Day - 95, March 5, 2025 (Falgun 21, 2081)

① Anomaly Detection vs. Supervised Learning

② Choosing what features to use

③ Making Recommendations

④ Anomaly Detection Algorithm

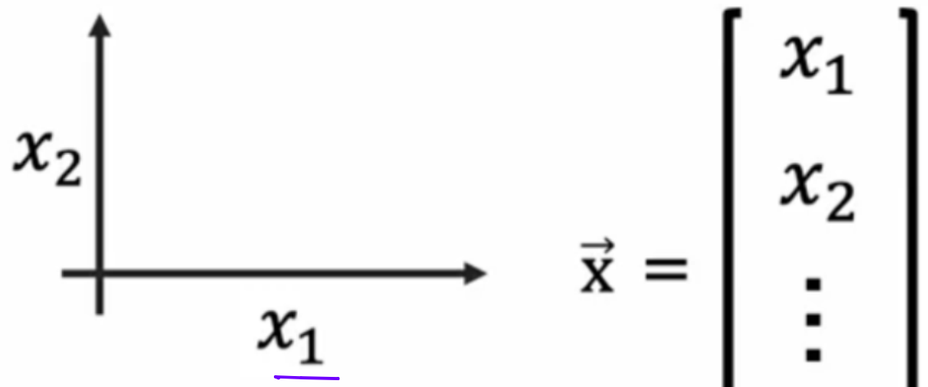
⑧ Developing and Evaluating an Anomaly Detection Algorithm.

Source: Unsupervised Learning, Recommendation, Reinforcement Learning
Coursera.

Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(m)}\}$

Each example $\vec{x}^{(i)}$ has n features



Assuming Independent Variables.

$$p(\vec{x}) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * p(x_3; \mu_3, \sigma_3^2) * \cdots * p(x_n; \mu_n, \sigma_n^2)$$

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

$\sum$ "add" $\quad$ $\prod$ "Multiply"

$p(x_1 = \text{high temp}) = 1/10$

$p(x_2 = \text{high vibra}) = 1/20$

$p(x_1, x_2) = p(x_1) * p(x_2)$

$\Rightarrow 1/20 \times 1/20$

$\Rightarrow 1/200$

Anomaly detection algorithm

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

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

vectorized formula.

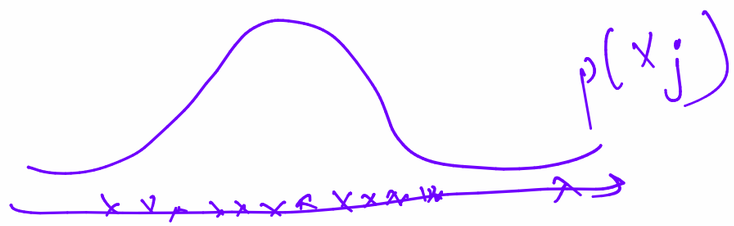
$$\vec{\mu} = \frac{1}{m} \sum_{i=1}^{m} \vec{x}^{(i)} \qquad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

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

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

$$= \prod_{j=1}^{n} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

→ Anomaly if $p(x) < \epsilon$



Steps:

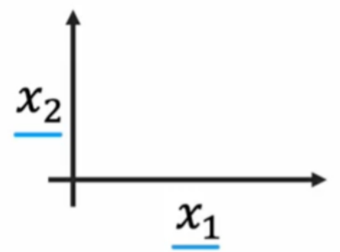
$p(\vec{x})$ = Independent probability.

