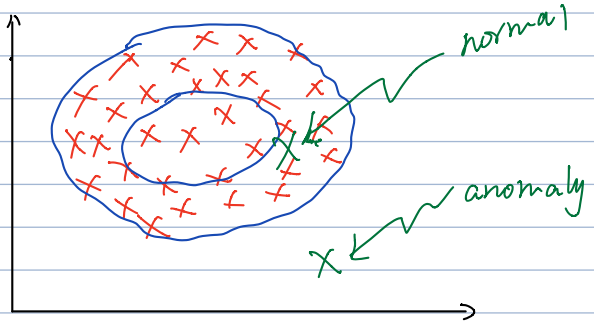
Anomaly detection

Aircraft engine features:
- x_1 = heat generated
- x_2 = vibration intensity

Dataset: $\{x^{(1)}, x^{(2)}, x^{(3)} \cdots, x^{(m)}\}$.

New engine: x_{test}.



we build a model for the probabilities of x, where x is the feature of aircraft.

$$\begin{cases} p(x) < \varepsilon \Rightarrow \text{anomaly} \\ p(x) \geq \varepsilon \Rightarrow \text{normal} \end{cases}$$

Gaussian Distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\delta} e^{-\frac{(x-\mu)^2}{2\delta^2}}$$

我们利用已有数据来预测总体中的 μ 和 δ^2 的计算方法，即

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

注：在机器学习中对于方差的计算我们通常使用 $\frac{1}{m}$ 而非统计学中的 $\frac{1}{m-1}$。

Anomaly Algorithm:

Step 1: Choose features x_i that you think might be indicative of anomalous example. given dataset $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$.

Step 2: $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \qquad \delta_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

Step 3: Assuming $x \in R^n$ (features), $x^{(i)} \sim N(\mu_i, \delta_i^2)$, then we'd like to model

$$\begin{aligned} p(x) &= p(x_1, \mu_1, \delta_1^2)\, p(x_2, \mu_2, \delta_2^2)\, p(x_3, \mu_3, \delta_3^2) \cdots p(x_n, \mu_n, \delta_n^2) \\ &= \prod_{j=1}^{n} p(x_j, \mu_j, \delta_j^2) \end{aligned}$$

317

In the density estimation function, we set the height of Gaussian distribution is the estimation of $p(x)$. we set a small value ε, for example $\varepsilon = 0.02$. then all points/examples which $p(x) \leq \varepsilon$ are anomaly. otherwise are normal.

Developing & evaluating an anomaly detection system

Anomaly detection is an unsupervised learning problem, which means that we cannot know whether the data are anomalous or not through y.

Assume we have some labeled data, of anomalous and not anomalous examples. ($y=0$ if normal, $y=1$ if anomalous)

We choose some normal data to build traing set: $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$
the rest of normal data and all non-anomalous data are in cross-validation and test sets.

Algorithm evaluation:

1. Fit model on training set $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$.

2. On a cross validation / test example x, predict

$$y = \begin{cases} 1, & \text{if } p(x) \le \varepsilon \quad \text{(anomaly)} \\ 0, & \text{if } p(x) > \varepsilon \quad \text{(normal)} \end{cases}$$

the metrics:
- True positive, false positive, false negative, true negative.
- precision / Recall
- F_1-score.

the classification is not a good metric in this problem, cause for the skew example, classifying it as $y=0$ will have a very high classification accuracy.

we can use the cross-validation to choose the best ε.

Anomalous detection V.S. Supervised learning.

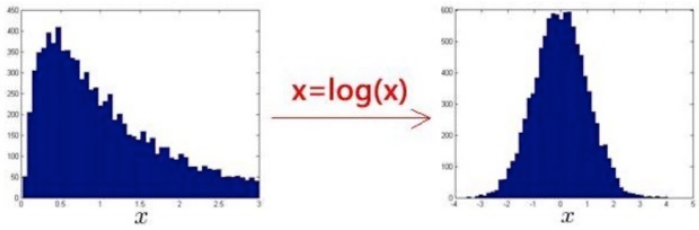
Anomaly detection	Supervised learning.
- Very small number of positive examples ($y=1$) very large number of negative examples ($y=0$)	- Large number of positive & negative examples.
- Many different types of anomalies. Hard for any algorithms to learn from positive examples what the anomaly looks like.	- Enough positive example to get a sense of what positive examples are like. future positive examples may look similar to ones in traing set.
- future anomalies may look like nothing like any of the anomalous examples weive seen so far.	

