Machine Learning class notes Documentation

Release 1.0.0

Dhananjay Nene

Aug 29, 2017

Contents

Contents:

CHAPTER 1

Lecture 16: Anomaly Detection

Video 16-1: Problem Motivation

- Like unsupervised problem though some aspects are similar to supervised learning
- Example being discussed of anomaly detection in aircraft engines.

Anomaly detection in aircraft engines

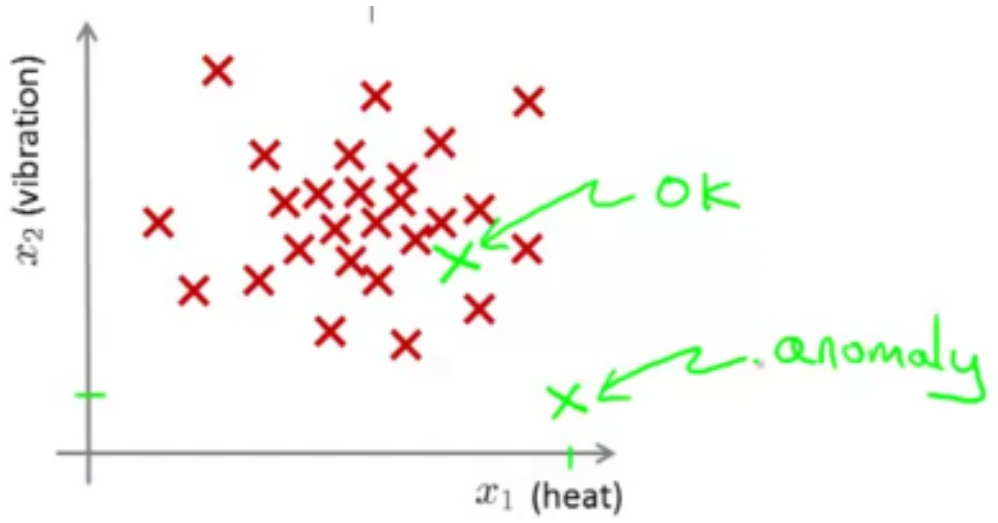
Aircraft engine features are :

- x_1 = heat generated
- x_2 = vibration intensity

Dataset = $x^{(1)}, x^{(2)}, ...x^{(m)}$

New engine rolls of the assembly line: x_{test}

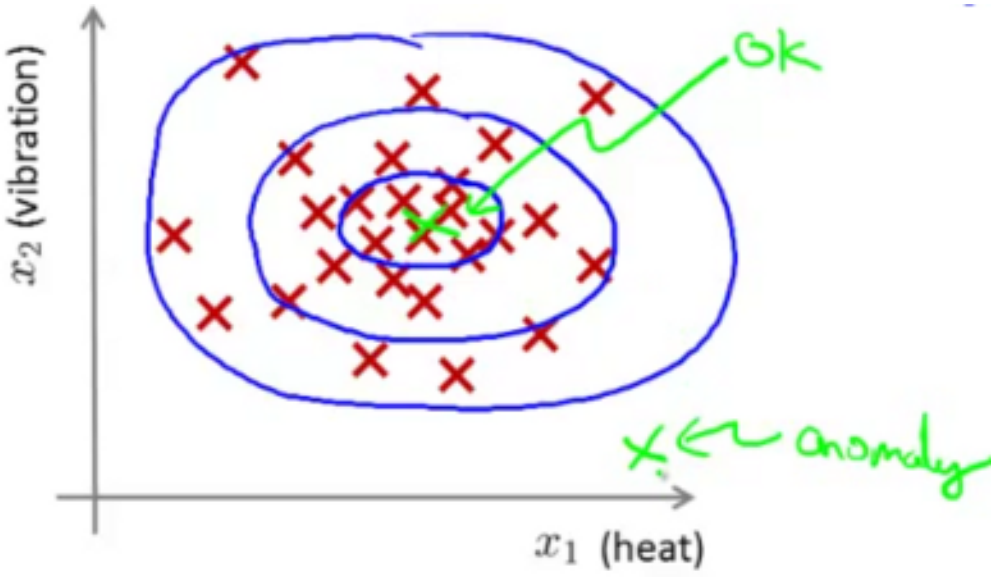
Question: Is the new engine anomalous or should it receive further testing as the two possibilities in the following graph.



- Assumption is that the dataset provided is non-anomalous or normal
- We need to build a model to predict the probability of the aircraft being appropriate.

  ie. if $p(x_{test}) < \epsilon$ ==> *then we flag an anomaly*

In the following example, the closer the point is to the inner circle, the higher is the likelihood of it being non-anomalous. On the other hand in the point which is far out (eg. the *x* near the bottom of the image), the likelihood of the engine being anomalous is high.



Video 16-2: Anomaly detection example

One of the most frequent usages of anomaly detection is that of fraud detection.

- $x^{(i)}$ = features of user i's activities
- Model $p(x)$ from data
- Identify unusual users by checking which may have $p(x) < \epsilon$

Another use case could be manufacturing (eg. as discussed earlier with aircraft engines).

Anomaly detection can also be used to monitor computers in a data center. eg.

- $x^{(i)}$ = 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 etc.

Identify machines that are likely to fail and flag off for attention.

Video 16-2: Gaussian Distribution

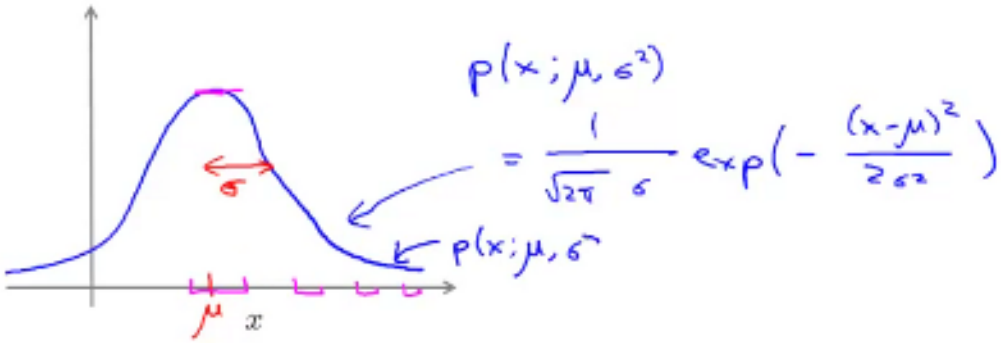
Note: This is more of an aside video focusing on Gaussian Distribution per se, rather than anomaly detection.

