Aprendizaje Supervisado

Docentes:

- **Dra. Ing. Karim Nemer**
- **Dr. Lic. José Robledo**

Segunda Clase

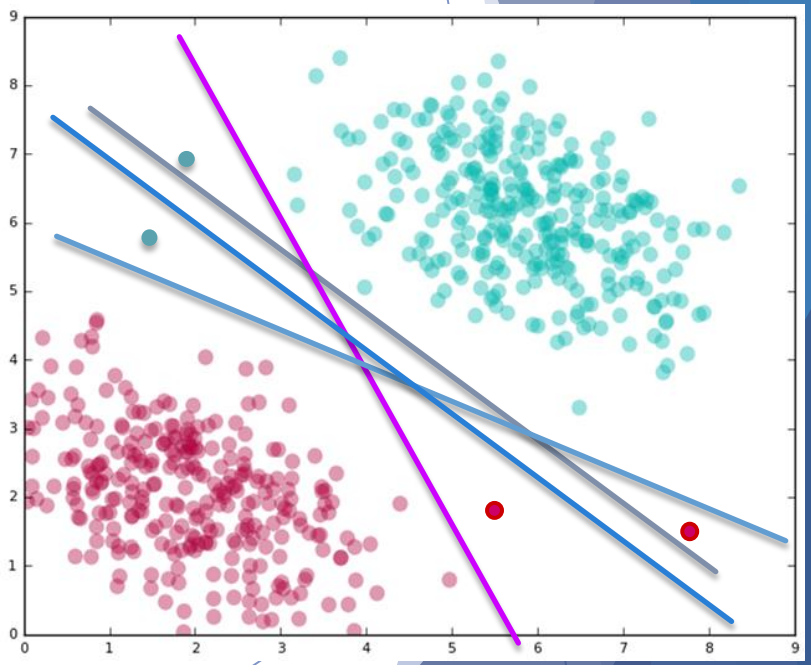
Segunda Clase: 25/06/2022

- Introducción al ML
- Etapas en la aplicación del ML
- Aprendizaje supervisado.
  - Repaso: Regresión Lineal y Polinomial, Regresión Logística, Naive Bayes.
  - Repaso: Perceptrón.
- Support Vector Machines.
  - Introducción
  - SVC/SVR. Datos no linealmente separables. Función de costo.
- Ensemble learning.
  - Repaso: Decision Trees
  - Random Forest
  - Bagging, Boosting, Voting.
- Redes neuronales.
  - Perceptrón multicapa.
- Sistemas de recomendación.
  - Filtrado colaborativo.
- Prácticas

Support Vector Machines

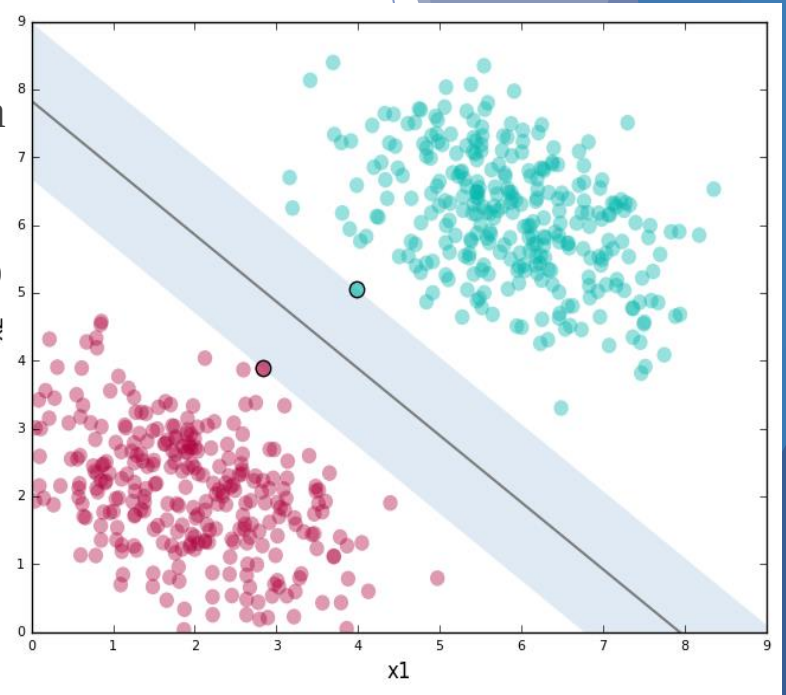
SVM: Fronteras de decisión en clasificación

- Un clasificador busca separar los datos de una y otra clase de la mejor manera.
- Esta separación se da mediante una frontera de decisión.
- ¿Qué determina que tan “buena” es una frontera de decisión?
- Cualquiera de las líneas separan los datos correctamente.
- Buscamos una línea que capture el patrón general entre los datos.
- La línea fucsia tiene menos margen entre ella y ambos clústeres de datos.
- La línea azul se encuentra bien a la mitad de ambos clústeres.



