

Introducción al aprendizaje automático

...

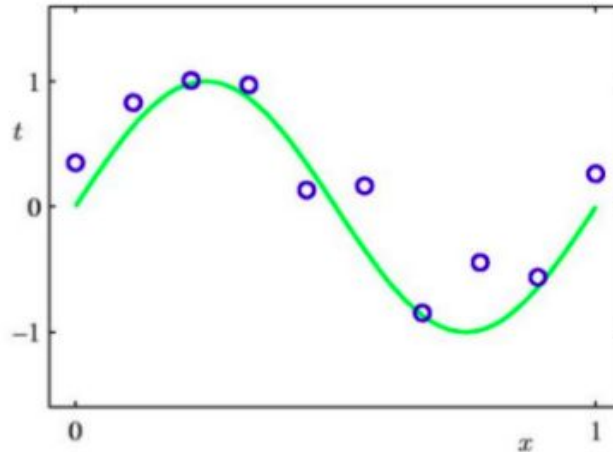
#2. Modelos probabilísticos y no paramétricos

Regresión

- Disponemos de N pares de entrenamiento (observaciones)

$$\{(x_i, y_i)\}_{i=1}^N = \{(x_1, y_1), \dots, (x_N, y_N)\}$$

- El problema de regresión consiste en estimar $f(x)$ a partir de estos datos



Regresión polinomial

- **Función de predicción lineal:**

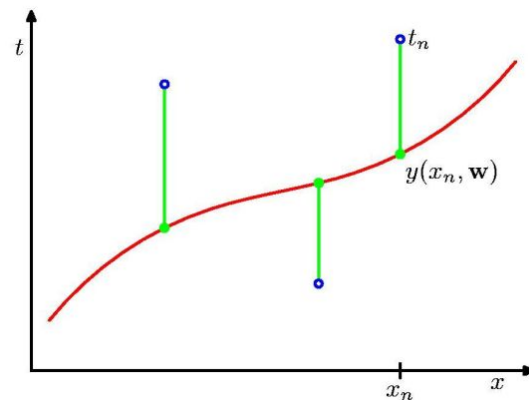
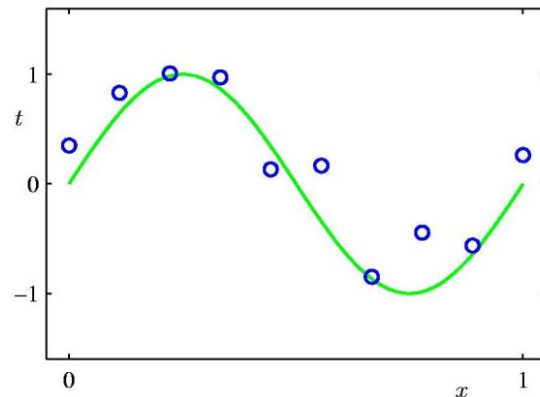
$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M$$

- **Función de costo:** error cuadrático

medida del error en la predicción de t mediante $y(x; w)$

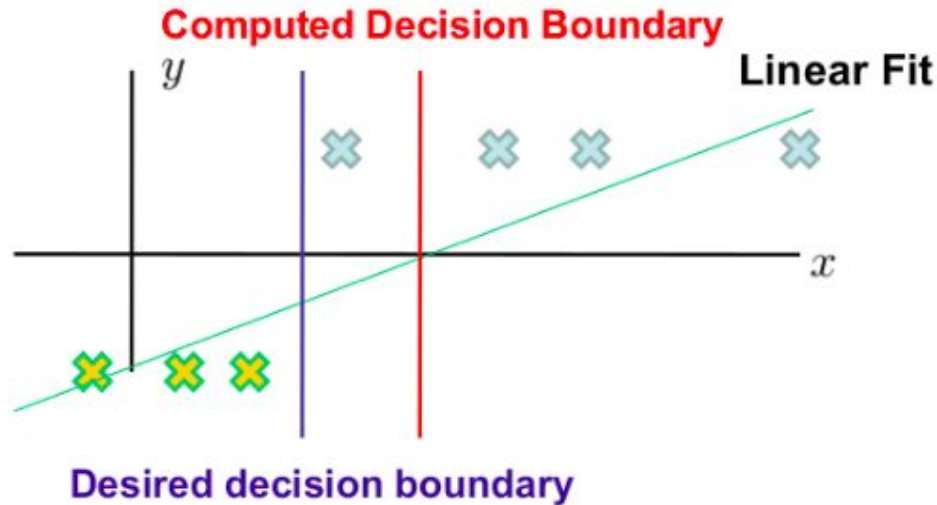
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

- ¿Se podría aplicar lo mismo en clasificación?