Say $xE\mathbb{R}$. x is a distributed Gaussian with mean μ and variance σ^2. The distribution is a bell shaped curve centred at μ. σ (or standard deviation) is indicative of the width of the bell curve.

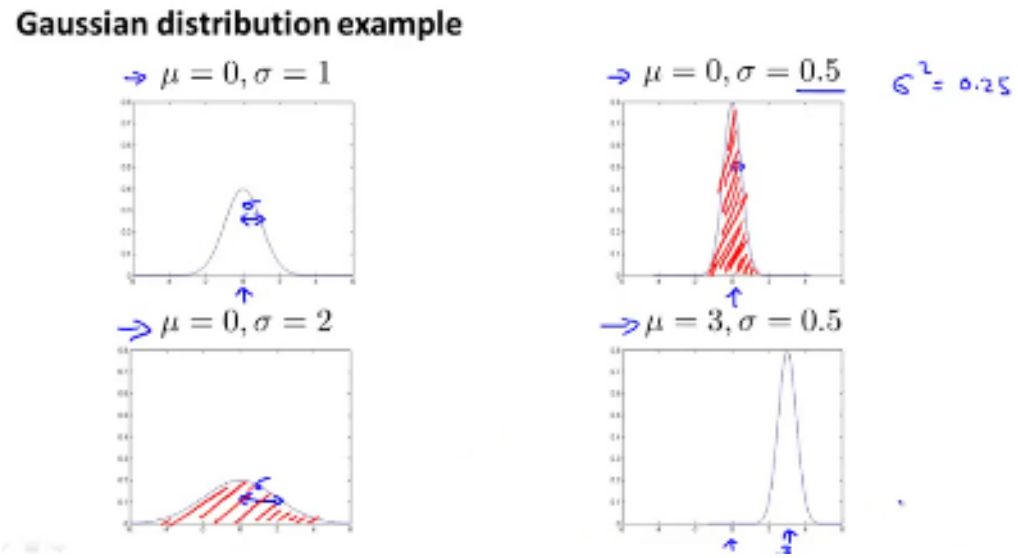
This is expressed as $x \sim N(\mu, \sigma^2)$

The equation for the probability distribution is :

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



The impact of varying μ and σ^2 on the distribution function is shown in the image below.



The equation for computing the mean is :

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

The equation for computing the variance is :

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

Video 16-3: Anomaly detection algorithm

Density estimation

Lets say we have a

- Training set : $x^{(1)}, ..., x^{(m)}$, and
- each example is $(x \epsilon \mathbb{R}^n)$ ie. has *n* features.

Assume that each feature is distributed as per gaussian probability distribution. ie. $x_1 \sim N(\mu_1, \sigma_1^2)$ and $x_2 \sim N(\mu_2, \sigma_2^2)$ and so on...

The computed probability is thus

$$p(x) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * ... * p(x_n; \mu_n, \sigma_n^2)$$

Note: Even if the above formula is that for computing probability for independent variables, in practice it works quite well even if the features are not independent.

The above expression could be summarised as

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

Note: The symbol $\prod_{j=1}^{n}$ is similar to $\sum_{j=1}^{n}$ except that it computes the product of all the values in the series rather than adding them up.

This computation of the probability $p(x)$ is often referred to as *Density Estimation*.

1. Choose features x_i that you think might be indicative of anomalous examples. Especially choose those for whom either unusually high or unusually low values of x_i might be indicative of existence of anomalies.

1. Fit parameters $\mu_1, ..., \mu_n, \sigma_1^2, ..., \sigma_n^2$ using

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

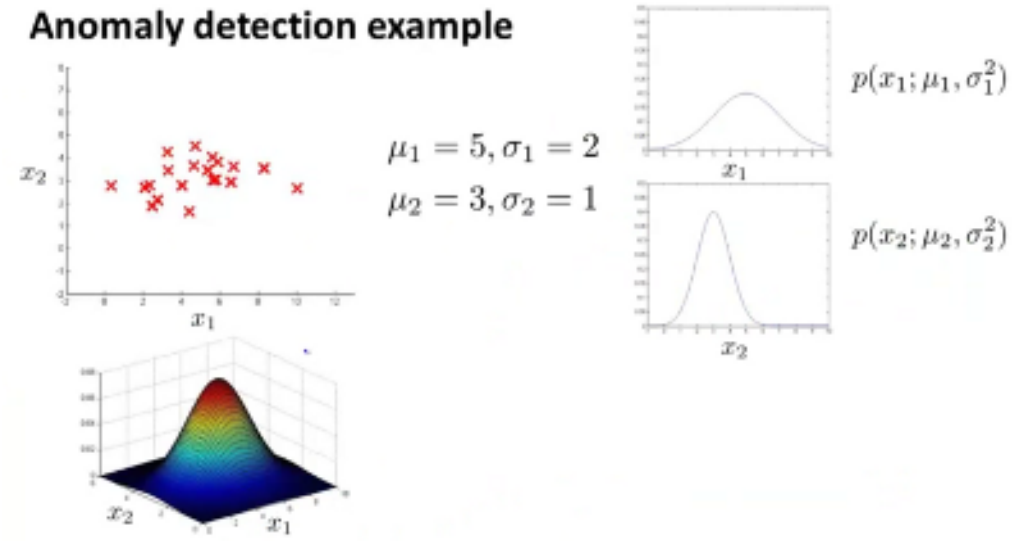
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$$

The corresponding vectorised implementations is $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ and $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$ where $xE\mathbb{R}^n$

1. Given new example x, compute $p(x)$

Anomaly detection example

Anomaly if $p(x) < \epsilon$



In the above example, x_1 and x_2 are two different features. The graphs on the right show their gaussian distribution curves, which are different from each other.

At the top left is the plot of the known combinations of x_1 and x_2 (which of course was used to compute the necessary μ and σ^2 values.

The figure at the bottom left shows the effective probability of occurrence of particular combinations of x_1 and x_2. Thus any points in this graph where the height of the point on the surface matching the particular point values of x_1 and x_2 is very low, can be viewed as likely anomalous.

