SEMANA 9 - DETECCIÓN DE ANOMALÍAS

1. ESTIMACIÓN DE DENSIDAD

- Motivación del problema

Principalmente en problemas de aprendizaje no supervisado.

Supuesto de ejemplos normales.

Con los datos de entrenamiento sin etiqueta generamos un modelo de probabilidad P(x).

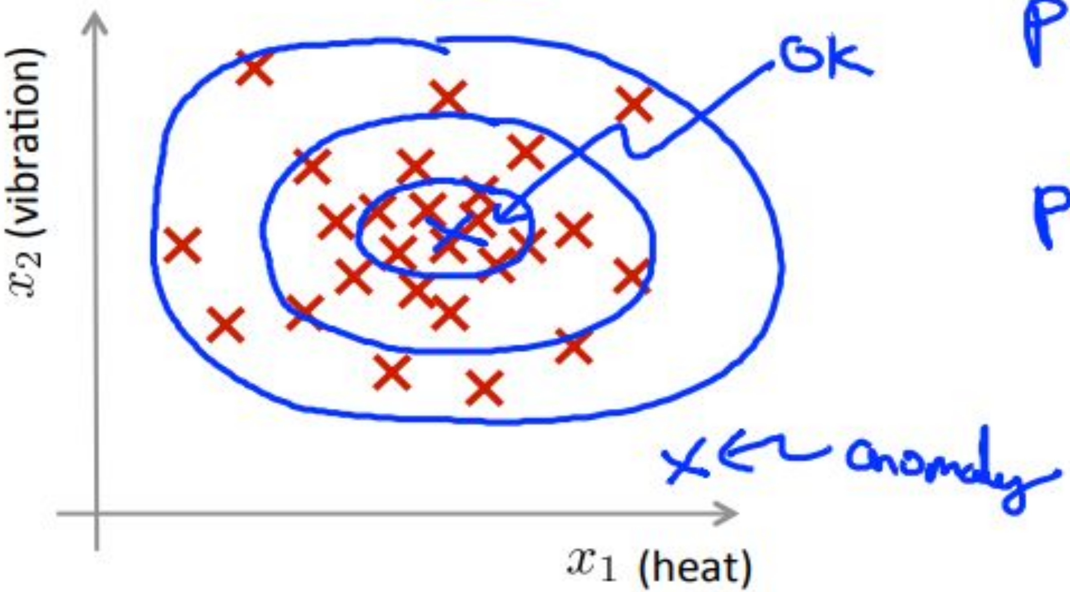
Introducimos nuevos datos (test) en el modelo de probabilidad y lo comparamos con un valor ‘exilon’ para er si es anormal o no.

Density estimation

→ Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

→ Is x_{test} anomalous?

Model $p(x)$.



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

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

- <u>Distribución Gaussiana</u>

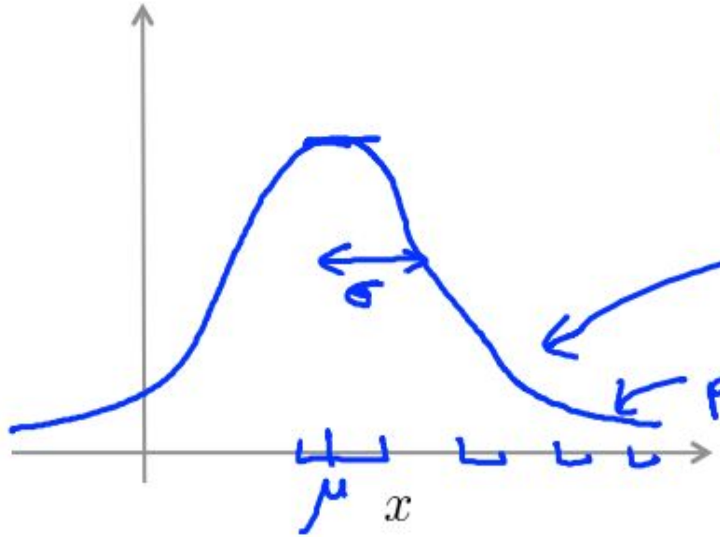
Gaussian (Normal) distribution

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

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

"distributed as"