Error cuadrático en clasificación

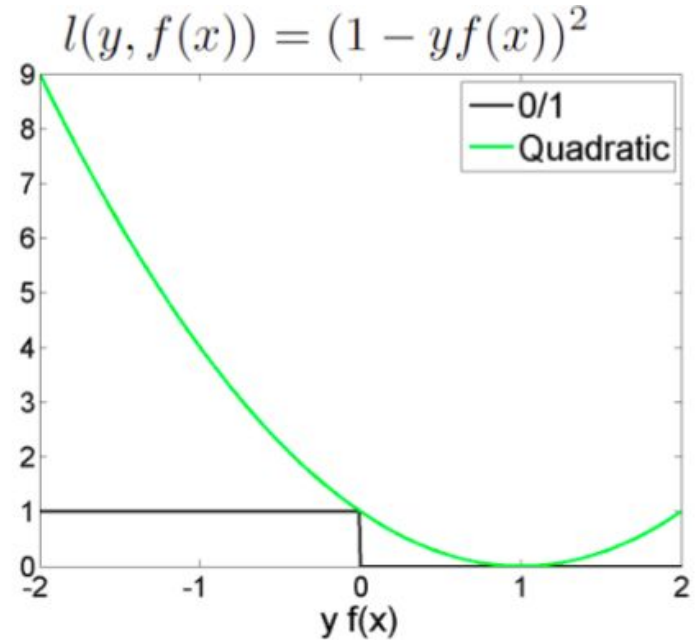
- Mínimo global único y solución en forma cerrada
- Pero, ¿es una medida del error de clasificación? ¿es adecuada?



Error cuadrático en clasificación

$$y_{\pm} \in \{-1, 1\}$$

$$\begin{aligned} l(y, f(x)) &= (y - f(x))^2 \\ &\stackrel{y^2=1}{=} y^2(y - f(x))^2 \\ &= (y^2 - yf(x))^2 \\ &\stackrel{y^2=1}{=} (1 - yf(x))^2 \end{aligned}$$



- No es robusta frente a *outliers*
- Penaliza predicciones que son muy buenas

Regresión logística

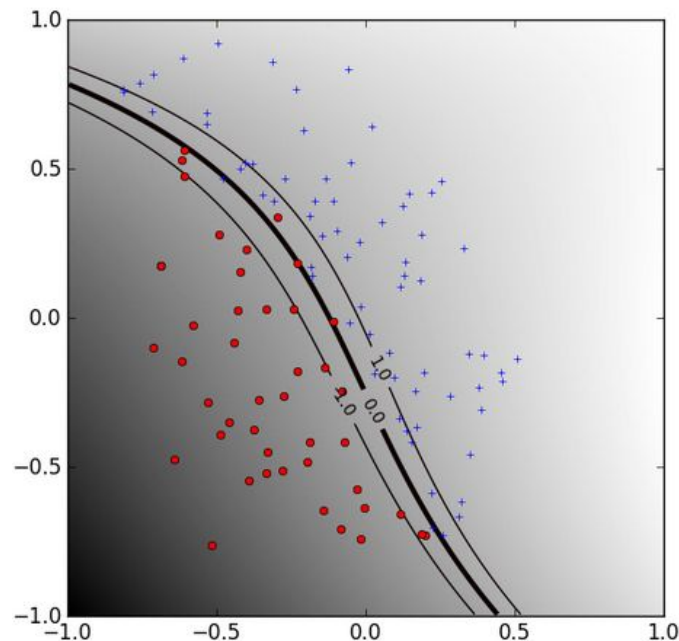
Clasificación basada en probabilidades

- Objetivo: dar la probabilidad de que una instancia x sea de una clase y , es decir, aprender $p(y|x)$

