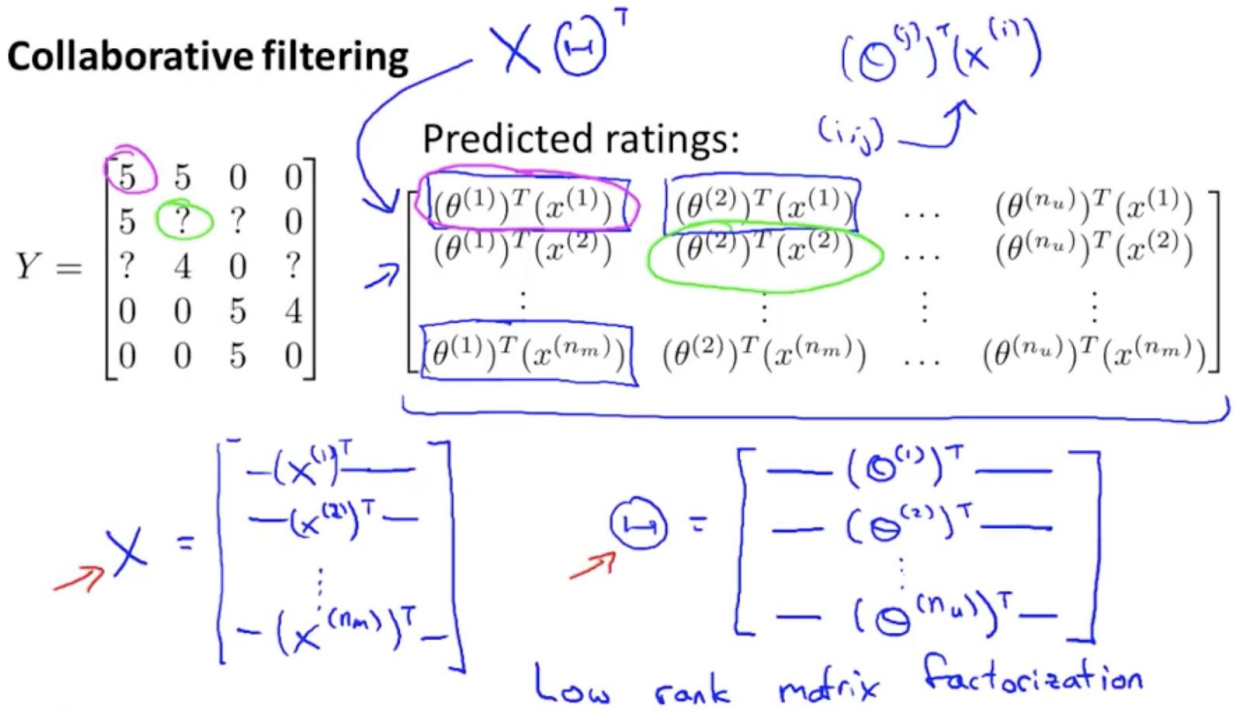
- This step is optional. Initializing to all 0's would work just as well.
- Random initialization is always necessary when using gradient descent on any problem.
- This ensures that $x^{(i)} \neq \theta^{(j)}$ for any i, j.
- This serves as symmetry breaking (similar to the random initialization of a neural network's parameters) and ensures the algorithm learns features $x^{(1)}, \ldots, x^{(n_m)}$ that are different from each other.

Correct

Low Rank Matrix Factorization

Vectorization: Low Rank Matrix Factorization

Collaborative filtering



Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

x_1 = romance, x_2 = action, x_3 = comedy, x_4 = ...

How to find movies j related to movie i?

small $\|x^{(i)} - x^{(j)}\|$ → movie j and i are "similar"

5 most similar movies to movie i:
Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.

Implementational Detail: Mean Normalization

Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

→ x_1 = romance, x_2 = action, x_3 = comedy, x_4 = ...

How to find movies j related to movie i?

small $\|x^{(i)} - x^{(j)}\|$ → movie j and i are "similar"

5 most similar movies to movie i:

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.

