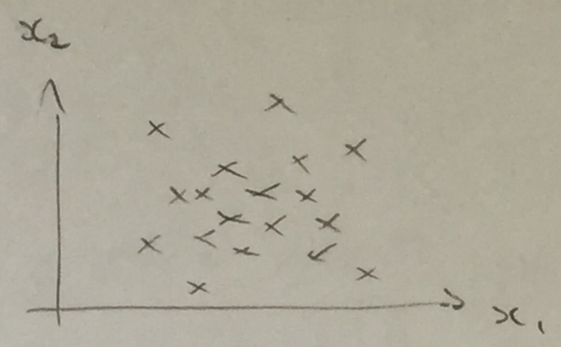
ML COURSERA - WEEK 9

LEO ROBINOVITCH

Anomaly Detection:

eg. Aircraft mnfg: x_1 = heat gen.
x_2 = vib. intensity



↳ how to determine what constitutes an anomaly above?

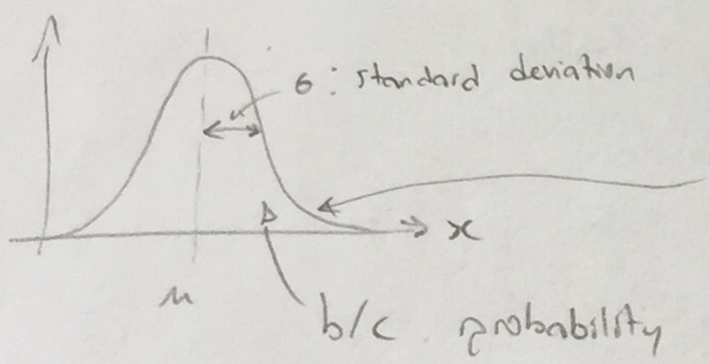
⇒ can build model $p(x)$ → if $p(x_{test}) < \varepsilon$ → flag anomaly!

Applications: fraud detection, manufacturing

Gaussian Distribution Review:

$x \in \mathbb{R}$, distributed gaussian w/ mean μ & variance σ^2 → $x \sim \mathcal{N}(\mu, \sigma^2)$

(distributed as; normal)



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

b/c probability dist., area = 1

⇒ given $x^{(i)} \sim \mathcal{N}(\mu, \sigma^2)$, can guess μ & σ^2 by a sample dataset:

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

← max likelihood estimations

↳ sometimes $(m-1)$

Anomaly Detection Algorithm

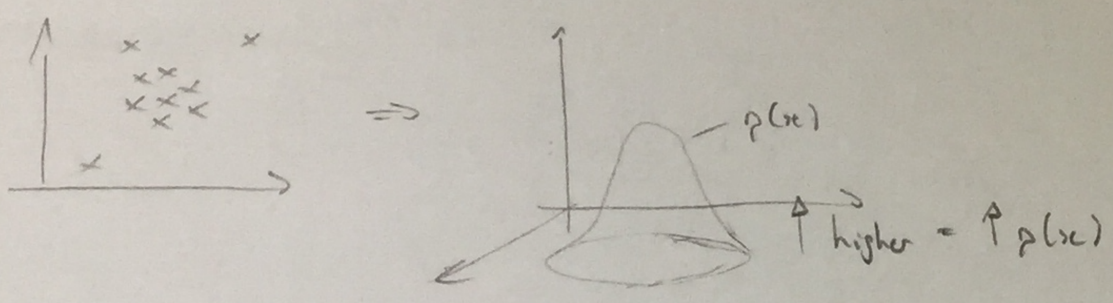
$$p(\underline{x}) = p(x_1) \cdot p(x_2) \cdot p(x_3) \cdots p(x_n) \quad \rightarrow p(x_i; \mu_i, \sigma_i^2)$$

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

① Choose features x_i indicative of anomalous examples

② Find μ_j, σ_j^2 for $j = 1 - n$ features

③ $p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$ → if $p(x) < \varepsilon$, anomaly!



⇒ making decisions in evaluating a learning algorithm always better if we have some metric

↳ eg we have labeled data (anomaly, non-anomaly)

↳ extract • training set (assume all non-anomalies)
- cross val set
- test set

eg 10 000 normal, 20 anomalous → train: 6000 normal
cv: 2000 normal, 10 anom
test: 2000 normal, 10 anom

① Fit model $p(x)$ on train

② Predict $y = 1$ (anomaly), $y = 0$ (normal) on C.V.