- Recordar:

$$0 \leq p(\text{evento}) \leq 1$$

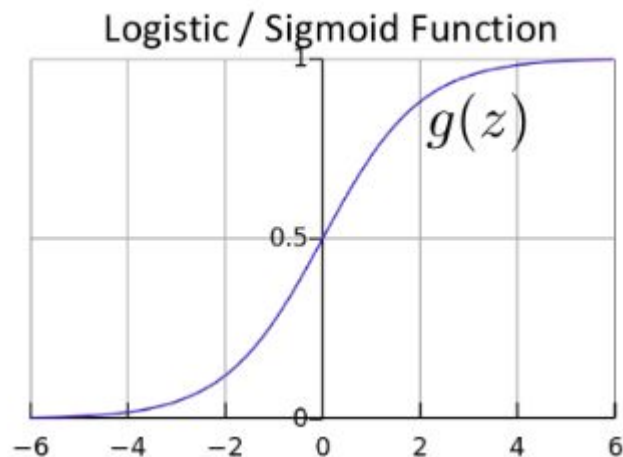
$$p(\text{evento}) + p(\neg \text{evento}) = 1$$



Regresión logística

- Aproximación probabilística al problema de clasificación
- La función de predicción $h_w(x)$ debe dar una aproximación de $p(y=1|x,w)$
- $0 \leq h_w(x) \leq 1$

$$h_w(x) = g(w^T x) = \frac{1}{1 + \exp(-w^T x)}$$



Regresión logística

- Given $\left\{ \left(\mathbf{x}^{(1)}, y^{(1)} \right), \left(\mathbf{x}^{(2)}, y^{(2)} \right), \dots, \left(\mathbf{x}^{(n)}, y^{(n)} \right) \right\}$
where $\mathbf{x}^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^{\top} \mathbf{x})$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

$$\mathbf{x}^{\top} = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

Regresión logística. Función de costo

- Can't just use squared loss as in linear regression:

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right)^2$$

- Using the logistic regression model

$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-\theta^{\top} \mathbf{x}}}$$

results in a non-convex optimization

Regresión logística. Función de costo

Training set: $\{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$, $\mathbf{x} \in R^M$, $y \in \{0, 1\}$

y: discrete observations: model as samples from Bernoulli distribution

$$P(y = 1|\mathbf{x}, \mathbf{w}) = f(\mathbf{x}, \mathbf{w})$$

$$P(y = 0|\mathbf{x}, \mathbf{w}) = 1 - f(\mathbf{x}, \mathbf{w})$$

$$P(y|\mathbf{x}) = (f(\mathbf{x}, \mathbf{w}))^y (1 - f(\mathbf{x}, \mathbf{w}))^{1-y}$$

Find \mathbf{w} that maximizes the likelihood of labels in the training set

$$\begin{aligned} -L(\mathbf{w}) = C(\mathbf{w}) &= \log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{i=1}^N \log P(y^i|\mathbf{x}^i, \mathbf{w}) \\ &= \sum_i y^i \log f(\mathbf{x}^i, \mathbf{w}) + (1 - y^i) \log(1 - f(\mathbf{x}^i, \mathbf{w})) \end{aligned}$$

Intuition Behind the Objective

$$J(\boldsymbol{\theta}) = - \sum_{i=1}^n \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) \right]$$

- Cost of a single instance:

$$\text{cost}(h_{\boldsymbol{\theta}}(\mathbf{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$

- Can re-write objective function as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^n \text{cost}(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), y^{(i)})$$

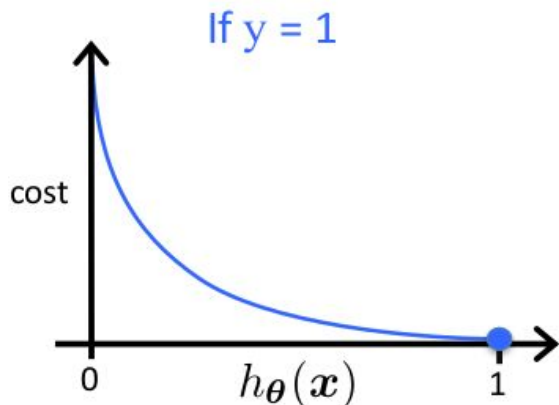
Compare to linear regression: $J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$

Intuition Behind the Objective

$$\text{cost}(h_{\theta}(\mathbf{x}), y) = \begin{cases} -\log(h_{\theta}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$

