Machine Learning - Week 9 - Anomaly Detection

T1 - Density Estimation

L1 - Problem Motivation

Anomaly Detection example:

Aircraft engine features:

x_1 = heat generated
x_2 = vibration intensity
⋮

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

New engine: x_{test}

Is this engine like the others?

Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

Is x_{test} anomalous?

Will build a model that outputs $p(x)$.

$p(x_{test}) < \varepsilon \rightarrow$ Flag anomaly

$p(x_{test}) \geq \varepsilon \rightarrow$ OK

Uses

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

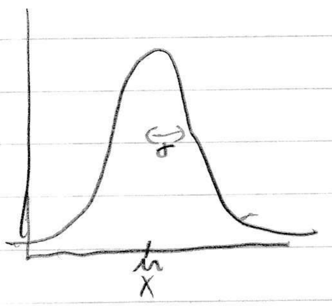
- Manufacturing
- Monitoring computers in a data center.
  - $x^{(i)}$ = features of machine i
  - x_1 = memory use
  - x_2 = # of disk accesses/second
  - x_3 = CPU load
  - x_4 = CPU load/network traffic

L2 - Gaussian Distribution (Normal Distribution)

Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ, variance σ^2

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

"distributed as"



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

Parameter estimation.

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$, $x^{(i)} \sim \mathcal{N}(\mu, \sigma^2)$

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

L3 - Algorithm

Density Estimation

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

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

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

$\vdots$

$x_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$

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

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

Anomaly detection algorithm:

1) Choose features x_i that ~~might take~~ might be indicative of anomalous examples.
2) Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

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

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

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

Anomaly if $p(x) < \varepsilon$

T2 - Building an Anomaly Detection System

L1 - 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: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples/not anomalous).

Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Aircraft engines motivation example

10 000 good (normal) engines

20 flawed engines (anomalous)

Training set: 6000 good engines ($y=0$)

CV: 2000 good engines ($y=0$), 10 anomalous ($y=1$)

Test 2000 good engines ($y=0$), 10 anomalous ($y=1$)

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

On a cross validation/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

L2 - Anomaly Detection vs Supervised Learning

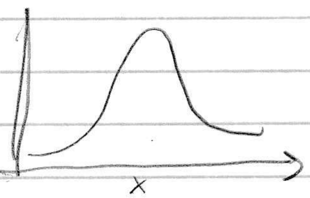
Anomaly Detection vs. **Supervised Learning**

Anomaly Detection
- Very small number of positive examples ($y=1$). (0-20 is common
- Large number of negative ($y=0$) examples.
- Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.

Supervised Learning
- Large number of positive and negative examples
- 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.

L3 - Choosing what features to use

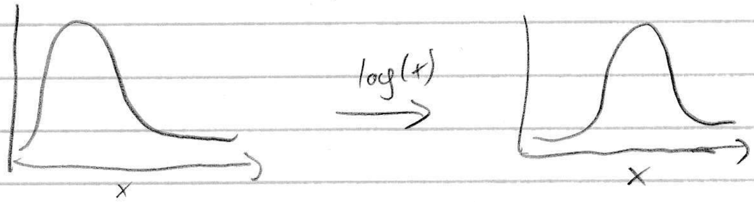
Non-gaussian features



$P(x_i ; \mu_i, \sigma_i^2)$

Plot histogram to see

Ex.



