.UBAfiuba
FACULTAD DE INGENIERÍA

Resumen Datos

Organización de datos, Curso Collinet - FIUBA

Grassano, Bruno	bgrassano@fi.uba.ar

Índice

1. Introducción

El presente archivo es un resumen que recopila el contenido dado en la materia Organización de datos, Curso Collinet 2C2020. Lo fui armando con los diferentes apuntes que tome durante las clases, contenido presentado durante las mismas, y en algunos casos busque algo mas de información para completar.

2. Análisis Exploratorio

Es un enfoque que comprende un conjunto de tareas para analizar conjuntos de datos, de forma tal de encontrar sus principales características.

Se comienza siempre formulando una pregunta interesante, se reúnen los datos, y se desarrolla el proceso necesario para poder responderla.

2.1. Feature Engineering

Forma a través de la cual se van tratando los datos. Comprende conversiones de tipos, missings, selección, creación de nuevas variables a partir de las que ya tenemos y reducciones dimensionales.

Good features allow a simple model to beat a complex model.
Peter Norving

2.1.1. Conversión

Los modelos entienden números, por lo que es necesario convertir los atributos a variables numéricas.

Variables categóricas

- One Hot Encoding: Por cada variable categórica crea una binaria. Tener cuidado con el problema de la colinealidad. Solución, crear para todos menos una la variable binaria. La ausencia de todas significa que se esta en presencia de la no codificada.
- Bin Counting Scheme: Convierte los valores categóricos en probabilidades. Útil para cuando se tiene muchas categorías.
- Hashing Trick: Mapea de un dominio muchísimo mayor a uno mas acotado. Muy bueno también para cuando se tienen muchas categorías. Se utiliza una funcion hash para conseguirlo.

Si se quiere conservar algun orden en particular, se puede utilizar Ordinal Encoder. Tener cuidado de que no se de el caso de agregar un orden donde no corresponda. *Ej. Azul <Rojo*

2.1.2. Missing values

Missing completely at random

En este caso la probabilidad de que un valor sea missing es la misma para todas las instancias. En este caso no depende de las medidas de otras variables. No tiene sentido intentar adivinar este caso. *Ej. Respuesta de una encuesta*

Missing at random

La probabilidad de missing depende de información que si tengo. *Ej. Edad relacionado con si una persona vive. Si no vive, y no tengo cargado el valor de edad ahí esta relacionado.*

Missing not at random

La probabilidad de missing esta relacionada con los valores perdidos. *Ej. La edad quizás prefieren no decirla, o el sueldo si es alto también.*

Soluciones

- Utilizar algoritmos que acepten datos faltantes.
- Eliminar los datos con problemas. ¿Filas? ¿Columnas? ¿Esta bien borrarlo? Depende siempre del caso.
- Convertir el valor faltante en una categoría. *Ej. No responde.*
- Rellenar el valor faltante con algún criterio. *Ej. Media, mediana, moda, etc* En estos casos puede convenir agregar una columna indicadora de que ese valor fue rellenado. Esto puede hacer que se le asigne menos peso al momento de mirarlo.

2.1.3. Selección

Esto busca reducir la dimensión mediante la eliminación de variables poco útiles. Favorece a la generalización y puede acelerar la velocidad del modelo mientras que mejora su interpretabilidad.

Técnicas

- Filter: Se realiza un ranking de cada variable generado a través de algún método estadístico. Se busca encontrar que variable afecta mas el valor a predecir.
- Wrapper: Encapsula la funcionalidad. Trata al problema de selección como una busqueda, ya que trata de conseguir la mejor combinación de variables. (Alto tiempo de computo). Tiene forward selection, backward selection, y random selection. Se puede utilizar también cualquier método que trabaje con problemas de combinatoria. *Ej Forward: Selecciono 1 al azar. Selecciono 2, ¿Mejoro?, selecciono mas y sigo preguntando.*
- Embedded: Rankea las variables segun métodos internos de cada algoritmo. Busca determinar cual es mas importante para predecir. *Ej. Lasso, Ridge, Elastic Net.* Ver 8.5

2.2. Visualización

- Para entender el trabajo y comunicarlo.
- Entender de forma mas rápida y eficiente.
- Comunicarlo de forma concisa y clara.
- Permite encontrar patrones y relaciones.
- Existe una amplia variedad de tipos de gráficos, cada uno con su objetivo.

Pagina con varios tipos de gráficos.

3. Clustering

Métodos de aprendizaje no supervisado. No se sabe que resultado esperar. Lo que hace es juntar los datos para ver como están relacionados.

Su objetivo es encontrar grupos de instancias similares entre si. Depende mucho de la naturaleza de los datos.

3.1. K-Means

Algoritmo que busca minimizar la ecuación de distorsión de clustering. Tengo los datos en p dimensiones.

Algoritmo

1. Particionar los datos en k dimensiones.
2. Elegir k centroides.
3. Asignar cada instancia al centroide mas cercano.
4. Repetir 2 y 3

Fortalezas

- Simple
- Eficiente
- Escala muy bien

Debilidades

- Especificar K
- Sensible a ruido
- Muy sensible a la elección de centroides iniciales.
- Solo encuentra clusters globales.

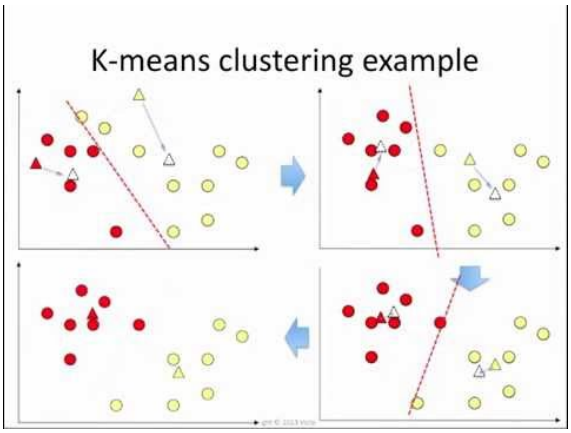


Figura 1: Ejemplo de K-Means, notar como se va moviendo el centroide.

3.2. Clustering Jerárquico Aglomerativo

Notas

- Cada instancia comienza en un cluster distinto.
- Cada instancia se va uniendo con el mas cercano.
- Bottom up

Fortalezas

- No tiene K
- Se reproduce una representación jerárquica (Dendograma)

Debilidades

- No busca optimizar una función
- Sensible a ruido

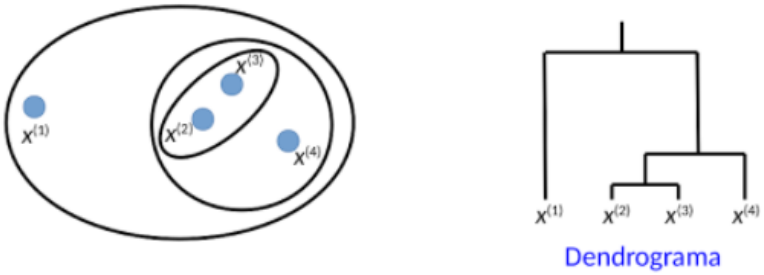


Figura 2: Ejemplo de Clustering Jerarquico Aglomerativo.

3.3. Clustering Jerárquico Divisivo

Similar al anterior en cuanto a procedimiento, solo que se comienza todo junto y se va dividiendo a medida que se va bajando.

3.4. DBSCAN

Se busca conseguir una mejora al ahorrar una cantidad de pasos en el algoritmo.

- **CORE POINTS** Son los puntos que tienen una densidad alta
- **BORDER POINTS** Vecinos de un core
- **NOISE POINTS** Ninguno de los otros dos

Fortalezas

- Robusto al ruido
- Clusters de forma arbitraria
- No tiene K

Debilidades

- Hay que elegir un ϵ y la cantidad mínima de puntos. Esto conlleva conocer los datos.
- Costoso en casos de alta dimensionalidad
- Funciona mal con datos de densidad variable

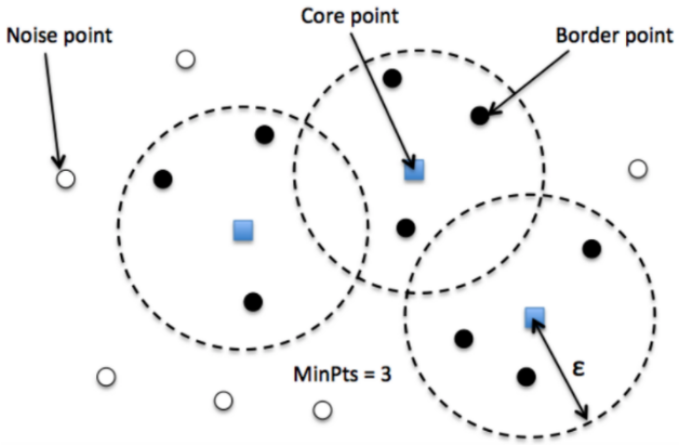


Figura 3: Ejemplo de DBSCAN.

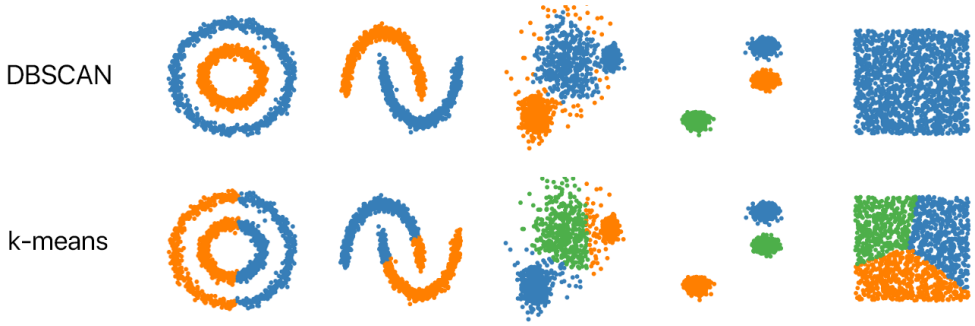


Figura 4: Comparación de la performance de DBSCAN con K-Means.

Algoritmo

1. Etiquetar cada punto como core, border o noise.
2. Eliminar los puntos noise.
3. Poner una arista entre los puntos que son vecinos entre si.
4. Los componenetes conexas son un cluster.
5. Asignar los puntos border.

3.5. HDBSCAN

Hereda del anterior. Es una mejora. Tiene un valor ϵ variable, densidad variable para evitar puntos que sean ruido, y mejora la velocidad.

