1. Introducción

Definición 1.0.1 (Machine learning (aprendizaje automático)) *Diremos que se aplica machine learning cuando un programa aprende de la experiencia, respecto de una determinada tarea, a través del uso de una unidad de medida.*

Se pueden diferenciar dos tipos de machine learning:

1. Supervised learning (Apredizaje supervisado).
2. Unsupervised learning (Apredizaje no supervisado).

Definición 1.0.2 (Supervised learning) *Es un tipo de machine learning que da la respuesta correcta de una variable.*

Ejemplo: Predicción del valor de una casa.

Definición 1.0.3 (Unsupervised learning) *Es un tipo de machine learning que proporciona las relaciones de nuestros datos.*

Ejemplo: Búsqueda de noticias similares.

A la hora de crear un algoritmo de machine learning es bueno considerar los siguientes consejos:

- Coleccionar muchos datos: la base de cualquier algoritmo son los datos. Cuantos más datos se tengan mejor.
- Empezar con un algoritmo simple: Al principio es conveniente elegir las características más relevantes. Posteriormente en función de los resultados erróneos, se eligen características que puedan solventen dichos errores.
- Hacer una lista de soluciones: Cuando un algoritmo no funciona es bueno realizar una lista de soluciones antes de tomar una decisión. Esto se debe a que algunas decisiones pueden llevar meses y no solventar el problema. Algunas soluciones que se consideran son: examinar más detenidamente los ejemplos que fallan, valorar si es mejor considerar más características o ejemplos, valorar la creación de datos artificiales (y el tiempo que supone)...

1.1. Machine learning pipeline

En ocasiones para obtener un resultado es necesario dividir el problema en varios sub-problemas de machine learning. Cada uno de estos sub-problemas serán tratados por separado en diferentes módulos. Para determinar en que módulo es necesario invertir más tiempo se utiliza el celling analisys. Este consiste en determinar el accuracy (exito) del modelo global según los diferentes módulos. Por ejemplo si tenemos cuatro módulos y obtenemos lo siguiente:

- Modelo global $\longrightarrow$ 72 % Accuracy
- Modelo global con módulo 1 perfecto $\longrightarrow$ 73 % Accuracy
- Modelo global con módulos 1 y 2 perfectos $\longrightarrow$ 84 % Accuracy
- Modelo global con módulos 1, 2 y 3 perfectos $\longrightarrow$ 85 % Accuracy
- Modelo global con módulos 1, 2, 3 y 4 perfectos $\longrightarrow$ 100 % Accuracy

Entonces los módulos más relevantes son el modulo 2 y el módulo cuatro, puesto que al estar perfectos (pasarles las respuestas correctas) obtenemos un aumento superior al 10 % en accuracy. Por lo tanto, sería necesario invertir más tiempo en estos.

1.2. Elementos de machine learning

Según el método que se utilice, se pueden observar los siguientes elementos:

- Sets (conjuntos)
  1. Training Set (conjunto de entrenamiento-60 %): conjunto de datos que permiten que un modelo aprenda.
  2. Cross Validation Set (conjunto de validación-20 %): conjunto de datos que permiten fijar los mejores parámetros del modelo.
  3. Test Set (conjunto de prueba-20 %): conjunto de datos que determinan el error de un modelo.

Nota 1.2.1 (Mean normalization) *En muchas ocasiones en lugar de tomar directamente los conjuntos, se realiza una normalización de estos:*

$$x_i \to \frac{x_i - \mu_i}{s_i} \tag{1}$$

donde:

- μ_i: *promedio de la variable* x_i.
- s_i: *derivación estándar o* máx(x_i) - mín(x_i)

Generalmente se utiliza cuando para varios i no se verifica:

$$x_i \in [-3, -1/3] \cup [1/3, 3].$$

- Hypothesis (hipótesis)
  - Función obtenida por el modelo para determinar la predicción de una solución al introducirle un ejemplo.
- Cost function (función coste)
  - Función que determina el error del modelo. En función del conjunto podemos distinguir tres tipos de errores:
    - Training Cost Function: Determina el error del aprendizaje del modelo.
    - Cross Validation Cost Function: Determina el error del modelo según el parámetro que se utilice.
    - Test Cost Function: Determina el error final del modelo (no depende de ningún parámetro).
- Regularization (regularización)
  - Control de las variaciones de los parámetros del modelo.