③ Use evaluation metric → · True Pos, etc.
· Precision / Recall
· F_1-Score
→ Use CV to choose ε

④ Evaluate on test

⇒ Why not just use a supervised learning algorithm?

↳ usually only have small number of $y = 1$ (anomaly) examples relative to $y = 0$!

↳ future anomalies may look nothing like current anomalies

↳ use supervised if we have lots of anomaly examples (eg spam)

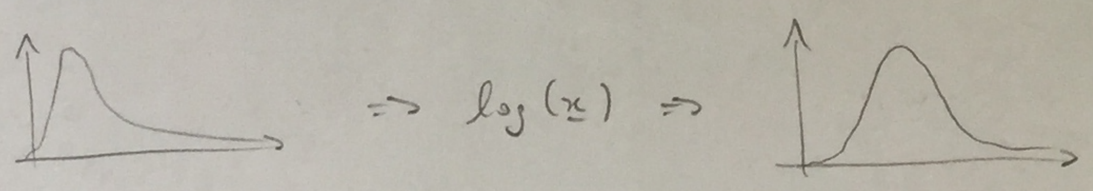
ANOMALY:
- fraud
- mnfg
- machines in data centres

SUPERVISED LEARNING:
- spam
- weather
- cancer prediction

Choosing the Right Features for Anomaly Detection:

⇒ plot histogram of data to make sure $\underline{x}$ is gaussian

↳ transform data to look gaussian (eg. $\log(\underline{x})$) if not gaussian!

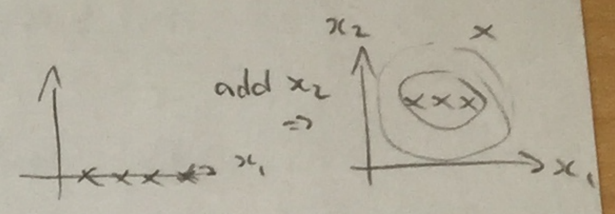


other transforms:
- $\log(\underline{x}+c)$
- $\underline{x}^c$

⇒ Want $p(x)$ large for normals, $p(x)$ small for anomalies:

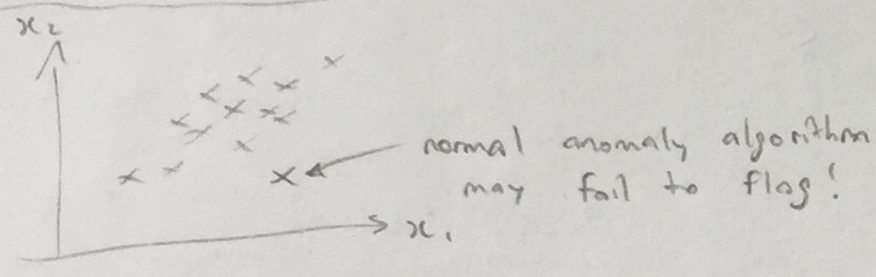
↳ most common problem: $p(x)_{norm} \approx p(x)_{anom}$

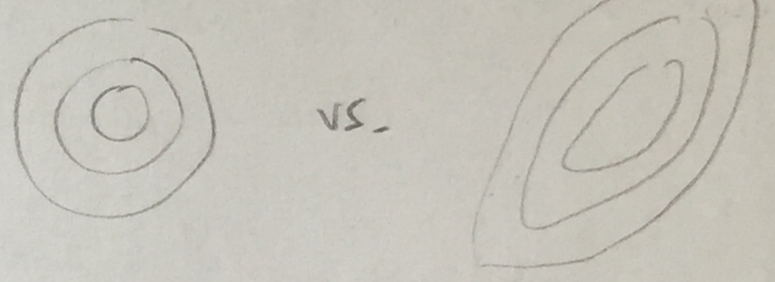
↳ can identify new/important features that solve this:



↳ choose features that might be large/small upon anomaly

Multivariate Gaussian Distributions:





⇒ $x \in \mathbb{R}^n$ → model $p(\underline{x})$ all at once rather than $p(x_1)$, $p(x_2)$, etc.