3.6. Evaluación de clusters

3.6.1. Matriz de similitud

- Cuadrada y simétrica.
- Similitud entre cada par de instancias.
- El resultado es bueno si la matriz es diagonal en bloques.
- **Cohesión** Mide cuan relacionados están los elementos de un cluster
- **Separación** Cuan lejos están los distintos clusters entre si.

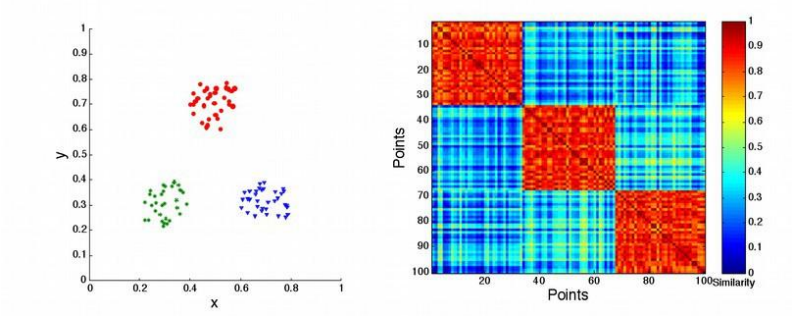


Figura 5: A la derecha se puede ver la matriz de similitud. Notar que es diagonal en bloques y los clusters estan bien clasificados.

3.6.2. Coheficiente de Silhouette

Mientras mas cercano al centro, mas cercano a 1. Se mueve de -1 a 1. Indica que tan bien definidos están los clusters.

$$s_i = \frac{b_i - a_i}{max\{b_i, a_i\}} \tag{1}$$

Siendo a la distancia promedio de x_i hacia otros puntos del mismo cluster, y b siendo la distancia promedio de x_i hacia los puntos del cluster mas cercano.

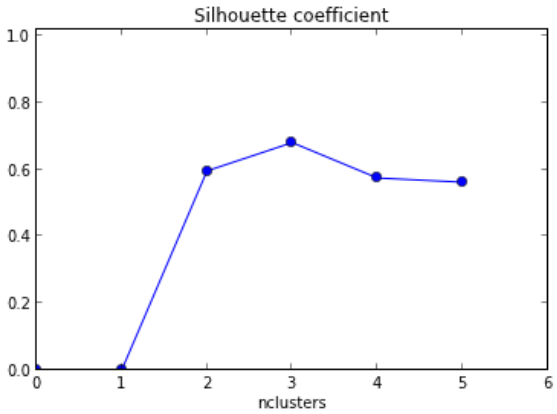


Figura 6: Grafico mostrando el avance con este coeficiente

3.6.3. Rand Index

Mide la similitud entre 2 resultados de clustering. Va de 0 a 1.

- a: son los pares que que están en un mismo cluster en X y en un mismo cluster en Y.
- b: pares que están en distintos clusters en X y en distintos Y.
- c: pares que están en un mismo cluster en X, pero en distintos en Y.
- d: pares que están en distintos clusters en X y en el mismo en Y.

$$R = \frac{a+b}{a+b+c+d} \tag{2}$$

3.7. Elbow Method

Método para encontrar el numero ideal de clusters. Se van graficando los resultados a medida que aumenta el K. Se termina eligiendo el numero a partir del cual no hay mejora.

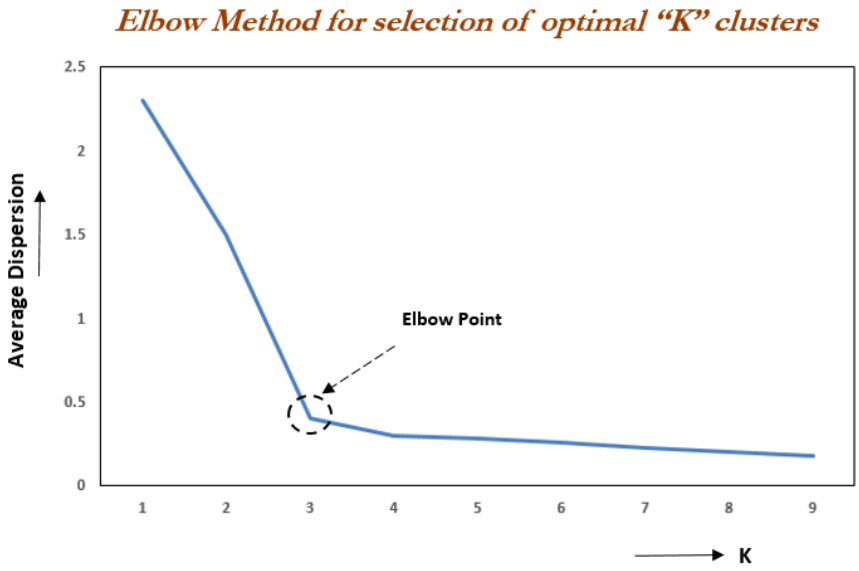


Figura 7: Método de Elbow. Se puede ver como a partir de 3 no hay una gran mejora.

4. Reducciones dimensionales

Son técnicas que sirven para cuando tenemos una alta dimensionalidad de los datos. Con ellas buscamos proyectar esos datos en un espacio de dimensión menor sin perder información.

Algunas reducciones posibles son:

- Seleccionando variables
- Transformando variables

¿Por que hacerlo?

- Sirve para visualizar mejor el espacio
- Puede ayudar a reducir el ruido
- Regulariza los datos
- Comprime la información
- Reduce el computo de los modelos

4.1. PCA

Dada una variable $X = (x_1, ..., x_n)$, buscamos un conjunto de proyecciones lineales ortogonales de X, donde las proyecciones estén ordenadas de forma decreciente según su varianza.

Notas

- La varianza es una medida que cuantifica la cantidad de información que se tiene.
- El resultado provoca una rotación o cambio del sistema de coordenadas.
- El método busca rotar el dataset de manera de maximizar la varianza de los datos en proyecciones ortogonales. Para esto asume que los datos se encuentran en un subespacio lineal de dimensión menor a la original.
- La dirección en que se maximiza es en la del autovector μ de $\sum$ (matriz de covarianza de los datos) cuyo autovalor λ asociado es mayor.
- La reducción PCA es única. Da siempre el mismo resultado. (Determinista)
- La proyección de X en k dimensiones esta dado en la matriz $Z = X * V$. Su reconstrucción es con $X = Z * V^t$
- Si la varianza en V es menor a 1, se perdieron datos y la reconstrucción no sera perfecta.
- *Si todos los valores son iguales, (Var = 0) esa columna no aporta mucho*

Fortalezas

- No tiene hiperparámetros
- No tiene iteraciones
- Sin óptimos locales.

Debilidades

- Limitado a proyecciones lineales.

Gif mostrando la conversión aquí.

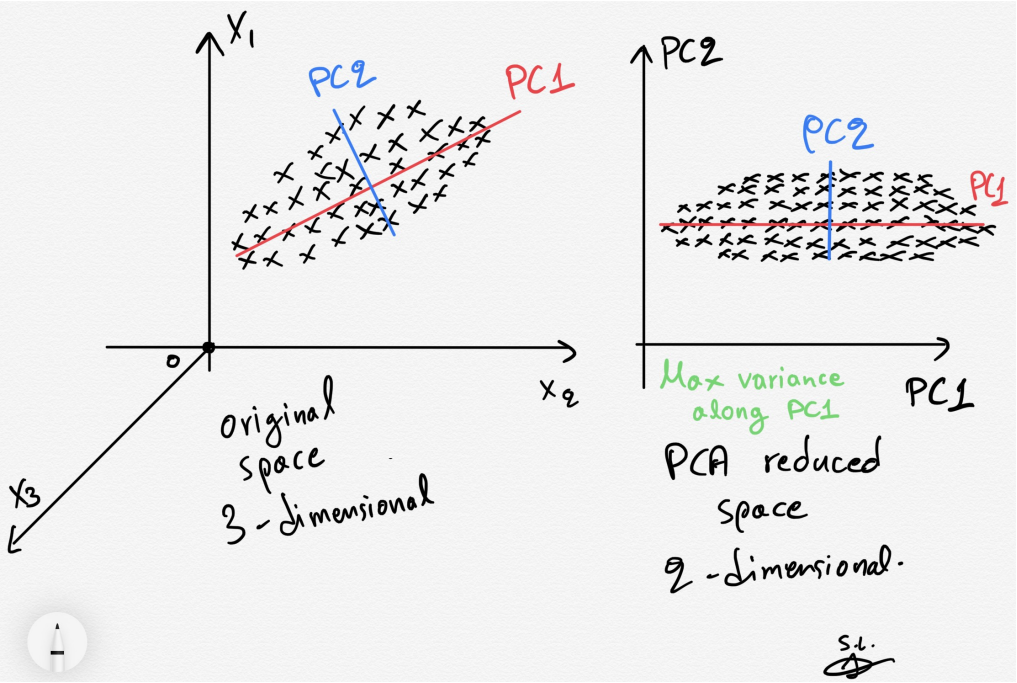


Figura 8: Conversión utilizando PCA

4.2. MDS

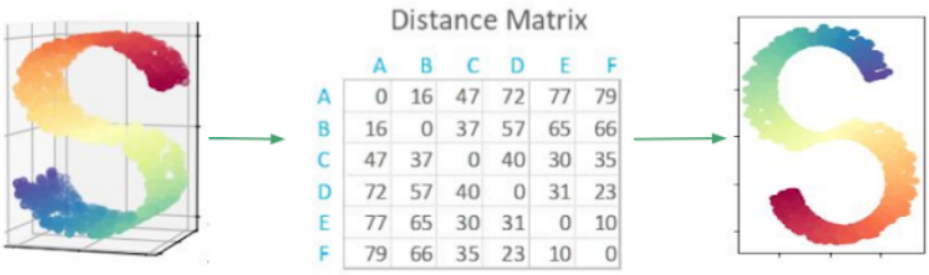
Es un método que consiste en preservar la distancia entre los puntos. Intenta ubicarlos en una dimensión menor tal que la distancia se parezca lo mas posible.

Fortalezas

- Soporta varios tipos de distancia.
- Permite transformaciones no lineales.

Debilidades

- Optimización iterativa con mínimos locales.
- Difícil determinar que distancia usar.



	A	B	C	D	E	F
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

Figura 9: Conversión utilizando MDS

4.3. ISOMAP

Similar al anterior en cuanto a preservar la distancia. Se diferencia en que este busca preservar la geométrica utilizando distancia geodésica.