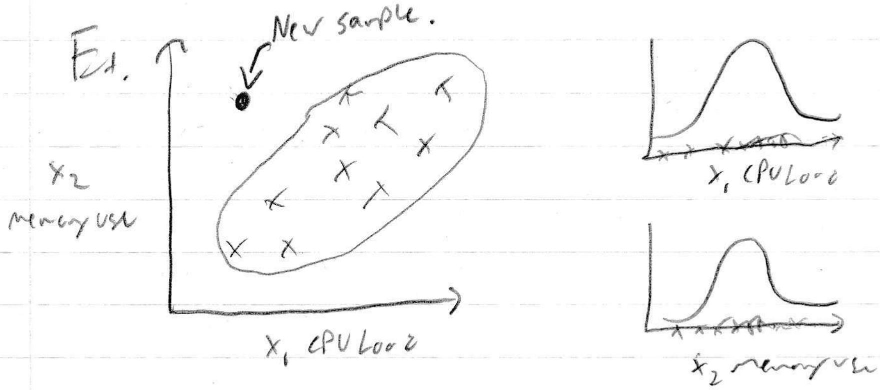
Error analysis for anomaly detection
Want $p(x)$ large for normal examples x.
$p(x)$ small for anomalous examples x.

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

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

T3- Multivariate Gaussian Distribution

L1- Multivariate Gaussian Distribution



Will not detect new sample as anomaly, since it is somewhat close to x_1 and somewhat close to x_2 (mutually explusive). But when looked at as a whole, it is not grouped with the other normal examples.

$x \in \mathbb{R}^n$. Don't model $p(x_1)$, $p(x_2)$, ..., etc separately.
Model $p(x)$ all in one go.
Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

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

$|\Sigma|$ = determinant of Σ

L2- Anomaly Detection using the multivariate Gaussian distribution

Watch Video

FIVE STAR

Machine Learning - Week 9 - Recommender Systems

T1 - Predicting Movie Ratings
L1 - Problem Formulation

Ex. Predicting Movie Ratings
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	?

n_u = no. users

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 $r(i,j)=1$)

Goal of the recommender system is to fill in the question marks based on the existing data

L2 - Content-based recommendations

Consider dataset from above.

Add columns:

x_1 (romance)	x_2 (action)
0.9	0
1.0	0.01
0.99	0
0.1	1.0
0	0.9

So we now have feature vectors.

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix} \leftarrow x_0 = 1$$

$\vdots$

$\Rightarrow$

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

Ex. $x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$ Say $\Theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$

∴ Predict Alice's rating for Cute puppies of love to be

$$(\Theta^{(1)})^T x^{(3)} = 5 \cdot 0.99 = 4.95$$

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

To learn $\Theta^{(j)}$: (parameter for user 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 learn $\Theta^{(1)}, \ldots, \Theta^{(n_u)}$:

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

Gradient Descent Update

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

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

T2 - Collaborative Filtering

L1 - Collaborative Filtering

Problem Motivation:

Movies	A(1) $\Theta^{(1)}$	B(2) $\Theta^{(2)}$	C(3) $\Theta^{(3)}$	D(4) $\Theta^{(4)}$	X_1 (Romance)	X_2 (action)
$x^{(1)}$ Romance	5	5	0	0	?	?
$x^{(2)}$ Romance	5	?	?	0	?	?
$x^{(3)}$ Romance	?	4	0	?	?	?
$x^{(4)}$ Action	0	0	5	4	?	?
$x^{(5)}$ Action	0	0	5	?	?	?

If we get $\Theta^{(j)}$ from our users

i.e. $\Theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ $\Theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ $\Theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$ $\Theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$

$\therefore$ we can find $X^{(1)}$ to fit

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

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

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

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

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

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

L2 - Collaborative Filtering Algorithm

Recall the algorithms from previous lectures.
You can minimize $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously.
See slides for formula

T3 - Low Rank Matrix Factorization

L1 - Vectorization

Collaborative Filtering

Movie	A(1)	B(2)	C(3)	D(4)
M_1	5	5	0	0
M_2	5	?	?	0
M_3	?	4	0	?
M_4	0	0	5	4
M_5	0	0	5	?

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

Predicted Ratings:

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

vectorized

Low rank matrix factorization

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

Finding related movies.

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

x_1 = romance, x_2 = action, x_3 = comedy, ... etc

How to find movies j related to movie i?

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

L2 - Implementation detail: Mean normalization

Watch lecture / look at slides