Video 16-4: Developing and evaluating an anomaly detection system

- One of the important aspects of being able to develop an anomaly detection system is being able to first have a way of evaluating the anomaly detection system. This can help decide later whether specific feature additions or removals are actually helping or hurting the anomaly detection system.
- The starting point would be some labeled data of anomalous and non-anomalous data (labels being whether the particular case is anomalous or non-anomalous).
- The training set should consist of the non-anomalous subset of the data referredt to above .. these would be $x^{(1)}, x^{(2)}, ..., x^{(m)}$, Ideally this data should *not* contain the anomalous data points. However if a few of them do seep through thats probably not such a big deal.
- On the other hand both the cross validation set $(x_{cv}^{(1)}, y_{cv}^{(1)}), ..., (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$ and the test set $(x_{test}^{(1)}, y_{test}^{(1)}), ..., (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$ should contain some elements which are known to be anomalous.

Example : Aircraft Engines

Let us consider a scenario where we have data for 10,000 good/normal engines and 20 flawed/anomalous engines. One may want to consider that the data should be split up as follows :

- Training set: 6000 good engines (unlabeled since all are considered good)
- Cross validation set: 2000 good engines, 10 anomalous
- Test set : 2000 good engines, 10 anomalous

Use the training set to compute $\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, ...$ and thus the density estimation as well, ie. fit the model $p(x)$.

Now on the cross validation/test example x, predict,

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

Note: y above is a prediction. You can now contrast it with the actual data in your cross validation set. Note that data is extremely skewed ie. #normal points are substantially greater than #anomalous. Thus classification accuracy would not be a good evaluation metric. Instead computing the following might be useful.

- % of True/False +ves/-ves, or
- Precision / Recall
- F_1 score

One could attempt to apply different values of ϵ on the cross validation set, and choose the value that maximises the F_1 score. Finally apply the selected value on the test set, and recompute the metrics above.

Video 16-5: Anomaly Detection vs. Supervised Learning

If there is labeled data ie. labeled anomalous or normal, why don't we just use techniques for supervised learning?

- Usually anomaly detection is likely to be useful in scenarios where there is a very very small number of positive (ie. anomalous or y = 0) scenarios.
- Anomaly detection might be more useful, where it is hard for an algorithm to learn from positive examples what the anomalies look like (could also cover situations where future anomalies may look nothing like the anomalies we have seen so far).

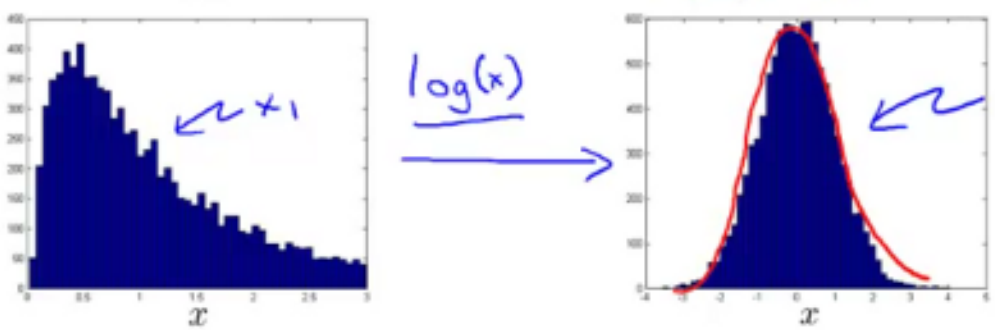
- Candidate uses for Anomaly detection : Fraud detection, manufacturing defects, monitoring machines in a data center.
- Candidate uses for supervised learning : Email spam classification, Weather prediction, (sunny / rainy etc)., Cancer classification.

Note: If you are a large online retailer and you have data about a large number of users who have been identified to commit fraud, then fraud detection in such a context might shift to a supervised learning approach rather than an anomaly detection one.

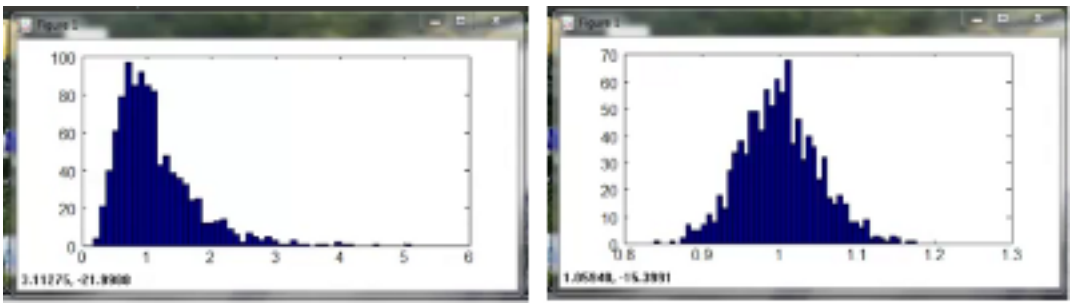
Video 16-6: Choosing what features to use

Non-gaussian features

Would be useful to plot a histogram of various features to get a sense if the features are gaussian. In many situations even if the feature is not showing gaussian distribution, it still might be Ok to consider to go ahead assuming it is so. However sometimes many features show themselves to be substantially non gaussian. In such a situation it might be useful to figure out a transformation to process the feature into a gaussian feature eg. $log(x)$ instead of x. Other options could be $log(x2 + c)$, $\sqrt{x_3}$, .. etc.



Above: Transformation of a non-gaussian to a gaussian distribution using log(x)



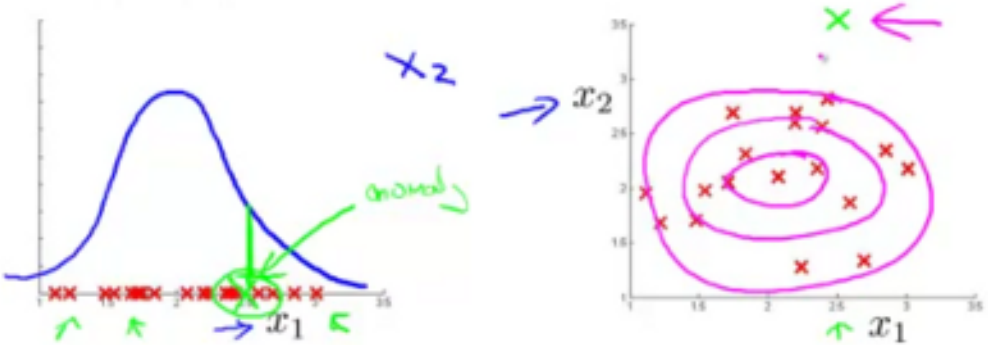
Above: Transformation using :math:‘xNew = x ^wedge 0.05‘

Error Analysis

How does one come up features appropriate for anomaly detection?

- Try to study the features by applying them on the cross validation set.