Algoritmo

1. Determinar vecinos mas cercanos.
2. Construir grafo de vecinos.
3. Computar el camino mínimo *Ej. Dijsktra.*
4. Generar matriz de distancias en base al anterior punto, y aplicar MDS.

Fortalezas

- Mantiene la estructura.
- Permite transformaciones no lineales.

Debilidades

- Determinar un k de vecinos.
- Sensible a ruido.
- Muchos vecinos puede hacer que se rompa la distancia geodésica y de algo mas como MDS.

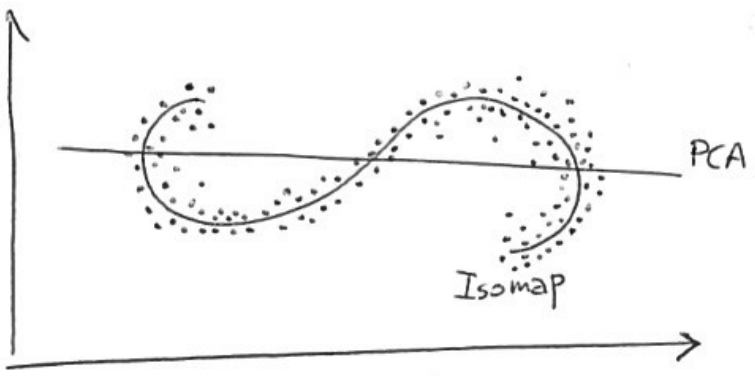


Figura 10: Conversión utilizando ISOMAP.

4.4. t-SNE

La idea detrás de este método es ir de $\mathbb{R}^n \Rightarrow \mathbb{R}^2$ respetando lo mejor posible los puntos de $\mathbb{R}^n$. *Ej. Conservar los grupos como estaban.*

Notas

- Se usan las distancias para construir una matriz de probabilidad de ser vecino. mientras mas grande, mas probable es que se este cerca.
- Tiene un parámetro, *perplexity*, que mientras mas grande mas vecino se es.
- Se utiliza el descenso del gradiente.
- No tiene sentido en *Machine Learning*.

Algoritmo

1. Iniciar los puntos al azar.
2. Matriz de probabilidad en X, Y.
3. Movemos los puntos de Y hasta que se parezcan lo mas posible.

Fortalezas

- Muy bueno para visualizar datos.
- Conserva estructuras no lineales.

Debilidades

- Es estocástico (al azar).
- Escala en tiempo con dimensiones y puntos.
- No sirve para nuevos puntos

5. Aprendizaje supervisado

5.1. Arboles de decisión

- Sencillo y fácil de comprender.
- Sirven como *baseline* trivial.
- Sirven como ejemplo para entender la estructura y funcionamiento de otros algoritmos.
- Cada nodo evalúa un atributo.
- El nodo tiene tantos hijos como posibles atributos tenga.
- Si las instancias están lo suficientemente bien clasificadas termino (Se vuelven hojas).

¿Cual es el mejor atributo?

Lo podemos medir con distintos indicadores.

- **Pureza de Gini** Se elige el que mas reduce la impureza.
  $Gini(S) = 1 - \sum_{c \in Clases}(\frac{|S_c|}{|S|})^2$
  Siendo S_c las instancias de la clase c.
  $GiniGain(S, A) = Gini(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} \cdot Gini(S_v)$
- **Ganancia de información** Se busca maximizar la ganancia.
  Entropía de una muestra: $H(S) = \sum_{c \in Clases} -p_c \cdot log_2 p_c$
  Siendo p_c la proporción de instancias S pertenecientes a C.
  $InfoGain(S, A) = H(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} H(S_v)$

Ambos criterios son numéricamente similares.

Fortalezas

- Interpretabilidad
- Similar a como decide el humano
- Acepta varios tipos de input
- Maneja bien los missing values

Debilidades

- Llegan a menos precisión
- Varianza alta

Overfitting

Sucede cuando se 'aprendió de memoria' los datos. En el caso de los arboles se puede ver cuando estos se vuelven muy profundos, esto puede llegar al caso de que en cada hoja haya incluso un solo dato. Como consecuencia de esto, el modelo generalizaría mal.

Las soluciones en arboles son, parar a partir de cierta profundidad, o realizar una poda (Pruning) (sacar ramas cuando mejore la performance). Otra opción es Post-Pruning.

5.2. KNN

Este modelo busca predecir la instancia de un nuevo punto a partir de sus vecinos.

- Mientras mas cerca a un punto, mas peso deben de tener. Ponderar sobre la distancia.
- Si se toman Ks mas grandes, se empieza a suavizar la variación.
- Se rompe mientras mas dimensionalidad se tenga.
- Es un modelo simple.
- El entrenamiento es muy rápido, ya que consiste en agarrar los puntos y ver solamente.
- La contraparte es que la consulta se vuelve lenta.
- También como funciona el modelo hace que se ocupe mucho espacio en disco.
- Es susceptible a escalas.

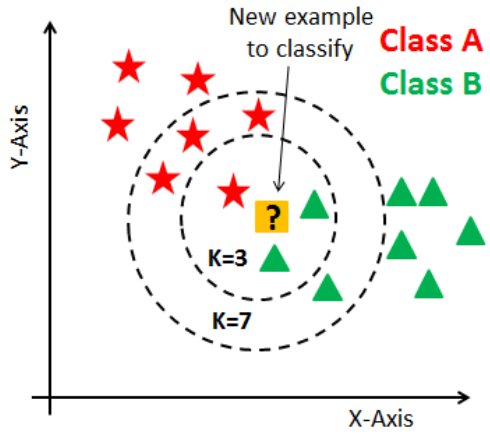


Figura 11: Ejemplo de KNN.

5.3. Naive Bayes

Este es un modelo que se basa en el calculo de probabilidades utilizando la teoría de probabilidad bayesiana. Asume que todos los sucesos son independientes para hacerlo.

- Cuenta la cantidad de veces que aparece cierta característica básicamente.
- Tiene de hiper-parámetro el smoothing (suavizado) que evita que haya casos con probabilidad 0.
- No tiene en cuenta el orden. Si se quiere, se pueden utilizar n-gramas.

5.4. SVM

Modelo que busca que una recta/hiper-plano separe las clases de la mejor forma posible. Busca esto sin tener que modelar la distribución de los datos de cada clase.

- Es iterativo, puede escalar en tiempo.
- Va a converger, no hay óptimos locales.
- Las instancias mas cercanas se vuelven 'Support Vectors'
- De hiper-parámetro tiene M, que determina el margen, es decir, el espacio entre los distintos tipos de puntos. Busca el caso que maximice esto.
- Si no encontró una recta/hiper-plano, los puntos no son linealmente separables.
- Se consigue la recta creando una nueva dimensión. Como esto es costoso se puede utilizar Kernel Trick, que hace que se 'piense que esta en otra dimensión'. *Ej. lineal, polinomial, sigmoideo, rbf*

Hard Margin vs Soft Margin

En Hard Margin los puntos están perfectamente separados. Esto hace que el modelo no funciones con outliers y ruido.

En Soft Margin se permite algo de ruido y outliers.

Este hiper-parámetro se controla con el C. A mayor C sae tiene HM, a menor SM.

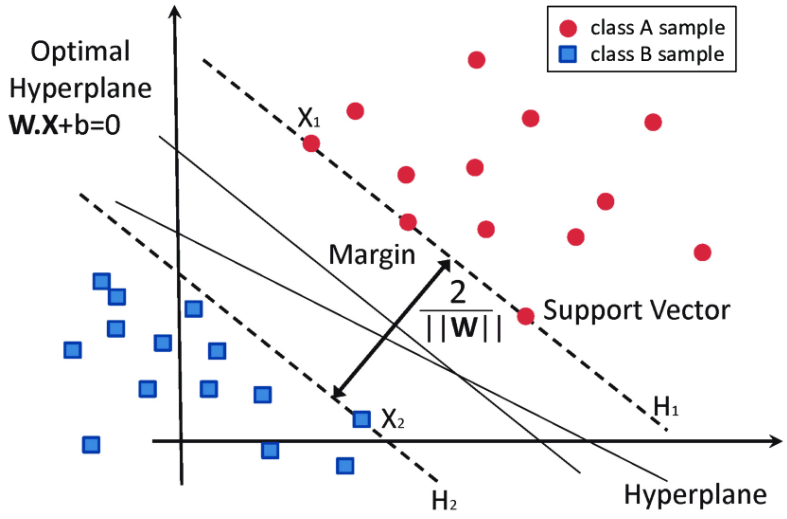


Figura 12: Ejemplo de SVM.

6. Evaluación y selección

La idea es evaluar sobre datos que no se hayan usado en el entrenamiento. Para ello, nos guardamos una parte de los mismos para usarlos mas adelante. (10 % a 25 %)

También se cuenta con una parte que se conoce como holdout, que es para evaluar cuando el modelo esta por salir a producción.

Algo a tener en cuenta, es evitar los *data leaks*, esto sucede cuando parte de lo que tenes en tu parte de entrenamiento aparece en la de evaluación.

6.1. ¿Como hacer el particionado?

El particionado tiene que ser random. No debe de agarrar las primeras x instancias y listo. Se le puede pedir cuando se esta partiendo que respete la distribución de las clases. Muy útil para cuando se tienen casos muy des-balanceados. *Ej. Clase positivo es el 10 % y el resto negativo.*

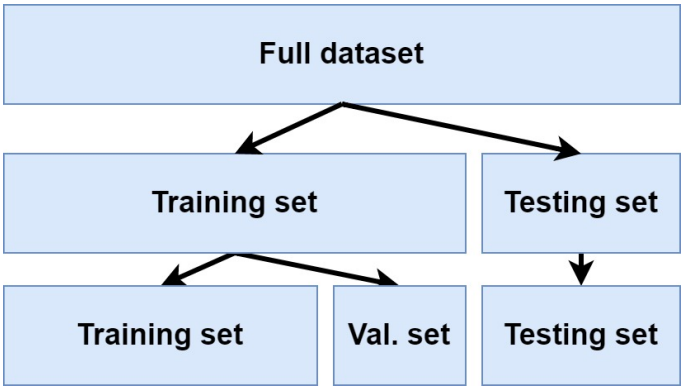


Figura 13: Division de los datos

6.2. Crossvalidation K-Fold

Consiste en dividir varias veces los datos, e ir cambiando el conjunto. Esto hace que se disminuya el riesgo de tener una mala partición.

