

Aplicaciones de Machine Learning

Camilo Arias Martelo

Universidad Iberoamericana.
Noviembre 28, 2019



- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

Este curso se basó en las siguientes clases:

- *Machine Learning for Public Policy*. Prof. Rayid Ghani. *Computer Science*, Universidad de Chicago. Primavera 2019.
- *Introduction to Machine Learning*. Prof. Kevin Gimpel. *Toyota Institute of Technology*, Universidad de Chicago. Otoño 2019.
- *Big Data & Development* Prof. Joshua Blumenstock. *School of Information* Universidad de California, Berkeley.

Principales textos fueron:

- Murphy, Kevin. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press.
- Bishop, Christopher. (2006). *Pattern Recognition and Machine Learning*. Springer Science.
- Athey, Susan. (2018). *The Impact of Machine learning on Economics*.

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

Field of study that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959)

A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E (Tom Mitchel, 1998)

¿Para qué?:

- Reducir el trabajo de un humano.
- Mejorar la precisión de los humanos.

Arthur Samuel, primer algoritmo de machine learning

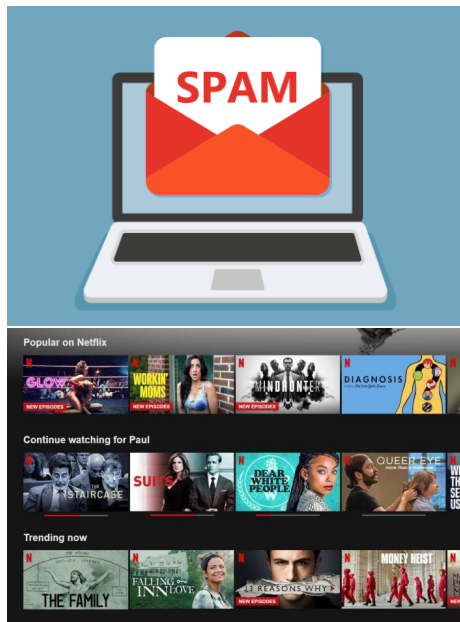
- En 1959, mientras trabajaba en IBM, creó el primer programa que aprendió a jugar ajedrez.
- El programa venció al jugador 4 de Estados Unidos.



Usos comunes de *machine learning*

- Clasificar correo electrónico como spam o no spam.
- Publicidad: Predecir la probabilidad de compra para colocar ads.
- Reconocimiento facial o de lenguaje (Amazon Alexa)
- Finanzas: Predecir pago de préstamos.
- Clasificar imágenes

<https://cloud.google.com/vision/#casos-prcticos>



Supervised Learning

- Dado un *input* X predecir un *output* Y .
- Objetivo: Construir una función $f : X \rightarrow Y$.
- Contamos con un *training set*: N Observaciones de las cuales conocemos tanto X como Y .
- Usaremos el *training set* para estimar la función f .
- Importante: La función f debe generalizar para observaciones fuera del *training set*.
- Dos tipos de tareas:
 - 1 Si Y es continua, es un problema de regresión.
 - 2 Si $Y = \{1, 2, \dots, C\}$ es un problema de clasificación.

Supervised Learning

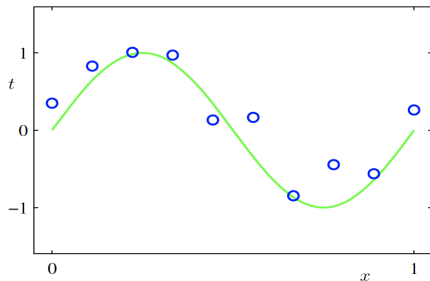
$$y = f(\mathbf{x}; \mathbf{w})$$

y es el valor predicho

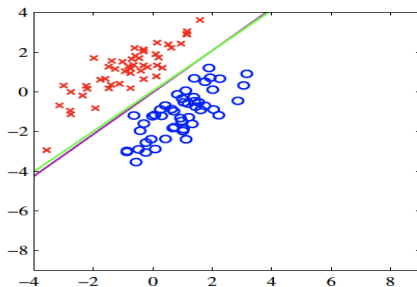
\mathbf{x} es un vector de d variables, inputs o *features*.