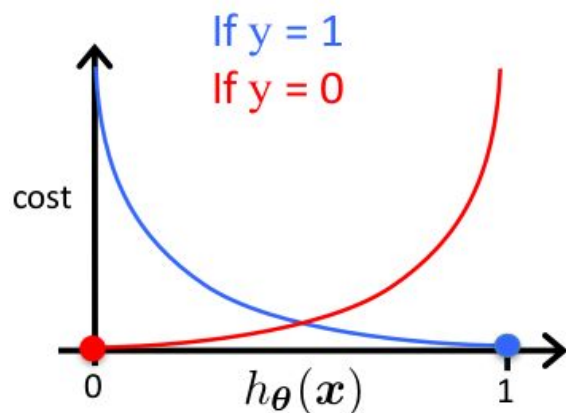
If $y = 1$

- Cost = 0 if prediction is correct
- As $h_{\theta}(\mathbf{x}) \rightarrow 0$, $\text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties
 - e.g., predict $h_{\theta}(\mathbf{x}) = 0$, but $y = 1$



Intuition Behind the Objective

$$\text{cost}(h_{\theta}(\mathbf{x}), y) = \begin{cases} -\log(h_{\theta}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$



If $y = 0$

- Cost = 0 if prediction is correct
- As $(1 - h_{\theta}(\mathbf{x})) \rightarrow 0$, $\text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties

Regularized Logistic Regression

$$J(\boldsymbol{\theta}) = - \sum_{i=1}^n \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) \right]$$

- We can regularize logistic regression exactly as before:

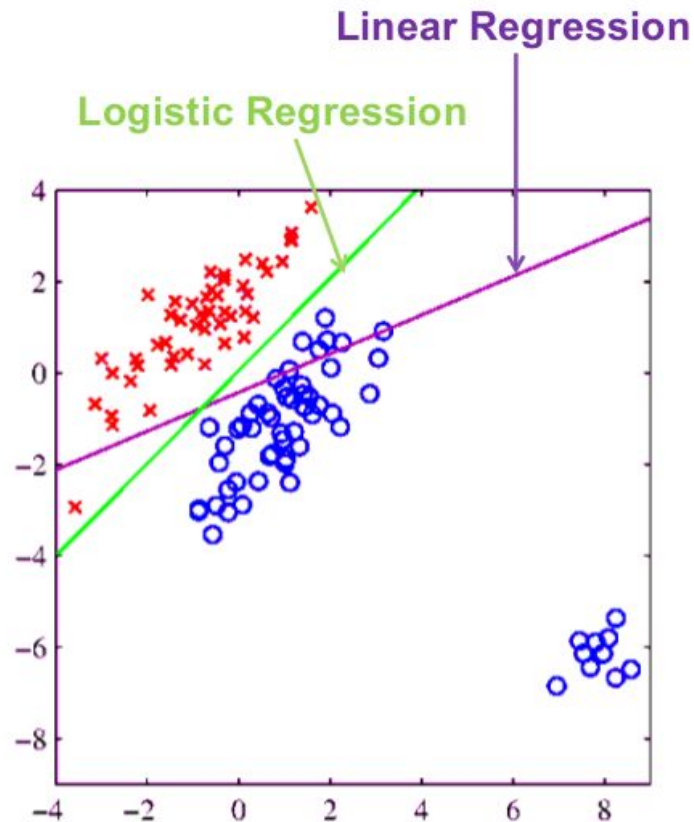
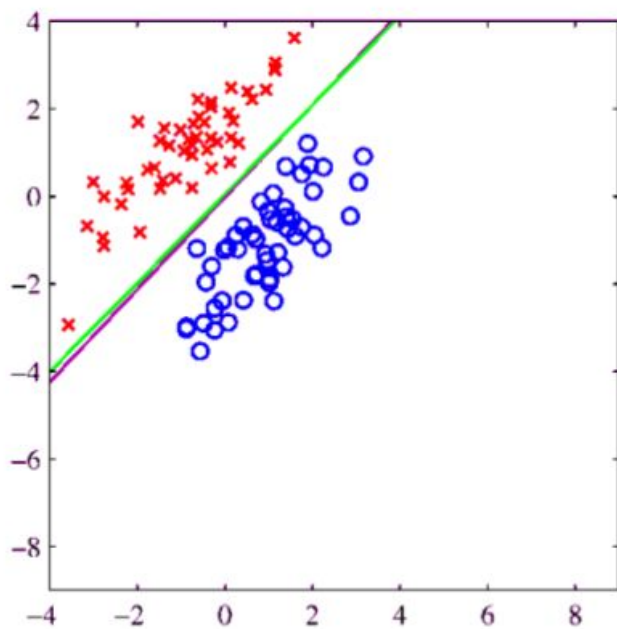
$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda \sum_{j=1}^d \theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

 [1:d] => exclude the bias!