Algoritmo

1. Desordenar los datos
2. Separar en k folds del mismo tamaño
3. De $i = 1, ..k$, entrenar con todos menos i. Después evaluar con i

Nota: Stratified K-fold mantiene las proporciones.

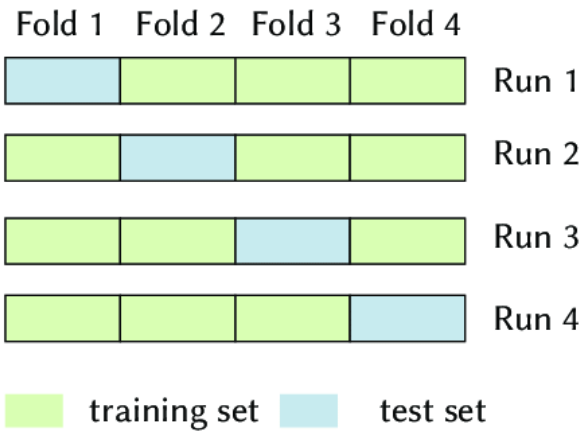


Figura 14: Funcionamiento de CV K-fold

6.3. Grid Search

Es un método con el cual se buscan los mejores hiper-parámetros de cada modelo. Escala mucho en tiempo si no se paraleliza, esto se debe a que el algoritmo consiste en ir agregando *for* por cada hiper-parámetro. Una alternativa es ir eligiendo aleatoriamente (Random Search).

```
best_score = None
best_hiperparams = None
for h1 in valores_hiperparametro_1:
    for h2 in valores_hiperparametro_2:
          ...
          for hn in valores_hiperparametro_n:
              kf = StratifiedKFold(n_splits=5)
              metrics = []
              for fold_idx, (train_index, test_index) in enumerate(kf.split(X, y)):
                  clf = Clasificador(h1, h2, ..., hn)
                  clf.fit(X[train_index], y[train_index])
                  metrics.append(metric(y[test_index], clf.predict(X[test_index])))
              if not best_score or np.mean(metrics) < best_score:
                  best_score = np.mean(metrics)
                  best_hiperparams = [h1, h2, ..., hn]
```

Código genérico de grid search en Python.

6.4. Métricas

- Accuracy: Evalúa directamente a cuantos le pegamos.
- Matriz de confusión: Ver imagen 15
- Precision: No le interesa si perdemos alguno, quiere saber a cuantos le pegamos de los que decimos que son.
- Recall: Nos importa la mayor cantidad de casos que predecimos reales.
- F1-Score: Permite tener un único numero para evaluar un modelo.

- Curva ROC: Es un gráfico que me dice de forma visual que tan buen modelo tengo. Ver imagen 16
- AUC-ROC: Valor que surge de medir el área bajo la curva ROC.
- Hay otros como True Positivity Rate o el True Negative Rate.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

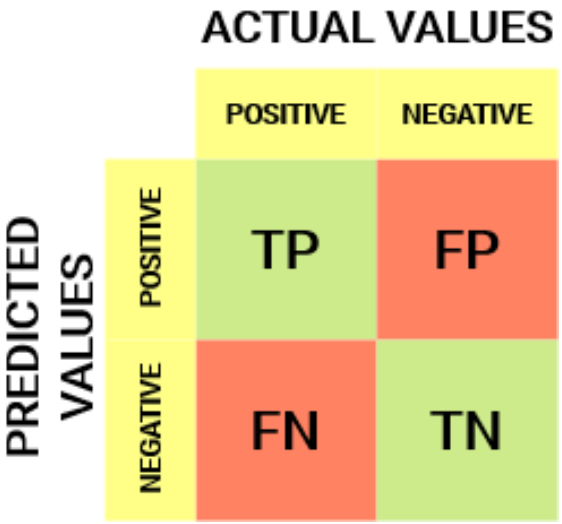


Figura 15: Matriz de confusión

Para encontrar la mejor combinación, hay que explorar el espacio de posibles combinaciones, usando idealmente *K-fold* para medir el desempeño.

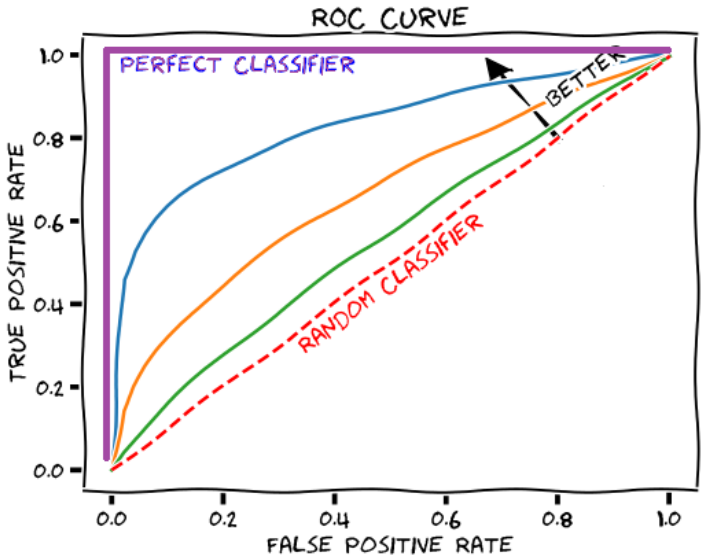


Figura 16: Gráfico mostrando posibles casos de ROC Curve

7. Ensambles

Un ensamble es una unión de muchos modelos juntos haciendo predicciones sobre un problema. Cada modelo ajustaría de forma distinta, agarrando una parte del concepto que se busca. Esto causa que se tenga un bajo sesgo y se reduzca la varianza.

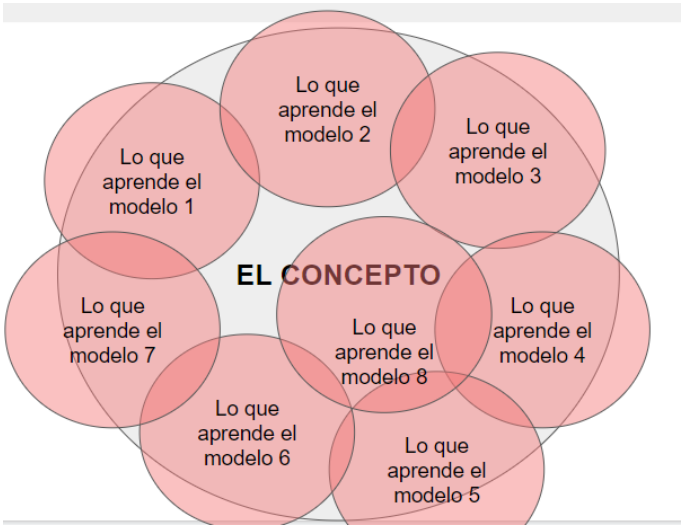


Figura 17: Funcionamiento de los ensambles

7.1. Bagging

Ensamble que funciona construyendo nuevos conjuntos de entrenamiento usando bootstrap. Realiza un muestreo con remplazo de las instancias.

La tasa de error reduce con el numero de clasificadores. Mientras mas tenga mas me voy a estar acercando al concepto. No suele aumentar el sobre-ajuste en este caso.

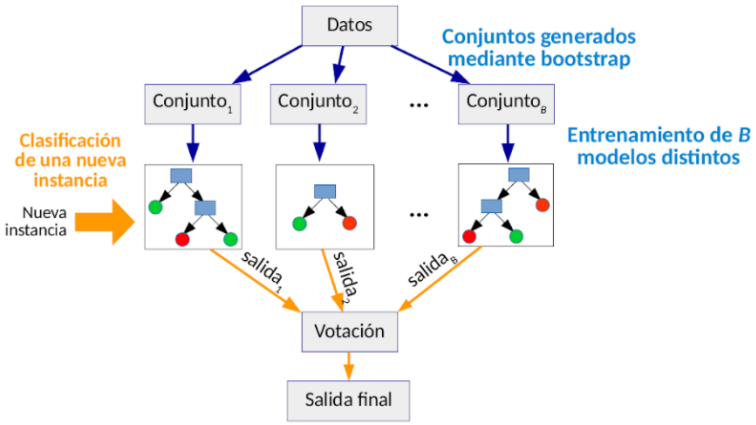


Figura 18: Ej. Bagging

Problema: Si se utiliza con arboles, y se tienen pocos atributos importantes, los arboles se van a parecer entre si. Por eso conviene Random Forest en ese caso.

7.2. Random Forest

- Igual a bagging, pero en cada nodo solo hay un subconjunto de atributos elegidos al azar.
- Es bueno para selección de features.
- Al aumentar la cantidad de arboles no aumenta el overfitting.

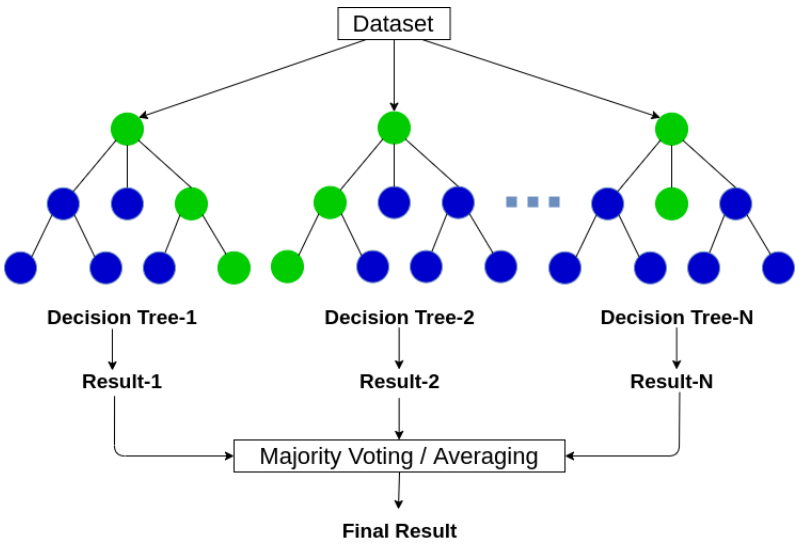


Figura 19: Forma de Random Forest

7.3. Boosting

- El sesgo baja en este caso, cada modelo mira mas los datos.
- Se comienza con un modelo simple entrenado sobre todos los datos.