1.3. Machine Learning Diagnostic

En función de los parámetros y datos del modelo podemos distinguir diferentes tipos de ajustes:

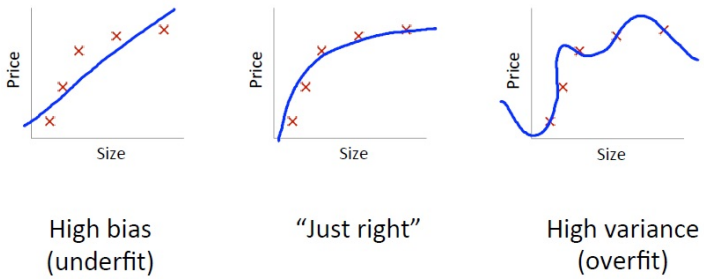

Figura 1: Tipos de ajustes

En función de los parámetros que elijamos podemos obtener cada uno de los ajustes. El mejor ajuste es el "just right", el cual se corresponde con el menor cross-validation error. En función de cada parámetro (según este crezca o decrezca), se suele obtener la siguiente gráfica:

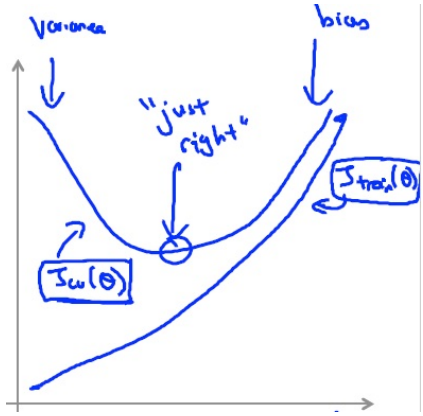

Figura 2: J_{cv}: Cross Validation Cost Function y J_{train}: Training Cost Function

De este modo se elegirán los parámetros del modelo para tener el mínimo cross-validation error. Además, para detectar si existe "high variance" o "high bios" se utilizan las curvas de apredizaje. Estas curvas tienen en cuenta las funciones J_{cv} y J_{train} en función del número de ejemplos:

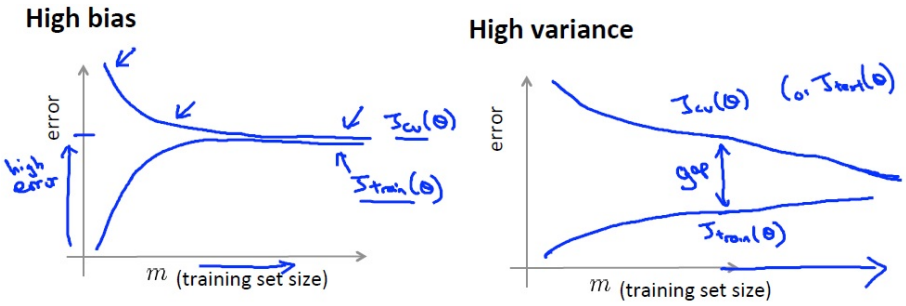

Figura 3: Curva de aprendizaje según el high bios y high variance

Cuando existe high bios las funciones convergen hacia un alto error. En cambio, cuando hay high variance existe un gran espacio entra las funciones. De este modo obtenemos dos opciones:

1. H.bios: El modelo no se ajusta a los datos. Es necesario usar una hipótesis más compleja: se añaden características, se añaden características polinómicas, se disminuye el parámetro de regularización...

2. H.variance: El modelo se sobre-ajusta a los datos. Es necesario usar una hipótesis más simple: se añaden más ejemplos, se quitan características, se aumenta el parámetro de regularización...

2. Modelos

En función del propósito podemos diferenciar varios modelos basados en machine learning:

- Value Prediction: Se utiliza para predecir valores de un ejemplo.

  Modelos:

  - **Linear/polynomial regression**.

  Ejemplos: predecir el valor de una casa...

- Clasification: Se utiliza para clasificar un ejemplo.

