Contents

Anomaly Detection (Unsupervised)
and
Recommender Systems

March 26, 2018

Part I

Week 9

0.1 Anomaly Detection: Intuition

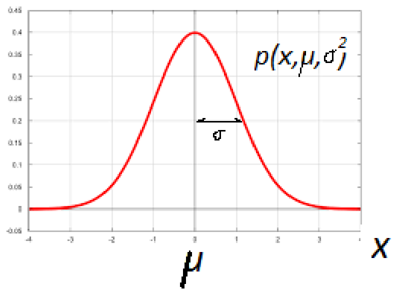
Given a dataset $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, the goal is to determine whether a new example x^{test} is anomalous.

We model $p(x)$ and then evaluate $p(x^{(\text{test})})$: $\begin{cases} \text{if } p(x^{(\text{test})}) \leq \epsilon & \rightarrow \text{flag anomaly} \\ \text{if } p(x^{(\text{test})}) > \epsilon & \rightarrow \text{OK} \end{cases}$

0.2 Review of Gaussian Distribution

If $x(\in \mathbb{R})$ is a distributed Gaussian with mean μ, variance σ^2: i.e $x \sim \aleph(\mu, \sigma^2)$, where '$\sim$' stands for 'distributed as'. The propability of x parametrized by μ and σ^2 ($x \sim \aleph(\mu, \sigma^2)$) is :

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



0.2.1 Parameter estimation

Given a dataset $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}$, for which we suspect each example come from a Gaussian distribution ($x^{(i)} \sim \aleph(\mu, \sigma^2)$:

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \\ \sigma^2 &= \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2 \end{aligned} \tag{2}$$

0.3 Anomaly detection

0.3.1 Density estimation: $p(x)$

Given a training set $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$. We model $p(x)$ from the dataset :

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = p(x_1; \mu_1, \sigma_1^2)p(x_2; \mu_2, \sigma_2^2)p(x_3; \mu_3, \sigma_3^2)....p(x_n; \mu_n, \sigma_n^2) \tag{3}$$

We assume that the probability of first feature, $p(x_1)$, 2nd feature $p(x_2)$, ... are Gaussian probability distributions, i.e : $x_1 \sim \aleph(\mu_1, \sigma_1^2)$, $x_2 \sim \aleph(\mu_2, \sigma_2^2)$, etc...$x_n \sim \aleph(\mu_n, \sigma_n^2)$

- The formulation of $p(x)$ assumes that the features are independent (not corrolated), but in fact the algorithm works for both (independent and not).

0.3.2 Anomaly Detection Algorithm

1. Given an unlabeled training set $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$

2. Fit parameters $\mu_1, \mu_2, ..., \mu_n, \sigma_1^2, \sigma_2^2, ..., \sigma_n^2$, which can be written in a vectorized form:

$$
\begin{aligned}
\mu &= \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \text{where: } \quad \mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \\
\sigma^2 &= \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \vdots \\ \sigma_n^2 \end{bmatrix}, \text{where: } \quad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j^{(i)})^2
\end{aligned}
\tag{4}
$$

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

$$
p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left[-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right] \tag{5}
$$

   If $p(x) < \epsilon$ then x is tagged as an anomaly

0.3.3 Evaluate an anomaly detection system

Example of aircraft engines: we have a dataset with 10,000 good/normal engines ($y = 0$) and 20 flawed engines/anomalous ($y = 1$)

1. Training set: 6,000 good engines ($y = 0$)
   Fit model $p(x)$ on training set $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$

2. Cross validation set: 2,000 good engines of dataset with both $y = 0$ and $y = 1$

3. Test set: 2,000 good engines of dataset with both $y = 0$ and $y = 1$

    - On CV and Test examples, predict $x^{(test)}, y^{(test)}$
    - Use CV to choose parameter ϵ
    - if $p(x)$ is about same when $y = 0$ and $y = 1$, use a new feature.

- Possible evaluation metrics:
    - True Positive/False Positive, True Negative/False Negative
    - Precision/Recall
    - F1-score

*Because the dataset is typically skewed (much less $y = 1$ count than $y = 0$), predicting $y = 0$ would have a very high accuracy.

