Anomaly Detection and Recommender Systems

Sidharth Baskaran

June 2021

Anomaly detection problem

- Decide if a example is an anomaly to dataset
- Build model for $p(x)$ which is probability of feature being anomalous
  - $p(x) < \epsilon \implies$ anomaly and $p(x) \geq \epsilon$ is fine
  - Ex: fraud detection where $x^{(i)}$ is features of user i

Gaussian/Normal Distribution

- If $x \in \mathbb{R}$, then if x is a distributed Gaussian with mean μ and variance σ^2
  - $x\tilde{}\mathcal{R}(\mu, \sigma^2)$
  - Formula is $p(x; \mu, \sigma^2)$
  - Density formula is $p = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- Distribution is centered at μ and σ determines width
- Area under curve is always 1, so $\sigma \propto \text{height}^{-1}$
- Parameter estimation
  - $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$
  - $\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)^2$
  - Tend to use $\frac{1}{m}$ instead of $\frac{1}{m-1}$, both work equally well

Anomaly detection algorithm

- Model probability of each feature vector as $p(x) = p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) \dots p(x_n; \mu_n, \sigma_n^2) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$
  - Assumes features are independent

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

- Is anomaly if $p(x) < \epsilon$

Developing and Evaluating an Anomaly Detection System

- Importance of real-number evaluation
  - Assume labeled data exists, anomalous and non-anomalous
  - Training set $x^{(i)}, \dots, x^{(m)} \rightarrow$ assume normal examples that are not anomalous
  - Define a cross validation and test set
- Fit model $p(x)$ on training set $\{x^{(i)}, \dots, x^{(m)}\}$
- On cross validation/test set example x, predict

$$y = \begin{cases} 1 \text{ if } p(x) < \epsilon \text{ (anomaly)} \\ 0 \text{ if } p(x) \geq \epsilon \text{ (normal)} \end{cases}$$

- Evaluation metrics
  - True positive, false positive, false negative, true negative
  - Precision/recall
  - F_1 score (if skewed)
  - Classification accuracy is not a good metric due to skewedness
- Can also use the CV set to choose ϵ

Anomaly Detection vs. Supervised Learning

- Anomaly Detection
  - Very small number of positive examples
  - Large number of negative examples
  - Many different types of anomalies $\rightarrow$ cannot discern what anomalies look like from small positive examples
- Supervised learning
  - Large number of positive and negative examples
  - Enough positive examples for algorithm to discern a positive example
    - Later positive examples are similar to those in training set

Choosing Features

- Plot a histogram of data to check normality
  - If skewed, can apply transform $x_i \rightarrow \log(x_i + c)$
    - Can also use polynomial transformations
  - Constant can be varied to make data more Gaussian
- Error analysis for anomalies
  - $p(x)$ large for normal examples and small for anomalous examples
  - Problem $\rightarrow$ $p(x)$ comparable for normal and anomalous
  - Can add features which are magnified $\rightarrow$ easier to capture anomalies

Multivariate Gaussian Distribution

- Allows for plotting multiple features and their probabilities
  - Peak is μ and height is $p(x)$
- Let $x \in \mathbb{R}^n$, and the multivariate function outputs $p(x)$
- Parameters are $\mu \in \mathbb{R}^n, \Sigma \in \mathbb{R}^{n \times n}$, the covariance matrix
  - $p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$
  - Where $|\Sigma|$ is determinant of Σ
- Covariance matrix Σ and μ
  - Decreasing diagonal values in Σ narrows distribution and increasing it widens (less height since $\sum p(x^{(i)}) = 1$)
  - Changing individual values of diagonals makes the contour plot ellipsoid affecting distributions of $x^{(i)}$ individually
  - Changing off-diagonal entries allow for "rotating" the contour plot in direction of sign of the entries (i.e $+ve = CW$ and $-ve = CCW$)
  - Changing μ shifts the peak

Anomaly Detection using Multivariate Gaussian Distribution

- Recall parameter fitting
  - $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
  - $\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$
- Given a new example x, compute $p(x)$, and flag an anomaly if $p(x) < \epsilon$
- Relationship to original model

$$p(x) = \prod_{i=1}^{n} = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \text{ if } \Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n^2 \end{bmatrix}$$

- Original model
  - Manually create features from those that have unusual combinations of values to capture anomalies where
  - Computationally cheaper
  - Fine if $m \leq n$
- Multivariate Gaussian
  - Automatically captures correlations between features
  - Computationally expensive (e.g. inverse matrix of Σ must be calculated)
  - Must have $m > n$, or Σ is singular
    * Singularity implies linearly dependent features

Predicting Movie Ratings

- Notation
  - $r(i,j) \in \{0,1\}$ represents whether or not user j has rated movie i
  - $y^{(i,j)}$ is the user's rating if $r(i,j) = 1$
  - n_u is number of users and n_m is number of movies
  - $\theta^{(j)}$ is parameter vector for user j and $x^{(i)}$ is feature vector for movie i
- For each user j, learn a parameter vector $\theta^{(j)} \in \mathbb{R}^3$, and predict user j as rating movie j with $(\theta^{(j)})^T x^{(i)}$ stars
- $i : r(i,j) = 1$ means all values of i such that user has given a rating

Learn $\theta^{(j)}$

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

Learn $\theta^{(1)}, \ldots, \theta^{(n_u)}$

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

Gradient descent update routine

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

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

Collaborative Filtering and Algorithm

- Given $x^{(1)}, \dots, x^{(n_m)}$ and movie ratings, can estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$

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

- Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, can estimate $x^{(1)}, \dots, x^{(n_m)}$

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

- Repeating these steps allows for learning features and parameters simultaneously

Efficient algorithm

- Combined cost function summation iterates over all $(i,j) : r(i,j) = 1$

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

Objective is $\min_{x^{(1)},\dots,x^{(n_m)},\theta^{(1)},\dots,\theta^{(n_u)}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$

- Minimize with respect to $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously
- Convention gets rid of $x_0 = 1$, therefore no θ_0, so $x, \theta \in \mathbb{R}^n$

Algorithm

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 advanced optimization algorithm

For example, for every $j = 1, \dots, n_u, i = 1, \dots, n_m$

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

3. For user with parameters θ and movie with learned features x, predict star rating of $\theta^T x$

Low Rank Matrix Factorization

- Let matrix Y be all given ratings including ?s
- Let predicted rating matrix be such that element (i,j) is $(\theta^{(j)})^T x^{(i)}$

- Define $X = \begin{bmatrix} -(x^{(1)})^T- \\ \vdots \\ -(x^{(n_m)})^T- \end{bmatrix}$ as the feature matrix
- Define $\Theta = \begin{bmatrix} -(\theta^{(1)})^T- \\ \vdots \\ -(\theta^{(n_u)})^T- \end{bmatrix}$
- To calculate prediction matrix, use $X\Theta^T$
- Algorithm called low rank matrix factorization
- Finding related movies from example
  - For each product i, we learn feature vector $x^{(i)} \in \mathbb{R}^n$
  - To find movies j related to movie i, want to find the smallest $||x^{(i)} - x^{(j)}||$

Mean normalization

- Average each row of Y to generate $\mu \in \mathbb{R}^{n_m}$
- Subtract each entry in μ from each value in corresponding row of Y, then use this to learn $\theta^{(i)}, x^{(i)}$
- When predicting for user j movie $i \rightarrow (\theta^{(J)})^T x^{(i)} + \mu_i$