Anomaly detection

Algorithm

Density estimation

Training set : $\{x^{(1)}, \ldots, x^{(m)}\}$

Each example is $x \in \mathbb{R}$

$x_1 \sim N(\mu_1, \sigma_1^2)$

$x_2 \sim N(\mu_2, \sigma_2^2)$

$$
\begin{aligned}
&p(x) \\
&= p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2)\, p(x_3; \mu_3, \sigma_3^2) \ldots p(x_n; \mu_n, \sigma_n^2) \\
&= \boxed{\prod_{j=1}^{n}} p(x_j; \mu_j, \sigma_j^2)
\end{aligned}
$$

taking product of set of value

$$\sum_{i=1}^{n} i = 1+2+3+\ldots+n$$

$$\prod_{i=1}^{n} i = 1 \times 2 \times 3 \times \ldots \times n$$

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples.
   Then, given data set $\{x^{(1)}, \ldots, x^{(m)}\}$
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \xrightarrow{\text{can be vectorized}} \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

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

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 $p(x) < \varepsilon$

No. ______ Date ______

Anomaly detection

Developing and evaluating an anomaly detection system

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

Assume we have some labeled data, of anomalous and non-anomalous examples. ($y=0$ if normal, $y=1$ if anomalous).

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (unlabeled) (assume normal examples / not anomalous)

Then, Define

Cross validation set: $(x^{(1)}_{cv}, y^{(1)}_{cv}), \dots, (x^{(m_{cv})}_{cv}, y^{(m_{cv})}_{cv})$
Test set: $(x^{(1)}_{test}, y^{(1)}_{test}), \dots, (x^{(m_{test})}_{test}, y^{(m_{test})}_{test})$
} to evaluate a particular anomaly detection algorithm.

We can include few examples into that contain examples that are known to be anomalous

Aircraft engine motivating example

10000 good (normal) engines
20 flawed engines (anomalous) 20-50 examples $y=1$

Training set: 6000 good engines ($y=0$) → fit $p(x)$: $p(x) = p(x_1; \mu_1, \sigma_1^2) \cdots p(x_n; \mu_n, \sigma_n^2)$
CV: 2000 good engines ($y=0$), 10 anomalous ($y=1$)
Test: 2000 good engines ($y=0$), 10 anomalous ($y=1$)
} put anomalous just into cv and test set

Alternative: (But not recommended)

Training set: 6000 good engines
CV: $\boxed{4000}$ good engines ($y=0$), $\boxed{10}$ anomalous ($y=1$)
Test: $\boxed{4000}$ good engines ($y=0$), $\boxed{10}$ anomalous ($y=1$)
even same set of anomalous examples in 2 sets

Put same 4000 into cv and test set, not good, as we like to think of cv and test set as completely different data sets to each other.

Using the same data in cv and test set, not considered a good ML practice

ε – a threshold that we would use to decide when to flag something as an anomaly.

No. ______ Date ______

Algorithm evaluation

Fit model $p(x)$ on training set $\{x^{(1)}, \dots, x^{(m)}\}$ — unlabeled examples but are those we're assuming are good and normal aircraft engines

On a cv/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.

keep trying many value and pick the ε that maximize F_1-score / or does well in cv set.

Then evaluate algorithm on cv set, and make all those decisions like what features did you use, how to set epsilon, use to evaluate algorithm on cv set, then when we've picked set of features, and value of epsilon that good, we take the final model and do final evaluation of algorithm on test sets.

Quiz: Suppose you have fit a model $p(x)$. When evaluating on CV or test set, your algorithm predict:
$y \begin{cases} 1 \text{ if } p(x) \leq \varepsilon \\ 0 \text{ if } p(x) > \varepsilon \end{cases}$

Is classification accuracy a good way to measure the algorithm's performance?

No, because of skewed classes (so an algorithm that always predict $y=0$ will have high accuracy).