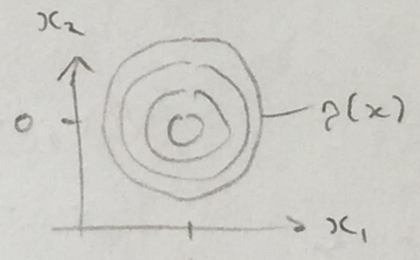
$$p(\underline{x}) = \frac{1}{(2\pi)^{n/2}|\underline{\underline{\Sigma}}|^{1/2}} \exp\left(-\frac{1}{2}(\underline{x}-\underline{\mu})^T \underline{\underline{\Sigma}}^{-1}(\underline{x}-\underline{\mu})\right)$$

$(p(\underline{x};\underline{\mu},\underline{\underline{\Sigma}}))$

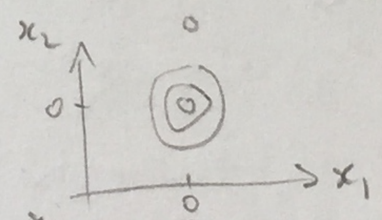
covariance matrix

where $|\underline{\underline{\Sigma}}|$ = determinant of $\underline{\underline{\Sigma}} \in \mathbb{R}^{n \times n}$

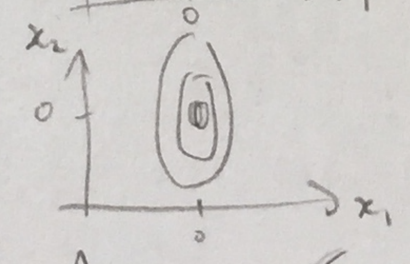
eg. $\underline{\mu} = \begin{bmatrix}0\\0\end{bmatrix}$, $\underline{\underline{\Sigma}} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \underline{\underline{I}}$ →



$\underline{\mu} = \begin{bmatrix}0\\0\end{bmatrix}$, $\underline{\underline{\Sigma}} = \begin{bmatrix}0.6 & 0\\0 & 0.6\end{bmatrix}$ →

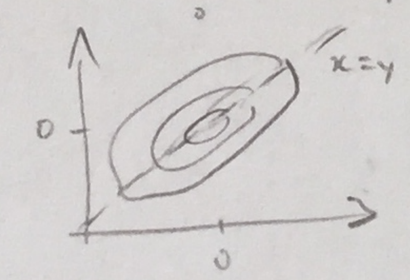


$\underline{\mu} = \begin{bmatrix}0\\0\end{bmatrix}$, $\underline{\underline{\Sigma}} = \begin{bmatrix}0.6 & 0\\0 & 1\end{bmatrix}$ →



⇒ x_2 has greater variance than x_1

$\underline{\mu} = \begin{bmatrix}0\\0\end{bmatrix}$, $\underline{\underline{\Sigma}} = \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}$ →



Note: $\underline{\mu}$ determines peak

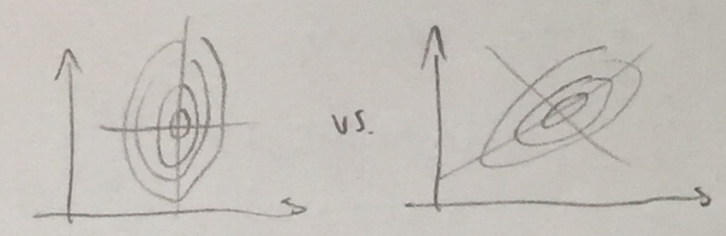
given $\underline{x} = [x^{(1)}, x^{(2)}, \ldots x^{(m)}]$, $\underline{x} \in \mathbb{R}^n$:

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

=> Algorithm: get μ & Σ → $p(x_{new}) < \varepsilon$ → anomaly!

=> In original model $p(x) = \underbrace{p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \cdots}$

each of these is multivariate special case:



AXIS ALIGNED! → $\Sigma = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$

↳ multivariate automatically captures correlations between features!

↳ computationally ↑ n → Σ^{-1} where $\Sigma \in \mathbb{R}^{n \times n}$ (large n ↑) $\Sigma \sim \frac{n^2}{2}$

↳ also needs $m > n$, otherwise Σ non-invertible (want $m \geq 10n$)

Recommender Systems:

eg. Predicting movie ratings: 0-5 stars

4 users (1-4), 5 movies

rating ↓ n_m	user → n_u			
	5	5	0	0
	5	?	?	0
	?	4	0	?
	0	0	5	4
	0	0	5	?