σ standard deviation



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

$p(x;\mu,\sigma^2)$

- <u>Algoritmo</u>

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples. $\{x^{(1)}, \ldots, x^{(m)}\}$
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

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

$p(x_j; \mu_j, \sigma_j^2)$

$\mu_1, \mu_2, \ldots, \mu_n$

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

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

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

Anomaly if $p(x) < \varepsilon$

Andrew Ng

CREACIÓN DE UN SISTEMA DE DETECCIÓN DE ANOMALÍAS

- Desarrollo y Evaluación de un sistema de detección de anomalías

  Tomar decisiones es mucho más fácil si tenemos una forma de evaluar nuestro algoritmo de aprendizaje.

  Training Set. Asumimos datos normales (y=0)

Aircraft engines motivating example

10000 good (normal) engines

20 flawed engines (anomalous) 2-50 $y=1$

$\mu_1, \sigma_1^2, \ldots, \mu_n, \sigma_n^2$

Training set: 6000 good engines $(y=0)$ $p(x) = p(x_1; \mu_1, \sigma_1^2) \cdots p(x_n; \mu_n, \sigma_n^2)$

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

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

Alternative:

Training set: 6000 good engines

CV: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Test: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Algorithm evaluation

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

$(x^{(i)}_{test}, y^{(i)}_{test})$

- On a cross validation/test example x, predict

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

$y = 0$

Possible evaluation metrics:

- - True positive, false positive, false negative, true negative
- - Precision/Recall
- - F_1-score

CV

Test set

Can also use cross validation set to choose parameter ε

- <u>Detección de anomalías VS Aprendizaje Supervisado</u>

Anomaly detection vs. Supervised learning

Anomaly detection

- Very small number of positive examples ($y = 1$). (0-20 is common).
- Large number of negative ($y = 0$) examples. $\boxed{p(x)}$
- 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.