Anomaly detection

Anomaly detection vs. supervised learning

Anomaly detection (save for CV and ~~test set~~)	vs. Supervised learning
Very small number of (positive examples)($y=1$). (0-20 is common)	Large number of positive and negative examples.
Large number of negative ($y=0$) examples. ↳ to fit $p(x)$	
Many different "types" of anomalies. Hard for any algorithm to learn from small sets of positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far. (Maybe in your set of positive examples, you've seen 5 or 10 or 20 ways that an aircraft engine could go wrong, but maybe on another day you need to detect a totally new set/new type of anomaly / totally new way for an aircraft engine to be broken that you've just never seen before) So it would be better to just model negative examples ($y=0$) with $p(x)$ rather than try to model positive examples, as another day anomaly may be nothing like the one you'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.
Application:	→ then Fraud detection can be in supervised learning problem. Application
• Fraud detection (← or maybe you have a lot $y=1$ people out there have a lot of ways to do fraud)	• Email spam classification
• Manufacturing (e.g. aircraft engines)	• Weather prediction (sunny/rain/etc)
• Monitoring machines in a data center	• Cancer classification
⋮	⋮

Some anomaly detection can be shifted to supervised learning if the positive and negative examples enough much.

Anomaly detection algorithm

- You run a power utility (supplying electricity to customers) and want to monitor your electric plants to see if any one of them might be behaving strangely.
- A computer vision/security application, where you examine video images to see if anyone in your company's parking lot is acting in an unusual way.

Anomaly detection - Choosing what features to use

If the graph looks non-gaussian, take the features to log it, make gaussian

$x_1 \to \log(x_1)$ or $x_2 \to \log(x_2+1)$ / $\log(x_2+c)$ / $x_4 \to x_4^{\frac{1}{4}}$ / $x_3 \to x_3^{\frac{1}{3}}$

(c ↳ constant)

So to make it looks as Gaussian as possible.

Plot data in Octave → hist(x) ← check whether Gaussian or not (hist ← histogram)

if not then, hist(x .^0.1, 50) ← to make the graph show in 50 bins

if it is Gaussian enough, then xNew = x.^0.1;

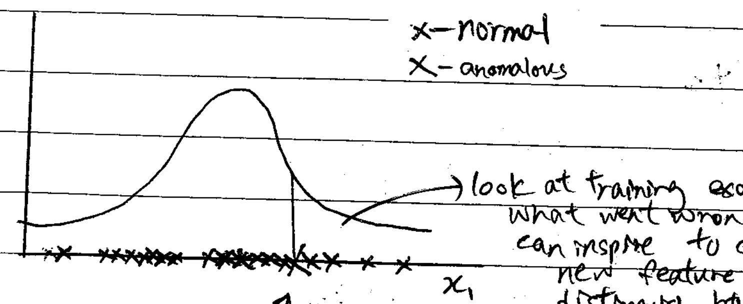
Error analysis for anomaly detection

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

$p(x)$ small for anomalous examples x.

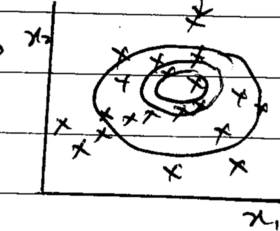
Most common problem:

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



↗ High probability and algorithm failed to flag this as anomalous example

→ look at training example, and check what went wrong, and check see can inspire to come up with new feature x_2 that helps to distinguish btwn bad example, compare to rest of the red example ⇒



maybe after plotting x_2, green point will be here

No.

Date

Monitoring computers in a data center

Choose features that might take on unusually large or unusually small values in the event of an anomaly.

x_1 = memory use of computer

x_2 = number of disk accesses/sec

x_3 = CPU load ←

x_4 = network traffic ← } this 2 work linearly, but maybe one failure case which code stuck in an infinite load, CPU load grow and network traffic doesn't

↳ new feature $x_5 = \frac{\text{CPU load}}{\text{network traffic}}$ ⇒ help in anomaly detection

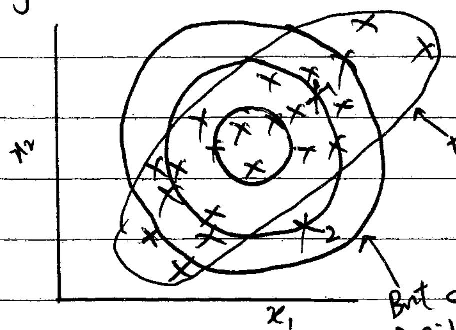
OR

$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$ (can come out with new features to capture different sort of anomaly examples.)

Quiz: Suppose your anomaly detection algorithm is performing poorly and outputs a large value of $p(x)$ for many normal examples and for many anomalous examples in your cross validation dataset. Which of the following changes to your algorithm is most likely to help?

- Try coming up with more features to distinguish between the normal and the anomalous examples.