Quiz

Suppose you run a bookstore, and have ratings (1 to 5 stars) of books. Your collaborative filtering algorithm has learned a parameter vector $\theta^{(j)}$ for user j, and a feature vector $x^{(i)}$ for each book. You would like to compute the "training error", meaning the average squared error of your system's predictions on all the ratings that you have gotten from your users. Which of these are correct ways of doing so (check all that apply)? For this problem, let m be the total number of ratings you have gotten from your users. (Another way of saying this is that $m = \sum_{i=1}^{n_m} \sum_{j=1}^{n_u} r(i,j)$). [Hint: Two of the four options below are correct.]

- [ ] $\frac{1}{m} \sum_{(i,j):r(i,j)=1} (\sum_{k=1}^{n} (\theta^{(j)})_k x_k^{(i)} - y^{(i,j)})^2$
- [ ] $\frac{1}{m} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} (\sum_{k=1}^{n} (\theta^{(j)})_k x_k^{(i)} - y^{(i,j)})^2$

Suppose you have two matrices A and B, where A is 5x3 and B is 3x5. Their product is $C = AB$, a 5x5 matrix. Furthermore, you have a 5x5 matrix R where every entry is 0 or 1. You want to find the sum of all elements $C(i, j)$ for which the corresponding $R(i, j)$ is 1, and ignore all elements $C(i, j)$ where $R(i, j) = 0$. One way to do so is the following code:

```
C = A * B;
total = 0;
for i = 1:5
  for j = 1:5
    if (R(i,j) == 1)
      total = total + C(i,j);
    end
  end
end
```

Which of the following pieces of Octave code will also correctly compute this total? Check all that apply. Assume all options are in code.

A. total = sum(sum((A * B) .* R))

B. C = A * B; total = sum(sum(C(R == 1)));

Week 10

Gradient Descent with Large Datasets

Learning With Large Datasets

Stochastic Gradient Descent

Linear regression with gradient descent

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

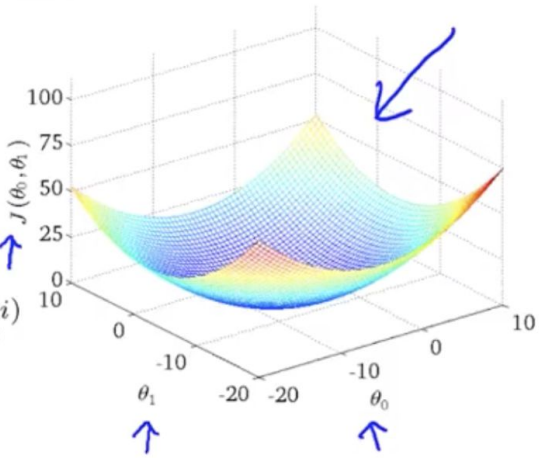
$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Repeat {

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

(for every $j = 0, \ldots, n$)

}



Batch gradient descent

$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Repeat {

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

$\frac{\partial}{\partial \theta_j} J_{train}(\theta)$

(for every $j = 0, \ldots, n$)

}

Stochastic gradient descent

$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$

1. Randomly shuffle dataset.

2. Repeat {

for $i = 1, \ldots, m$ {

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

(for $j = 0, \ldots, n$)

}

}

$\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots$

Mini-Batch Gradient Descent

Mini-batch gradient descent

- Batch gradient descent: Use all m examples in each iteration
- Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

b = mini-batch size. $b = 10$. 2-100

Get $b = 10$ examples $(x^{(i)}, y^{(i)}), \ldots (x^{(i+9)}, y^{(i+9)})$

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) \cdot x_j^{(k)}$$

$i := i + 10$

Stochastic Gradient Descent Convergence

Checking for convergence

Batch gradient descent:

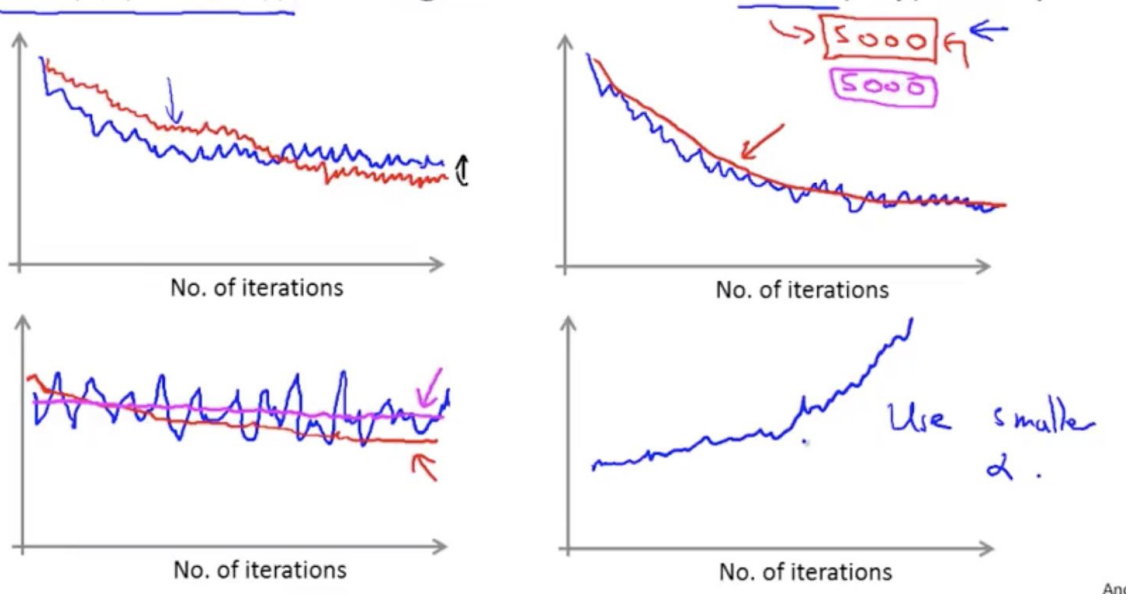
- Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.
- $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ $\quad m = 300,000,000$

Stochastic gradient descent:

- $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$ $\quad (x^{(i)}, y^{(i)}), (x^{(i+1)}, y^{(i+1)}), \ldots$
- During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.
- Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

Checking for convergence

Plot $cost(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1000 (say) examples



Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\texttt{const1}}{\texttt{iterationNumber + const2}}$)

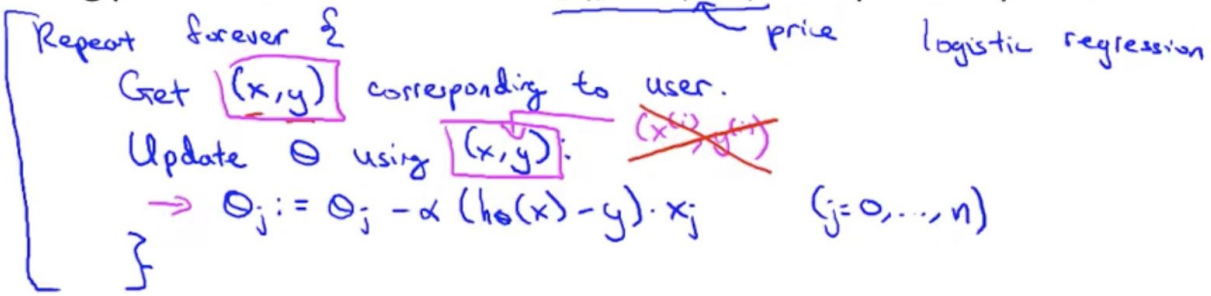
Advanced Topics

Online Learning

Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x; \theta)$ to optimize price.



Can adopt to changing user preferences.

Other online learning example:

Product search (learning to search)

User searches for "Android phone 1080p camera"

Have 100 phones in store. Will return 10 results.

→ x = features of phone, how many words in user query match name of phone, how many words in query match description of phone, etc.

→ $y = 1$ if user clicks on link. $y = 0$ otherwise.