- You might find situations where say $p(x)$ is high for anomalous examples as well. Lets say you find an example where $p(x)$ is high for a clearly anomalous situation. Study that particular example to identify perhaps an additional feature that would lead to this particular situation getting flagged as an anomaly.

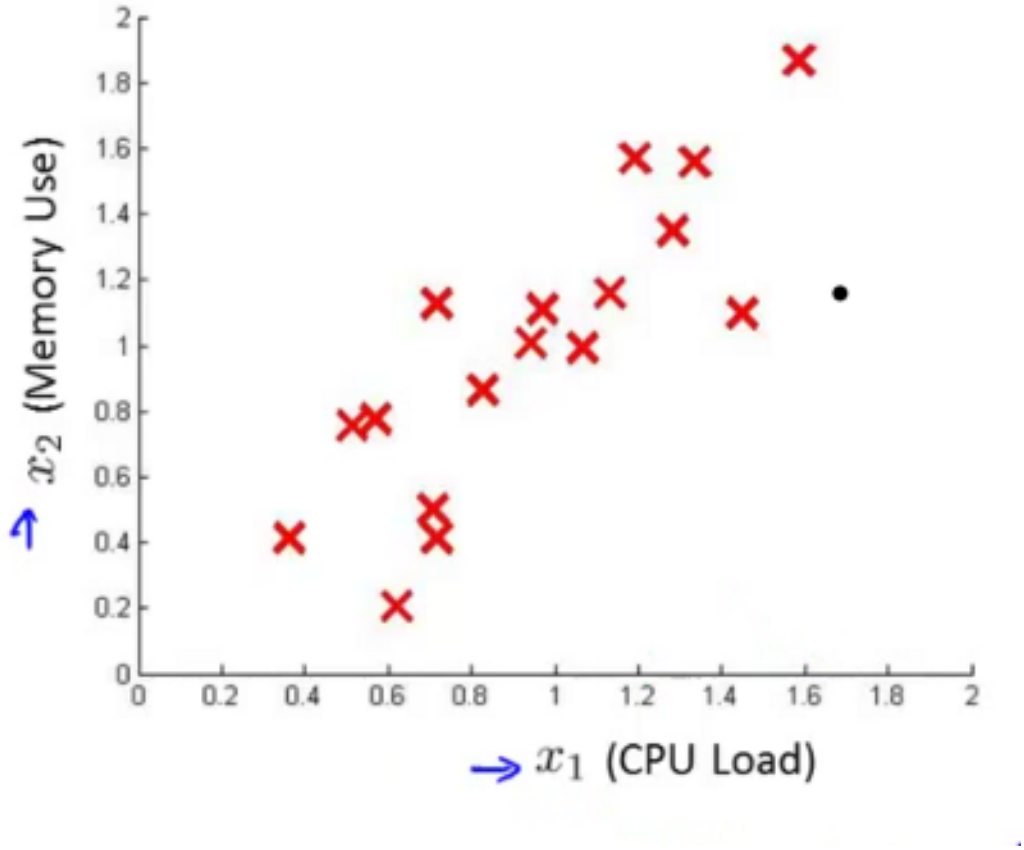


Above: identifying a new feature by looking at anomalies with a high p(x)

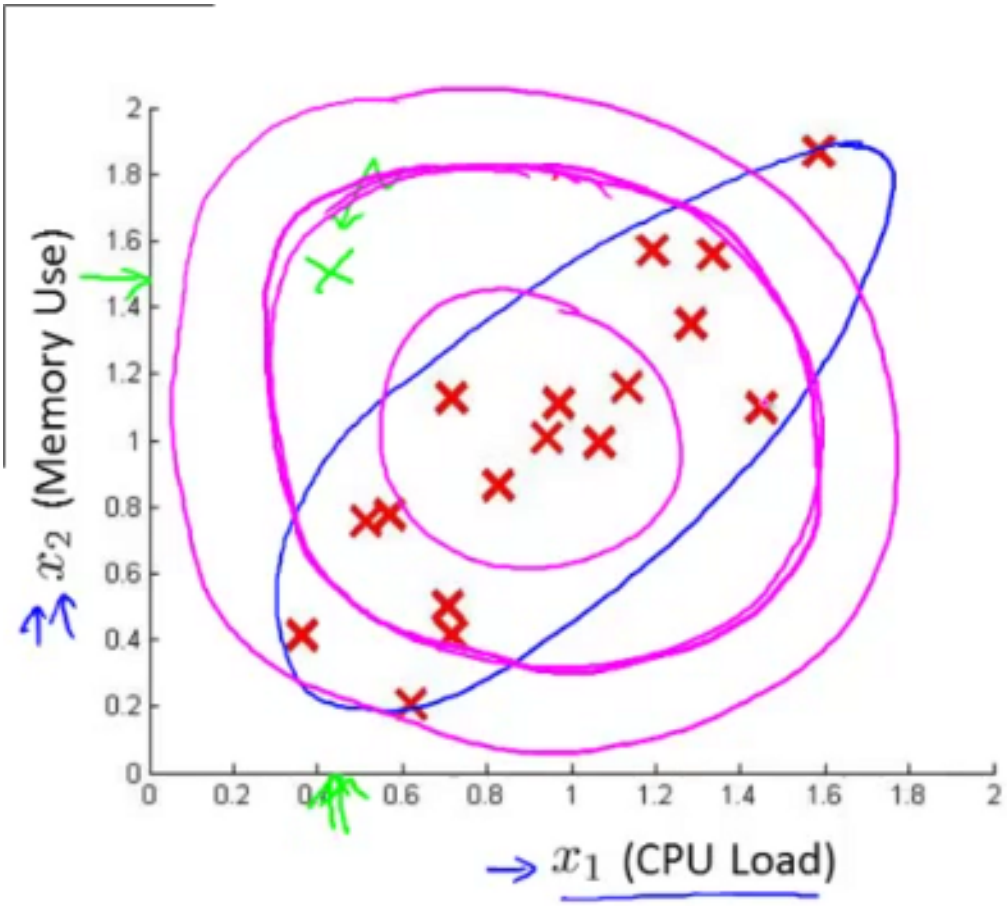
- Also prefer to choose features that might take on unusually large or small values in event of an anomaly. Let us imagine memory, disk acceses, cpu load and network traffic are features being looked at for monitoring computers in a data center. Lets imagine that anomalies are more likely to occur when the computer gets stuck in a long while loop, in which case the CPU load is likely to be quite high and the network traffic quite low. This is a candidate case for identification of yet another feature which is the ratio of CPU load to network traffic. (or perhaps even square of cpu load to network traffic). This will help you spot anomalies which are based on unusual combination of features.