1. Feature Selection ($x_1, x_2 \dots x_n$)
2. Model Assumption (Normal dist).

Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(m)}\}$

Each example $\vec{x}^{(i)}$ has n features



$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$p(\vec{x}) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * p(x_3; \mu_3, \sigma_3^2) * \cdots * p(x_n; \mu_n, \sigma_n^2)$$

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

$\sum$ "add" $\prod$ "multiply"

$p(x_1 = \text{high temp}) = 1/10$

$p(x_2 = \text{high vibra}) = 1/20$

$p(x_1, x_2) = p(x_1) * p(x_2)$

$= \frac{1}{10} \times \frac{1}{20} = \frac{1}{200}$

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

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

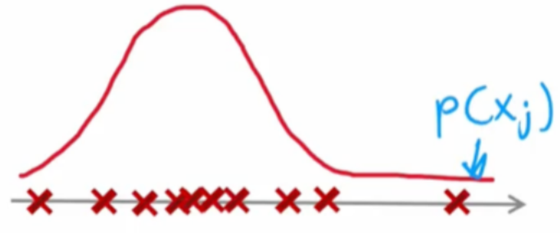
Vectorized formula

$$\vec{\mu} = \frac{1}{m}\sum_{i=1}^{m} \vec{x}^{(i)} \qquad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_n \end{bmatrix}$$

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

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

Anomaly if $p(x) < \varepsilon$

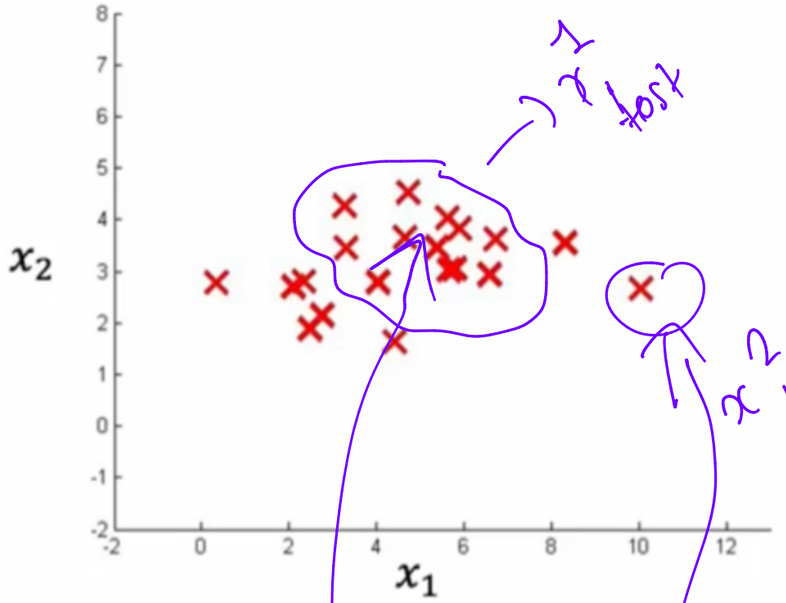


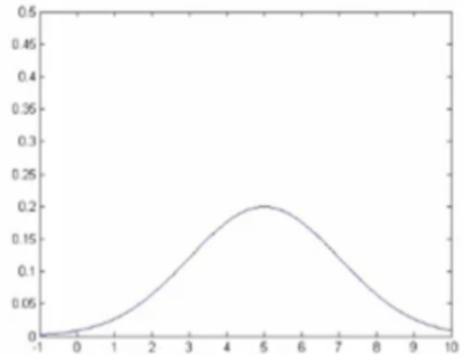
3. Parameter Estimation μ_j and σ_j^2
4. Probability Calculation:

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

5. Anomaly Detection
6. Intuition: See distribution.

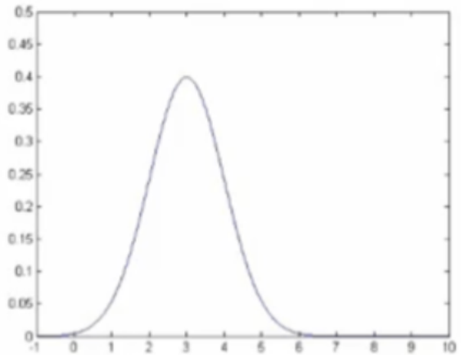
Anomaly detection example





$\mu_1 = 5, \sigma_1 = 2$

$p(x_1; \mu_1, \sigma_1^2)$



$\mu_2 = 3, \sigma_2 = 1$

$p(x_2; \mu_2, \sigma_2^2)$

$\varepsilon = 0.02$

$p(x_{test}^{(1)}) = 0.0426$ "Ok"

$p(x_{test}^{(2)}) = 0.0021$ ↓ Not Okay.