0.3.4 Anomaly detection vs. supervised Learning

Anomaly Detection	Supervised Learning
Very small number of positive examples ($y = 1$): 10-50, and Large Number of Negative examples	Large Number of positive and Negative examples
Many different types of anomalies (hard for algorithm to learn from positive examples, what the anomalies look like; future anomalies may look different that the anomalous examples seen so far)	Enough positive examples for the algorithm to get a sense of what positive example is : future positive examples likely to be similar to ones in training set
Application: Fraud Detection, Manufacturing (aircraft engines, etc...), Monitoring Machines in a data center...	email spam classification, weather prediction (sunny/rainy, etc...), cancer classification

0.3.5 Features selection

- It is important to check whether the data is Gaussian.
- if the data is not Gaussian, transform it to make it look Gaussian. Some examples of feature Transformation:

$$\begin{aligned} x_2 &\leftarrow log(x_2) \\ x_2 &\leftarrow log(x_2 + c) \\ x_2 &\leftarrow \sqrt{x_2} \\ x_2 &\leftarrow x_2^{1/3} \end{aligned} \tag{6}$$

0.4 Multivariate Gaussian (Normal) distribution

0.5 Density Estimation

Instead of modeling $p_x = p(x_1)p(x_2)...p(x_n)$ where $p(x_j)$ are calculated separately, Multivariate Gaussian models $p(x)$ all in one-go, using the parameters $\mu \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n\times n}$ (covariance matrix):

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{7}$$

where $|\Sigma|$ is the determinant of the covariance matrix.
* Octave uses det(Sigma) for determinant of Sigma

0.5.1 Algorithm

1. Given a training set $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$ where $x \in \mathbb{R}^n$ comes from a multivariate Gaussian distribution.
2. Model:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{8}$$

where

$$
\begin{aligned}
\mu &= \begin{bmatrix} \mu_1 \\ \mu_2 \\ .. \\ .. \\ \mu_n \end{bmatrix} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \\
\Sigma &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T
\end{aligned} \tag{9}
$$

3. Given a new example x, compute :

$$
p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{10}
$$

- if $p(x) < \epsilon$, x is an anomaly

0.5.2 Original Model versus Multi-Variate

- The original model corresponds to the multivariate model when the off-diaginal elements of Σ are zero, i.e:

$$
\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0.. & 0 \\ 0 & \sigma_2^2 & 0.. & 0 \\ 0 & 0 & .... & 0 \\ 0 & 0 & .... & 0 \\ 0 & 0 & 0.. & \sigma_n^2 \end{bmatrix} \tag{11}
$$

Original Model	**Multivariate Gaussian**
Manually create features to capture anomalies ($x_3 = x_1/x_2$)	Automatically captures correlations between features
Computationally cheap	Computationally more expensive
scales better to large n (10K, 100K)	Must have $m > n$ or else Σ is non-invertible (and Σ^{-1} cannot be calculated)
OK even if training set size m is small	duplicate features ($x_1 = x_2$) or redundancy $x_3 = x_1 + x_4$ result in Σ not invertible

0.6 Recommender Systems

0.6.1 Problem formulation: Predicting movie rating (0 to 5)

movie	**Alice**	**Bob**	**Carol**	**Dave**
'Love at Last'	5	5	0	0
'Romance for ever'	5	?	?	0
'Cute puppies of Love'	?	4	0	?
'Non-Stop Car chase'	0	0	5	4
'Swords versus Karate'	0	0	5	?

0.6.2 How to predict the missing ratings

We will for now consider that we have 2 features x_1 and x_2 that gives the degree to which a movie is a 'romance' or 'action' movie.

movie	Alice [$\theta^{(1)}$]	Bob [$\theta^{(2)}$]	Carol [$\theta^{(3)}$]	Dave [$\theta^{(4)}$]	x_1 romance	x_2 action
'Love at Last' [$x^{(1)}$]	5	5	0	0	0.9	0
'Romance for ever' [$x^{(2)}$]	5	?	?	0	1.0	0.01
'Cute puppies of Love' [$x^{(3)}$]	?	4	0	?	0.99	0
'Non-Stop Car chase' [$x^{(4)}$]	0	0	5	4	0.1	1.0
'Swords versus Karate' [$x^{(5)}$]	0	0	5	?	0	0.9

Each movie can therefore be represented by a feature vector (including the bias term $x_0 = 1$). For example, for movie 1: $x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$.