Video 16-7: Multivariate Gaussian distribution

Sometimes the features have some correlation with each other.



You can see the positive seemingly linear correlation between the two features x_1 and x_2. Yet the algorithm above largely assumed these features to be independent. This creates a difficulty as shown in the diagram below.



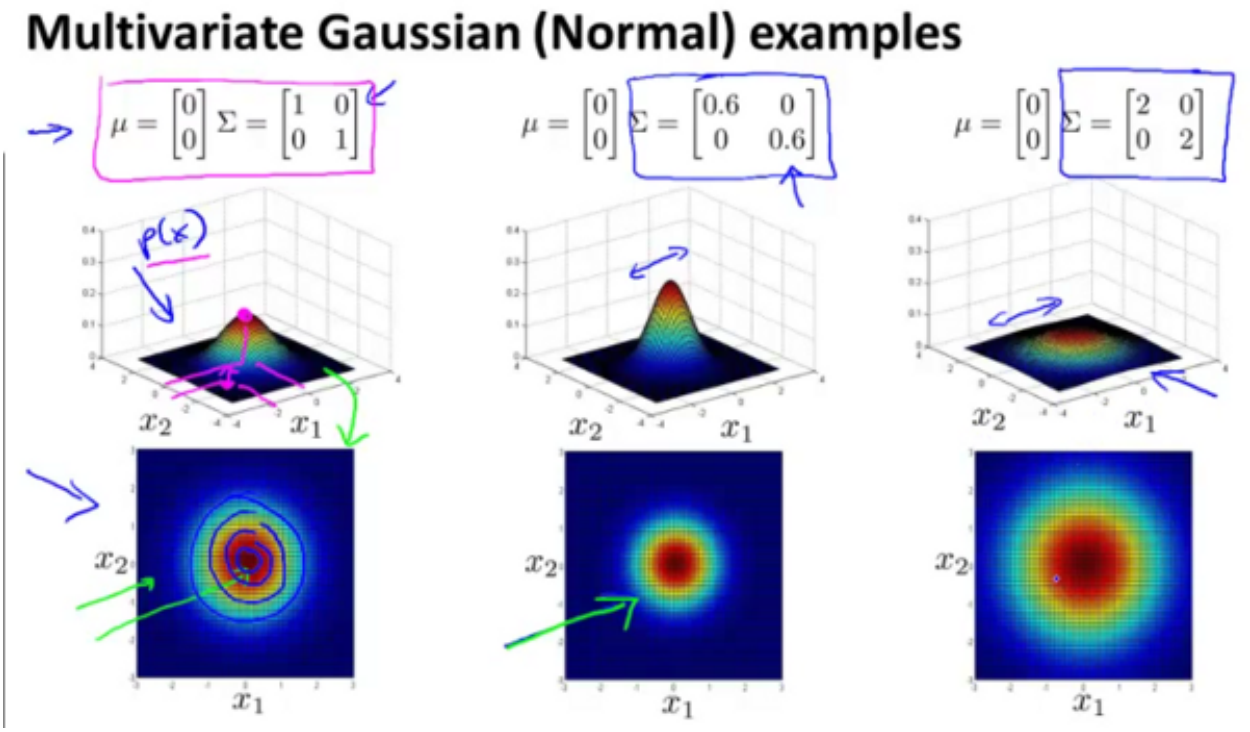
The contours of the probability function computed by independent gaussian variables are similar to the magenta circles drawn above. Yet a casual look can convince us that the contours need to be more along the lines of the contour drawn in the blue line. Thus if you focus on the point marked in green, it should ideally get flagged off as an anomaly, but given the seemingly circular contours, it in this case will not. For this enter - multivariate gaussian distribution.

So for $x \epsilon \mathbb{R}^n$, do not model $p(x_1), p(x_2), ...$ separately assuming them to be independent variables. Model $p(x)$ in one go. The parameters to such computations here are $\mu \epsilon \mathbb{R}^n$ and $\Sigma \epsilon \mathbb{R}^{nxn}$. Note that we have now introduced Σ which is the covariance matrix instead of σ^2 which was just a vector.

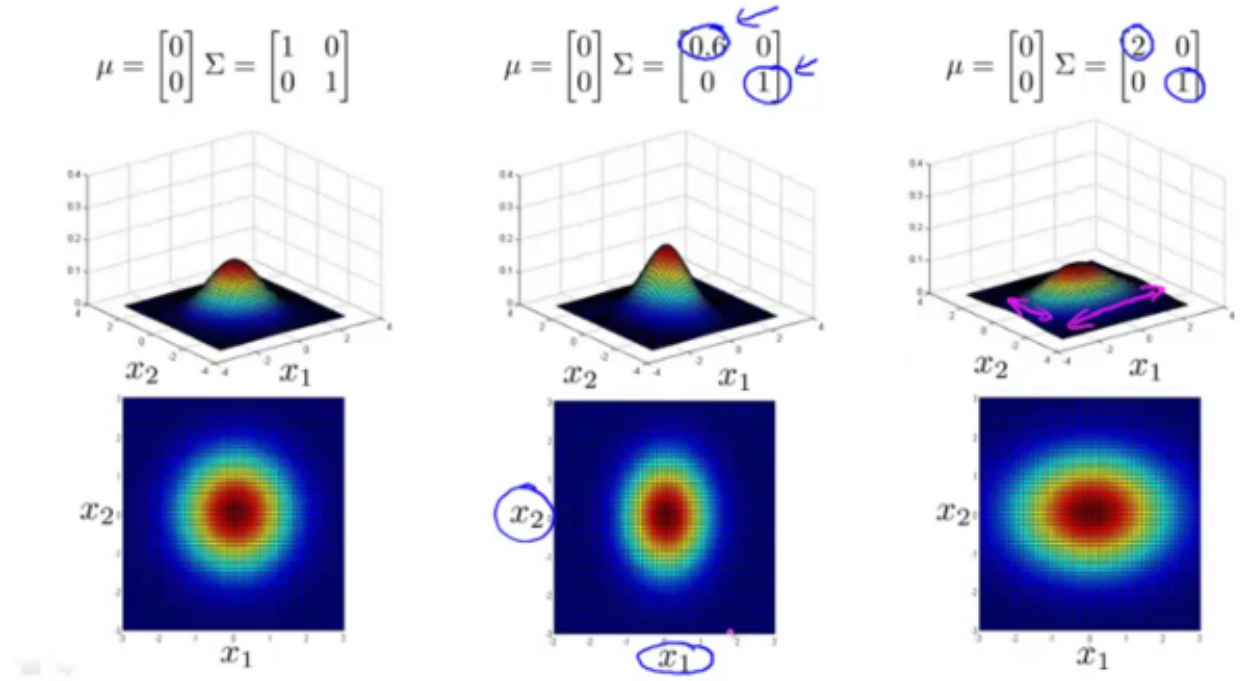
In such a case the probability function will be computed as follows :