  Modelos:

  - **Logistic regression**: Para $n >> m$ ($n = 10000$ y $m = 10 - 1000$) o $n << m$.
  - **Suport vector machine**: Para $n >> m$ ($n = 10000$ y $m = 10 - 1000$) -sin núcleo-. Para $n < m$ ($n = 1 - 1000$ y $m = 10000 - 50000$) -con núcleo-. Para $n << m$ -sin núcleo-.
  - **Neural networks**: Obtiene de funciones complejas con menos coste computacional.

  Ejemplos: email spam classification, predicción del tiempo, cancer classification...

- Anomaly Detection: Se utiliza para detectar si un ejemplo es anómalo. A diferencia de los de clasificación, estos se utilizan cuando el número de ejemplos con $y = 1$ es muy pequeño y $y = 0$ es grande. También se usa cuando se esperan que las futuras anomalías no sean como las actuales.

  Modelos:

  - **Normal distribution**: Cuando m es pequeño o el coste computacional es alto.
  - **Multivariate gaussian distribution**: Cuando $m > n$ y el coste computacional es bajo.

  Ejemplos: detección de usuarios distintos, fraude, monotorización de dispositivos...

- Collaborative filtering: Se utiliza para obtener características de los ejemplos.

  Modelos:

  - **Collaborative filtering**: Se utiliza cuando tienes la opinión de varias personas sobre un mismo producto. Se pueden utilizar incluso si la mayoría de usuarios presentan pocas puntuaciones (la mayoría son 0).

  Ejemplos: recomendar a usuarios películas en función de sus valoraciones, buscar libros (relacionados) que le guste a un usuario...

- Principal component analysis: Se utiliza para reducir el tamaño de los datos.

  Modelos:

  - **Principal component analysis**.

  Ejemplos: reducir el coste computacional de un algoritmo, representar datos en 2D y 3D... No se utiliza para evitar el overfitting.

- Online Learning: Se utiliza para tratar con datos que se reciben constantemente. Se adapta a la sociedad cambiante.

  Modelos:

  - **Online Learning**.

  Ejemplos: Captación de tendencias de usuarios con las que realizar recomendaciones y ofertas especiales.

2.1. Linear/Polinomial regression

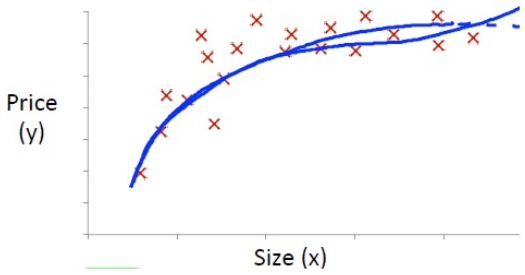

Figura 4: Linear regression

- Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \theta x$$

donde $x_0 = 1$.

Nota 2.1.1 *Para el caso polinómico realizamos lo siguiente:* $x_1 = x_1$, $x_2 = x_1^2$, ..., $x_d = x_1^d$. *De este modo obtenemos* $h_\theta(x) = h_\theta(x_1)$.

- Cost Function:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) + \frac{\lambda}{2m}\sum_{i=1}^{n}\theta_j^2$$

- Parámetros desconocidos:
  - $\theta \longrightarrow$ Training Set (Automático).
  - $\lambda \longrightarrow$ Cross Validation Set (Seleccionar).
  - $d \longrightarrow$ Cross Validation Set (Manual).
- Consejos:
  - Hay que fijarse que la regularización no afecta al parámetro θ_0.
  - La normalización de las variables es importante en el caso polinómico.

2.2. Logistic regression

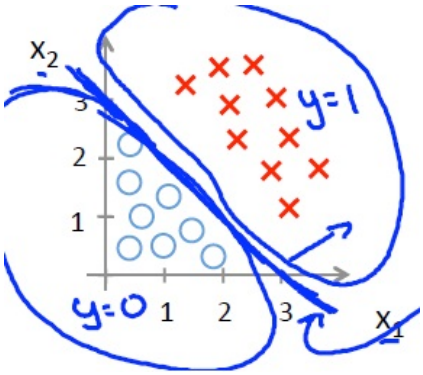

Figura 5: Logistic regression

- Hypothesis:

$$h_\theta(x) = g(\theta x)$$

donde $x_0 = 1$ y $g(z) = \dfrac{1}{1+e^{-z}}$ (sigmoid function).

Nota 2.2.1 *Para el caso polinómico realizamos lo siguiente:* $x_1 = x_1$, $x_2 = x_1^2$, ..., $x_d = x_1^d$. *De este modo obtenemos* $h_\theta(x) = h_\theta(x_1)$.