- En cada iteración se entrena dando mayor importancia a los datos mal clasificados por las iteraciones anteriores.
- Puede sobre ajustar.
- Algunas implementaciones son AdaBoost, GradientBoost, XGBoost.

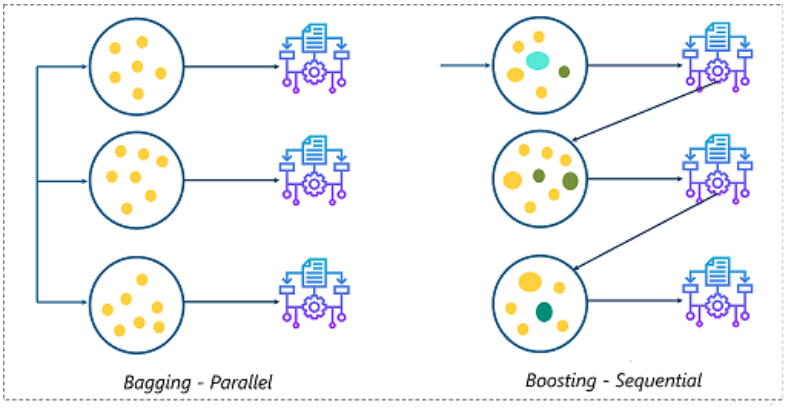


Figura 20: Comparación Boosting y Bagging

7.4. Híbridos

7.4.1. Voting

Consiste en construir N modelos usando los mismos datos, y luego tomar la predicción mayoritaria.

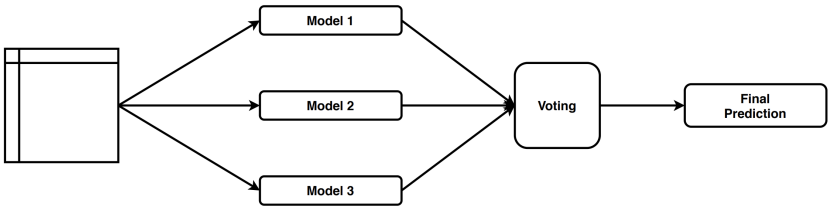


Figura 21: Sistema de Voting

7.4.2. Stacking

Entrenar distintos modelos base, y uno mas que al final recibe las predicciones realizadas. Este ensamble es una mejora del método de votación. Este agregado que remplaza el sistema de votación es de meta-aprendizaje. Generalmente son arboles, NB, SVM, o Perceptron para el modelo final.

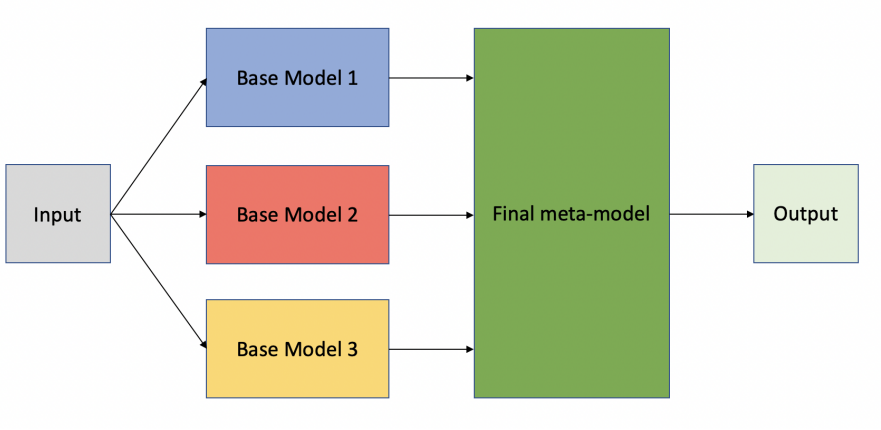


Figura 22: Sistema de Stacking

7.4.3. Cascading

En este ensamble se pasan sucesivamente los datos de un modelo a otro. Cada capa tendría un solo modelo, el cual se entrenaría sobre las instancias con baja certeza del modelo anterior. Se suele usar cuando se necesita una alta certeza de la predicción. *Ej. Robo de tarjeta de crédito*

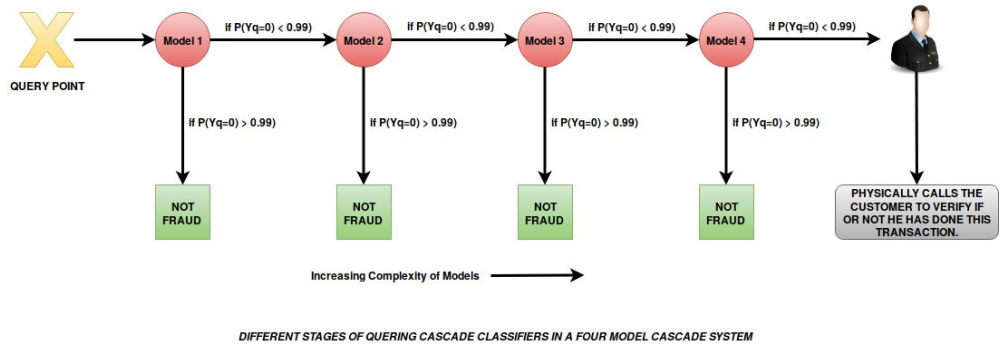


Figura 23: Ejemplo de Cascading

Nota: Cascadas muy profundas pueden producir overfitting.

8. Regresiones

Cada instancia tiene un valor numérico, se quieren predecir una o mas respuestas a partir de varios atributos. (la funcion es desconocida.)

$$Y \approx f(X_1, X_2, ..., X_n) \tag{6}$$

Para evaluarlo tenemos un rango de incerteza.

Error cuadrático medio

$$MSE = \frac{1}{n}\sum^{n}(y^i - \hat{y}^i)^2 \tag{7}$$

8.1. Arboles de regresión

- En cada nodo usar reducción del desvío estándar.
- Al llegar a una hoja se devuelve el promedio de las Y sobre las instancias que tenga.

8.2. KNN de regresión

Dada una instancia, se devuelve el promedio ponderado de los valores de sus vecinos. (Z)

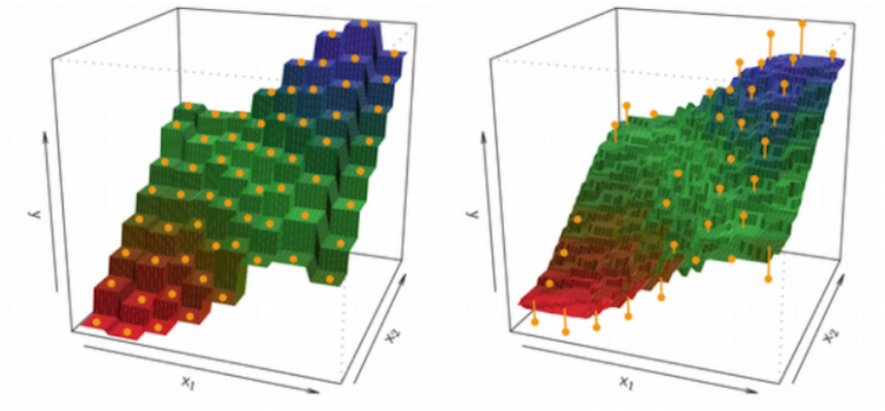


Figura 24: Ejemplo de KNN en regresiones.

8.3. Regresión lineal

Se supone una relación lineal con una variable y se procede en la cuenta. Cuadrados mínimos es una variante.

$$y \approx \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{8}$$

Para estimar los coeficientes, se igualan a 0 las derivadas. Si algún coeficiente da 0, significa que esa feature no tiene importancia para lo que se quiere predecir.

Si hay error, existe la posibilidad de que sea debido a que la relación no es lineal.

Residual Sum of Squares

$$RSS = \sum^{n}(y^i - \hat{y}^i)^2 \tag{9}$$

8.4. Residual Plot

Forma de observar los residuos que nos da el modelo. Si da algo que no es razonable, dudar de que sea lineal. El gráfico nos tiene que dar una nube de puntos. No se tienen que ver relaciones.

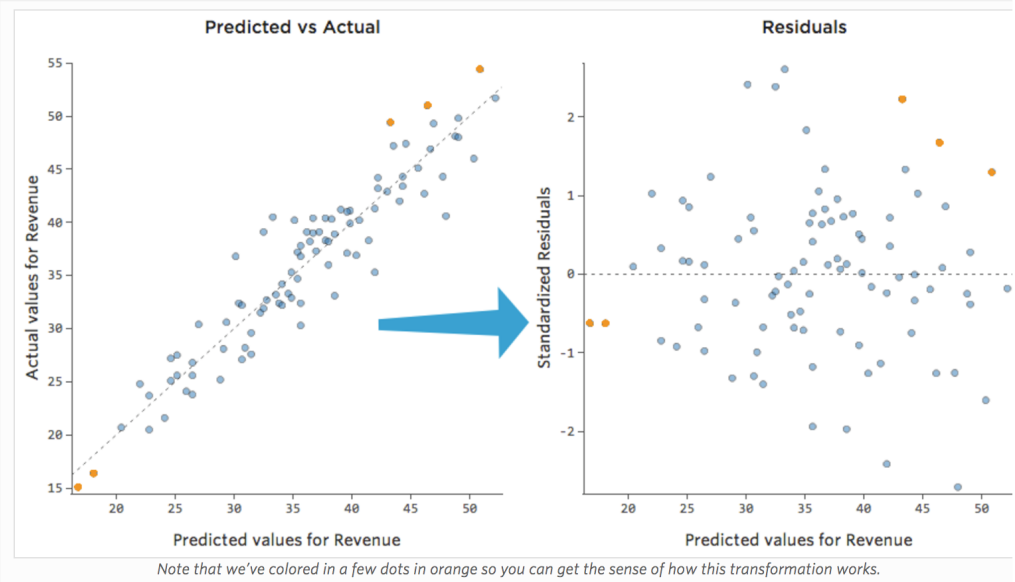


Figura 25: Ejemplo mostrando un gráfico de residuos. Notar que se ve que es como una nube de puntos. No se nota algún patrón o estructura en particular.

8.5. Regularización

- Técnica para evitar el overfitting.
- Busca hacer mas homogéneos los modelos.
- Reduce varianza pero aumenta el sesgo de los estimadores.
- Hay que estandarizar las variables.

$$Z = \frac{X_i - \mu}{\sigma} \tag{10}$$

Causas de overfitting

- Features irrelevantes: No tienen relacion con el target. *Tener cuidado cuando el numero de caracteristicas esta cerca del de observaciones.*
- Features correlacionados: El modelo se hace asumiendo independencia.
- Coeficientes muy grandes: A mayor valor absoluto, mas puede cambiar la respuesta prevista.