\mathbf{w} son los parámetros del modelo

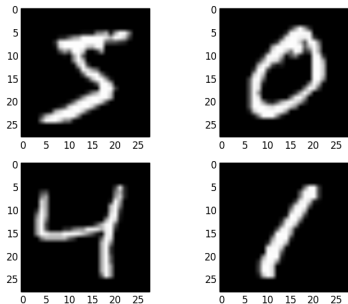
Regresión



Clasificación



Reconocimiento de dígitos



- X : Imagen de 28 X 28 pixeles, cada uno con un nivel de brillo.
- $Y = \{1, 2, \dots, 9\}$

Procedimiento:

- 1 Clasificar a mano 20,000 imágenes.
- 2 Representar cada imagen i como un vector $\mathbf{x}_i = (x_1, x_2, \dots, x_d)$.
- 3 Utilizar 15,000 observaciones para entrenar modelo f . (*Training set*).
 - 1 10,000 *training*.
 - 2 5,000 para validación.
- 4 Evaluar f en 5,000 observaciones restantes (*Testing set*).

Como entrenar modelo f

El proceso de estimación de un modelo tiene 3 componentes:

- ① Representación. Debemos seleccionar la familia F a la cual pertenecerá la función f que estimaremos. Hoy veremos 6 diferentes familias F
 - ▶ Árboles de Decisión
 - ▶ *K Nearest Neighbors*
 - ▶ Regresión logística
 - ▶ *Support Vector Machines*
 - ▶ LASSO
- ② Función objetivo. Necesitamos una función a optimizar.
 - ▶ Error cuadrático medio
 - ▶ Error 0 si $f(\mathbf{x}_i) = y_i$, 1 si $f(\mathbf{x}_i) \neq y_i$
 - ▶ Costo / utilidad.
- ③ Algoritmo de optimización. No se cuenta con una solución cerrada como en regresión lineal. Hay que seleccionar método para encontrar f óptima de forma iterativa.

(Domingos, P. 2012)

Lo importante es generalizar

Función de error: $l(y, \mathbf{x}; \mathbf{w})$

- Utilizando el error empírico:

$$L(\mathbf{y}, \mathbf{X}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n l(y_i, \mathbf{x}_i, \mathbf{w})$$

- Buscamos minimizar el error esperado o el riesgo:

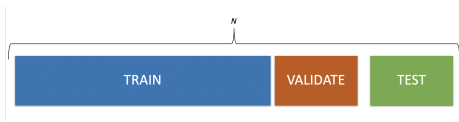
$$R(\mathbf{w}) = E[l(y_0, \mathbf{x}_0, \mathbf{w})]$$

Donde y_0 es la etiqueta de \mathbf{x}_i que desconocemos al momento de modelar.

- Supuesto: El *training set* es representativo de la distribución de \mathbf{y} y \mathbf{X} .

Training y Test data

- Training set: Estimar modelo f .
- Validation set: Evaluar precisión de f y comparar con otras estimaciones. Hacer selección final f^* final
- Test set: Evaluar modelo f^* final y obtener aproximación de riesgo.

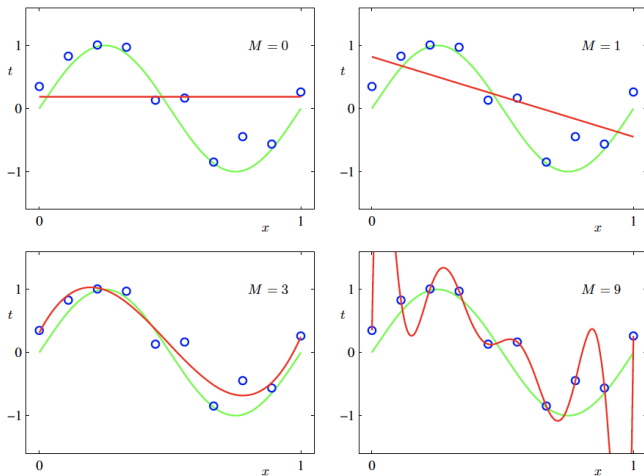


Independencia del test set.

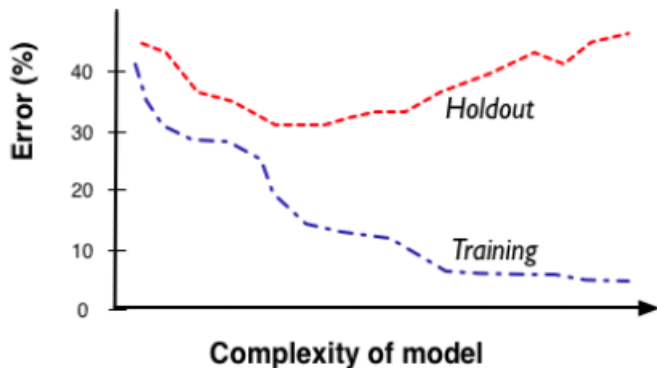
Solo usar test set para evaluar modelo final. Queremos estimar el error esperado. No queremos optimizar según el test set.