Support Vector Machines

- Es un algoritmo que busca separar los datos mediante la mejor frontera de decisión. Esta frontera de decisión es conocida como **hiperplano**.
- En este caso, “mejor” se refiere a aquella que esté lo más separada posible de los puntos más cercanos a ella. Estos puntos son conocidos como **vectores de soporte**, y el espacio entre ellos y el hiperplano se conoce como **margen**.
- En términos más técnicos, un algoritmo de SVM encuentra el hiperplano que devuelva el mayor margen entre sí mismo y los vectores de soporte.
- Este tipo de clasificador a veces es conocido como “clasificador por márgenes” (margin classifier).



Link

SVM: Función de costo y a optimizar

- Los SVM utilizan una función de costo conocida como ***Hinge loss***.
- A diferencia de regresión logística, los datos se anotan con {-1, 1} de acuerdo al valor de la etiqueta.
- La función de costo de Hinge se define como:

$$\boldsymbol{c\big(x, y, f(x)\big) = max(0, 1 - y * f(x))}$$

- Donde el costo es 0 si el valor real y el predicho tienen el mismo signo y están dentro del margen de error (por lo general 1).

- La función que buscamos minimizar es la siguiente:

$$\boldsymbol{min_{\omega} \sum_{i=1}^{n} \max(0; 1 - y_i \langle x_i, \omega \rangle) + \lambda \, \|\omega\|^2}$$

- Dónde $\lambda \, \|\omega\|^2$ es el parámetro de regularización.

SVM: Gradientes

Tenemos dos factores en la función de costo que hay que derivar:

$$\frac{\delta}{\delta \omega_k} = \lambda \|\omega\|^2 = 2\lambda\omega_k$$

$$\frac{\delta}{\delta \omega_k} \max(0; 1 - y_i \langle x_i, \omega \rangle = \begin{cases} 0 & si\ y_i \langle x_i, \omega \rangle \geq 1 \\ -y_i\, x_{ik} & c.c. \end{cases}$$

Al actualizar los pesos, de acuerdo al signo de la predicción, tendremos para el caso donde el signo sea el mismo:

$$\omega = \omega - \alpha(2\lambda\omega)$$

Mientras que cuando el signo entre la predicción y el valor real es diferente:

$$\omega = \omega - \alpha(y_i x_i - 2\lambda\omega)$$

SVM con outliers

SVM: Outliers

La mayoría de los casos, los datos no son linealmente separables.

En algunos casos, existen outliers.

Hay un parámetro que define qué tan tolerante puede ser SVM sobre la clasificación incorrecta de datos.

El "***parámetro C***", define un *tradeoff* entre clasificar mejor los datos de entrenamiento y tener una mejor "***separación***" (un margen más amplio).