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

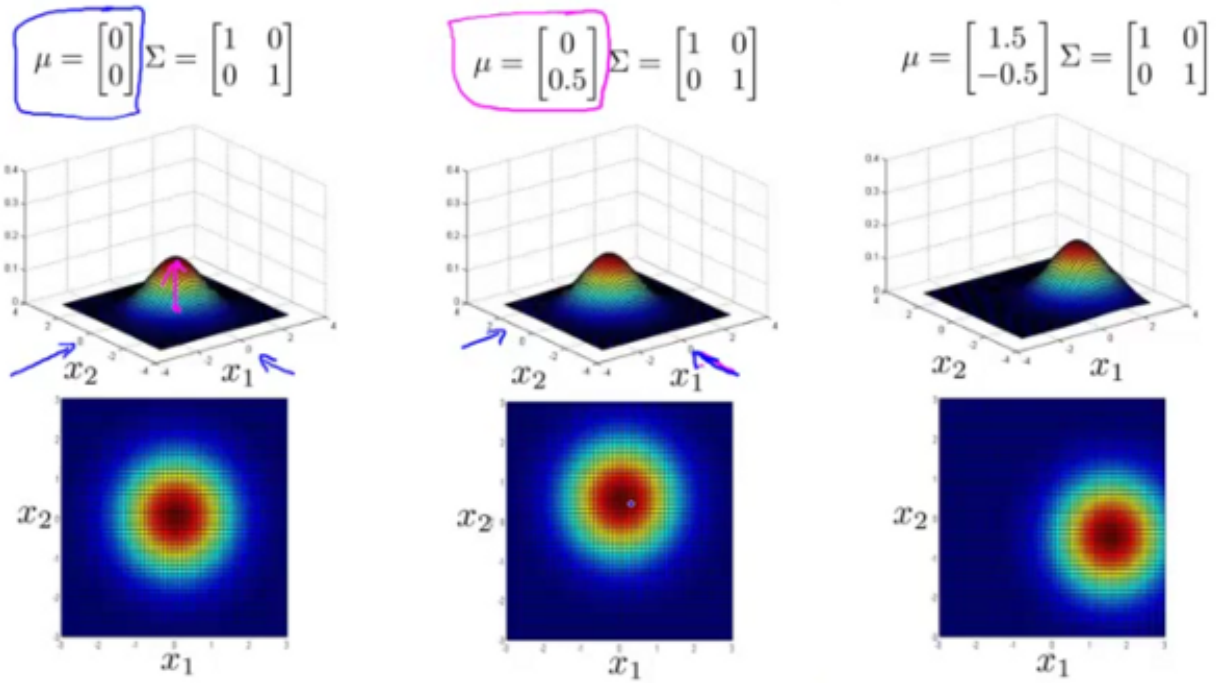
where $|\Sigma|$ is the determinant of Σ



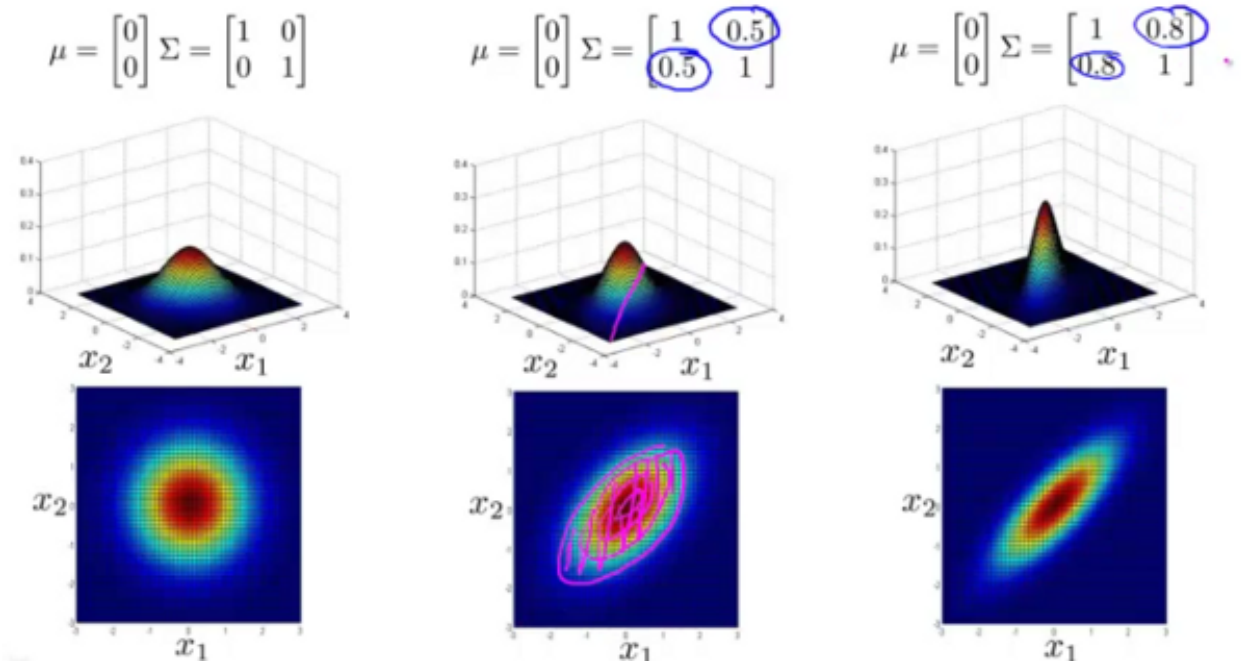
In the figure above it can be seen that by changing the values on the diagonal of the covariance matrix simultaneously, the contours can be made either broader or narrower.



In this figure it can be seen that by independently changing the values on the diagonal of the covariance matrix, the contour profiles can be made to be elliptical along the horizontal and vertical axes.