Overfitting

Overfitting ocurre cuando seleccionamos un f que se ajusta tanto al *training set* que deja de generalizar para la distribución de y y \mathbf{X} .



Overfitting

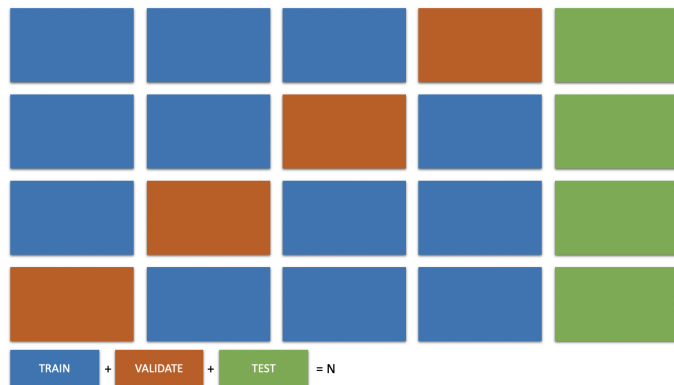


(Provost y Fawcett, 2013)

Como evitar Overfitting

- Incrementar N
- Restringir la complejidad del modelo. Cada familia F tiene parámetros que sirven para restringir complejidad.
- Penalizar función objetivo por complejidad.
- Usar más de un modelo: *ensembles*
- Cross Validation

K folds Cross Validation

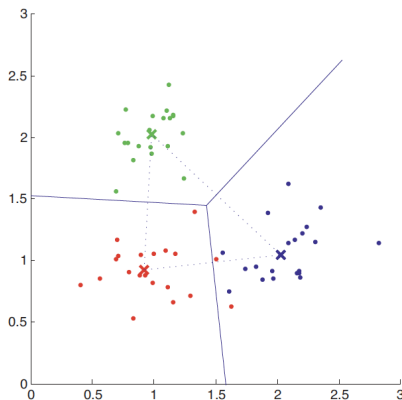


Dado K validation sets, con labels \mathbf{y}_k y features \mathbf{X}_k Seleccionar $f^* \in F$ que minimiza el error medio

$$L(\mathbf{y}, \mathbf{X}, \mathbf{w}) = \frac{1}{K} \sum_k L(\mathbf{y}_k, \mathbf{X}_k, \mathbf{w})$$

- Set de datos X sin *labels*.
- El objetivo es comprender los datos, no predecir.
- Objetivos:
 - ▶ Agrupar datos en *clusters*
 - ▶ Estimar distribución de los datos.
 - ▶ Reducir dimension de los datos para visualizar.
- Se puede utilizar en una etapa exploratoria.

Ejemplo de K clustering, $K = 3$



(Flach, 2012)

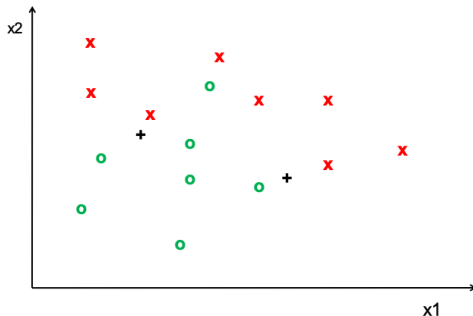
- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

K Nearest Neighbors

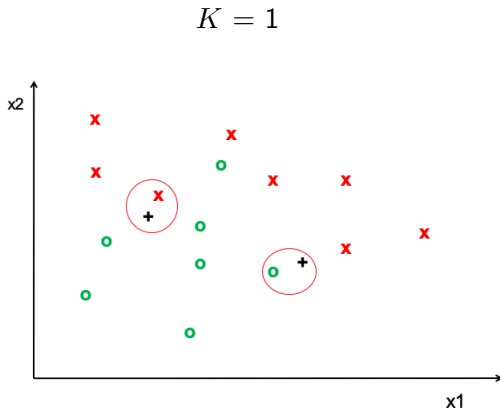
- Encontrar los K nearest neighbors (KNN) de \mathbf{x}_0
- Predecir y_0 como la media, mediana or moda y_k para $k \in KNN$

Clasificar 2 nuevas obs.