- Cost Function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

donde

$$cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

- Boundary

$$h_\theta(x) = P(y = 1|x, \theta) \tag{2}$$

Si consideremos que:

$$h_\theta(x) \geq \alpha \Rightarrow y_{pred} = 1.$$

la frontera vendrá dada por:

$$h_\theta(x) = \alpha.$$

Por lo tanto:

- Para predecir únicamente si se está seguro se aumenta el α.
- Para predecir si existe una mínima posibilidad se disminuye el α.

- Multi-class (one vs all)

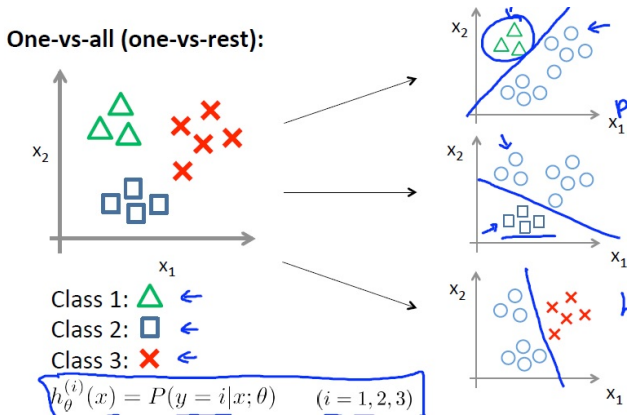

Figura 6: Logistic regression

Si y puede tener más de dos valores (1,2,3,4...) se utiliza multi-class clasification. Para ello se realizan tantas logistic regression como clases. Las logistic regression se realizan totalmente por separado considerando cada tipo como $y = 1$ y el resto $y = 0$. De este modo la hipótesis de predicción será:

$$h_\theta(x) = \max_i h_\theta^{(i)}(x).$$

donde $h_\theta^{(i)}(x)$ son cada una de las hipótesis.

- Parámetros desconocidos:
  - $\theta \longrightarrow$ Training Set (Automático).
  - $\lambda \longrightarrow$ Cross Validation Set (Seleccionar).
  - $d \longrightarrow$ Cross Validation Set (Manual).

2.3. Neural networks

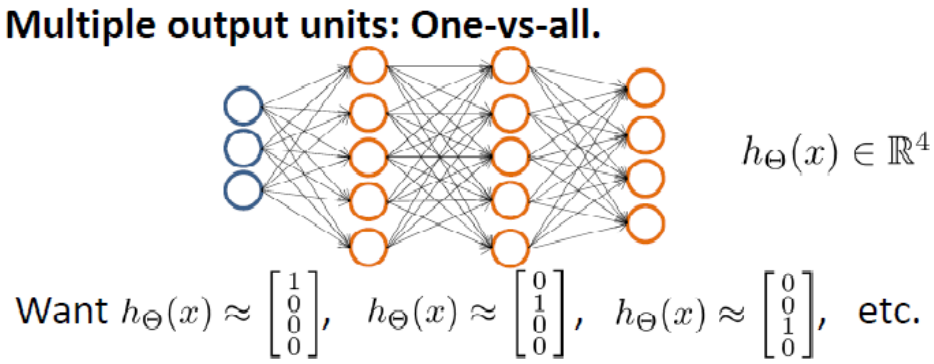

Figura 7: Neural network

La layer 1 (capa 1) se denomina input layer. Por otro lado la capa final se denomina output layer. El resto de capas intermedias son las hidden layers.

- Hypothesis:

  Forward propagation (ejemplo con 4 layers):

$$\begin{aligned} a^{(1)} &= x \\ z^{(2)} &= \theta^{(1)} a^{(1)} \\ a^{(2)} &= g(z^{(2)}) \\ z^{(3)} &= \theta^{(3)} a^{(3)} \\ a^{(3)} &= g(z^{(3)}) \\ z^{(4)} &= \theta^{(3)} a^{(3)} \\ h &= g(z^{(4)}) \end{aligned}$$

- Cost Function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}(y_k^{(i)}\log(h_\theta^k(x^i)))+((1-y_k^{(i)})\log(1-h_\theta^k(x^i)))+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}(\theta_{ji}^{(l)})^2$$

- Parámetros desconocidos:
  - $\theta \longrightarrow$ Training Set (Automático).
  - $\lambda \longrightarrow$ Cross Validation Set (Seleccionar).