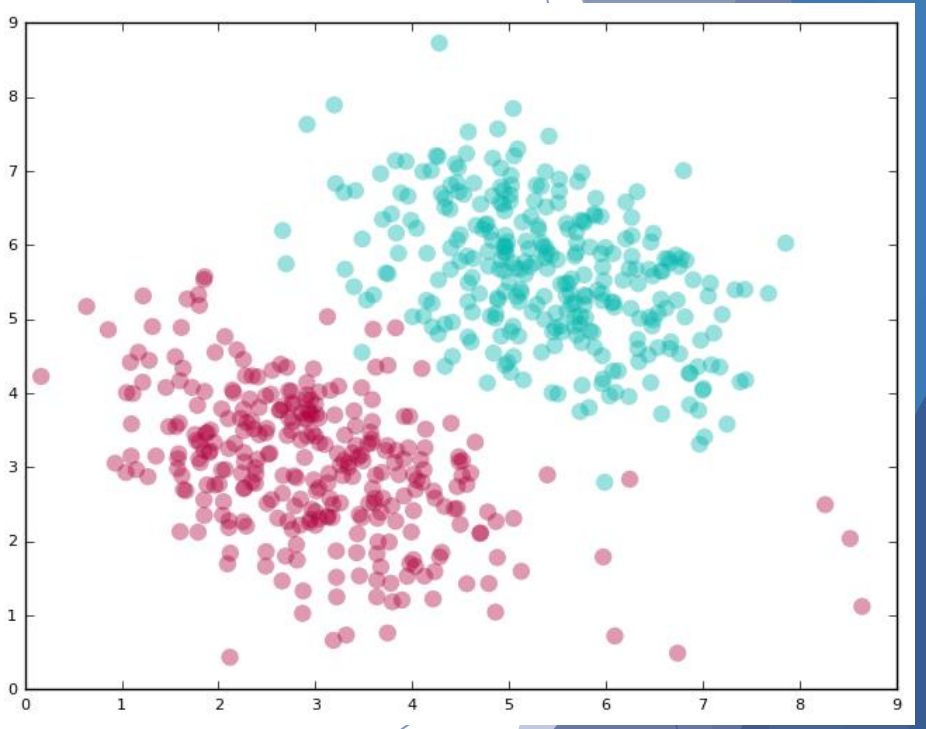


Image from https://blog.statsbot.co/

SVM: Parámetro C

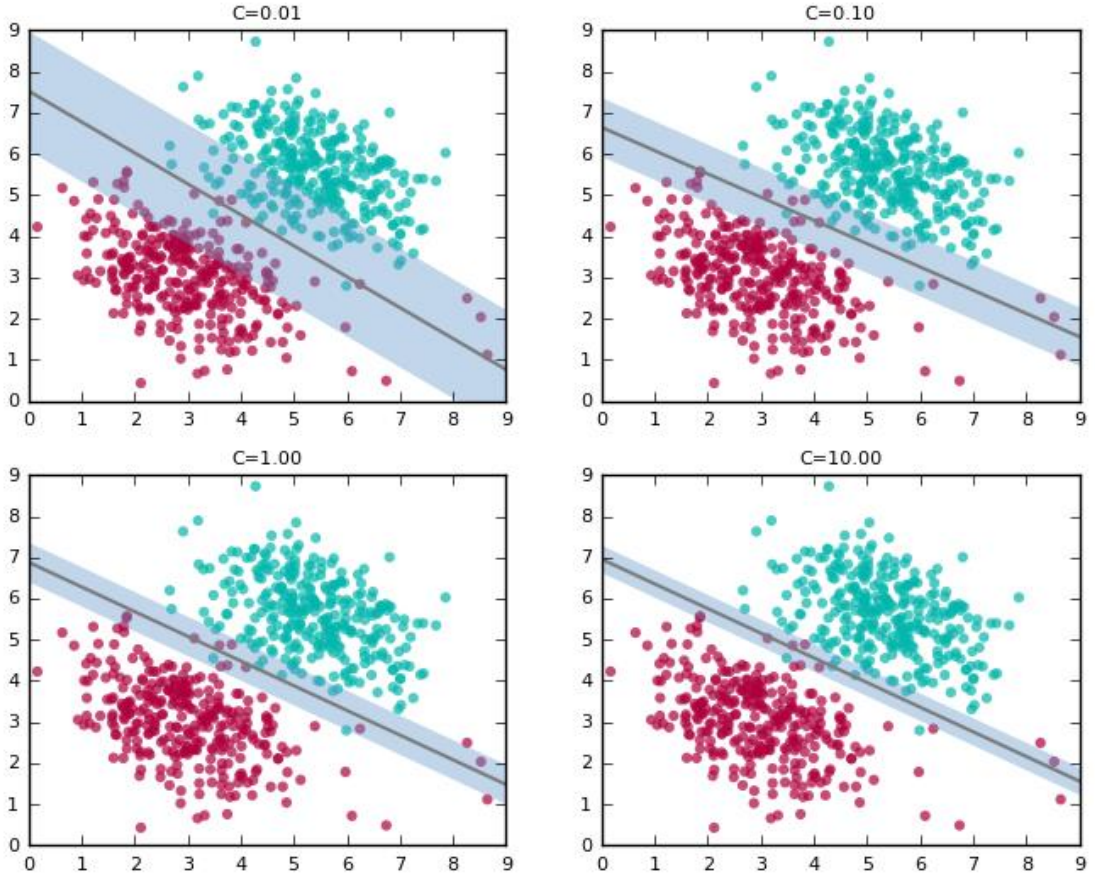


Image from https://blog.statsbot.co/

Demo Time (demo_5_svm)

SVM con datos no linealmente separables

¿Qué hacer con datos no linealmente separables?

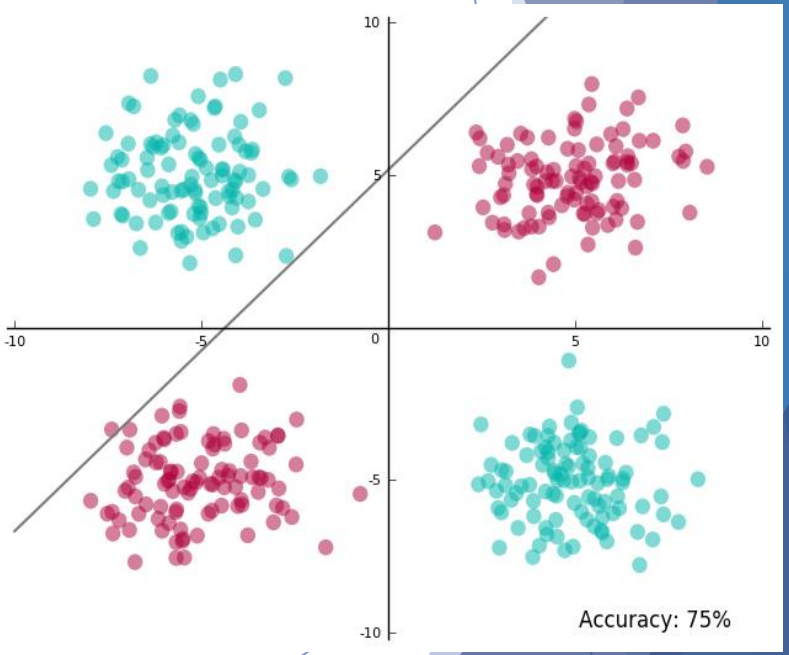
SVM es una técnica para separar los datos mediante un hiperplano.