Choosing what features to use.

Non-gaussian features.

In anomaly detection, we assume features satisfying Gaussian distribution. If the data are non-gaussian, we should convert them to gaussian at first.

In Python, we usually use np.log1p( ) (log(x+1)) to avoid negative values.



Error analysis for anomaly detection

Want p(x) large for normal examples x.
p(x) small for anomaly examples x.

Most common error.

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

In order to solve this problem, we should find a new feature or some unusual features to let algorithm frog data correctly.

Recommend System

User rates movies using zero to five stars.

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	?

4.5

Note: n_u = Number of users
n_m = # of 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 when $y^{(i,j)} = 1$).
$y^{(i,j)} = 0, \cdots, 5$.

What the recommend system does is predict those undefined rates based on defined rates, then recommend the similar movies to uses.

Content Based Recommendations.

problem formulation

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

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

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

$m^{(j)}$: number of movies rated by user j.

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

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

因为对每个用户来说，$m^{(j)}$ 是个常数，所以在这里可将其去掉

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

针对所有用户：

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

Gradient Descent:

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

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

Collaborative Filtering (协同过滤):

If we only know the preference of each user (parameter), we can learn the feature vector of each movie.

Given $\theta^{(1)}, \cdots, \theta^{(n_u)}$, to learn $x^{(i)}$: 已知用户的参数，学习电影的特征.

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

If we don't know neither user parameters nor feature vectors, collaborative filtering may help a lot, the user parameters and feature vectors can learn from each other.

$$\theta \to x \to \theta \to x \to \theta \to x \cdots$$

we don't use $x_0 = 1$

note: $x \in \mathbb{R}^n$ rather than $x \in \mathbb{R}^{n+1}$.

Minimize $x^{(1)}, \cdots x^{(n_m)}, \theta^{(1)}, \cdots, \theta^{(n_u)}$ simultaneously:

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

Algorithm:

1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random value.
2. Minize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent, since there is no $x_0 = 1$, all θ are needed to be regularized.

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

given $\theta^{(1)}, \dots, \theta^{(n_u)}$, learn $x^{(1)} \dots x^{(n_m)}$

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

given $x^{(1)} \dots x^{(n_m)}$, learn $\theta^{(1)} \dots \theta^{(n_u)}$.

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating $(\theta^{(j)})^T (x^{(i)})$. ← user j rates movie i.

Vectorization: low rank matrix factorization.

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

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

predicted ratings. ← low rank matrix.

Finding related movies/products.

① 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 = ...

② How to find movie j related to movie i?

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

保证两部电影的特征向量之间的距离 $x^{(i)}, x^{(j)}$ 很小.

Mean Normalization:

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	?	?

if we want to add a new user into this system, and she doesn't rate any movie.

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

$$\min_{\substack{x^{(1)},\dots,x^{(n_m)} \\ \theta^{(1)},\dots,\theta^{(n_u)}}} = \frac{1}{2}\sum_{(i,j):r(i,j)=1} \underbrace{\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2}_{1/0} + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\underbrace{(\theta_k^{(i)})^2}_{1/0} + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

$$= \frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$$

because she doesn't rate any movie. $r(i,j)=0$

So, $\theta^{(5)} = \begin{bmatrix}0\\0\end{bmatrix}$, since there is only regularization term.

then $(\theta^{(5)})^T \cdot x^{(i)} = 0$. ← this doesn't make sense.

Mean normalization

① We get the Y matrix. $Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$ $(n_m \times n_u)$

② Calculate the mean rate of each movie. $\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$

③ each rate subtracts by mean value. $Y = Y - \mu = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$

④ Use the new Y matrix to learn $\theta^{(j)}$ & $x^{(i)}$ through collaborative Filtering Algorithm.

⑤ For user j on movie i predict: $(\theta^{(j)})^T x^{(i)} + \mu_i$.

Notice: the mean normalization doesn't need to divide by the range (max-min) because all the movie ratings are already comparable (eg. 0~5 stars), so they are already on similar scales.