- Consejos:
  - Para calcular las derivadas de la función coste se puede realizar Backpropagatión (mejora la velocidad para minimizar J).
  - Es necesario inicializar los parámetros de forma aleatoria: $\theta \in [-\epsilon, \epsilon]$.
  - Lo más usual es utilizar una capa oculta.
  - Cuantas más características tenga la capa oculta más compleja es la función hipótesis.
  - Si se utilizan varias capas ocultan, se suelen tomar del mismo tamaño.

2.4. Suport Vector Machine

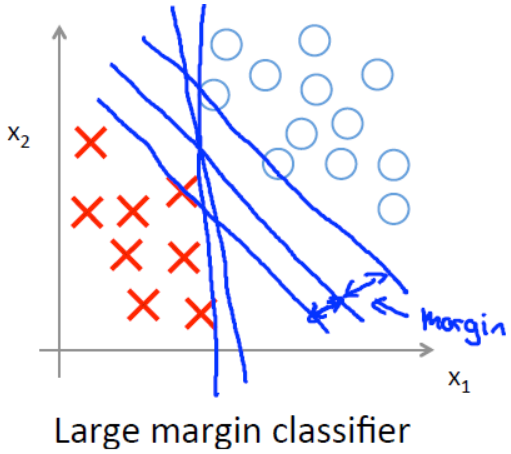

Figura 8: Suport Vector Machine

- Hypothesis:

$$h_\theta(x) = \begin{cases} 1 & \text{si } \theta x \geq 0; \\ 0 & \text{en otro caso.} \end{cases}$$

- Cost Function:

$$J(\theta) = C\sum_{i=1}^{m} y^{(i)} cost_1(\theta x^{(i)}) + (1 - y^{(i)}) cost_0(\theta x^{(i)}) + \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

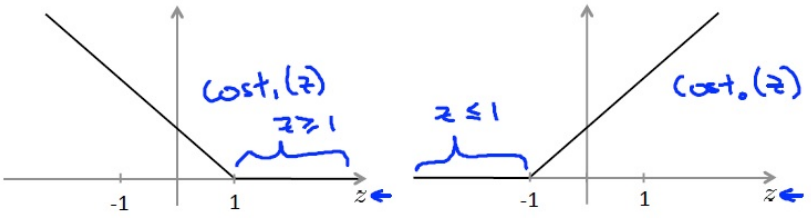

Figura 9: Suport Vector Machine Cost

- Núcleos:
  - Se cambian las características para obtener decisiones no lineales.

$$x_1, x_2, ..., x_n \longrightarrow f_1, f_2, ..., f_n$$

donde f_i son los núcleos.

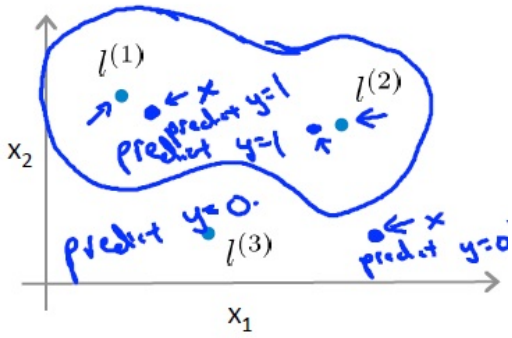

Figura 10: Gaussian Kernels

- Gaussian Kernel:

$$f_s = \exp(-\frac{||x - l^{(s)}||}{2\sigma^2})$$

  donde $l^{(s)}$ es un ejemplo. De este modo mide la distancia entre el punto $l^{(s)}$ y el ejemplo x, obteniéndose m características.

- Otros núcleos: Polynomial kernels: $k(x, l) = (xl + c_1)^{c_2}$, ... (satisfacen "Mercer's Theorem").

- Parámetros desconocidos:
  - $\theta \longrightarrow$ Training Set (Automático).
  - $C \longrightarrow$ Cross Validation Set (Seleccionar).
  - $\sigma \longrightarrow$ Cross Validation Set (Seleccionar).
- Consejos:
  - Utilizar librerías para implementarlo (falta por hacer).
  - Es importante normalizar las características antes de hacer los núcleos.
  - Los núcleos solo se aplican en Suport Vector Machine.