Si los datos no son linealmente separables, dicho hiperplano no existe.

Solución: Proyectar los datos a una dimensión donde sí sean linealmente separables.

En el ejemplo anterior, tomamos el conjunto de datos en dos dimensiones, y lo proyectamos a tres dimensiones con la siguiente ecuación:

$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2} x_1 \, x_2$$



¿Cómo se ve el plano proyectado?

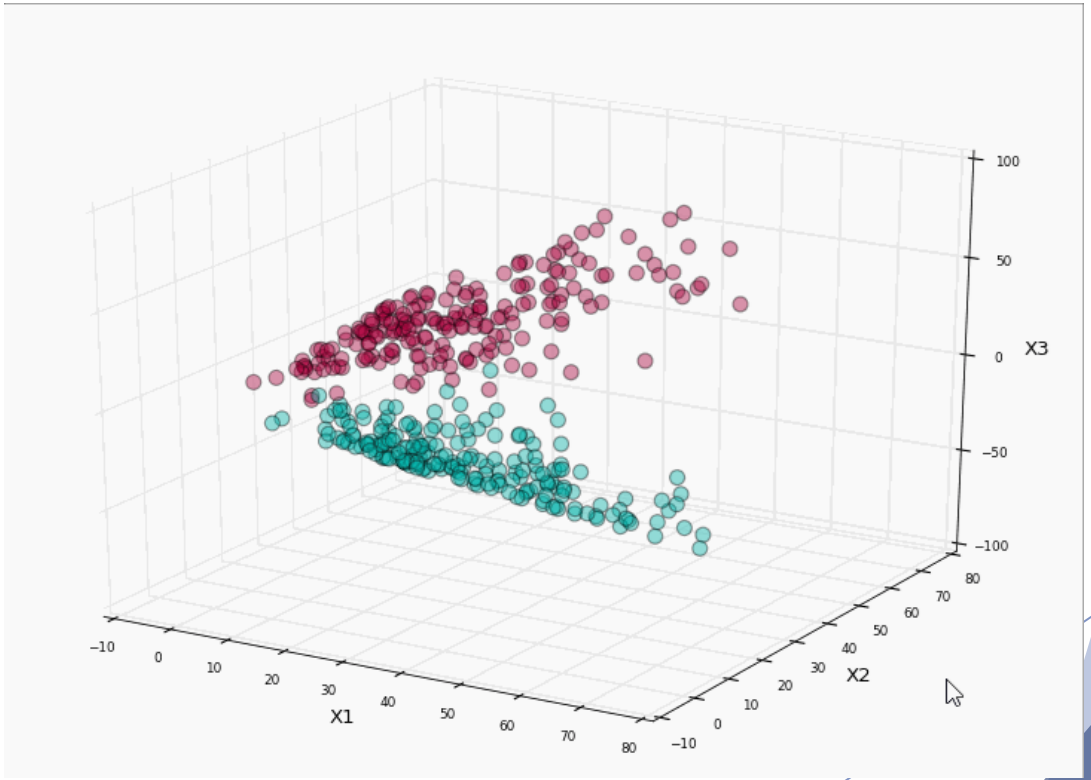


Image from https://blog.statsbot.co/

¿Cómo se ve el plano proyectado?

