Density Estimation
Anomaly Detection

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

Is x_{test} anomalous?

Model $p(x)$

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

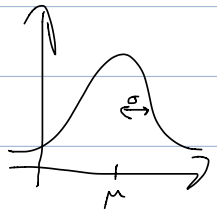
$p(x_{test}) \geq \varepsilon \rightarrow$ OK

Gaussian Distribution (Normal)

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

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

($\sim$ = 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

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

Anomaly Detection Algorithm

+ training set: $\{x^{(1)}, \dots, x^{(m)}\}$

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

$p(x)$

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

$x_2 \sim N(\mu_2, \sigma_2^2)$

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

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

1. Choose features x_i that you think might be indicative of anomalous examples

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

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

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

$p(x_j; \mu_j, \sigma_j^2)$ $\quad \mu_1, \dots, \mu_n \quad \mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$

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

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

Anomaly if $p(x) < \varepsilon$

Developing and Evaluating an ADS

Importance of real number evaluation

Some labeled data, of anomalous & non-anomalous examples

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)

CV & test DO have anomalous examples

$y = 0 \sim$ anom

$y = 1 \sim$ non-anom

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

On 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

Anomaly Detection vs. Supervised Learning

Very small # of positive examples ($y=1$) (0-20 is common) AD

Large # of negative ($y=0$) examples

Many different "types" of anomalies. Hard for any algo to learn from ($+$) examples what anomalies look like

Large # of positive & negative examples

Enough ($+$) for algo to get a sense of what ($+$) ex are like, future will be similar

Choosing What Features to Use

Error analysis for AD

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

Multivariate Gaussian Distribution

$x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2)$... etc. seperately

Model $p(x)$ all in one go.

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

$p(x; \mu, \Sigma)$

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

↑ determinant

A1) Using Multivariate Gaussian Distribution

parameters μ, Σ

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

Parameter fitting:

Given training set $\{x^{(1)}, \ldots, x^{(m)}\}$

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

1. Fit model $p(x)$ by setting

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

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

2. Given a new example x, compute

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

Flag an anomaly if $p(x) < \varepsilon$

Relationship to original model

Original model: $p(x) = p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$

Corresponds to multivariate Gaussian

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

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

Original Use case

- used more often

Manually create features to capture anomalies where X_1, X_2 take unusual combinations of values

Computationally cheaper (alternatively, scales to larger n)

OK even if m (training set size) is small

Multivariate case

Automatically captures correlations between features

Computationally more expensive

$X_1 = X_2$

$X_3 = X_4$

Must have $m > n$, or else Σ is non-invertible. In practice $m \geq 10n$

Recommender Systems

Problem Formulation

n_u = # users

n_m = # 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 $r(i,j)=1$)

Content Based Recommendations

$\theta^{(j)} \in \mathbb{R}^{n+1}$

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

$r(i,j) = 1$ 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 $\theta^{(j)}$:

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

Optimized)

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

To learn $\theta^{(1)}, \dots, \theta^{(n_u)}$

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

Gradient descent update:

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

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\underbrace{\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \theta_k^{(j)}}_{\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})} \right) \quad (\text{for } k \neq 0)$$

Collaborative Filtering (Basic)

Don't know any features

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

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

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(1)}, \dots, x^{(n_m)}$:

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

Can guess $\theta \to x \xrightarrow{\text{new}} \theta \xrightarrow{\text{new}} x \cdots$ Converge reasonably

---

Collaborative Filtering Algorithm

solve simultaneously

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

$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) \leftarrow \text{minimize simultaneously}$$

~~$x_0 = 1$~~

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

$$\theta \in \mathbb{R}^n$$

1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values
2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization)
3. For a user parameters θ and a movie with (learned) features x, predict a star rating $x^{(i)}$

Vectorization: Low Rank Matrix Factorization

Matrix Y of all ratings

Predicted ratings:

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

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

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

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, x_4 = ...

How to find movies j related to movie i?

$\|x^{(i)} - x^{(j)}\|$ small $\rightarrow$ movies j and i are "similar"

Implementational Detail: Mean Normalization

User has not rated any movies

$(\theta^{(j)})^T x^{(i)} = 0$ for all movies

Mean Normalization

Matrix Y $\quad \mu$ = average of rows Y $\rightarrow$ $Y - \mu = Y_{new}$ (normalize)

learn $\theta^{(j)}, x^{(i)}$

For user j, on movie i predict:

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