Here's the data set with features x 1 and x 2.

DeepLearning.AI Stanford ONLINE Andrew Ng

Summary of Gaussian Anomaly Detection Algorithm

1. **Feature Selection**: Choose features $x_1, x_2, \ldots, x_n$ that may indicate anomalies (e.g., engine heat and vibrations).
2. **Density Estimation**: Model the probability $p(x)$ of a feature vector x using the product of individual feature probabilities:

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

   This assumes feature independence, but the algorithm works even if this assumption is not strictly true.

3. **Parameter Estimation**: Compute mean μ_j and variance σ_j^2 for each feature x_j using training data:

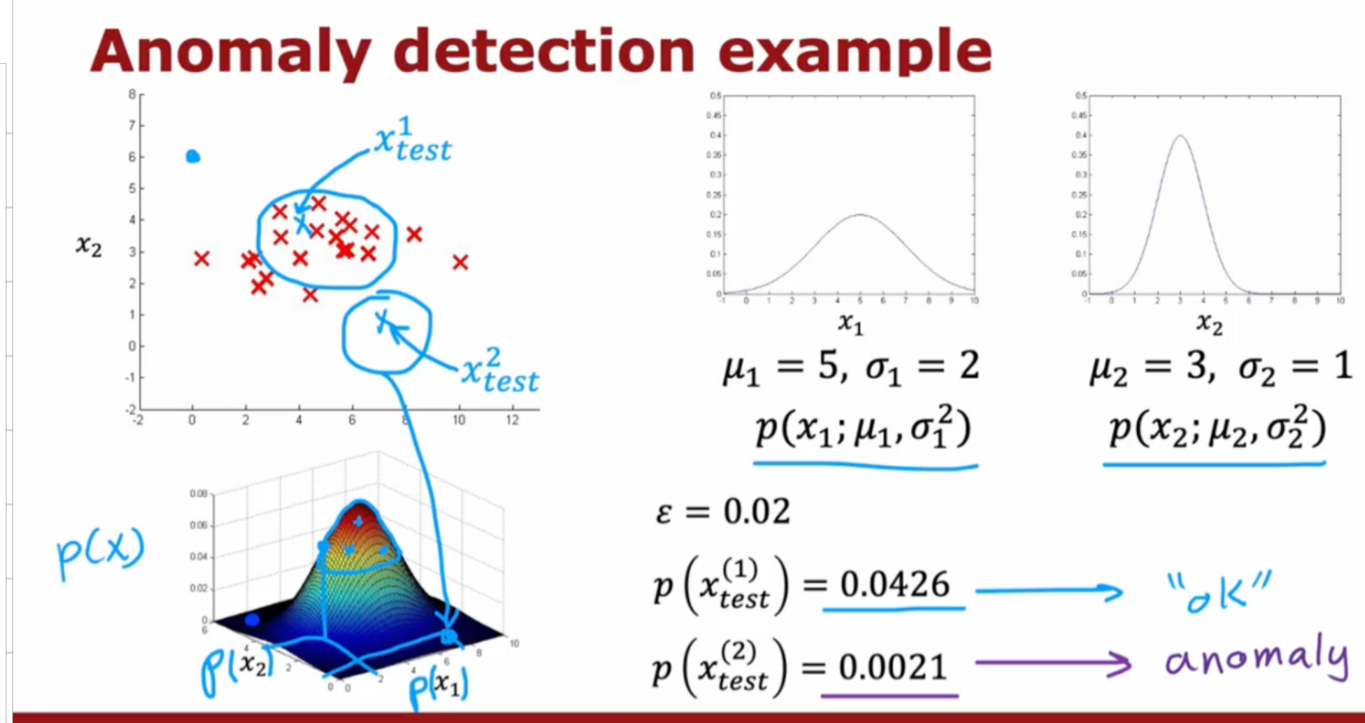
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}, \quad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_{ij} - \mu_j)^2$$