Anomaly detection — Multivariate Gaussian distribution



this is what we think the probability in

But computer will think inside circle has probability

But actually x_1 has much more higher probability to be normal than x_2

We are going to fix this with Multivariate Gaussian

No.

Date

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.

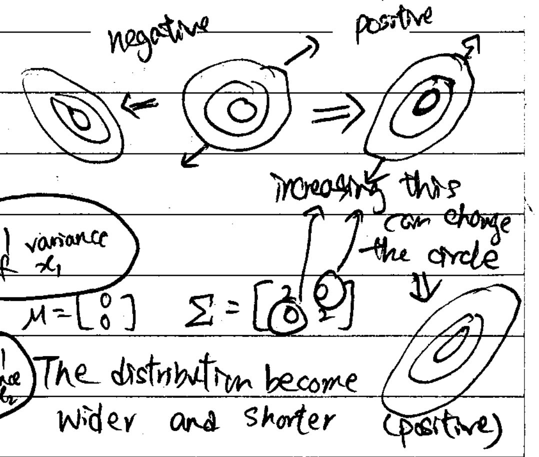
Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

$p(x; \mu, \Sigma)$

Probability

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

$|\Sigma|$ = determinant of Σ / Octave: det(sigma)



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

Change the peak of the distribution

normal distribution

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

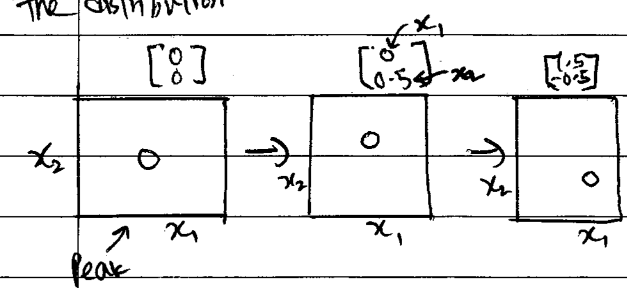
Control variance of x_1

Control variance of x_2

The distribution become narrower and taller.

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

The distribution become wider and shorter (positive)



Anomaly detection

Anomaly detection using the multivariate Gaussian distribution

Multivariate Gaussian (Normal) distribution

Parameters μ, Σ $\quad \mu \in \mathbb{R}^n \quad \Sigma \in \mathbb{R}^{n \times n}$

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

⇒ Can vary μ and Σ, can get a range of different distributions

Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ ← $x \in \mathbb{R}^n$

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

Anomaly detection with the multivariate Gaussian

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

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

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

will be flagged anomaly

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 $p(x) < \varepsilon$

Relationship to original model

Original model: $p(x) = p(x_1;\mu_1,\sigma_1^2) \times p(x_2;\mu_2,\sigma_2^2) \times \dots \times p(x_n;\mu_n,\sigma_n^2)$

- The contours of $p(x)$ are axis aligned

Corresponds to multivariate Gaussian

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

where
$$\Sigma = \begin{bmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \sigma_3^2 & \\ 0 & & & \ddots \ \sigma_n^2 \end{bmatrix}$$

all the numbers in the off diagonal are "0" zero

Original model	vs. Multivariate Gaussian
$p(x_1;\mu_1,\sigma_1^2) \times \dots \times p(x_n;\mu_n,\sigma_n^2)$	$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}\lvert\Sigma\rvert^{\frac{1}{2}}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$
Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.	Automatically captures correlations between features
Create new feature $x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}$	if $n = 10{,}000$, $\Sigma \in \mathbb{R}^{n \times n}$ ⇐ will be
Computationally cheaper (alternatively, scales better to large n)	Computationally more expensive
e.g. $n = 10{,}000$, $n = 100{,}000$ still work fine	$m < n$ / Redundant features like $x_1 = x_2$, $x_3 = x_4 + x_5$ — Can make Σ non-invertible
OK even if m (training set size) is small	Must have $m > n$, or else Σ is non-invertible.
	* will only use when $m >> n$ / $m \geq 10n$

Recommender Systems — Problem formulation

Example: Predicting movie ratings

User rates movies using ~~one~~ zero to five stars

? – Didn't watch and didn't rate, doesn't have rating

	Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Romantic comedy movies	Love at last	5	5	0	0
	Romance forever	5	(predict) maybe → ? 4.5	? 0	0
	Cute puppies of love	? 5	4	0	? 0
Action movies	Nonstop car chases	0	0	5	4
	Swords vs. karate	0	0	5	? 4 ← predict

n_u = no. users $\qquad n_u = 4 \quad n_m = 5$

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 $r(i,j) = 1$)

$0, \dots, 5$

⇒ Recommender system problem is given this dataset, that give these $r(i,j)$ and $y^{(i,j)}$ to look through data and look at all movie rating that are missing, try to predict what value of "?" should be.