Debido a estos problemas, se busca restringir a los coeficientes. Se consigue modificando la función de perdida.

$$RSS = \sum^{n}(y^i - \hat{y}^i)^2 + \lambda \sum^{M} |\hat{\beta}_j|^q \tag{11}$$

8.5.1. Ridge

Se tiene cuando q = 2. Hace que el sesgo crezca y la varianza baje a medida que se aumente λ.
Teorema: Siempre existe un valor de λ tal que el MSE de Ridge sea menor que el MSE de mínimos cuadrados.

8.5.2. Lasso

Es el caso en que q = 1. Esta regularización puede llegar a anular coeficientes, por lo que puede usarse como selección de variables.

8.5.3. Elastic Net

Es una combinación de Lasso y Ridge. Agrega ambos términos a la cuenta, haciendo que cada uno aporte lo valioso que tiene.

8.6. Robustez

Outlier

- Observación que se desvía tanto de las otras observaciones como para levantar sospechas de que fue generada por otro mecanismo.
- En algunos casos nos puede interesar encontrarlo. *Ej. Robo de tarjeta de crédito*
- En otros casos es simplemente molesto.

Para agregarlo hay que cambiar la función de perdida.

8.6.1. L1

Remover el cuadrado y usar directamente el modulo. Lo malo de este caso es que no es bueno para derivarla.

8.6.2. M estimadores

$$\hat{\beta} = argmin_{\beta} \sum^{n} \rho(\frac{y_i - \beta_0 - \sum^{p} \beta_j \cdot X_{ij}}{\hat{\sigma}}) \tag{12}$$

Para ρ se usan distintas funciones. *Ej. Cuadratica, Huber, Bicuadrada*
Para β se deriva nuestra función de perdida y se ve que β la igualan a 0.
Esto hace que los outliers tengan poca influencia en el modelo.

8.6.3. RANSAC

Forma mas algorítmica. Se seleccionan n valores random, se hace un ajuste de regresión con esos valores, y con esto se evalúa cuales puntos restantes pueden pertenecer con una distancia 'd' (hay que definir también la cantidad mínima de puntos internos para considerarlo un buen modelo). Este algoritmo es iterativo, sirve mucho para cuando hay muchos outliers o ruido.
Gif mostrando el algoritmo en funcionamiento aquí.

8.6.4. THEIL-SEN

Otra forma algorítmica. Se toman la mediana de las pendientes calculadas de a pares y me quedo con esa, despues es repetir de forma similar para la ordenada al origen. Es 'insensitivo' a outliers. No depende de hiper-parámetros.

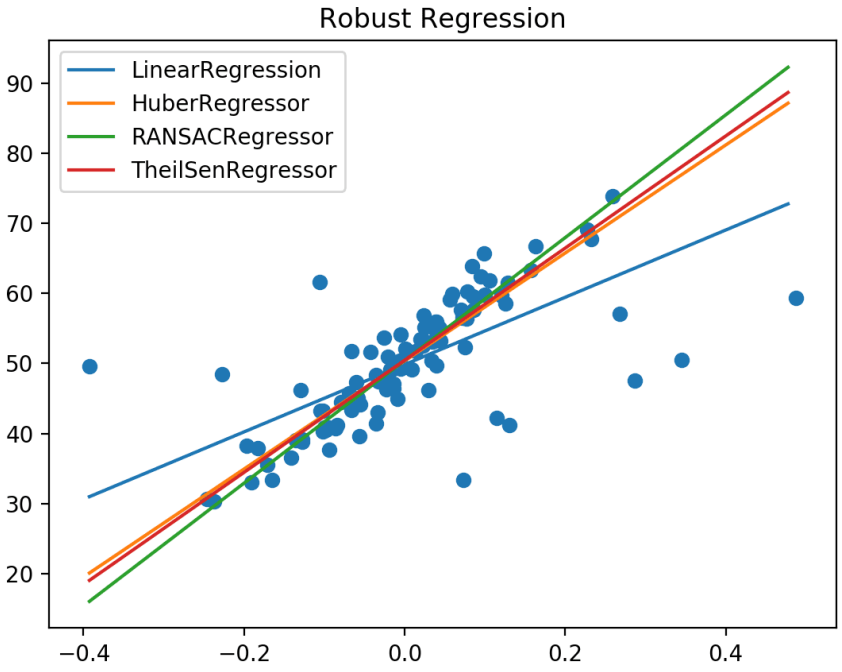


Figura 26: Diferentes tipos de regresiones. Notar como la lineal (sin ser robusta) se desvía de lo esperado.

9. Redes Neuronales

- Basado en neuronas con conexiones con distinta intensidad.
- Se activa una neurona cuando recibe suficiente estimulo de las conexiones entrantes.
- El primer modelo que surgió fue el Perceptron, a partir de varias entradas daba una salida con una función de activación.
- Tienen también una variable Bias, la cual indica que tan fácil es que devuelva 1 la neurona.
- El Perceptron funcionaba para casos lineales. *Ej. OR, AND.*
- Unir varios Perceptron hace que actúen como si fuera uno solo.
- La solución, fue utilizar una función de activación no lineal. *Función Sigmoidea.* Esta función es aplicada al resultado de la ecuación.

Para entrenar una red era necesario poder ajustar los pesos y bias. Esto se hace encontrando los mejores valores de cada uno, y se va consiguiendo a medida que se va reduciendo una función de costo. *Ej. funciones de costo: MSE para regresión, Binaria para clasificación.* Gracias a esta función de costo, ahora la red sabe cuando se equivoco.

Le falta saber como actualizar los pesos. Esto se consigue utilizando el descenso del gradiente, que apunta a la dirección en la cual la función crece mas rápido. Esta velocidad con la que cambia se conoce como *learning rate.* Si es muy alto puede ir hacia algún mínimo local, si es bajo puede no moverse.

Para cuando se tiene mas de una capa, se utiliza Back Propagation, ya que una neurona depende de los valores de las activaciones de las neuronas de la capa anterior. Con esto se busca modificar las activaciones anteriores para conseguir minimizar aun mas la función de costo.

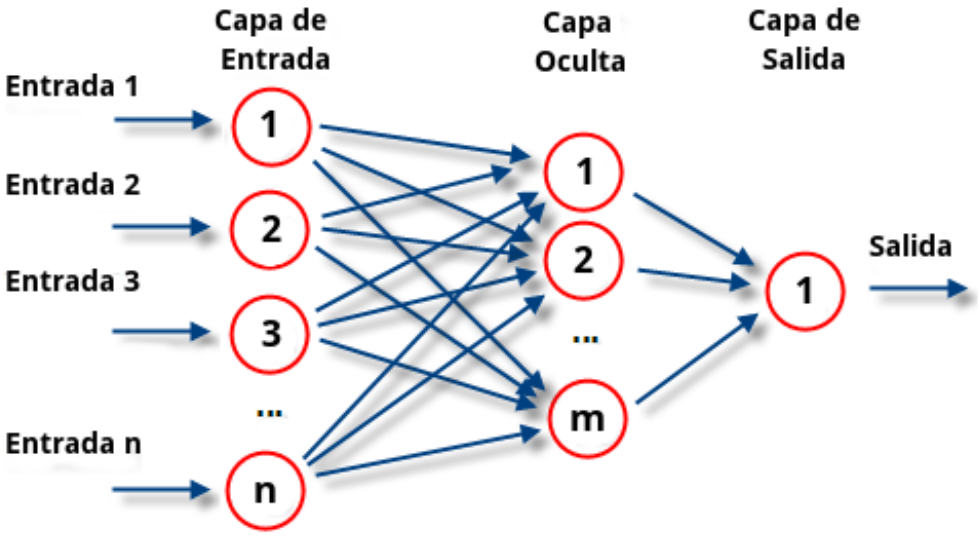


Figura 27: Modelo de red neuronal.

¿Que necesita la red?

- Son necesarias tantas neuronas de salida como cantidad de clases a predecir.
- La arquitectura, la cantidad de neuronas y la cantidad de capas.
- Funciones de activación. *Ej. Sigmoidea, RELU, Softmax*
- Optimizadores que determinen como modificar los pesos.*SGD (descenso del gradiente), RMS-Prop, Adam.*
- Una función de costo que se va a estar minimizando.*MSE, MAE, Cross Entropy.*(Depende del problema)

Todas estas configuraciones dependen del problema.

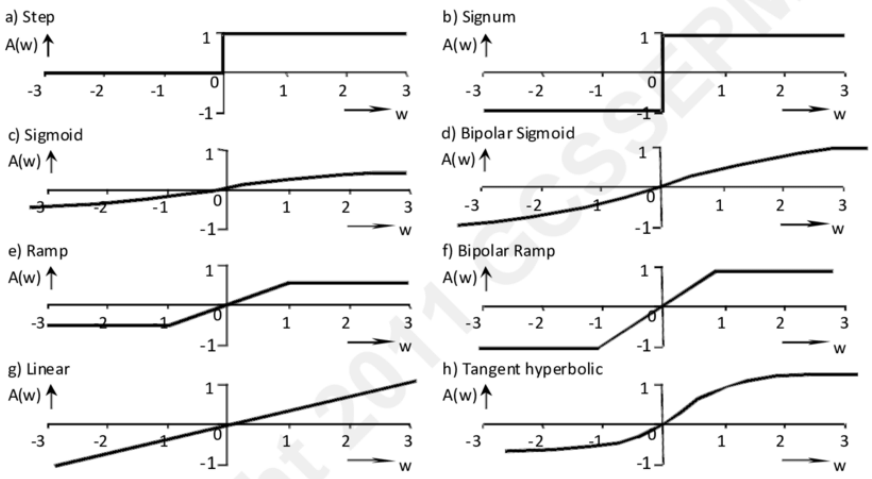


Figura 28: Diferentes funciones de activación.