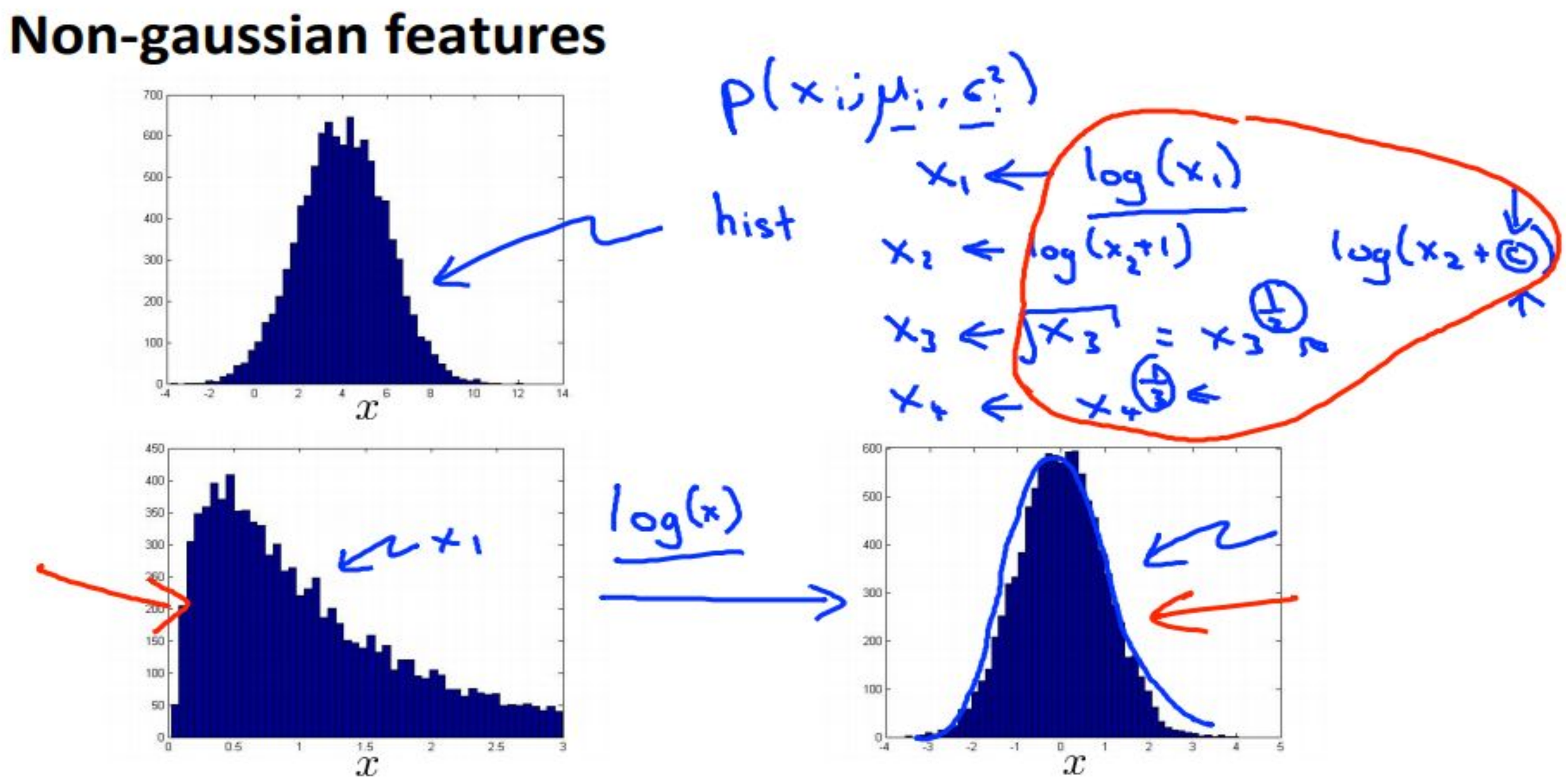
Supervised learning

- Large number of positive and negative 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.

Spam

- <u>Elegir qué variables usar</u>

  Realizar un histograma de las variables, si presenta asimetría transformar hasta obtener la forma de Campana de Gauss.



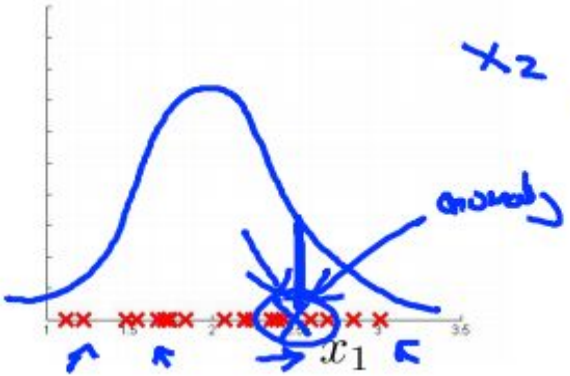
Error analysis for anomaly detection

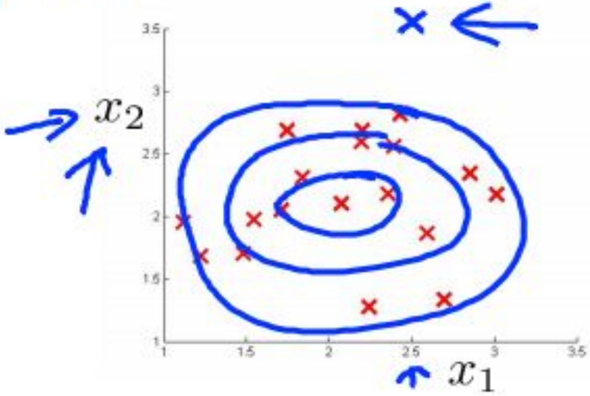
Want $p(x)$ large for normal examples x.

$p(x)$ small for anomalous examples x.