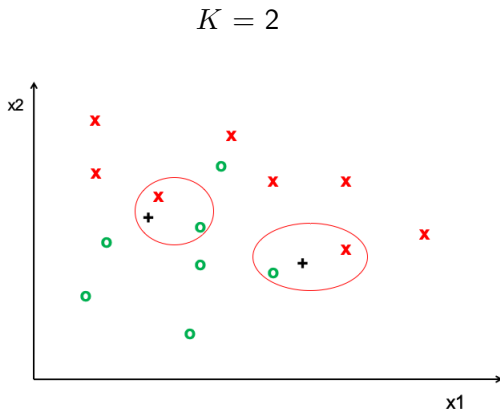
K Nearest Neighbors

- Encontrar los K nearest neighbors (KNN) de \mathbf{x}_0
- Predecir y_0 como la media, mediana or moda y_k para $k \in KNN$



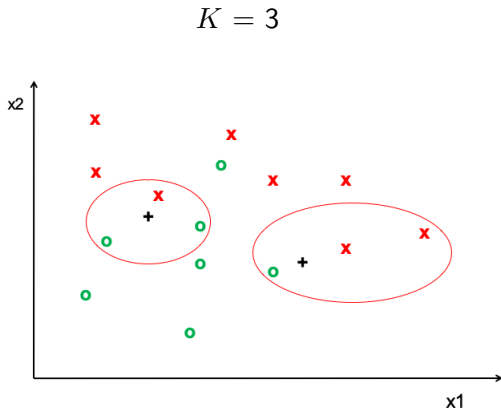
K Nearest Neighbors

- Encontrar los K nearest neighbors (KNN) de \mathbf{x}_0
- Predecir y_0 como la media, mediana or moda y_k para $k \in KNN$



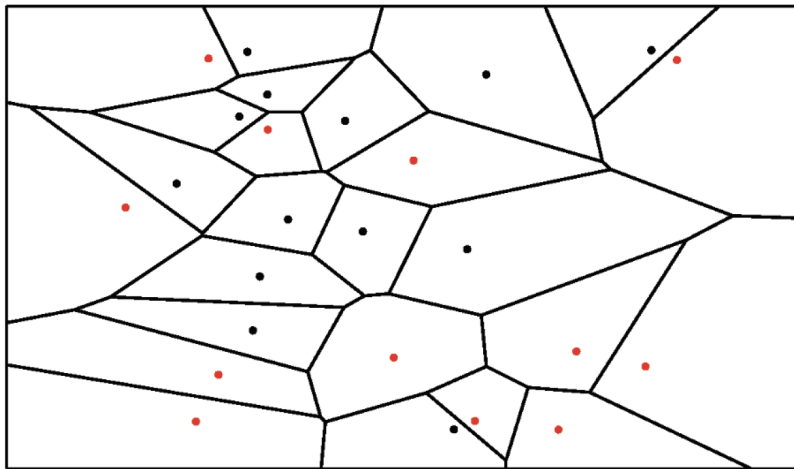
K Nearest Neighbors

- Encontrar los K nearest neighbors (KNN) de \mathbf{x}_0
- Predecir y_0 como la media, mediana or moda y_k para $k \in KNN$



Partición del espacio con 1-NN

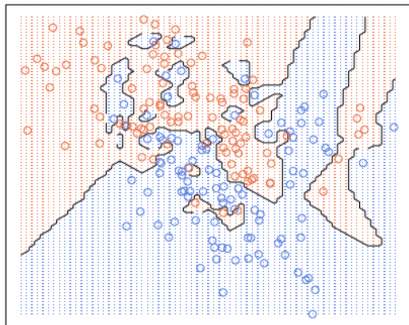
Voronoi Tessellation



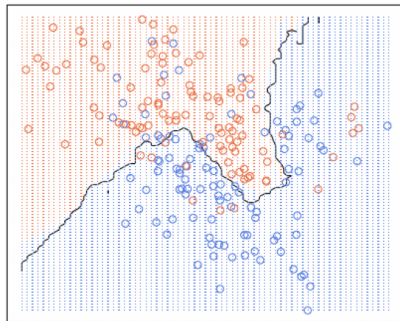
(Murphy, 2012)

Parámetros clave de KNN

- 1 K . Número de vecinos más cercanos.
nearest neighbour ($k = 1$)



20-nearest neighbour




- 2 Métrica de distancia.
 - 1 Euclideana
 - 2 Manhattan

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

Decision trees

Crear una función f discreta que minimice el error.



| Name | Balance | Age | Employed | Write-off |
|---------|-----------|-----|----------|-----------|
| Mike | \$200,000 | 42 | no | yes |
| Mary | \$35,000 | 33 | yes | no |
| Claudio | \$115,000 | 40 | no | no |
| Robert | \$29,000 | 23 | yes | yes |
| Dora | \$72,000 | 31 | no | no |

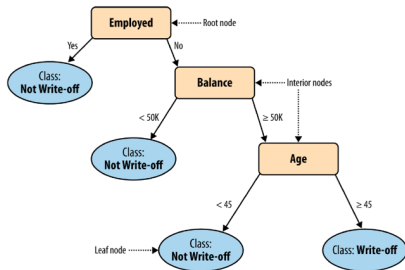
Decision trees

Crear una función f discreta que minimice el error.