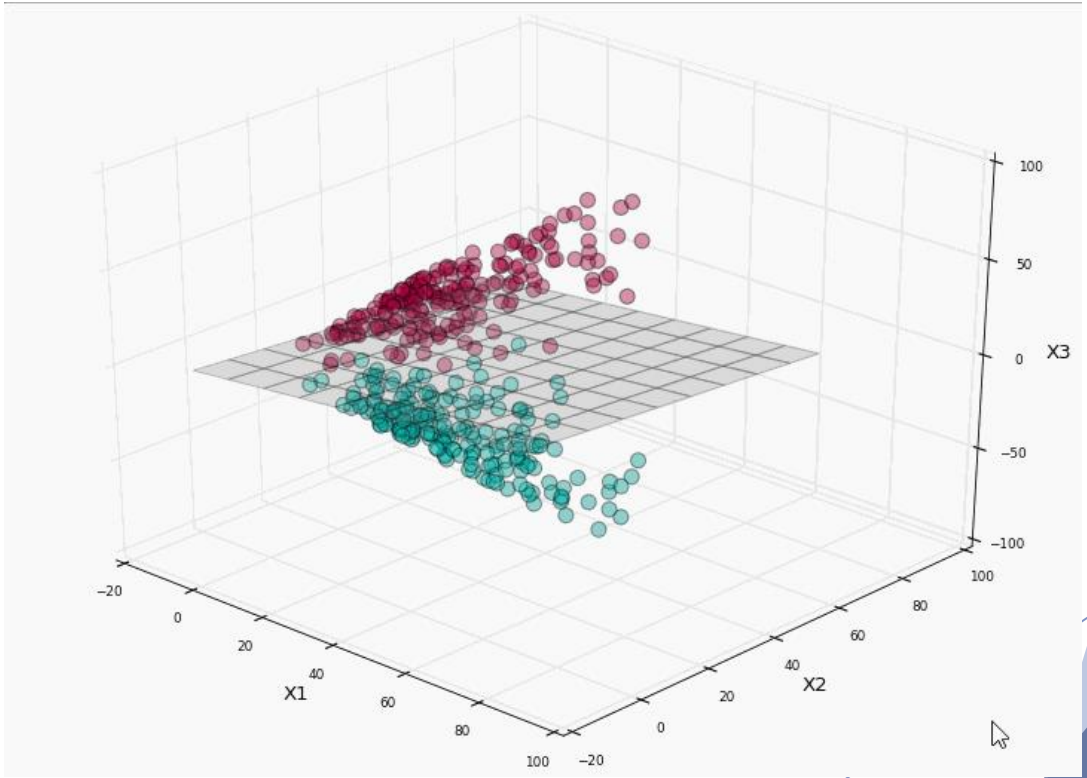


Image from https://blog.statsbot.co/

¿Y en 2 dimensiones?

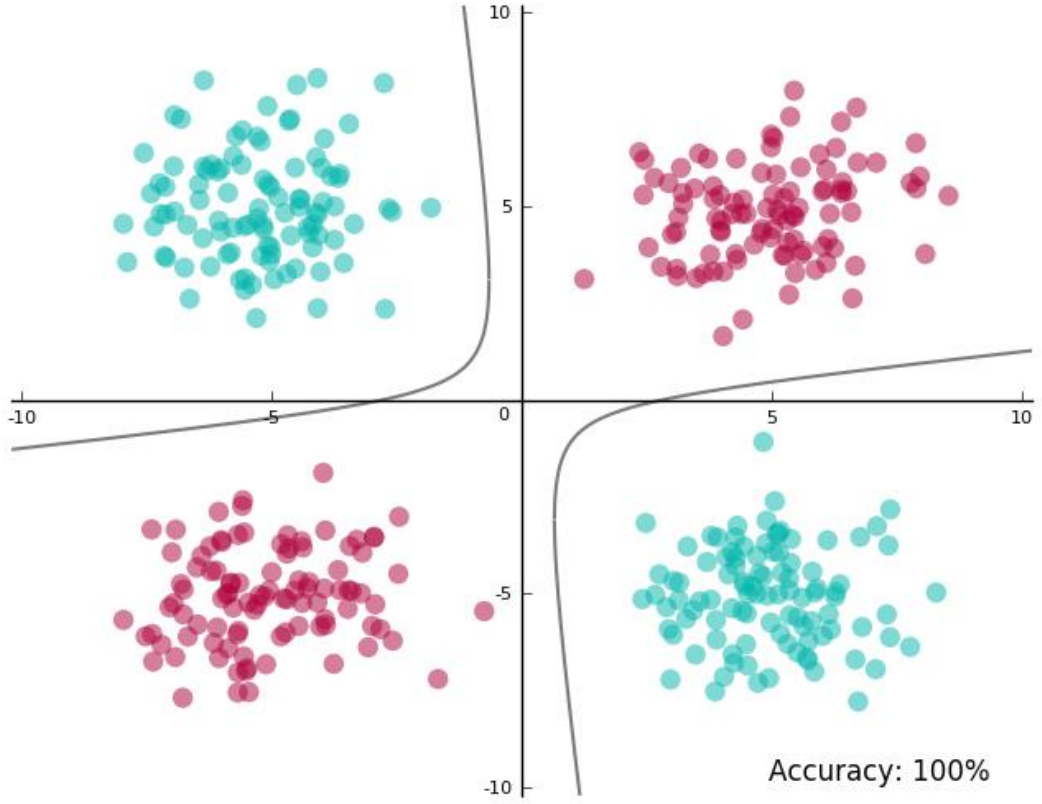


Image from https://blog.statsbot.co/

SVM: Kernels

La manera en que el algoritmo de SVM realiza la proyección es mediante el uso de kernels.

Las funciones de kernel toma dos puntos del espacio original, y devuelve el producto punto en el espacio proyectado.

Este producto punto es lo que la función de SVM necesita para calcular el costo.

En el ejemplo anterior, el kernel es:

$$K(x_i, x_j) = \langle x_i, x_j \rangle^2$$

¿Cómo elegir el kernel?

Este no es un problema trivial. Requiere mucho conocimiento matemático encontrar la proyección correcta.

En general, los frameworks más utilizados para hacer SVM tienen algunos kernels bastante comunes:

- **Polinomial**: $K(x, z) = (\langle x, z \rangle + c)^d$
- **Radial Basis Functions (RBF)**: $K(x, z) = \exp\left(\frac{-(x-z)^2}{1\sigma^2}\right)$
- **Sigmoid**: $K(x, z) = \tanh(c\,\langle x, z \rangle + h)$

Support Vector Regression

Se basa en la idea de SVMs de buscar los vectores de soporte, pero en este caso el valor de y_i es un número real.