→ Learn $p(y = 1|x; \theta)$. ← predicted CTR

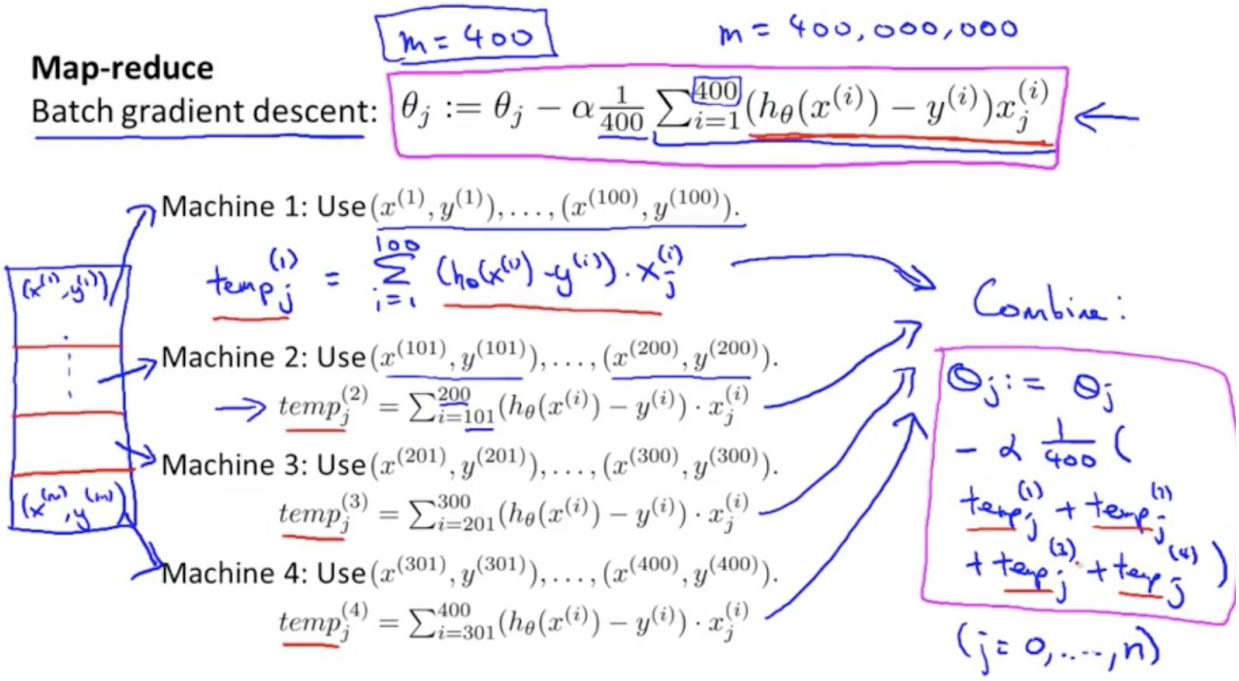
→ Use to show user the 10 phones they're most likely to click on.

(x, y)

CTR: click through rate

Other examples: Choosing special offers to show user; customized selection of news articles; product recommendation; ...

Map Reduce and Data Parallelism



Map-reduce and summation over the training set

Many learning algorithms can be expressed as computing sums of functions over the training set.

E.g. for advanced optimization, with logistic regression, need:

$$J_{train}(\theta) = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

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

$temp_j^{(1)}$ $temp_j^{(2)}$ ←

Suppose you apply the map-reduce method to train a neural network on ten machines. In each iteration, what will each of the machines do?

- Compute either forward propagation or back propagation on 1/5 of the data.
- Compute forward propagation and back propagation on 1/10 of the data to compute the derivative with respect to that 1/10 of the data.

Correct

Quiz

4. Assuming that you have a very large training set, which of the following algorithms do you think can be parallelized using map-reduce and splitting the training set across different machines? Check all that apply.

- CORRECT A neural network trained using batch gradient descent.
- WRONG Logistic regression trained using stochastic gradient descent.
- CORRECT Linear regression trained using batch gradient descent.
- WRONG An online learning setting, where you repeatedly get a single example (x,y), and want to learn from that single example before moving on.
- WRONG Computing the average of all the features in your training set

5. Which of the following statements about map-reduce are true? Check all that apply.

- WRONG Running map-reduce over N computers requires that we split the training set into N^2 pieces.
- WRONG In order to parallelize a learning algorithm using map-reduce, the first step is to figure out how to express the main work done by the algorithm as computing sums of functions of training examples.

right

- CORRECT When using map-reduce with gradient descent, we usually use a single machine that accumulates the gradients from each of the map-reduce machines, in order to compute the parameter update for that iteration.
- CORRECT If you are have just 1 computer, but your computer has multiple CPUs or multiple cores, then map-reduce might be a viable way to parallelize your learning algorithm.