Métodos de regularización

- L1, L2, Elastic Net: Penalizan el valor de los pesos de la red.
- Dropout: Apaga activaciones aleatoriamente durante el entrenamiento. Esto evita que la red no dependa de unas pocas neuronas.
- Early Stopping: Evita el sobreajuste parando el entrenamiento a partir de cierto tiempo.
- Data Augmentation: Agrega datos usando los que ya se tienen. (Aplica transformaciones). *Ej. Imagen de un gato, se rota, sigue siendo un gato.*

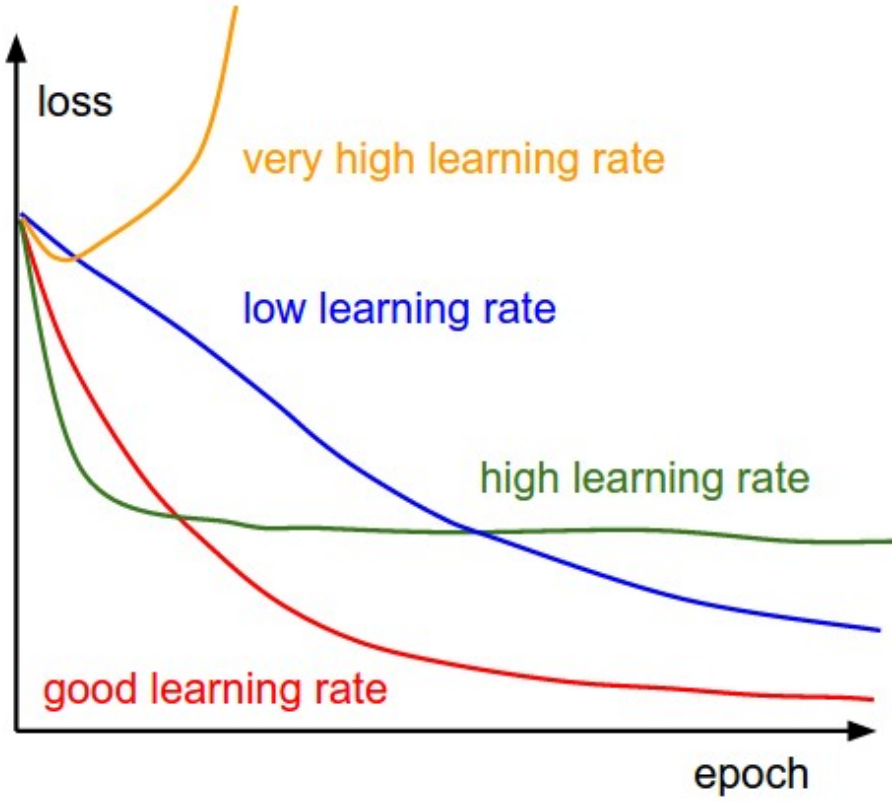


Figura 29: Ejemplos de entrenamiento de una red.

10. PLN : Procesamiento del Lenguaje Natural

Área enfocada a procesar y entender el lenguaje humano. Tiene un montón de usos. *Ej. Traducciones, modelado de tópicos, clasificación de texto, generador de texto...*

10.1. Enfoques

- Basado en reglas. Busca expresiones regulares, trata de encontrar si un texto cumple con determinadas características.
- Modelos probabilísticos. Son clasificadores lineales. Maximizan el likelihood
- Deep Learning. Son redes neuronales recurrentes o convolucionales.

10.2. Semantic Slot Filling

Es una tecnica que asigna categorías semánticas a distintas partes de un texto. Permite que el programa entienda mejor que hacer. Una forma de encararlo es con gramáticas libres de contexto.*Ej. Puede ser Show me..., I want..., Can i see...* .

Los problemas con esta técnica son:

- Que debe de hacerse manualmente y hay que escribir todas las reglas. Estas son definidas por el humano.
- Como funciona la técnica, se consigue un presicion alto pero un recall bajo.
- Es necesario un corpus de entrenamiento que tenga asignado para cada set de palabras su correspondiente categoría semántica.
- Requiere de un proceso de feature engineering, ya que hay que preguntarse si las palabras están en mayúscula, cual es el siguiente slot, ...
- En Deep Learning hay que usar un corpus de datos aun mas grandes.

10.3. Pre-procesamiento

Tokenización

El texto tiene distintas formas de verse, como caracteres, palabras, frases, oraciones,...

Para hacer esto, podemos utilizar 'Tokens' *Ej. Espacios, comas, cualquier unidad útil para separar*. En otros lenguajes esto se queda corto.

Normalización

Se busca reducir la dimensionalidad. *Ej. Lobos puede pasar a ser Lobo directamente, no nos interesaría el plural.*

Para esto tenemos 2 formas distintas de trabajarlo.

- Stemming: Elimina sufijos y prefijos. Trata de llevar la palabra a la raiz. *Ej.*
  - *Feet* $\Rightarrow$ *Feett*
  - *Wolves* $\Rightarrow$ *Wolv*
  - *Talked* $\Rightarrow$ *Talk*
- Lematización: Realiza un análisis morfológico y de vocabulario. *Ej.*
  - *Feet* $\Rightarrow$ *Foot*
  - *Wolves* $\Rightarrow$ *Wolf*
  - *Talked* $\Rightarrow$ *Talk*

Extracción de información

Busca organizar la información de manera útil. Distribuirla de forma precisa para el modelo.

Name Entity Recognition

Busca reconocer y clasificar el texto en categorias.
Ej. Ana se recibió de ingeniería informática en la UBA.

Colocaciones

Son frases compuestas por mas de una palabra. *Ej. Buenos Aires*

10.4. Feature Extraction

Bag of Words

Cuenta la cantidad de ocurrencias de un token en el texto. El objetivo que tiene es buscar palabras distintivas. *Ej. Excelente, buenísimo,...*

Por cada token se crea una columna de características y se va contando.

Los problemas con esta forma, es que se pierde el orden, que los contadores no están normalizados, y que se tiene una alta dimensionalidad.

Una posible solución es contar de a pares de tokens, pero esto hace que crezca aun mas la dimensionalidad.

Soluciones

Eliminar N-Gramas. Estos serian los de alta frecuencia y los de baja. *Ej. y, o, algún nombre.* El objetivo es quedarse con las palabras que tienen una frecuencia media.

TF-IDF

Se puede usar para una mejora de Bag of Words remplazando a los contadores. Al aplicarlo habria que normalizar cada fila.

- $TF(t, d)$ - Cuenta la frecuencia de un termino t en el documento d.
- $IDF(t, D)$= $\log(\frac{|D|}{OcurrenciasDet})$ - Cantidad total de documentos en los que aparece un termino. (Inverse Document Frequency)
- $TF - IDF(t, d, D) = IDF \cdot TF$

10.5. Hipotesis distribucional

Se asume que la semántica de una palabra esta determinada por el contexto en el que aparece. Esto significaría que se puede obtener el significado de las palabras si las resumimos en los contextos en las cuales aparecen

10.6. Word2Vec

Es una forma por la cual se agrupan palabras. Si las palabras tienen significados similares debe de significar que están en espacios vectoriales cercanos. (Esto incluso permite que se puedan realizar operaciones con ellas. *Ej.* $Rey - Hombre + Mujer = Reina$)

El objetivo que tiene es entrenar una red neuronal con las palabras del texto, de forma tal de que dada una palabra, nos diga la probabilidad de que cada palabra del vocabulario sea su vecina. (Entrenamos la red y nos quedamos con los pesos de la capa ocula, los 'word vectors')

Window size: parámetro que determina el numero de palabras que define un contexto.

La desventajas que tiene este modelo es que no puede trabajar con palabras que no conoce.

10.7. FastText

Mejora de Word2Vec. Para esto trata a las palabras como n-gramas. *Ej. n=2, Hola se representa como Ho,ol,la. Hola se interpreta como la suma de esos 3 n-gramas.* Con este cambio el modelo es capaz de generar embeddings de palabras que no habia visto durante el entrenamiento.

El problema que trae este cambio, es que modelarlo trae un crecimiento de dimensionalidad y memoria enorme. *Ej. 'Se ha visto que para un tamaño de corpus de 50 millones de palabras únicas, el tamaño de RAM requerido puede ser de hasta 256 GB de RAM.'*

Para evitar que se pase, esta limitado en la cantidad máxima de palabras (30 millones). Si se esta alcanzando el limite empieza a hacer remplazos (podas).

La implementación de este modelo esta hecha en C++, por lo que funciona rápidamente debido a la buena optimización de Threads que tiene.

Otros ejemplos similares son Bert y GPT-3.

11. Redes Convolucionales

Tipo especial de red que se usa para problemas abstractos o difíciles de explicitar. Se basan en ir detectando conceptos e ir resolviendo. Empieza con capas simples y va llegando a las complejas Es muy útil para la clasificación de imágenes.

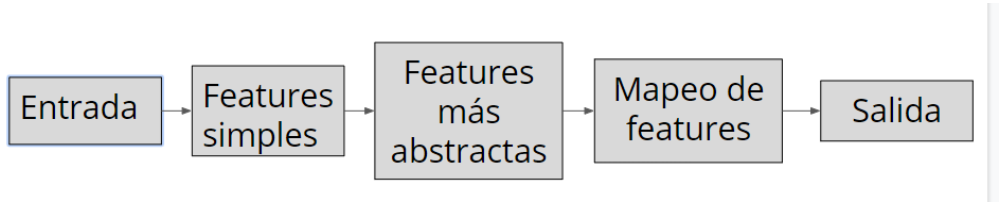


Figura 30: Sistema de trabajo de estas redes.

Tiene distintos usos:

- Clasificación
- Localización
- Detección de objetos
- Segmentación

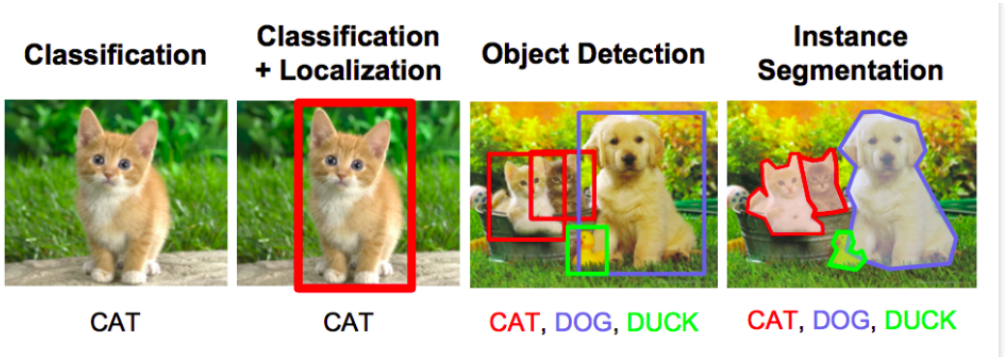


Figura 31: Ejemplos con algunos de los usos posibles.