4. **Probability Calculation**: Compute the Gaussian probability density function (PDF) for each feature:

$$p(x_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

   Then compute $p(x)$ by multiplying probabilities for all features.

5. **Anomaly Detection**: Compare $p(x)$ with a threshold ϵ:
   - If $p(x) < \epsilon$, flag x as an anomaly.
   - If $p(x) \geq \epsilon$, classify x as normal.
6. **Intuition**: If any feature value is significantly different from the training distribution, $p(x)$ becomes very small, indicating an anomaly.



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

$p(x_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)}$

7. Visualization:
   - 2D Gaussian distributions show high-probability regions in the center
   - Anomalies lie in low-probability regions, far from many data centers.

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.

$y = 1$ $y = 0$

Training set: $x^{(1)}, x^{(2)} \dots x^{(m)}$ (assume normal examples/ not anomalous)
$y = 0$ for all training examples

Cross Validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}) \dots (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$ } Include a few anomalous examples $y = 1$

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

Mostly Normal examples $y = 0$.

Aircraft engines monitoring example

10000 good (normal) engines
~~20~~ 2 flawed engines (anomalous)

2 to 50

$y=1$ from Algorithm in training set.

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

CV: 2000 GE ($y=0$) 10 anomalous ($y=1$)

Test: 2000 GE ($y=0$), 10 " ($y=1$)

Cross validation set tune ϵ

(Using dense flawed Engine is beneficial)

Alternative: No test set

Training Set: 6000 good engines 2

CV: 4000 good engines ($y=0$), ~~20~~ anomalous ($y=1$)

tune ϵ tune x_j.

→ Smaller Data-Size are prone to the higher risk of overfitting.

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.

$y=1$ $y=0$

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

$y=0$ for all training examples

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

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

include a few anomalous examples $y=1$

mostly normal examples $y=0$

Real-Number Evaluation:

↳ Quickly Assess and tune parameters like Epsilon

Labelled Anomalies for Validation:

↳ Use a small set of known anomalies to refine the model.

Aircraft engines monitoring example

10000 good (normal) engines — 2 to 50

~~20~~ 2 flawed engines (anomalous) — $y=1$

$y=0$

Training set: 6000 good engines — train algorithm on training set

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

use cross validation set — tune ε tune x_j

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

Alternative: No test set — Use if very few labeled anomalous examples

Training set: 6000 good engines 2

CV: 4000 good engines ($y=0$), ~~20~~ anomalous ($y=1$) — higher risk of overfitting

tune ε tune x_j

- Data Partitioning:
  - Training: Mostly Normal Data
  - Cross-Validation: Mix of Normal and Anomalies for tuning.
  - Test: Similar to Cross-Validation for final evaluation.

↳ Merge Cross-Validation and test sets but be aware of Overfit.

course 2 week 3

skewed datasets

Algorithm evaluation

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
On a cross validation/test example x, predict

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

10

2000

possible evaluation metrics:

- TP, FP, FN, TN
- Precision/Recall
- F_1-Score:

Use Cross-Validation Set to choose parameter ε.

More Uncertainties.

Anomaly detection vs. Supervised learning

Very small number of positive examples ($y = 1$). (0-20 is common).
Large number of negative ($y = 0$) examples.

p(x)

y = 1

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.

Fraud.

Many different types/scenarios where engine might go wrong

Large number of positive and negative examples.

20 positive 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.