The above figure shows, that by changing the values of μ, the overall position of the contour profile could be moved.



Finally the image above shows that by changing the values on the covariance matrix not on the diagonal, the contour profile changes to an elliptical shape along arbitrary axes. In fact the right most profile is probably closest to the one that we started with. And setting the non-diagonal elements of a correlation matrix to be non-zero is an admission of the fact that these elements are correlated and not independently variable.

So multivariate gaussian distribution should help us model the situation to better fit the actual behaviour of the two features x_1 and x_2 that we started out with. Thus by using the modified probability function above we can better predict anomalies when the features show some correlation within themselves.

Video 16-8: Anomaly detection using the multivariate gaussian distribution

In computing the probability function using a multivariate gaussian distribution the following could be used.

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

We would need to start by first computing μ and Σ as follows

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

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

Then compute $p(x)$. And flag an anomaly if $p(x) < \epsilon$

Relationship to the original model

It turns out that gaussian distribution is simply a special case of multivariate gaussian distribution with the constraint that all the non-diagonal elements of the covariance matrix should be set to zero.

Howeever gaussian distributed still tends to be used more frequently than its multivariate cousin given that the former is computationally cheaper, and can even deal with situations where m (the training set size) is small or even less than n (the number of features). If $m \leq n$, multivariate gaussian distribution cannot be used since Σ is non-invertible. It is preferred to generally have $m \geq 10n$.

Quite often (as in the case above of the two correlated features), it might still be helpful to model additional features by creating new features eg. $x_1 - x_2$ or $\frac{x_1}{x_2}$ and using gaussian distribution rather then the multivariate gaussian because of the additional computation complexity or if m is not substantially larger than n.

CHAPTER 2

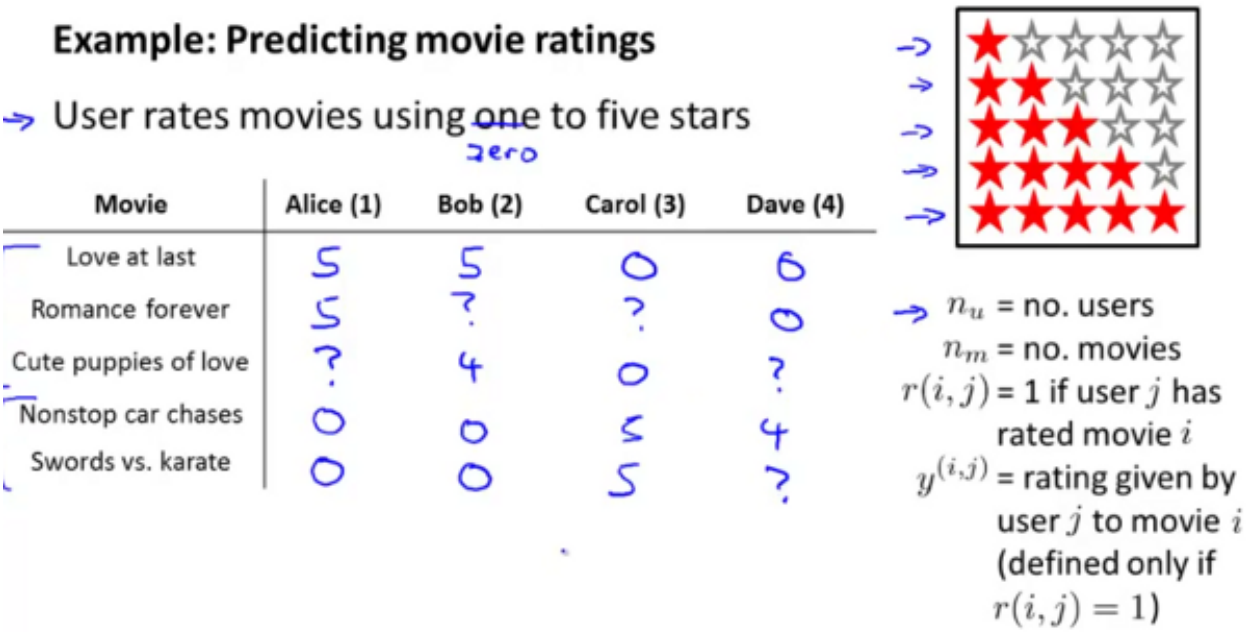
Recommender Systems

Video 17-1: Problem Formulation

- Many websites in silicon valley attempting to build better recommender systems. eg. Amazon recommends books, Netflix recommends movies etc.
- Often these systems are responsible for influencing a substantial portion of their revenues and thus their bottom-lines as well.
- Recommender systems receive relatively little attention within academia but a lot from commercial enterprises.

Predicting movie ratings

Lets say you allow your users to rate movies using zero (just for the sake of this example) to five stars.



Notations

- n_u = number of users (
- n_m = number of movies
- $r(i, j)$ = 1 if user j has rated movie i
- $y^{(i,j)}$ = rating given by user j to movie i

In the above example, you might find Alice & Bob giving high ratings to romatic movies and low ratings to action movies. Carol and Dave rate in exactly the opposite manner.

The problem definition is to look through this data and try to predict what the values of the cells with *?* should be. That in turn will form the basis of recommending movies to the users.

Video 17-2: Content-based recommendation

Here's the data.

Content-based recommender systems

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	?

In a content based recommender system, one will have features described for the content of the films which can be used in recommendation. Lets introduce two features $x_1 \& x_2$ which respectively quantify the extent of romance and action in the movies and provide them appropriate values as follows.

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
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
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

Now one could create a feature vector for the first movie as follows (note an additional feature for the interceptor has been introduced) :

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix} x^{(4)} = \begin{bmatrix} 1 \\ 0.1 \\ 1.0 \end{bmatrix}$$

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

Lets say we want to predict what Alice will think of Cute puppies of love. The parameter vector for the movie is as follows

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

Let us also assume that some as yet unspecified learning algorithm has learned Alice's preferences as the vector :

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

The prediction for Alice's rating for Cute puppies of love shall be

$$(\theta^{(1)})^T x^{(3)} <== evaluates\ to\ 4.95$$

Lets also use yet another variable $m^{(j)}$ to refer to the number of movies rated by user j. This can be treated as a linear regression problem.