2.5. K-Means Algorithm

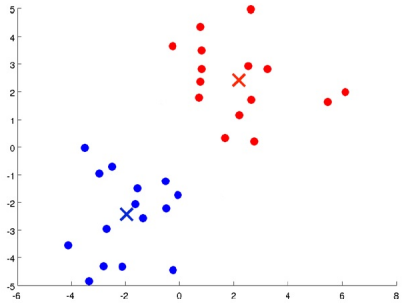

Figura 11: K-Means algorithm

- Process:
  La función coste se minimiza en dos pasos:

  1. Clasifica los puntos en función del centro que esté más cerca: $\text{mín}_K \, ||x^{(i)} - \mu_k||$
  2. Una vez los puntos tienen asignado un centro, se mueve cada centro al punto promedio de los puntos con dicho centro asignado.

  $$\mu_k = \text{ promedio de los puntos asignados al claster K}$$

- Cost Function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{C^{(i)}}||$$

  donde $C^{(i)}$ es el índice del centro $\mu_{C^{(i)}}$ al que $x^{(i)}$ ha sido asignado.

- Parámetros desconocidos:

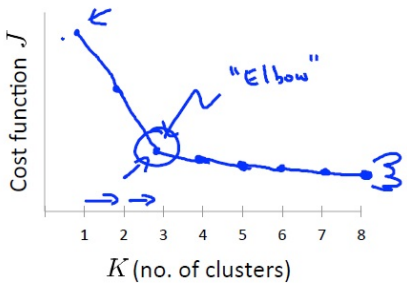

Figura 12: K-Means elbow

- K: El número de clasters se elige de dos modos:
  - Método del codo: Se elige el número que se encuentre en el codo según el error para cada K (no siempre existe).
  - En función del propósito: Por ejemplo en las ventas más clasters se corresponden con una mayor oferta pero más caro. Menos clasters se corresponden con una menor oferta pero más barato.

- Consejos:
  - La inicialización se realiza tomando como centros algunos puntos de $x^{(i)}$ de forma aleatoria.
  - Es bueno inicializar varias veces (puesto que en ocasiones falla) para que no coja soluciones no muy buenas, y coger la de menor coste.

2.6. Principal Component Analysis

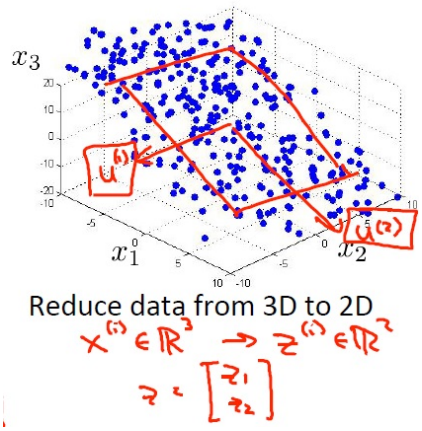

Figura 13: PCA

- Calculo de vectores U:

  $[U, S, V] = svd(Sigma)$ (también se puede utilizar $eig(Sigma)$).

- Calculo de nuevos valores en dimensión reducida:

  $$Z = U_{reduce}X \text{ donde } U_{reduce} \text{ es } u_1, u_2, ..., u_r.$$

- Cálculo de las proyecciones:

  $$x_{approx} = U_{reduce}Z$$

- Parámetros desconocidos:

  - r: El número de dimensiones finales. Se puede elegir de dos formas:

    1. Se elige el primer r que verifique:

       $$1 - \frac{\sum_{i=1}^{r} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq p$$

       de modo que:

       - Si $p = 0{,}01 \implies 99\,\%$ de varianza retenida.
       - Si $p = 0{,}05 \implies 95\,\%$ de varianza retenida.
       - Si $p = 0{,}1 \implies 90\,\%$ de varianza retenida.