Example: Spam

Anomaly detection vs. Supervised learning

- Fraud detection
- Manufacturing - Finding new previously unseen defects in manufacturing.(e.g. aircraft engines)
- Monitoring machines in a data center

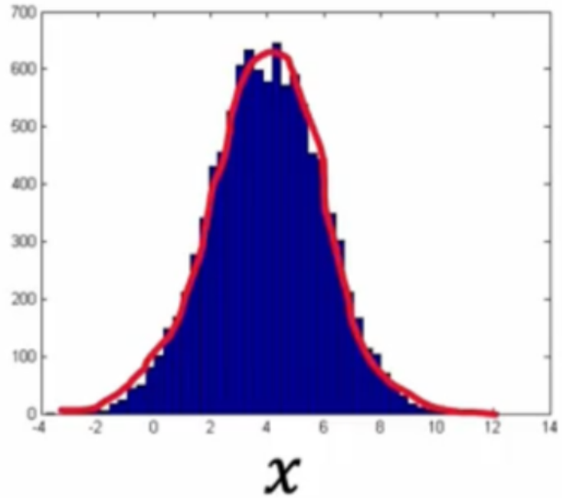
⋮

- Email spam classification
- Manufacturing - Finding known, previously seen defects
- Weather prediction (sunny/rainy/etc.)
- Diseases classification

⋮

Always use Gaussian features (Make features into Normal distribution).

Non-gaussian features



$p(x_1; \mu_1, \sigma_1^2)$

```
plt.hist(x)
```

$x_1 \leftarrow \log(x_1)$

$x_2 \leftarrow \log(x_2+1)$ $\log(x_2+c)$

$x_3 = \sqrt{x_3} \Rightarrow x_3^{1/2}$.

$x_4 \leftarrow x_4^{1/4}$.

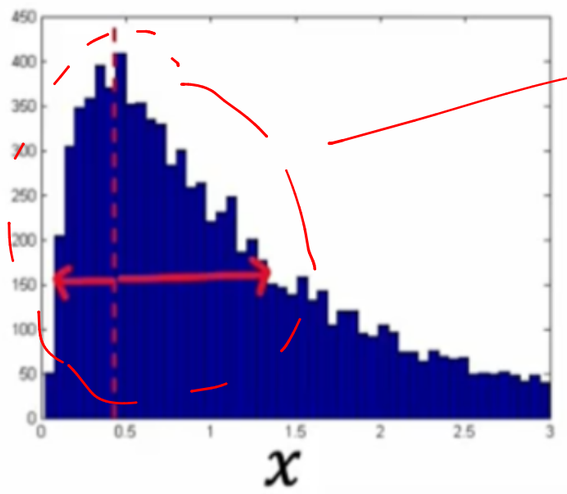
Use

plt.hist(x)

↓

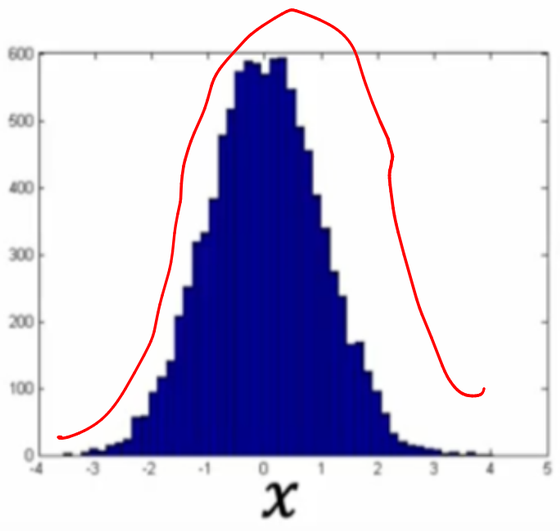
Normalized

np.log(x).



use this.

