Week 9 : Anomaly Detectⁿ

- We'll try to model $P(x)$ for the given data.
- If $P(x) \le \varepsilon$, the given input x will be considered an anomaly & $p(x) > \varepsilon$ will correspond to normal behaviour.

- **Gaussian (Normal) distributⁿ:**

Suppose $x \in \mathbb{R}$ & $X \sim \mathcal{N}(\mu, \sigma^2)$ then,

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

$\mu \rightarrow$ mean
$\sigma \rightarrow$ std dev.

For a dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$,

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

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

Density estimatⁿ:

Suppose $x \in \mathbb{R}^n$.

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

(Assuming features $\{x_1, x_2, \ldots, x_n\}$ are indep.)

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

- **Evaluating an ~~Anomaly~~ detection algorithm:**

(i) We assume that we have labelled data i.e. $y=0$ if normal, $y=1$ if anomalous.

Splitting into training set, CV set & Test set

Suppose there are 10 000 good engines, } labelled.
20 flawed engines }

Training set: 6000 good engines

CV set: 2000 good engines, 10 flawed engines

Test set: 2000 good engines, 10 flawed engines.

Possible evaluat^n metrics:

(i) True positive, False pos, false neg, true neg

(ii) Precision / Recall

(iii) F1 score

* Classific^n accuracy isn't a good metric because of the skewed nature of the data.

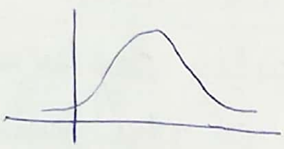
Note: We can also choose ε using the CV set.

Anomaly detect^n	Supervised learning
- Small no. of +ve examples (y=1) Large no. of -ve examples (y=0)	- Large no. of both +ve & -ve examples.
- Hard for any algo to learn how the future anomalies look like given the small no. of +ve examples.	- Enough no. of +ve examples for an algo to get a sense of what +ve examples are like.
eg - Fraud detect^n, ~~Ma~~ can shift here as well → Manufacturing, Monitoring machines in a data center.	eg - Spam classific^n, weather predic^n, cancer classific^n.

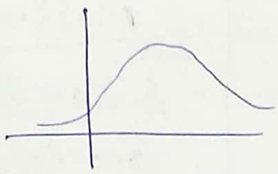
Choosing the features to use

If feature x is a bit skewed, 
then we can transform x to $\log x$ or $x^{1/2}$ or $x^{1/3}$
to get a more bell shaped distributⁿ.

$\log x$



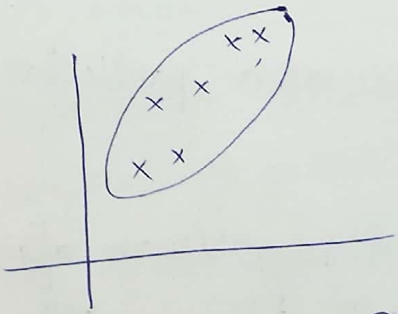
$x^{1/2}$



Error analysis for anomaly detectⁿ

- Most common problem: $p(x)$ is comparable for both normal & anomalous examples.
- For ex: if there's a training example very close to a bunch of normal training examples but is actually abnormal, we should take a look at that ex closely & come up with features specific to it

Multivariate Gaussian distⁿ



Sometimes when we model many gaussian dist^ns, the model fails to get the right boundary for the training set, so we use multivariate gaussian distⁿ.

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

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

$$\Sigma = \begin{bmatrix} a & c \\ d & b \end{bmatrix}$$

a, b – σ of x_1, x_2

c, d – Covariance of x_1, x_2 if it's more then we get high prob. across $y = x$.

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

* Individual gaussian distribut^n's are special cases of multi-variate gaussian distribut's where the ellipses are axis aligned.

Original model	Multivariate model
- Have to manually create features to capture anomalies	- Automatically capture anomalies in the features.
- Computationally cheaper	- computationally expensive
- Should be used if m is small	- Should be used if $m \geq n$ or preferably $m \geq 10n$.

Recommender Systems

- Suppose there are m movies, u users who rate the movies.
  $r(i,j) = 1$ if user j has rated movie i.
  $y(i,j)$ = rating value
- We want to find the ratings of movies as a particular user would have rated
- We basically train different linear regression algos for diff. users. & come up with $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(u)}$.

Rating for a movie i by user j: $(\theta^{(j)})^T(x^{(i)})$ | $x^{(i)} \rightarrow$ feature vector of movie i.

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

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

with change:

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

Over all $\theta^{(j)}$'s i.e. $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$

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

Gradient Descent:

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

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

Repeat this for all values of $j : 1, 2, \dots, n_u$.

Collaborative filtering

- Each user tells us their θ vector & we also know the ratings given by these users to some of the movies.
- We need to find an algo that comes up with the necessary features.

Algo:

Given $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$, to learn $x^{(i)}$.

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

To learn $x^{(1)}, \dots, x^{(n_m)}$

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

Procedure:

1. Guess θ
2. Learn features x
3. Get better estimate for θ.
4. . . . . .

$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \dots$

Collaborative filtering algorithm

$$J(x^{(1)}, x^{(2)}, \dots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)})$$

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

$$+ \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

* Combining both optimising θ & optimising x.
* minimise over $x^{(1)}, x^{(2)}, \dots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$
* $x_0 = 1$ isn't used bcoz the algo learns the features for itself.

Collaborative filtering algo:

1. Initialise $x^{(1)}, x^{(2)}, \ldots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n_u)}$ to small random values.

2. Gradient descent for $j = 1, 2, \ldots, n_u$. $i = 1, 2, \ldots, n_m$.

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

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

3. Predict: star rating $= (\theta^{(j)})^T x^{(i)}$.

Vectorizⁿ of algo: (Low rank matrix factorizⁿ)

$Y = n_m \times n_u$.

$X = n_m \times n$ $\qquad \theta = n_u \times n$

$\text{Pred} = X \theta^T$.

Finding related movies:

For each product i, we learn a feature vec $x^{(i)} \in \mathbb{R}^n$.
Movie j is similar to movie i, if

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

Mean normalizⁿ

- ~~Suppose~~ For each movie i, $x^{(i)} \rightarrow x^{(i)} - \mu^{(i)}$

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