Most common problem:

$p(x)$ is comparable (say, both large) for normal and anomalous examples



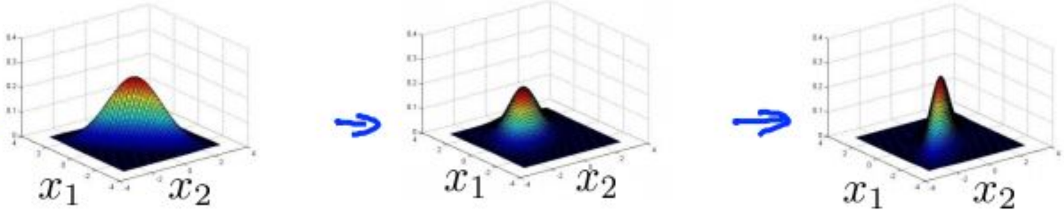


Distribución Gaussiana Multivariante

Multivariate Gaussian (Normal) distribution

Parameters $\underline{\mu, \Sigma}$ $\mu \in \mathbb{R}^n$ $\Sigma \in \mathbb{R}^{n \times n}$

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



Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $x \in \mathbb{R}^n$

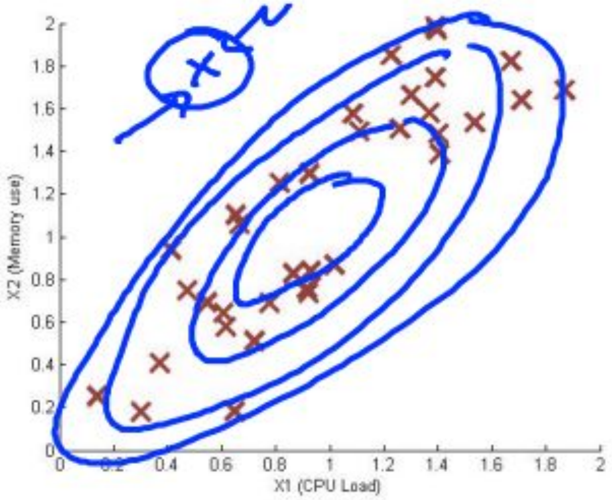
$$\boxed{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \boxed{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

Anomaly detection with the multivariate Gaussian

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

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

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



2. Given a new example x, compute

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

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

Original model vs. Multivariate Gaussian

$$p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$$

Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.

$\boxed{x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}}$

Computationally cheaper (alternatively, scales better to large $n = 10{,}000$, $n = 100{,}000$)

OK even if m (training set size) is small

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

Automatically captures correlations between features

$\Sigma \in \mathbb{R}^{n \times n}$ $\quad \Sigma^{-1}$

Computationally more expensive

$\Sigma \sim \frac{n^2}{2}$

$x_1 = \cancel{x_2}$

$\cancel{x_3} = x_4 + x_5$

Must have $\boxed{m > n}$ or else Σ is non-invertible. $\boxed{m \geq 10n}$

SISTEMAS DE RECOMENDACIÓN

Content-based recommender systems

$n_u = 4$, $n_m = 5$

$x_0 = 1$

$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$
$\boxed{\text{Love at last}}$ 1	5	5	0	0
Romance forever 2	5	?	?	0
Cute puppies of love 3	? 4.95	4	0	?
Nonstop car chases 4	0	0	5	4
Swords vs. karate 5	0	0	5	?

$n = 2$

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.

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

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \longleftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \qquad (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

Andrew Ng

Optimization algorithm:

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

Gradient descent update:

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

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

$\cancel{\frac{1}{m^{(j)}}}$

Andrew Ng

$\cancel{x_0 = 1}$ $\quad x \in \mathbb{R}^n, \ \theta \in \mathbb{R}^n$

Collaborative filtering algorithm

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$\cancel{\theta_0}$, θ_1, $\vdots$, θ_n

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

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

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

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

Collaborative filtering

$X \Theta^T$

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

(i,j)

Predicted ratings:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \boxed{(\theta^{(1)})^T(x^{(1)})} & \boxed{(\theta^{(2)})^T(x^{(1)})} & \ldots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \ldots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{(\theta^{(1)})^T(x^{(n_m)})} & (\theta^{(2)})^T(x^{(n_m)}) & \ldots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}}$$

$$X = \begin{bmatrix} - (x^{(1)})^T - \\ - (x^{(2)})^T - \\ \vdots \\ - (x^{(n_m)})^T - \end{bmatrix} \qquad \Theta = \begin{bmatrix} - (\theta^{(1)})^T - \\ - (\theta^{(2)})^T - \\ \vdots \\ - (\theta^{(n_u)})^T - \end{bmatrix}$$

→ Low rank matrix factorization