Attributes

Target attribute

| Name | Balance | Age | Employed | Write-off |
|---------|-----------|-----|----------|-----------|
| Mike | \$200,000 | 42 | no | yes |
| Mary | \$35,000 | 33 | yes | no |
| Claudio | \$115,000 | 40 | no | no |
| Robert | \$29,000 | 23 | yes | yes |
| Dora | \$72,000 | 31 | no | no |



(Provost y Fawcett, 2013)

Decision trees

Un árbol T con M hojas, en el cual la hoja m corresponde a la región R_m que predice el valor f_m

$$f(\mathbf{x}) = \sum_{m=1}^M f_m \pi[\mathbf{x} \in R_m]$$

Donde $\pi = 1$ si $\mathbf{x} \in R_m$ es verdadero.

$f_m = \text{moda de labels } y_i \{y_i | x_i \in R_m\}$

Error $\{0, 1\}$:

$$l(y_0, \mathbf{x}_0, f) = \pi[f(\mathbf{x}_0) \neq y_0]$$

Entropía

Supongamos $y_i = 0, 1$

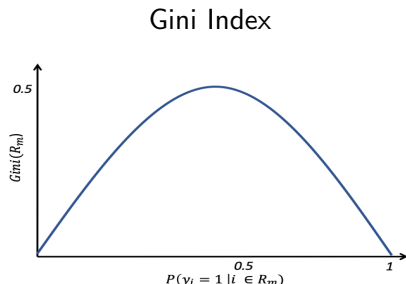
Una medida común de entropía es el índice de Gini.

$$Gini(R_m) = 2p(y = 1|R_m)(1 - p(y = 1|R_m))$$

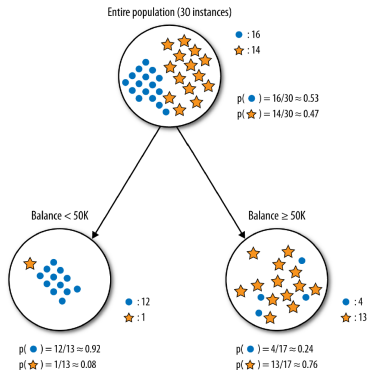
Donde

$$p(y = 1|i \in R_m) = \frac{1}{N_m} \sum_{i \in R_m} y_i$$

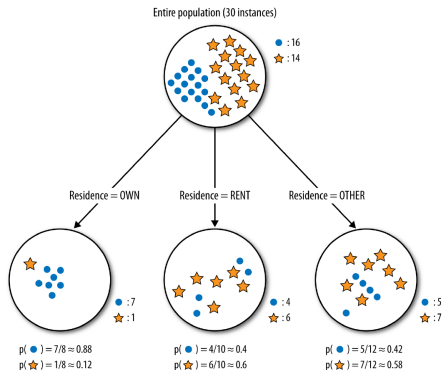
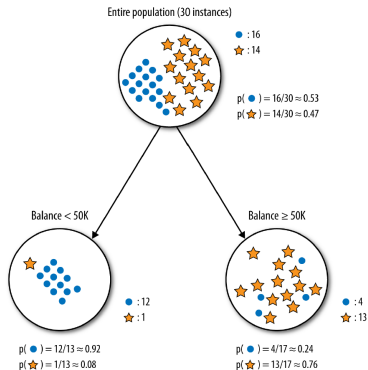
La medida de entropía determina el mejor *split*.



Comparar reducción en entropía



Comparar reducción en entropía



(Provost y Fawcett, 2013)

Algoritmo para llegar a T

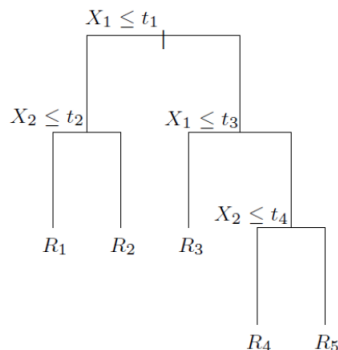
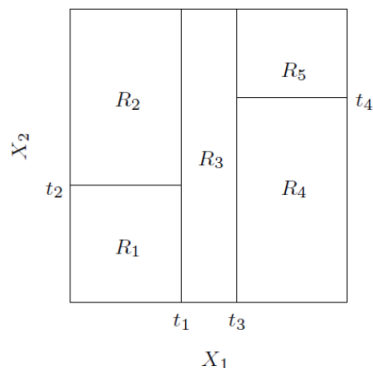
Tenemos d features binarias.