Week 11

Application Example: Photo OCR

Problem Description and Pipeline

Photo OCR: Photo Optical Character Recognition

Pipeline: text detection, character segmentation, character classification

Sliding Windows

Getting Lots of Data and Artificial Data

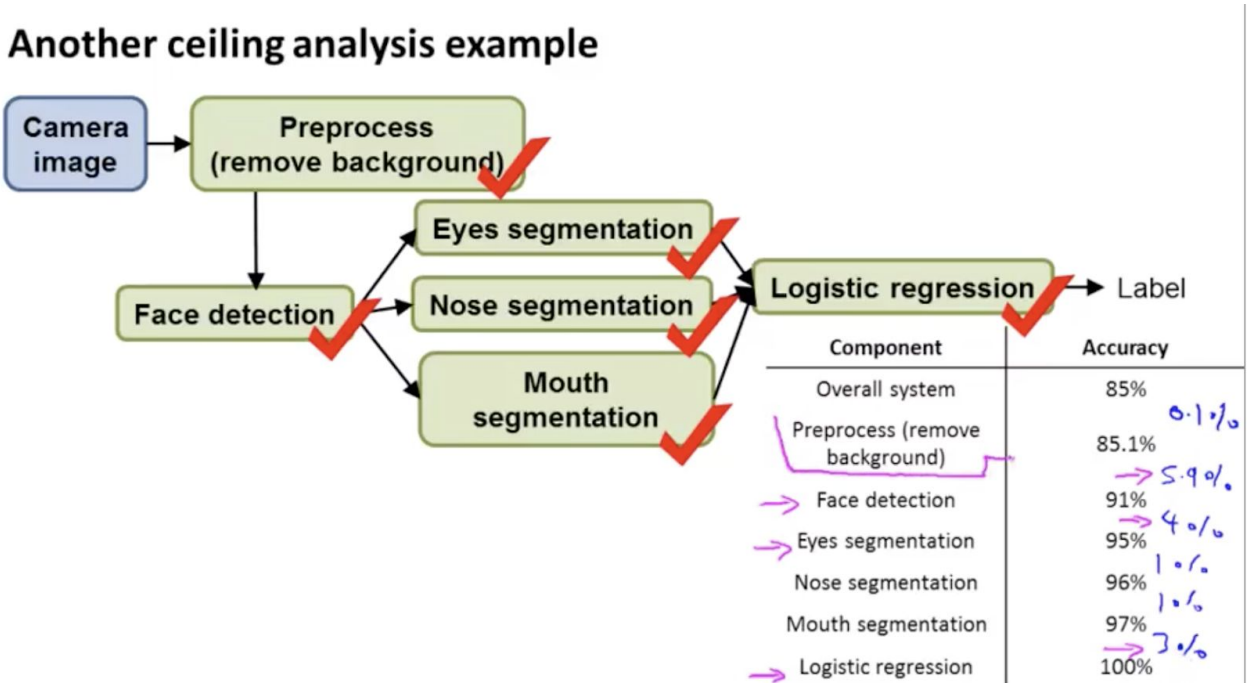
Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. "How much work would it be to get 10x as much data as we currently have?"
   - Artificial data synthesis
   - Collect/label it yourself
   - "Crowd source" (E.g. Amazon Mechanical Turk)

Ceiling Analysis: What Part of the Pipeline to Work on Next

Component	Accuracy
Overall system	72%
Text detection	89%
Character segmentation	90%
Character recognition	100%

Text detection对整个系统有17%的提升, character segmentation(1%), character recognition(10%)



Summary

Summary: Main topics

- Supervised Learning
  - Linear regression, logistic regression, neural networks, SVMs

  Unsupervised Learning
  - K-means, PCA, Anomaly detection

  Special applications/special topics
  - Recommender systems, large scale machine learning.

- Advice on building a machine learning system
  - Bias/variance, regularization; deciding what to work on next: evaluation of learning algorithms, learning curves, error analysis, ceiling analysis.