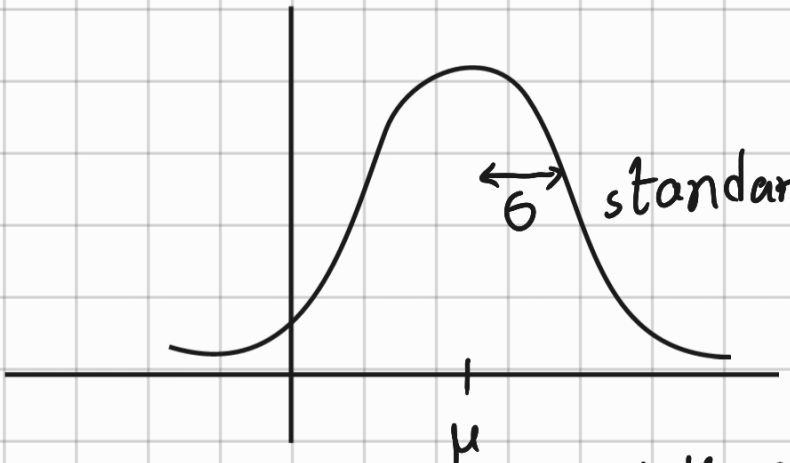
Anomaly detection:

→ Fraud detection
→ Manufacturing
→ Monitoring computers in a data center

We'll build a model which can determine the probability of being a new x_{test} as anomalous

Gaussian Distribution:

bell-shaped curve $\mathcal{N}(\mu, \sigma^2)$ μ = mean, σ^2 = variance



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

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

Algorithm:

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

Anomaly if $p(x) < \varepsilon$

Dataset division: Train - CV - Test
60 - 20 - 20 for good
0 - 50 - 50 for bad

Evaluation:

Fit model $p(x)$ on training set

On CV/Test set predit

$p(x) < \varepsilon$, $y = 1$ (anomaly)

$p(x) \geq \varepsilon$, $y = 0$ (normal)

Using, Precision / Recall / F_1-Score evaluate accuracy of model

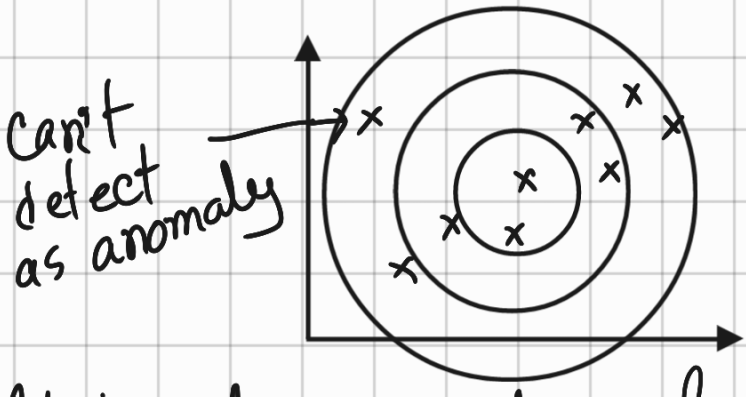
"We use CV for choosing parameter ε"

Anomaly detection	Supervised learning
→ Small number of positive example	→ Large number of both example to train the model
→ Many types of anomaly → Future anomalies may not look like previous	→ Enough positive examples to get sense of what new positive will look like
→ Fraud detection	→ Email Spam Classification
→ Manufacturing defect	→ Weather prediction
→ Monitoring machine in a data center	→ Cancer classification

To make anomaly detection greatly work features need to be gaussian.

If x is a non-gaussian, we can convert it to gaussian by $\log(x)$, $\log(x+c)$, $x^{\frac{1}{c}}$. We need to choose features carefully so that it takes very small or large value in the event of an anomaly.

Normal Gaussian Distribution sometimes fail to detect anomalies like



It is always alinged to axes. To rotate the shape we need to use Σ instead of σ^2

Now, we'll use Multivariated Gaussian Dist

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

Varing Σ changes the shape, width and orientation of the contours. Changing μ move the center of distribution.

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

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

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

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

Original model	Multivariated Gaussian
→ Manually create features for unusual combinations	→ Automatically capture correlations between features
→ computationally cheaper	→ Computationally more expensive $(\Sigma^{-1}) \rightarrow \mathbb{R}^{n \times n}$
→ Better scaling option	→ If features are linealy dependent then Σ is non-invertable
→ OK even with small m.	→ Must have $m > n$ otherwise Σ is non-inv. (Essentially $m \geq 10n$)

Recommender Systems:

n_u = # of users

n_m = # of movies

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

$y(i,j)$ = rating given by user j to movie i (only if $r(i,j) = 1$)

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vector for movie i

For, user j, movie i, predicted rating: $(\theta^{j})^T (x^{(i)})$

$m^{(j)}$ = number of movies rated by user j

$$\min (\theta^{(1)}, \ldots \theta^{(n_u)}) = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1}^{m^{(j)}} \left((\theta^{j})^T (x^{(i)}) - y^{(i,j)}\right)^2 +$$

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

↳ Same as linear regression except $\frac{1}{m}$

Collaborative Filtering:

To infer features from given parameter, we use the squared error function with regularization over all users:

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

Firstly, randomly guess θ, then

$$\theta \to x \to \theta \to x \to \theta \to x \to \theta \to \dots$$

Algorithm:

1. Initialize $x^{(i)},\dots x^{(n_m)}$, $\theta^{(1)},\dots\theta^{(n_u)}$ to small random (break symmetry) value.

2. Minimize $J(x,\Theta)$ using gradient descent

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

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

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

$$Y = X \Theta^T$$

Most similar two moive smallest $\|x_i - x_j\|$

In this recommendation system, if a new user is added without any rating, we automatically assign him/her rating 0 for all movies. This is intuitively incorrect.

We better assign "mean" values.

So modified linear regression will be

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