Utiliza necesariamente “márgenes blandos”, requiere un parámetro adicional ε para calcular la función de costo.

En general la regresión lineal es más popular, pero con el uso de kernels, se pueden lograr regresiones no lineales muy interesantes.

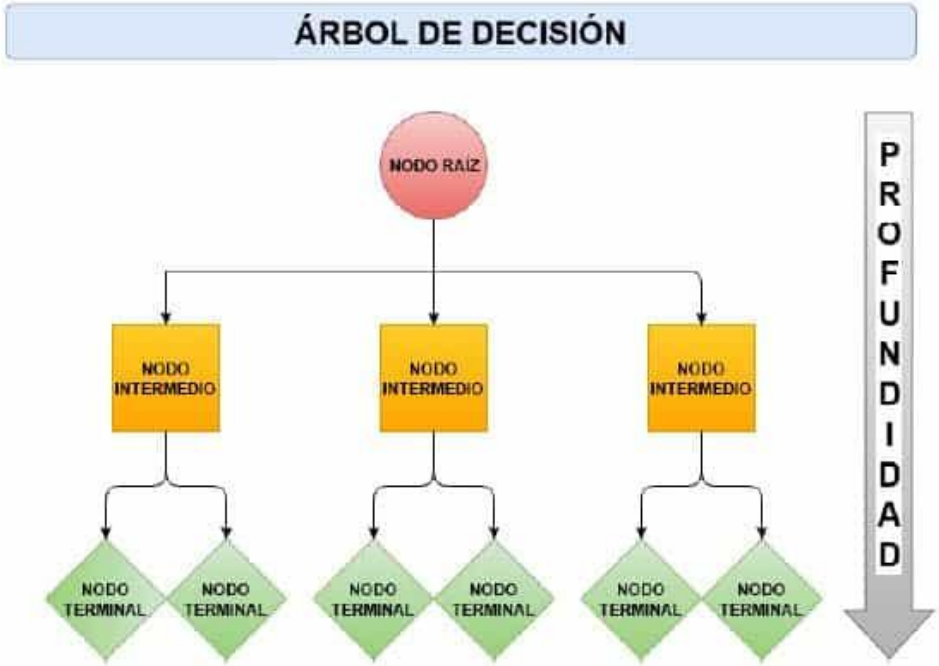
Link

Demo Time (demo_6_kernels)

Árboles de decisión

Árboles de decisión

- Aprende a diferenciar los datos en base a reglas de decisión.
- Los nodos del árbol representan las reglas. Las hojas asignan la clase o el valor.
- El árbol se particiona recursivamente. Para obtener el resultado, simplemente se siguen los nodos de decisión de acuerdo a los datos y se asigna la clase final.
- Es un algoritmo de “caja blanca”, ya que puede visualizarse fácilmente e interpretarse por los humanos (a diferencia de un algoritmo de “caja negra” como son las redes neuronales).
- Son buenos con datos de mucha dimensionalidad (high dimensional data)



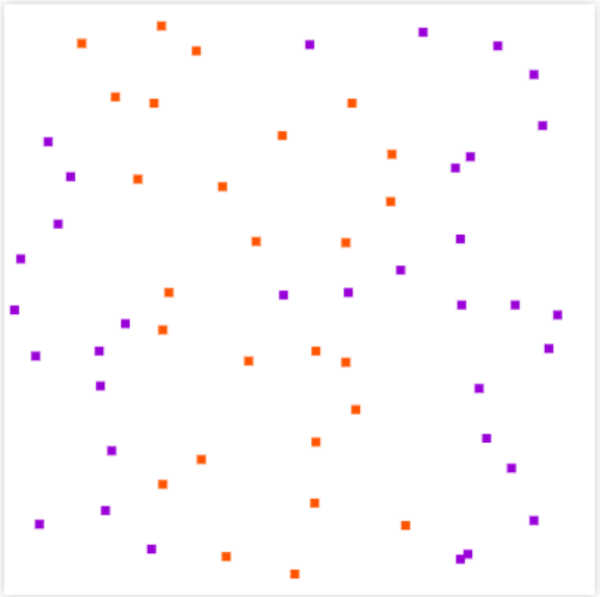
Árboles de decisión: Algoritmo

1. Selecciona el mejor atributo de acuerdo a alguna métrica (e.g. ganancia de información).
2. Hacer un nodo de decisión con ese atributo, que particione los datos en subconjuntos de menor tamaño.
3. Repetir recursivamente el procedimiento para cada nodo hijo.
4. El algoritmo se detiene si:
    - Todos los ejemplos del subconjunto son de la misma clase.
    - Todos los elementos del subconjunto son constantes con respecto al atributo/s de interés del nodo actual.