⇒ Automatically fill in missing value, so can recommend new movie that users haven't watch to them, predict what can be interesting to users.

Recommender Systems

Content-based recommendations

Content-based recommender systems $n_u = 4$, $n_m = 5$

$x_0 = 1$ features

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

x_1 - measures degree to which a movie is a romantic movie

x_2 - measures degree to which a movie is an action movie

Have features, each movie can be represented with a feature vector.

movie 1

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$$

n - number of features not counting x_0

In this case, $n = 2$

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$ ($\mathbb{R}^{n+1}$ ($+x_0$)). Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$$

For this example, let's say we found out parameter vector for Alice

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

Our prediction for "?" in Alice $= (\theta^{(1)})^T x^{(3)}$

$= 5 \times 0.99$

$= 4.95$

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: $(\theta^{(j)})^T (x^{(i)})$

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

$\theta^{(j)} \in \mathbb{R}^{n+1}$ number of features we have per movie

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

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

To simplify the subsequent math

prediction of user j rating on movie i

actual user rating

Don't regularize bias term

Like linear regression

Just a constant, doesn't change $J(\theta)$

Sum over all movies user j has rated (summation over all values of i, so $r(i,j)$ is equal to 1)

Optimization objective

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

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

To learn $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n_u)}$:

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

Optimization algorithm:

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

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)}_{\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \ldots, \theta^{(n_u)})} \quad (\text{for } k \neq 0)$$

Can plug into more advanced optimization algorithm

Recommender Systems

Collaborative filtering

Problem motivation

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

~~Suppose we don't have~~ the features data set

Let's say we've gone to each of our users, each of the users told us how much they like the romantic movies and how much they like action packed movies.

$\theta^{1} = \begin{bmatrix}0\\5\\0\end{bmatrix}, \theta^{(2)} = \begin{bmatrix}0\\5\\0\end{bmatrix}, \theta^{(3)} = \begin{bmatrix}0\\0\\5\end{bmatrix}, \theta^{(4)} = \begin{bmatrix}0\\0\\5\end{bmatrix}$

If can get these, can infer value of x_1 and x_2

Both of them love romantic movie but hate action movie

Both of them love action movie but hates romantic movie

What $x^{(1)}$ should be so that

$(\theta^{(1)})^T x^{(1)} \approx 5$

$(\theta^{(2)})^T x^{(1)} \approx 5$

$(\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(4)})^T x^{(1)} \approx 0$

$x^{(1)} = \begin{bmatrix}1.0\\0.0\end{bmatrix}$

can conclude $x^{(1)}$ as $[x_1 = 1.0,\ x_2 = 0.0]$

and can keep on to figure out x_1, x_2 in movie $x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}$

Quiz

	User 1	User 2	User 3	(romance)
Movie 1	0	1.5	2.5	?

Note that there is only one feature x_1, suppose that:

$\theta^{(1)} = \begin{bmatrix}0\\0\end{bmatrix}\quad \theta^{(2)} = \begin{bmatrix}0\\3\end{bmatrix}\quad \theta^{(3)} = \begin{bmatrix}0\\5\end{bmatrix}$

What would be a resonable value for $x_1^{(1)}$?

$\Rightarrow 0.5$

Optimization algorithm

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(i)}$:

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

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(1)}, \dots, x^{(n_m)}$:

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

Quiz Suppose you use gradient descent to minimize:

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

Correct gradient descent update rule for $i \neq 0$

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

Collaborative filtering

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

Parameters (By user) Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, can estimate $x^{(1)}, \dots, x^{(n_m)}$

Can start by making initial guess on $\theta \xrightarrow{\text{fit}} X \xrightarrow{\text{Make better changes}} \theta \xrightarrow{\text{Better fit}} X \to \theta \to X \to \dots$

keep iterating and optimizing

$\Rightarrow$ will cause the algorithm to converge

Recommender Systems

Collaborative filtering algorithm

Collaborative filtering optimization objective

Given $x^{(1)},\dots,x^{(n_m)}$, estimate $\theta^{(1)},\dots,\theta^{(n_u)}$:

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

Given $\theta^{(1)},\dots,\theta^{(n_u)}$, estimate $x^{(1)},\dots,x^{(n_m)}$:

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

Minimizing $x^{(1)},\dots,x^{(n_m)}$ and $\theta^{(1)},\dots,\theta^{(n_u)}$ simultaneously:

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

$$\min_{\substack{x^{(1)},\dots,x^{(n_m)}\\ \theta^{(1)},\dots,\theta^{(n_u)}}} J(x^{(1)},\dots,x^{(n_m)},\theta^{(1)},\dots,\theta^{(n_u)})$$

instead of going

$\theta \to x \to \theta \to x \to \theta$ ...... we minimize with respect to both sets of parameters simultaneously

~~x_0~~ $x \in \mathbb{R}^n$ ~~$x \in \mathbb{R}^{n+1}$~~

$\theta \in \mathbb{R}^n$

no need $x_0 = 1$ as if the algorithm wants the feature to be 1, it can learn itself and can set $x_1 = 1$

Collaborative filtering algorithm

~~x_0~~ ~~$x_0=1$~~ $x \in \mathbb{R}^n$ $\theta \in \mathbb{R}^n$

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

   E.g. for every $j=1,\dots,n_u$, $i=1,\dots,n_m$:

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

   Partial derivative value

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

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

Predict User j is going to rate movie i with $(\theta^{(j)})^T(x^{(i)})$

Quiz: In algorithm, we initialized $x^{(1)},\dots,x^{(n_m)}$ and $\theta^{(1)},\dots,\theta^{(n_m)}$ to small random values. Why?

This serves as symmetry breaking (similar to the random initialization of a neural network's parameters) and ensures the algorithm learns features $x^{(1)},\dots,x^{(n_m)}$ that are different from each other.

Vectorization: Low rank matrix factorization

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs karate	0	0	5	?

$n_m = 5$
$n_u = 4$

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

5x4

$y^{(i,j)}$

Collaborative filtering

Predicted rating: $\underbrace{(\theta^{(j)})^T(x^{(i)})}_{(i,j)}$

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \cdots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \cdots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \cdots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

$X\Theta^T$

$$X = \begin{bmatrix} - (x^{(1)})^T - \\ - (x^{(2)})^T - \\ \vdots \\ - (x^{(n_m)})^T - \end{bmatrix} \qquad \Theta = \begin{bmatrix} - (\theta^{(1)})^T - \\ - (\theta^{(2)})^T - \\ \vdots \\ - (\theta^{(n_u)})^T - \end{bmatrix}$$

Low rank matrix factorization

Finding related movies

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

⇒ Let's say we have x_1 = romance, x_2 = action, x_3 = comedy, ...

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 $\|x^{(i)} - x^{(j)}\|$.

smallest distance between features between these different movies

Recommender Systems Date

Implementational detail: Mean normalization

Users who have not rated any movies

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	Eve(5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs karate	0	0	5	0	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

$$\min_{\substack{x^{(1)},\dots,x^{(n_m)} \\ \theta^{(1)},\dots,\theta^{(n_u)}}} \underbrace{\frac{1}{2}\sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2}_{\text{Don't have value, so going to be 0}} + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2}_{\frac{\lambda}{2}[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2]}$$

$n = 2$ $\quad \theta^{(5)} \in \mathbb{R}^2$

$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ← will regularize and make it near to 0 as possible

$(\theta^{(5)})^T x^{(i)} = 0$

Not good.

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \begin{matrix} \leftarrow 2.5 \\ \leftarrow 2.5 \\ \leftarrow 2 \\ \leftarrow 2.25 \\ \leftarrow 1.25 \end{matrix} \quad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

$$Y - \mu = Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

↓ Use as actual rating

Learn $\theta^{(j)}, x^{(i)}$

For user j, on movie i predict:

$$(\theta^{(j)})^T(x^{(i)}) + \mu_i$$

∴ ~~Movie~~ rating are comparable (0–5). are on similar scales so no need to do feature scaling.

User 5 (Eve):

$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\quad (\theta^{(5)})^T(x^{(i)}) + \mu_i$

$= 0 + \mu_i$

⇒ If user hasn't rated any movies, we just predict for each of the movies, the average rating that those movies got