2. Para representar los datos en 2D o 3D (r=2,3).

- Consejos:
  - Es necesario normalizar los datos para hacer la reducción.
  - Si se aplica antes de realizar un algoritmo se aplica solo sobre el training set. Para los otros casos se utiliza U_{reduce} (para trasformar los datos a menor dimensión).
    - $x_1, x_2, ..., x_n \longrightarrow^{PCA} z_1, z_2, ..., z_r$ para Training Set.
    - $x_1, x_2, ..., x_n \longrightarrow^{U_{reduce}} z_1, z_2, ..., z_r$ para Cross and Test Sets.
  - Siempre es mejor probar primero sin PCA y posteriormente con PCA (se pierde cierta información al utilizar PCA).

2.7. Normal/Gaussian Distribution

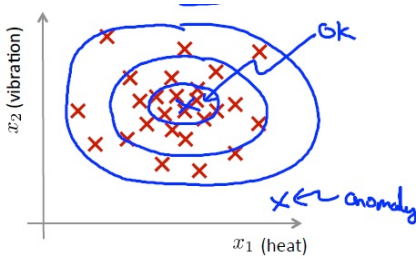

Figura 14: Normal/Gaussian Distribution

Caso - Normal distribution:

$$
\begin{aligned}
X &\sim N(\mu, \sigma^2) \\
\mu &= \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \\
\sigma^2 &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2 \equiv \frac{1}{m-1}\sum_{i=1}^{m}(x^{(i)} - \mu)^2 \\
p(x) &= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})
\end{aligned}
$$

- Detección de casos anómalos:

$$
\begin{aligned}
p(X_t) &< \epsilon \longmapsto \text{Caso anómalo} \\
p(X_t) &\geq \epsilon \longmapsto \text{Caso normal}
\end{aligned}
$$

- Parámetros desconocidos:
  - $\epsilon \longmapsto$ Cross validation set mediante el uso del F1 Score (si se tienen valores).
- Consejos:
  - Es importante la elección manual de características que obtenga relaciones de estas (Ej: $x_5 = \frac{x_4^2}{x_2}$).
  - El algoritmo funciona incluso sin independencia de soluciones.
  - Es bueno tener o crear variables normales (sin normalizar). Para ello se pueden realizar transformaciones: $x_1 \longleftrightarrow \log(x_1 + c), x_1 = x_1^{\frac{1}{3}}$...

Caso - Multivariate Normal distribution:

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \\ \sigma^2 &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)^2 \equiv \frac{1}{m-1}\sum_{i=1}^{m}(x^{(i)}-\mu)^2 \\ p(x) &= \frac{1}{(2\pi)^{n/2}|\sigma|^{1/2}}\exp(-\frac{1}{2}(X-\mu)\sigma^{-1}(X-\mu)) \end{aligned}$$

- Detección de casos anómalos:

$$\begin{aligned} p(X_t) &< \epsilon \longmapsto \text{Caso anómalo} \\ p(X_t) &\geq \epsilon \longmapsto \text{Caso normal} \end{aligned}$$

- Parámetros desconocidos:
  - $\epsilon \longmapsto$ Cross validation set mediante el uso del F1 Score (si se tienen valores).
- Consejos:
  - Es bueno tener o crear variables normales (sin normalizar). Para ello se pueden realizar transformaciones: $x_1 \longleftrightarrow \log(x_1 + c), x_1 = x_1^{\frac{1}{3}}$...
- F1 Score:

$$\begin{aligned} P &= \frac{True\ positives}{True\ positives + False\ positives}\ \text{(Precision)} \\ R &= \frac{True\ positives}{True\ positives + False\ negatives}\ \text{(Recuperation)} \\ F1\ Score &= 2\frac{PR}{P+R} \end{aligned}$$

2.8. Collaborative filtering

Collaborative filtering

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	?

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

Figura 15: Collaborative filtering

- Hypothesis:

$$h_\theta(x) = \theta^{(j)} x^{(i)} \text{ opinión del usuario j para la “pelicula” i}$$

- Cost Function:

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

- Parámetros desconocidos:
  - $\theta, x \longrightarrow$ Training Set (Automático).
  - $\lambda \longrightarrow$ No se elige.
- Consejos:
  - Se inicializa θ y x con valores aleatorios pequeños.
  - No existe θ_0 ni x_0.
  - Es aconsejable realizar la normalización.
  - Si una película no tiene suficientes calificaciones, es mejor no incluirla.
  - Si se buscan encontrar películas relacionadas se utiliza: $||x^{(i)} - x^{(j)}||$.