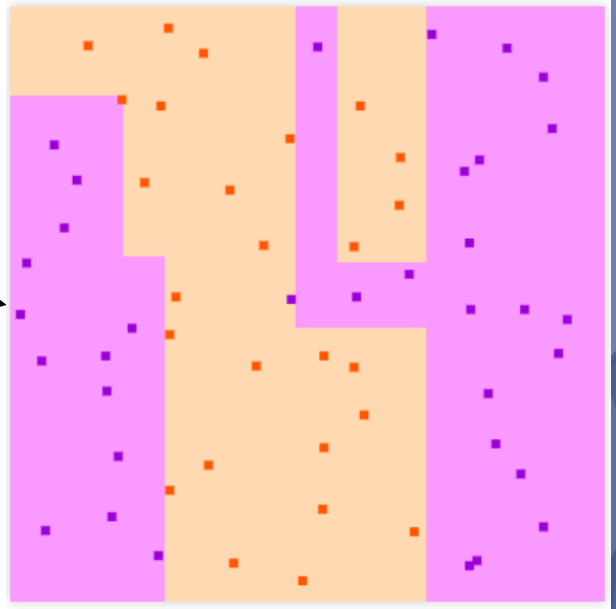
Árboles de decisión: Métodos de decisión

1. Definen la forma de particionar el conjunto de datos (es una heurística).
2. Buscan rankear cada atributo de acuerdo a ciertos parámetros.
3. Los criterios más populares son:
    - **Information Gain:** calcula la diferencia entre la entropía del conjunto antes de dividirse y la entropía media luego de dividir el conjunto de datos, para cada atributo.
    - **Gini Index:** Es una métrica que mide cuánto un elemento del subconjunto elegido al azar sería identificado incorrectamente. Busca los atributos que tienen menor índice de Gini.
    - **Mean Squared Error:** Este método sirve para casos de regresión (en lugar de clasificación). Busca minimizar el error cuadrático medio en los nodos hijos a la hora de particionar los datos.

Árboles de decisión: Cómo son las fronteras



"Frontera de Decisión"



Link

Imagen de Yisong Yue

Demo Time (demo_7_trees)

Ensemble Learning

Ensemble Learning: Motivación

Se basan en la idea de que el trabajo en conjunto debería dar mejores resultados.

De tal forma, un conjunto de modelos, al combinarse, deberían tener mejor performance.

Consideremos el caso de tres modelos: M1, M2 y M3:

Cuando las predicciones son iguales, es sencillo definir cómo predecir; qué hacemos en el caso en que difieran?

- Le creemos al mejor de los modelos?
- Votamos por mayoría?

Ensemble Learning

Un modelo "***ensemble***" se constituye como un conjunto de diferentes modelos.

Habitualmente, un modelo "***ensemble***" es más preciso que los modelos que lo constituyen. Intuitivamente, esto se debe a que "dos aprenden mejor que uno" (Sinergia).

Ensemble Learning: Aproximación de justificación

- Asumimos que tenemos un modelo M, formado por n modelos: M_1, M_2,... M_n.
- Cuando un modelo recibe un dato *x*, el modelo predice M(*x*) a partir de las predicciones $M_i(x)$, a partir de votación (clase más votada).
- Cómo determinamos cuán bien funciona el modelo?
- Para responder, consideramos:
  - Todos los clasificadores M_1, M_2,... M_n son igualmente precisos (precisión **p**)
  - Los errores en la clasificación hecha por cada clasificador son independientes: **P**(M_j erróneo | M_k erróneo) = **P**(M_j erróneo)

Ensemble Learning: Aproximación de justificación

Para hacer el ejemplo más concreto, consideremos:

- **p** = 0.8 (probabilidad de acertar de todos los modelos)
- **n** = 5 (cantidad de modelos)

Entonces,

P (la predicción de M es correcta) =

= P(al menos 3 de los 5 M_i predicen correctamente)

$$= \binom{5}{5} 0{,}8^5\ 0{,}2^0 + \binom{5}{4} 0{,}8^4 0{,}2^1 + \binom{5}{3} 0{,}8^3\ 0{,}2^2 = 0{,}942$$

Ensemble Learning: Bagging

Bagging

Es un método para hacer aprendizaje por "ensemble".

Si usamos el mismo modelo sobre los mismos datos, obtendremos los mismos resultados (*salvo inicializaciones aleatorias*). Entonces, de cierta forma, debemos introducir variaciones en los datos:

Si **D** es el dataset inicial; repetimos **k** veces lo siguiente:

- Generar $\mathbf{D_i}$ a partir de las entradas de **D**, seleccionando aleatoriamente y con reposición $|\mathbf{D}|$ instancias de **D**.
- Entrenamos el modelo $\mathbf{M_i}$ a partir de $\mathbf{D_i}$.

Nuestro modelo **M** selecciona las predicciones más frecuentes de $\{\mathbf{M_i}\}_\mathbf{i}$.