=> want to fill in ?'s in order to recommend movies!

n_u: # users

n_m: # movies

$r(i,j) = 1$ if user j rated movie i

$y^{(i,j)}$ = rating by user j for movie i (if $r(i,j) = 1$)

=> Have 2 features, x_1 (romance) & x_2 (action) for movies

↳ movie 1: $x^{(1)} = \begin{bmatrix} x_0 & x_1 & x_2 \\ 1 & 0.9 & 0 \end{bmatrix}$ → feature vectors for movie i

$x^{(2)} = [1 \quad 1.0 \quad 0.01]$ etc.

=> for each user, learn $\theta^{(j)} \in \mathbb{R}^{n+1}$ ($=3$ here) → rating of user j = $(\theta^{(j)})^T x^{(i)}$

↳ parameter vector for user j

Formulation:

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

↳ to learn all $\theta^{(j)}$'s:

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

Gradient Descent for This:

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

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

Collaborative Filtering ; feature learning (self-learns features)

↳ where do x_1 (romance) & x_2 (action) come from?

↳ instead of having x_1, x_2 vals, survey users → $\theta^{(j)}$ given by user

↳ given ratings of users & self-reported $\theta^{(j)}$'s, can classify movie type!

⇒ given $\theta^{(1)} \dots \theta^{(n_u)}$, learn $x_1 \dots x^{(n_m)}$:

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

So given x's (ratings) → predict θ's (user preferences)

& given θ's (user preferences) → predict x's (ratings)

↳ can guess θ → get x → update θ → get x etc.

⇒ more efficient algorithm (simultaneous x & θ):

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

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

j: users (n_u) $\quad r(i,j)=1$ → rated / exists

i: movies (n_m)

k: # genres (n) → (features in θ)

① Initialize $x^{(n_m)}, \theta^{(n_u)}$'s to small random vals

② Minimize $J(x, \theta)$ → gradient descent: $x_k^{(i)} = x_k^{(i)} - \alpha \left[\sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right]$

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

NOTE: no $x_0 = 1$, $x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^n$

③ For user w/ parameters $\underline{\theta}$ & movie w/ learned features (genres) $\underline{x}$,
predict rating of $\theta^T x \rightarrow (\theta^{(j)})^T x^{(i)}$

Vectorization of this Algorithm:

movie ↓ i, users → j, $n_u = 4$, $n_m = 5$

$$\Rightarrow \underline{\underline{Y}} = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & - & - & 0 \\ - & 4 & 0 & - \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix} \in \mathbb{R}^{n_m \times n_u}$$

$y^{(i,j)}$ entries (actual)

$$\Rightarrow \text{predicted rankings: } \begin{matrix} \\ \downarrow \\ i \end{matrix} \begin{bmatrix} (\theta^{(1)})^T x^{(1)} & \cdots & (\theta^{(n_u)})^T x^{(1)} \\ \vdots & \ddots & \vdots \\ (\theta^{(1)})^T x^{(n_m)} & \cdots & (\theta^{(n_u)})^T x^{(n_m)} \end{bmatrix} \rightarrow \text{what user } j \text{ predicts on movie } i$$

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

↳ pred. rankings $= \underline{\underline{X}}\,\underline{\underline{\Theta}}^T$ (Low-Rank Matrix Factorization)

⇒ for each product (movie), we learn feature vector $\underline{x}^{(i)} \in \mathbb{R}^n$

↳ eg. x_1 = romance, x_2 = action, etc.

↳ how to find movies j related to movie i?

↳ want $\|x^{(i)} - x^{(j)}\|$ small → then movies i & j similar!

Mean Normalization:

⇒ If no ratings for user → $\theta^{(user)} = \underline{0}$, $(\theta^{(user)})^T x^{(movie)} = 0$

↳ to fix this, get average for each movie, then subtract this from all rated vals → avg. rating of each movie now 0

⇒ Now, prediction $= (\theta^{(j)})^T x^{(i)} + \mu_i$ → for user w/ no ratings, give average rating μ_i instead of 0

(If movie w/ no ratings, don't recommend it anyway)