2.9. Online learning

- X: Parámetros del usuario.
- θ: Parámetros del modelo.
- Y: Acción del usuario.
- Process:

  $(X, Y) \longmapsto \theta$ (se actualiza) $\longmapsto (X, Y)$ (se desecha y se toma el del siguiente)

- Hypotesis y cost function:
  Logistic regression con un ejemplo.
- Consejos:
  - Las características a tener en cuenta se pueden obtener a partir de un sistema colaborativo.

3. Tratamiento de datos masivos

Recolectar datos masivos ayuda si con estos un experto sería capaz de sacar el resultado, es decir, si las características están relacionadas con la predicción que se quiere hacer. En general, es mejor tener más datos, sin embargo, el coste computacional es mayor.

Una solución para tratar con datos masivos sería reducir las dimensiones de las características (PCA). Aún así, hay varios ejemplos que pueden suponer un problema. En los modelos, el principal problema del coste computacional se encuentra en la minimización de la función coste, para lo cual se suele utilizar el método del gradiente. Por ejemplo para el método linear regression el método del gradiente es:

$$\begin{aligned} \theta_j &= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \\ j &= 1, 2, ..., n. \end{aligned}$$

Alternativamente existen dos variantes con menor coste coputacional:

- Stochastic Gradient Descent

  1. Ordenamiento aleatorio de los ejemplos.
  2. $$\begin{aligned} \theta_j &= \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \\ j &= 1, 2, ..., n. \end{aligned}$$
  3. Se hace para $i = 1, ..., m$.
  4. Se repiten los dos pasos anteriores entre 1-10 veces.

- Mini-Batch Gradient Descent

  1. Ordenamiento aleatorio de los ejemplos.
  2. Se considera b=2-100 (usual b=10)
  3. $$\begin{aligned} \theta_j &= \theta_j - \alpha \frac{1}{b} \sum_{k=1}^{i+(b-1)} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)} \\ j &= 1, 2, ..., n. \end{aligned}$$

4. Se hace para $i = 1, b+1, 2b+1, ...$
5. Se repiten los dos pasos anteriores entre 1-10 veces.

Además en los sumatorios de los métodos del "gradiente descensoz "mini-batch gradient descent"se puede realizar un procesamiento en paralelo por núcleos o dispositivos (Multi-Core Machine). Al final una maquina junta los anteriores resultados de la computación en paralelo.

3.1. Cost Function

La función coste en el caso de stochastic gradient descent es:

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h(x^{(i)}) - y^{(i)}) \text{ (se toma antes del ejemplo)}$$

Dibujando la función coste cada 1000 iteraciones (o más en caso de no observar una tendencia ascendente o descendente) obtenemos dos opciones:

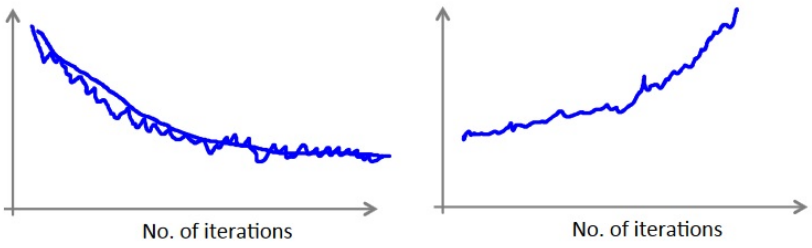

Figura 16: Collaborative filtering

Si la tendencia es descendente el algoritmo funciona bien. En otro caso, es necesario elegir un α más pequeño. Las variantes no convergen con un α fijo sino que se quedan cerca del mínimo. Si se quiere que estos converjan se toma lo siguiente:

$$\alpha \longmapsto \frac{\alpha}{cte_{iter} + cte_2}$$

A. Notación

- (X, Y): Training Set donde X son las características e Y son las respuestas correctas.
- J: Cost Function.
- θ: Parámetros de aprendizaje.
- $h_\theta(x)$: Hypothesis para el ejemplo $x = \{x_1, x_2, ..., x_n\}$.
- x_i: Característica i.
- $x^{(i)}$: Ejemplo i.
- $x_j^{(i)}$: Característica j del ejemplo i.
- $y^{(i)}$: Respuesta del ejemplo i.
- λ: Parámetro de regularización.
- m: Número de ejemplos del training set.
- n: Número de características.