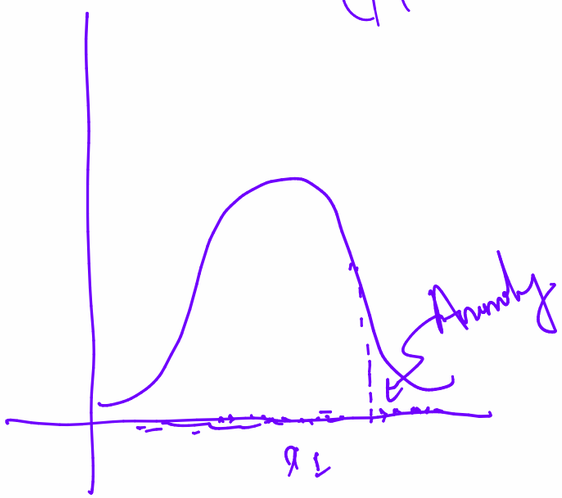
```
np.log(x)
```

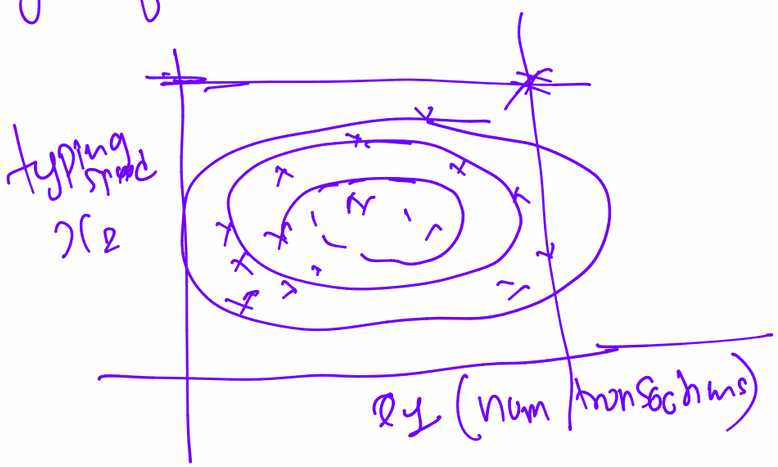


Error analysis for anomaly detection

Want $p(x) \geq \varepsilon$ large for normal examples x.
$p(x) < \varepsilon$ small for anomalous examples x.

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





Monitoring computers in a data center

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

x_1 = memory use of computer

x_2 = no. of disk accesses/sec

x_3 = CPU load

x_4 = network traffic

Computer Behaving Anomaly?

high / low: ← not unusual.

$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$

$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$

Deciding feature choice based on $p(x)$.

Making Recommendation

Predicting movie ratings

User rates movies using one 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. of users

n_m = no. of movies

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

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

$n_u = 4$

$n_m = 5$

$r(1,1) = 1$

$r(3,1) = 0$

Goal: Predict missing ratings to recommend items.

Notation:

n_u, n_m: num of users and items.

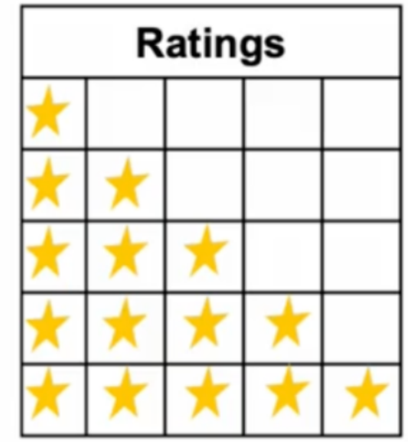
$r(i,j)$: 1 if user j rated movie i, else 0

$y(i,j)$: User j's rating for movie i.

Predicting movie ratings

User rates movies using ~~one~~ 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. of users

n_m = no. of 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 $r(i,j)=$

$n_u = 4$ $r(1,1) = 1$

$n_m = 5$ $r(3,1) = 0$ $y^{(3,2)} = 4$

DeepLearning.AI Stanford ONLINE Andrew Ng

Start with known features (eg. genres) later explore feature-free methods.