- A medida que se avanza, las capas mas profundas implican conceptos mas complejos.
- Se llaman redes convolucionales porque aplican la operación de convolución. Gif
- Las redes neuronales comunes no escalan bien en este tipo de problema.

Arquitectura

Las redes estas tienen 2 tipos de capas:

- Classification layers: Son neuronas que se conectan con todo el volumen de entrada. (Capa densa comun)
- Feature extraction layers: Formadas por capas de convolucion y de filtros.

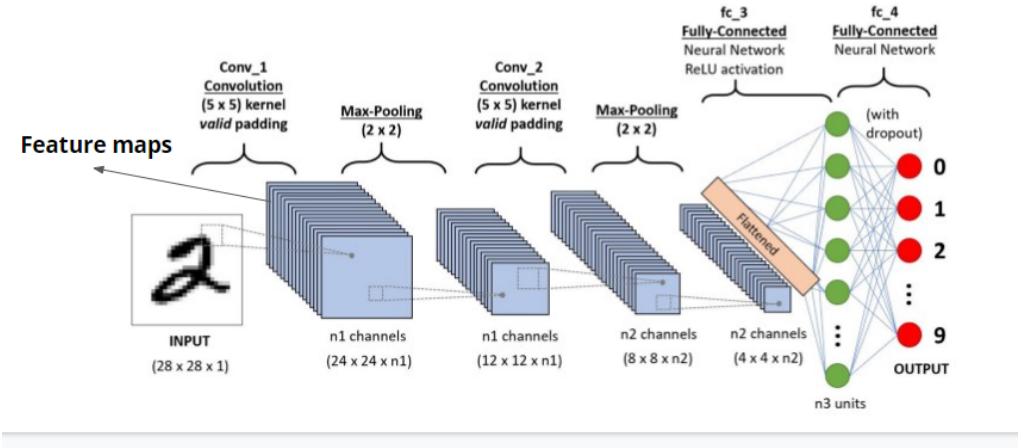


Figura 32: Arquitectura de la red

En las capas de convolución se aplican filtros(o kernel) también. Estos filtros reducen la dimensionalidad y tienen sus parámetros que son aprendidos en el entrenamiento. *Ej. MaxPool, se queda con el valor mas alto.*

En los filtros hay que tener en cuenta 3 propiedades:

- Stride: Determina los pasos del filtro.
- Padding: Hace el rellenado de la imagen.
- Depth Volume: Determina la profundidad.

Estas redes tienen también métodos de regularización. Los ya vistos en 8.5.

12. Redes Sociales

- Tienen un crecimiento exponencial.
- Son importantes.
- Ubicuas.
- Modelan distintas cosas.
- Crecientes, siempre estuvieron. A partir de la globalización fue exponencial.
- Creció la capacidad de mapearlas y estudiarlas.

12.1. ¿En cuales participamos?

- Sociales
  - Reales: Familia, amigos, trabajo...
  - Virtuales: Twitter, Facebook,...
- Tecnológicas: Internet
- De información: www.
- De transporte, comercio...
- Neuronas

12.2. ¿Que modelamos?

- Conectividad
- Comportamiento, interacciones
- Evolución

12.3. ¿Quienes se ocupan?

Tienen distintas escalas, distintas visiones

- Estructura, complejidad, sistemas conectados $\Longleftrightarrow$ Computación, matemática, física
- Estratégico $\Longleftrightarrow$ Economía, psicología
- Grupales, agregado $\Longleftrightarrow$ Sociología
- Sistemas gigantes, Big data $\Longleftrightarrow$ Computación

Cada uno tiene sus teorías subyacentes, de grafos, juegos, y redes.

12.4. Análisis de las redes sociales

- Diferencias entre amigos y conocidos. *Ej. Cambio de trabajo. Mas probable que venga de conocidos. Actúan como puentes entre grupos*
- Principio de clausura triadica.
- Mundo pequeño: 6 niveles de separación. *Ej. Carta a Boston.*

Puente

- Eje cuya eliminación desconecta un grafo.
- En los mundos pequeños (grafos muy conexos) son poco comunes.

Puentes Locales

Su eliminación no desconecta al grafo, pero aumenta el recorrido.

Homofilia

- Los nodos conectados tienden a ser parecidos.
- *Ej. Amigos con gustos parecidos.*
- 'Te metes en la burbuja'.
- Existe la homofilia inversa.
- Se puede medir al comparar cuanto se parecen los ejes de los grafos.

Relaciones +/-

- Amigo o enemigo
- Existen 4 tipos de triángulos. Ver la imagen 33

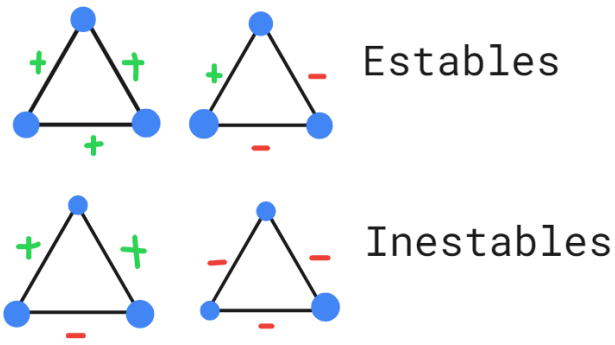


Figura 33: Los 4 tipos de triángulos posibles. Los + representan a los 'amigos' y los - a los 'enemigos'.

Hay un balanceo estructural si todos los triángulos de una estructura son estables. Caso contrario se tiende a romper y generar problemas.

13. Aprendizaje por refuerzo

Un agente (similar a modelo) tiene sensores para observar el estado de su entorno. Con el cual puede hacer acciones para alterar ese estado. Estas acciones conllevan una recompensa numérica.

Estas recompensas que el agente recibe vienen con cierta demora. Ya que hay que discernir entre cuales fueron las meritorias.

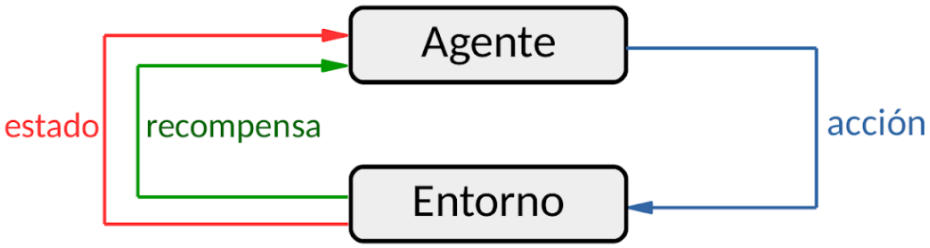


Figura 34: Sistema de aprendizaje por refuerzo.

13.1. Q-Learning

Algoritmo con el cual se controla el aprendizaje. Se busca aprender la función óptima fruto de la política óptima.

Se le indica un α (learning rate) que indica que tanto aprende. Si se aumenta mucho puede llevar a un mínimo local. Si es muy bajo el agente no se va a estar moviendo.

¿Como determinar la acción en base al estado? Hay que buscar un equilibrio entre la explotación y la exploración. Hay 2 estrategias para esto.

- $\epsilon - greedy$ Con la probabilidad ϵ elegimos al azar uno, y con 1-ϵ elegimos la acción que mejor conocemos.
- $\epsilon - first$ Elegimos al azar cierta cantidad de casos (1-ϵ) %, y después el que mejor conocemos (ϵ) %.

Este algoritmo (Q-Learning) tiene los problemas de que no aprende la noción de estados similares y de que a nivel practico no escala bien, se vuelve intratable (no es el caso teórico). Una alternativa para evitar esto, es aproximar la función Q del algoritmo con una red neuronal de convolución.

Algoritmo

1. Inicializar $Q(s, a)$
2. Inicializar s
3. Elegir la acción a en s y ejecutarla
4. Observar la recompensa r en el nuevo estado s'
5. Repetir a partir de 2 hasta que el estado s sea terminal.

Gif mostrando un ejemplo. Este seria el caso ya entrenado, donde lo que aprendió es a mover la pelota para el fondo del juego, de forma tal de que rebote sola y limpie el mapa. Al comienzo apenas responde.

14. Ética y moral en ML

14.1. ¿Por que discutirlo?

- Impacto económico
  - Hay mucha materia prima (datos)
  - Mucha capacidad de procesamiento y almacenamiento
  - Centrales de datos

14.2. Posicionamientos

- Optimista
  - No confinar o intentar controlar a la IA.
  - Enseñarle los valores humanos.
  - Hay que tener un planteo simbiótico.
- Pesimista
  - Que nos va a pasar y atentar contra nosotros.
- Realistas
  - Es falso el desarrollo exponencial de la IA.
  - Es falso que constituyen un peligro.
  - Los progresos son lineales.

14.3. Usos

La IA tiene muchos usos. Estos varían desde asistentes personales, redes de transporte, buscadores, publicidad, etc.

14.4. Preguntas a hacerse

- ¿Quien guarda la información?
- ¿Violación de derechos?

Hay que hacer los problemas y cuestiones éticas explícitos. Evitar los sesgos y las violaciones de privacidad.

Estas cuestiones no pueden quedar en pasarle la responsabilidad al usuario directamente, ya que el usuario es el ultimo en la cadena de desarrollo. Las empresas disponen de una mayor cantidad de herramientas para tratar los temas.

Una forma es incorporar las cuestiones éticas en los métodos de diseño, y alinearlos a los valores humanos. El tema que surge es: ¿Cuales? ¿Quien los decide? ¿Con que interpretación? Son cuestiones que hay que definir. Dejar de forma clara que es desleal y que no.

14.5. Teorías éticas

- Meta-ética: Estudia los principios. *Ej. Objetivismo vs Relativismo*
- Ética Normativa: La que genera reglas y nos interesa en *Machine Learning*
- Ética Aplicada: Trata las situaciones sociales. *Ej. Derechos de animales, eutanasia...*

14.6. Ética Normativa

- Ética de la virtud: Busca la moralidad de una acción. Hábitos vs Reglas.
- Ética consecuencialista: Evalúa el resultado.
- Ética deontológica: Propone evaluar la moralidad de una acción

Tiene 2 enfoques:

- Top-Down: Se basa en el deber, toma la teoría previa y la deriva.
- Bottom-Up: Se intenta capturar la teoría a partir de la observación. Aprendizaje, emoción.

Ambas tienen el problema del marco. El cual se basa en el impacto en el contexto. Se puede medir que sucede en el local, pero no en los colaterales.