The problem can now be narrowed down to that of minimising over $\theta(j)$ for

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

Since $m^{(j)}$ is a constant here, and one attempts to minimise the optimisation objective, the equation above can be simplified to

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

If one wants to simultaneously learn θ for all the users, the optimisation objective J which needs to be minimised, can be further stated as

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

This optimisation function can be then used with gradient descent to incrementally obtain the next value of θ as follows :

$$\begin{aligned} \theta_k^{(j)} &:= \theta_k^{(j)} - \alpha \textstyle\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} && (for\ k = 0) \\ \theta_k^{(j)} &:= \theta_k^{(j)} - \alpha \left(\textstyle\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right) && (for\ k \neq 0) \end{aligned}$$

Video 17-3: Collaborative Filtering

- This algorithm has an interesting property, *feature learning*, ie. the algorithm learns what features to use.
- In this case (as shown in the image below) we no longer have explicit rating of the features. Instead each user has told us how much they like romance movies and how much they like action movies (via their θ values).

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
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	?	?	?

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

In this case we know the values of θ, but do not know the values of the features $x_1 \& x_2$. The question therefore to be answered is what is the value of the vector x_1 (or similarly x_2) which will satisfy the following equations,

$$\begin{aligned}(\theta^{(1)})^T x^{(1)} &\approx 5\\ (\theta^{(2)})^T x^{(1)} &\approx 5\\ (\theta^{(3)})^T x^{(1)} &\approx 0\\ (\theta^{(4)})^T x^{(1)} &\approx 0\end{aligned}$$

Due to the simplicity of this particular problem one can probably reason quickly that the appropriate value would be

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

Stating the problem formally, given $\theta^{(1)}, ..., \theta^{(n_u)}$, we are required to learn the feature vector $x^{(i)}$ such that,

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

Generalising further, across all features, the problem statement now is, given $\theta^{(1)}, ..., \theta^{(n_u)}$, we are required to learn the feature vectors $x^{(i)}, ...x^{(n_m)}$ such that,

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

In content based rating, we saw that given a value of feature vectors x, we could compute θ, while later we saw that given θ, we could compute x. Thus it is possible that we may apply these methods alternately to converge these values from a starting random value. ie

$$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta....$$

This mechanism of alternatively applying the transformations is called collaborative filtering.

Video 16-4: Collaborative Filtering Algorithm

Combining the two optimisation objectives shown earlier, the combined optimisation cost function is

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

And the optimisation exercise shall then be to minimise the cost function as follows

$$\underset{\substack{x^{(1)},...,x^{(n_m)}\\ \theta^{(1)},...\theta^{(n_u)}}}{min}\ J(x^{(1)},...,x^{(n_m)},\theta^{(1)},...,\theta^{(n_u)})$$

Note that now $x \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^n$ (earlier it was n+1. There is no reason to hard code an extra feature since the algorigthm is going to learn the feature by itself. If one does want to introduce the feature corresponding to the interceptor, one could always start by specifying $x^{(1)} = 1$

Using the collaborative filter algorithm

To apply the algorithm we shall

1. Set $x^{(1)},...,x^{(n_m)}$
$\theta^{(1)},...\theta^{(n_u)}$ to small random values. 1. Minimise J by applying gradient descent (or an advanced optimisation algorithm). Thus

$$\begin{array}{l} for\ every\ j = 1,...,n_u \\ for\ every\ i = 1,...,n_m \\ x_k^{(i)} := x_k^{(i)} - \alpha\left(\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda x_k^{(i)}\right) \\ \theta_k^{(j)} := \theta_k^{(j)} - \alpha\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)}\right) \end{array}$$

As earlier the two terms that we multiply with α are the partial derivatives of the cost function. Also note, the special cases such as x_0 and θ_0 are not present.

Once the θ and x matrices are known, we can predict that the rating a user j assigns to a movie i will be $(\theta^{(j)})^T(x^{(i)})$

Video 17-5: Vectorisation and Low Rank Matrix Factorisation

If we model the composite matrices as follows,

$$X = \begin{bmatrix} (x^{(1)}) \\ (x^{(2)}) \\ \vdots \\ (x^{(n_m)}) \end{bmatrix} \quad \Theta = \begin{bmatrix} \theta^{(1)} & \theta^{(2)} & .... & \theta^{(n_u)} \end{bmatrix}$$

Then the prediction matrix can simply be written as $X\Theta$. This resultant matrix is a low rank matrix and hence the name *(did not explain what low rank meant)*.

After having learned *n* features, these could theoretically map to romance, action, comedy etc. However in reality, its pretty difficult to derive a human understandable interpretation of what these features are.

The next question we want to focus on is what movies to recommend to a user. In other words, how to find movies *j* related to movie *i*?

Turns out if we have the corresponding feature vectors for the movies represented as $x^{(i)}$ and $x^{(j)}$, and if we identify that the distance between these feature vectors $\|x^{(i)} - x^{(j)}\|$ is pretty small, then the two movies are likely to be quite similar.

So if you want to find 5 most similar movies to movie *i*, that would be equivalent to finding the 5 movies *j* with the smallest $\|x^{(i)} - x^{(j)}\|$.

Video 17-6: Implementational detail - Mean normalisation

Let us consider a situation where we add a new user *Eve* to the situation we have been discussing so far. Eve has not rated any movies so far.

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	?	?

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

Recall the cost function we used was

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

In this case, since $r(i,j) = 1$ will not be true for any value of $j = 5$ since eve has not rated any movies. So the first term in the function above will have not affect $\theta^{(5)}$ at all. $\theta^{(5)}$ does not appear in the second term. The only place where it will appear is in the third term. as $(\theta^{(j)})^2$. Minimising this will obviously lead to a $\theta^{(5)}$ to have all its values being zero. Hence the all the predictions for Eve given by $(\theta^{(5)})^T x^{(i)}$ will also have the value zero.

To accomodate for this issue, we will perform mean normalisation as follows. Let us start with a matrix *Y* of all the ratings as shown in the image above. We compute a vector μ to be the mean of each row in Y. We finally recompute Y by subtracting μ from itself. This is shown in the image below.

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \quad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

Now we can use this matrix for actually learning $\theta^{(j)}$ and $x^{(i)}$. But when we have to compute the final prediction, we need to add back μ_i as follows

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

As is obvious, $\theta^{(5)}$ is still set to all zeroes, but the predictions for eve will no longer be zero. They will be those specified by μ. That seems rather natural, since if we have no idea about a particular new user being introduced, then the prediction we are going to make is that of the average rating.

Note that the technique could also be used to instead account for situations where one introduced a new movie which has no prior ratings and one wanted to predict the ratings for it for each user. But that is rather questionable in the first place, and in any case it is likely to be more important to account for introduction of new users rather than new movies.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search