Bagging gráficamente (para árboles de decisión)

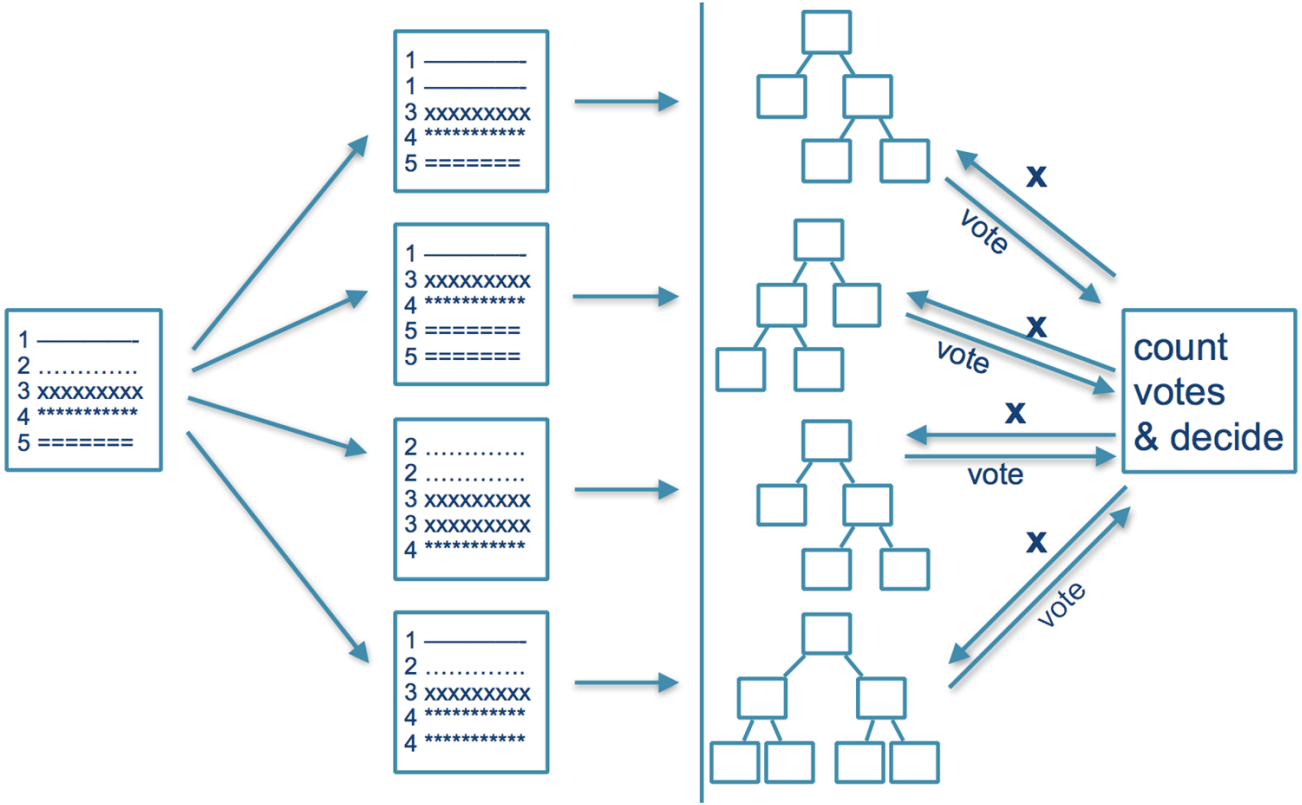


Imagen de Hendrik Blockeel

Bagging para árboles de decisión

Bagging generalmente funciona bien para algoritmos "inestables":

- *Un algoritmo es inestable si pequeñas variaciones en el dataset pueden generar modelos muy diferentes.*

Resulta que los árboles de decisión son inestables.

Si bien lo anterior incentiva al uso de bagging para árboles de decisión, tenemos una desventaja importante: entrenar ***k*** árboles es ***k*** veces más caro que entrenar uno solo.

Random Forests

Random Forests

Las Random Forests son una modificación a Bagging para Árboles de Decisión. Para evitar la sobrecarga, se simplifican los modelos:

- Como vimos antes, cuando se crean los árboles (cuando se entrenan), todas las características son consideradas o evaluadas al crear cada nodo.
- Para los Random Forests, en cada nodo se consideran sólo M atributos elegidos aleatoriamente (parámetro *max_features* en sklearn).
- Usualmente, se toma:

$$M = \sqrt{número\ de\ atributos}$$

Random Forests

El modelo se puede resumir en los siguientes pasos:

Repetir **k** veces:

- Construir un dataset a partir del original, como se hace con bagging.

- Construir un árbol de decisión:
  - Para cada árbol, seleccionar **M** atributos y construir el arbol "*óptimo*" entre esas features.
  - Repetir hasta que el árbol esté completamente construido (no se hace pruning).

Random Forests

"Los Random Forests son uno de los métodos de aprendizaje más eficientes y precisos a la fecha" (2008): Caruana: *An empirical evaluation of supervised learning in high dimensions. ICML 2008.*

El algoritmo es sencillo, fácil de implementar, fácil de usar y requiere de poco ajuste de parámetros.

Es relativamente sencillo debuggear los Random Forests; aunque comparado con los Árboles de Decisión, pueda resultar menos interpretable.

Demo Time (demo_8_random_forest)

Fin de la segunda clase