- 1 Obtener *Gini* de todo el *training set*
- 2 Para t en $\{1, 2, maxdepth\}$:
 - 1 Seleccionar $j \in d$ que logran la mayor reducción en Gini.
 - 2 Crear nodo con j como variable de split.
 - 3 Obtener Gini de cada uno de los dos nodos.
 - 4 Repetir para cada nodo.

Criterios para detenerse:

- 1 Cero entropía
- 2 Máxima profundidad alcanzada
- 3 Observaciones en hoja menores a mínimo para hacer split.
- 4 Reducción en GINI menor a mínima reducción en GINI.

Partición del espacio con DT



(Gimpel, 2019)

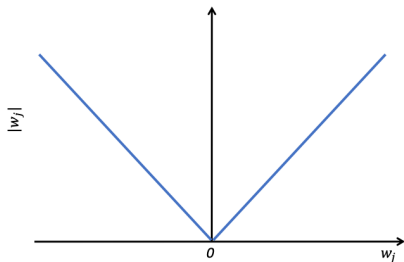
- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - ***LASSO Y Ridge regression***
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- Los Al regularizar penalizamos la complejidad de \mathbf{w}
- Algunas penalizaciones:
 - 1 Número de parámetros: Akaike Information Criterio
 - 2 $\|\mathbf{w}\| = \sum_{j=1}^d |w_j|$. LASSO
 - 3 $\|\mathbf{w}\|^2 = \sum_{j=1}^d w_j^2$. Ridge
- Por método de máxima verosimilitud: Maximizar: $p(\mathbf{y}, \mathbf{X}, \mathbf{w})$, penalizando por el tamaño de \mathbf{w} .

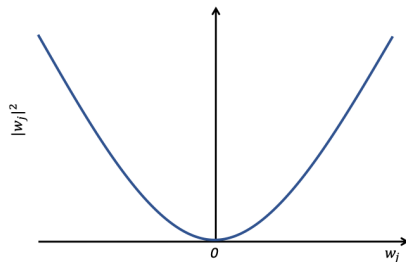
$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \left\{ \frac{1}{n} \sum_{i=1}^n \log p(y_i, \mathbf{x}_i, \mathbf{w}) - \text{penalty}(\mathbf{w}) \right\}$$

LASSO y RIDGE

LASSO



RIDGE



LASSO: *least absolute shrinkage and selection operator*

- LASSO:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|$$

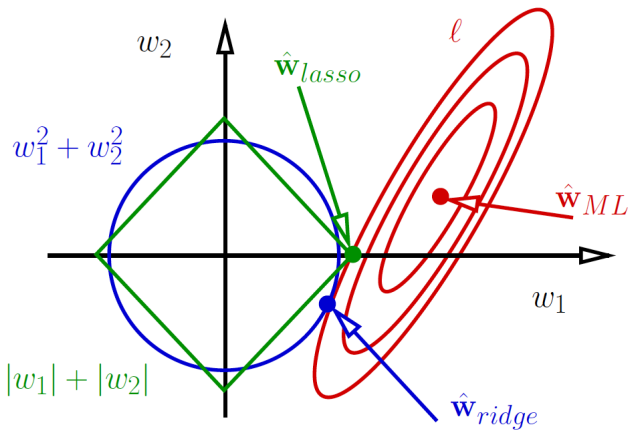
- Ridge:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|^2$$

Importante

Es importante normalizar cada variables $\mathbf{x}_j \in \mathbf{X}$ pues magnitudes afectan $\|\mathbf{w}\|$

Effecto de LASSO y Ridge en coeficientes



(Gimpel, 2019)

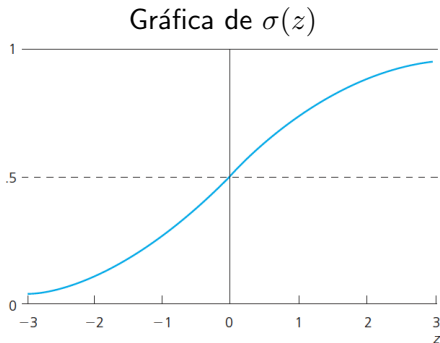
- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - ***Logistic Regression***
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

Logistic Regression

Suponemos $y_i \in \{0, 1\}$

$$p(y_i = 1 | \mathbf{x}_i) = \sigma(\mathbf{w} \cdot \mathbf{x}_i)$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