In order to predict the rating **for each user**, we can learn a parameter $\theta^{(j)} \in \mathbb{R}^3$, and then predict for user j, the rating of movie i with $(\theta^{(j)})^T x^{(i)}$.

- For example, for movie $x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$, let's say that the user 1 has the following parameters: $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$. The predicted rating of user $j = 1$ for movie $x^{(3)}$ is: $(\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$.

- **Parameters Definition**:
  - n_u: number of users
  - n_m number of movies
  - n number of movie features
  - $r(i,j) = 1$ is user j has rated movie i
  - $y^{(i,j)}$ (0 to 5) rating given by user j to movie i (defined only if $r(i,j) = 1$
  - $\theta^{(j)}$: parameter vector for user j
  - $x^{(i)}$ feature vector for movie i
    Note that the predicted rating of movie i by user j is : $(\theta^{(j)})^T x^{(i)}$ where $\theta^{(j)} \in \mathbb{R}^{n+1}$

- Optimization objective to learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} (\theta_k^{(j)})^2 \tag{12}$$

  For simplification, $m^{(j)}$ can be taken out of the equation , hence we can write:

- Optimization objective to learn $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$ for all users:

$$\min_{\theta^{(1)},\dots,\theta^{(n_u)}} \underbrace{\frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2}_{J(\theta^{(1)},\theta^{(2)},\dots,\theta^{(n_u)})} \tag{13}$$

- The gradient descent update:

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

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left[\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} + \lambda \theta_k^{(j)}\right]}_{\frac{\partial J(\theta^{(1)},\dots,\theta^{(n_u)})}{\partial \theta_k^{(j)}}} \text{ for } k \neq 0 \tag{14}$$

0.6.3 Collaborative filtering - partI

Previously, we assume set values for the features $x^{(1)}, x^{(2)}$, but this data might not be available. One way to get the feature values is by asking the user if they like 'action' or 'romantic movies. W might get user vector such as : $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ (multiplier '5' is associated with x_1), $\theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, $\theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$, $\theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$.

From $\theta^{(j)}$, we can infer values of $x^{(1)}$ and $x^{(2)}$ by solving $\theta^T x$. For movie 1, we will get for each user:

$$\begin{aligned}
&(\theta^{(1)})^T x^{(1)} \approx 5 \rightarrow x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0 \end{bmatrix} (\text{ with } \ x_0 = 1) \\
&(\theta^{(2)})^T x^{(1)} \approx 5 \\
&(\theta^{(3)})^T x^{(1)} \approx 0
\end{aligned} \tag{15}$$

- **Optimization algorithm:** learning features $x^{(i)}$ for movie # i (given $(\theta^{(1)}), ..., (\theta^{(n_u)})$):

$$\min_{x^{(i)}} \left[\frac{1}{2} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2 \right] \tag{16}$$

  i.e sum over all the users j for which we have a rating for movie i.

- Learn all features for all movies (predict value of how user j rate movie i):
  Given $(\theta^{(1)}, ..., \theta^{(n_u)}))$ to learn $x^{(1)}, ..., x^{n_m}$:

$$\min_{x^{(1)},...,x^{(n_m)}} \left[\frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 \right] \tag{17}$$

0.6.4 Collaborative filtering - partII

- Given the features $x^{(1)}, ..., x^{(n_m)}$, we can estimate the users vectors $\theta^{(1)}, ..., \theta^{(n_u)}$:

$$\min_{\theta^{(1)},...,\theta^{(n_u)}} \left[\frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2 \right] \tag{18}$$

- Given the users parameters $\theta^{(1)}, ..., \theta^{(n_u)}$, we can estimate the features $x^{(1)}, ..., x^{(n_m)}$:

$$\min_{x^{(1)},...,x^{(n_m)}} \left[\frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)}) - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 \right] \tag{19}$$

- For the first minimization, it is the sum over all user j and all movies i rated by user j, whereas the 2nd optimization objective is a sum over all users j which have rated movies i. The 2 optimization objectives can be combine into one, by summing over the pair (i, j): we need to minimize $(x^{(1)}, ..., x^{(n_m)})$ and $(\theta^{(1)}, ..., \theta^{(n_u)})$ simultaneously:

$$\begin{aligned}
J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}) = &\frac{1}{2} \sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 \\
&+ \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2
\end{aligned} \tag{20}$$

- Minimization:

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)},\\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}) \tag{21}$$

  Note that here we do not use $x_0 = 1$, so features $x \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^n$. This is because we are learning all features, so no need to hardcode one of the features to be equal to 1.