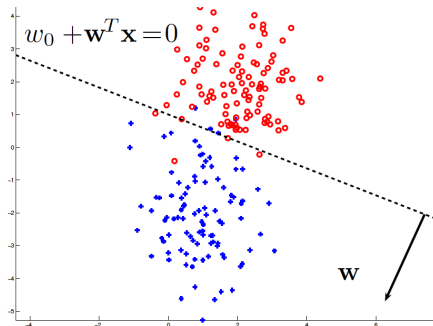
(Wooldridge, 2013)

Partición del espacio

$$f(\mathbf{x}_0 : \mathbf{w}) = \begin{cases} 0 & \sigma(\mathbf{w} \cdot \mathbf{x}_0) \leq 1/2 \\ 1 & \sigma(\mathbf{w} \cdot \mathbf{x}_0) > 1/2 \end{cases}$$

Lo cual genera una frontera lineal:

$$\frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})} = 1/2 \implies \mathbf{w} \cdot \mathbf{x} = 0$$



(Gimpel, 2019)

Máxima verosimilitud

$$p(y_i = 1 | \mathbf{x}_i) = \sigma(\mathbf{w} \cdot \mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)}$$

Log verosimilitud:

$$\log p(\mathbf{y}, \mathbf{X}; \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n (y_i \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i)))$$

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} [-\log p(\mathbf{y}, \mathbf{X}; \mathbf{w})]$$

Con regularización

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \{-\log p(\mathbf{y}, \mathbf{X}; \mathbf{w}) + \lambda \|\mathbf{w}\|^2\}$$

Máxima verosimilitud

$$p(y_i = 1 | \mathbf{x}_i) = \sigma(\mathbf{w} \cdot \mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)}$$

Log verosimilitud:

$$\begin{aligned}\log p(\mathbf{y}, \mathbf{X}; \mathbf{w}) &= \frac{1}{n} \sum_{i=1}^n (y_i \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))) \\ &= \frac{1}{n} \sum_{i=1}^n \log\left(\frac{1}{1 + \exp(-y_i(\mathbf{w} \cdot \mathbf{x}_i))}\right)\end{aligned}$$

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} [-\log p(\mathbf{y}, \mathbf{X}; \mathbf{w})]$$

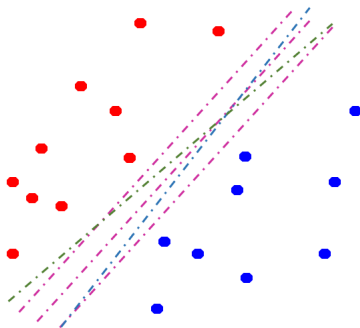
Con regularización

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \{-\log p(\mathbf{y}, \mathbf{X}; \mathbf{w}) + \lambda \|\mathbf{w}\|^2\}$$

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - ***Support Vector Machines***
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

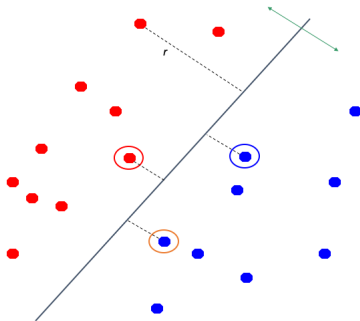
Support vector machines

- ¿Qué frontera seleccionar?



Support vector machines

- ¿Qué frontera seleccionar?

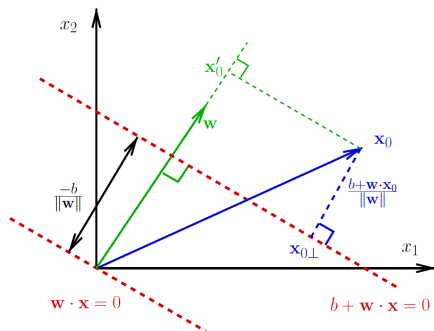


SVM

Seleccionar la frontera que maximiza la distancia a los puntos más cercanos clasificados correctamente.

La frontera con el margen máximo

Proyecciones



- Distancia de una observación x_i correctamente clasificada:

$$\frac{1}{\|w\|} y_i (w \cdot x_i + b)$$

- Distancia a la observación mas cercana:

$$\min_i \frac{1}{\|w\|} y_i (w \cdot x_i + b)$$

(Gimpel, 2019)

$$w^* = \operatorname{argmax}_w \left\{ \frac{1}{\|w\|} \min_i y_i (w \cdot x_i + b) \right\}$$

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \right\}$$

Podemos reescalar \mathbf{w} y b para que

$$\min_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

. Por lo tanto,

$$\begin{aligned} \mathbf{w}^* &= \operatorname{argmax}_{\mathbf{w}} \left\{ \frac{1}{\|\mathbf{w}\|} \right\} \\ &= \operatorname{argmin}_{\mathbf{w}} \|\mathbf{w}\|^2 \end{aligned}$$