The algorithm:

1. Initialize $x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}$ to small random values (symmetry breaking).
2. Minimize $J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)})$ using gradient descent for every $j = 1, .., n_u, i = 1, ..., n_m$
3. The gradient descent update (reminder: no θ_0, and x_0)

$$x_k^{(i)} := \theta_k^{(i)} - \alpha \underbrace{\left[\sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right]}_{\frac{\partial J(....)}{\partial x_k^{(i)}}} \tag{22}$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left[\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right] \tag{23}$$

4. for a user with parameters θ and a movie with (learned) features x, one can predict a star rating for movie i: $(\theta^{(j)})^T x^{(i)}$

0.6.5 Low rank matrix factorization

movie	Alice $[\theta^{(1)}]$	Bob $[\theta^{(2)}]$	Carol $[\theta^{(3)}]$	Dave $[\theta^{(4)}]$
'Love at Last' $[x^{(1)}]$	5	5	0	0
'Romance for ever' $[x^{(2)}]$	5	?	?	0
'Cute puppies of Love' $[x^{(3)}]$	?	4	0	?
'Non-Stop Car chase' $[x^{(4)}]$	0	0	5	4
'Swords versus Karate' $[x^{(5)}]$	0	0	5	?

- The table of movie rating can be vectorized (i-row = movie, and j-col=user), where $y^{(i,j)}$ is an element of Y:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & ? \end{bmatrix} \tag{24}$$

- The table of movie can be vectorized:

$$X = \begin{bmatrix} --(x^{(1)})^T -- \\ --(x^{(2)})^T -- \\ .... \\ .... \\ --(x^{(n_m)})^T -- \end{bmatrix} \tag{25}$$

- Users list can be vectorized:

$$\Theta = \begin{bmatrix} --(\theta^{(1)})^T-- \\ --(\theta^{(2)})^T-- \\ .... \\ .... \\ --(\theta^{(n_u)})^T-- \end{bmatrix} \tag{26}$$

- Predicting ratings:

$$X\Theta^T = \begin{bmatrix} (\theta^{(1)})^T x^{(1)} & (\theta^{(2)})^T x^{(1)} & ... & .... & .... & (\theta^{(n_u)})^T x^{(1)} \\ (\theta^{(1)})^T x^{(2)} & (\theta^{(2)})^T x^{(2)} & ... & .... & .... & (\theta^{(n_u)})^T x^{(2)} \\ ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... \\ (\theta^{(1)})^T x^{(n_m)} & (\theta^{(2)})^T x^{(n_m)} & ... & .... & .... & (\theta^{(n_u)})^T x^{(n_m)} \end{bmatrix} \tag{27}$$

  where $(\theta^{(1)})^T x^{(1)}$ is the predicted rating from user 1 for movie 1, and $(\theta^{(1)})^T x^{(2)}$ is the predicted rating from user 2 for movie 1.

- **Finding relate movies**:
  For each movie i we learn feature vectors $x^{(i)} \in \mathbb{R}^n$.
  How to find movies j related to movie i?

$$\text{small}||x^{(i)} - x^{(j)}|| \Rightarrow \text{movie} \;\; j \;\; \text{and} \;\; i \;\; \text{are} \;\; \text{similar} \tag{28}$$

0.6.6 Mean Normalization

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & ? & ? \end{bmatrix} \tag{29}$$

The last user $j = 5$ has no rated movies. If one calculate predicted rating from $(\theta^{(5)})^T x^{(i)}$, one would predict all raying to be zero.
To prevent this type of 'incident', all the Y data are mean normalized, i.e

$$Y := Y - \mu \tag{30}$$

where μ is the average rating of movie i over all users: $\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.525 \\ 1.25 \end{bmatrix}$ For user j, the predicted rating on movie i is:

$$(\theta^{(j)})^T x^{(i)} + \mu_i \tag{31}$$