Datos no separables linealmente

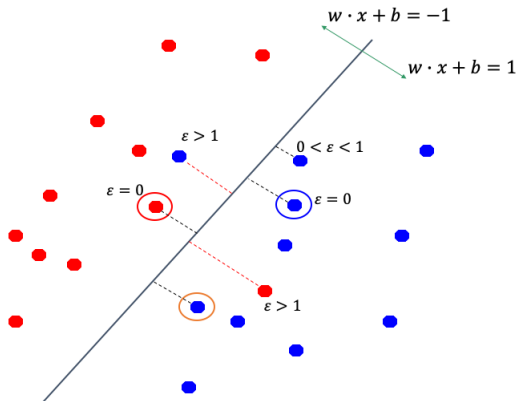
Imposible lograr que

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

Introducimos *slack* variables

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

$$\xi_i = \max[0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)]$$



$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \{ \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \}$$

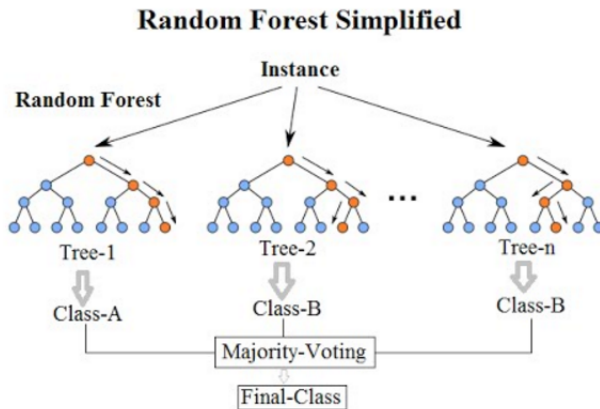
- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - ***Ensembles***
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- En vez de construir un solo modelo f^* , construir M modelos y combinarlos.
- Objetivo: Mejorar precisión sobre nuevas observaciones.
- Agregación más sencilla: Promedio

$$\hat{f}(\mathbf{x}) = \frac{1}{M} \sum_{j=1}^M f_j(\mathbf{x})$$

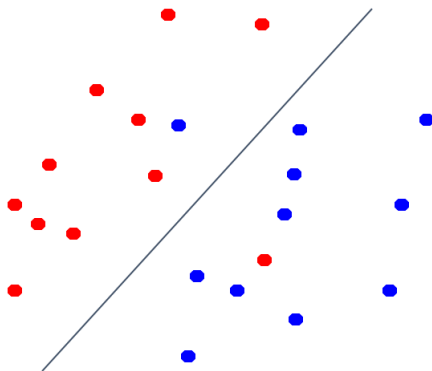
Random Forests

- Construir M decision trees
- Entrenar cada árbol T_m usando una muestra aleatoria del *training* set. Usar muestras con reemplazo.
- En cada nodo, únicamente considerar b de las d variables.



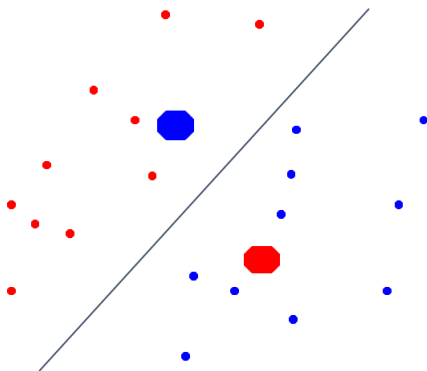
Boosting

- 1 Obtener f_t para todo el *training* set.
- 2 Evaluar precision de f_t sobre *training* set.
- 3 Incrementar peso de observaciones misclasificadas usando factor $1 + r$.
- 4 Estimar f_{t+1} utilizando pesos ajustados.
- 5 Repetir hasta $t + i = T$



Boosting

- 1 Obtener f_t para todo el *training* set.
- 2 Evaluar precision de f_t sobre *training* set.
- 3 Incrementar peso de observaciones misclasificadas usando factor $1 + r$.
- 4 Estimar f_{t+1} utilizando pesos ajustados.
- 5 Repetir hasta $t + i = T$



- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID

- 1 Introducción a *machine learning*
- 2 Principales algoritmos, metodología e intuición
 - *K Nearest Neighbors*
 - *Decision Trees*
 - *LASSO Y Ridge regression*
 - *Logistic Regression*
 - *Support Vector Machines*
 - *Ensembles*
- 3 Aplicaciones
 - Reduciendo intoxicaciones por plomo
 - Algoritmos como jueces
 - Mapeando pobreza
 - Evaluando MEDICAID