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Logistic vs Linear Regression


Logistic regression is more robust



Multi-Class Logistic Regression

- For 2 classes:

$$h_{\theta}(x) = \frac{1}{1 + \exp(-\theta^T x)} = \frac{\exp(\theta^T x)}{\boxed{1} + \boxed{\exp(\theta^T x)}}$$



weight assigned to $y = 0$ weight assigned to $y = 1$

- For C classes $\{1, \dots, C\}$:

$$p(y = c \mid x; \theta_1, \dots, \theta_C) = \frac{\exp(\theta_c^T x)}{\sum_{c=1}^C \exp(\theta_c^T x)}$$

– Called the **softmax** function

Implementing Multi-Class Logistic Regression

- Use $h_c(\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_c^\top \mathbf{x})}{\sum_{c=1}^C \exp(\boldsymbol{\theta}_c^\top \mathbf{x})}$ as the model for class c
- Gradient descent simultaneously updates all parameters for all models
 - Same derivative as before, just with the above $h_c(\mathbf{x})$
- Predict class label as the most probable label

$$\max_c h_c(\mathbf{x})$$

Naïve Bayes

Regla de Bayes

- Dos formas de factorizar una distribución en dos variables:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

- Operando:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

- ¿Porqué es útil?
 - Nos permite "revertir" el condicional
 - A veces una dirección es difícil de calcular, pero la otra no
 - Es la base de muchos modelos



El clasificador de Bayes

- Distribución conjunta sobre X_1, \dots, X_n e Y
- Podemos definir una función de predicción de la forma:

$$\arg \max_Y P(Y|X_1, \dots, X_n)$$

- por ejemplo: ¿cuál es la probabilidad de que una imagen represente un "5" dado el valor de sus píxeles?
- Problema: ¿cómo computamos $P(Y|X_1, \dots, X_n)$? ...

El clasificador de Bayes

- ... ¡Usando regla de Bayes!

$$P(Y|X_1, \dots, X_n) = \frac{\overset{\text{Likelihood}}{P(X_1, \dots, X_n|Y)} \overset{\text{Prior}}{P(Y)}}{\underset{\text{Normalization Constant}}{P(X_1, \dots, X_n)}}$$

- Ahora podemos pensar en modelar cómo los píxeles de la imagen son "generados" dado el número "5".

Naïve Bayes

- Hipótesis: los X_i son independientes dado Y

$$\begin{aligned}P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\ &= P(X_1|Y)P(X_2|Y)\end{aligned}$$

- O en forma más general:

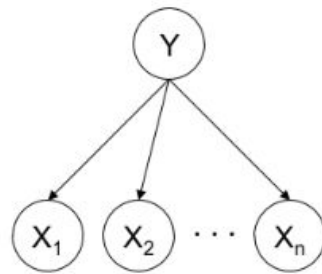
$$P(X_1 \dots X_n | Y) = \prod_i P(X_i | Y)$$

- Si los X_i consisten en n valores binarios, ¿cuántos parámetros necesito especificar para $P(X_i | Y)$?

El clasificador naïve Bayes

- Dado:
 - Distribución a priori $P(Y)$
 - n features X_i condicionalmente independientes dada la clase Y

- Para cada X_i , especificar $P(X_i | Y)$

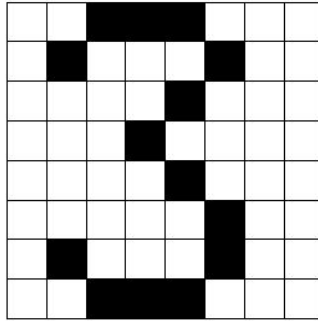


- Función de decisión:

$$\begin{aligned} y^* = h_{NB}(\mathbf{x}) &= \arg \max_y P(y) P(x_1, \dots, x_n | y) \\ &= \arg \max_y P(y) \prod_i P(x_i | y) \end{aligned}$$

Ejemplo: reconocimiento de dígitos

- Input: pixel grids

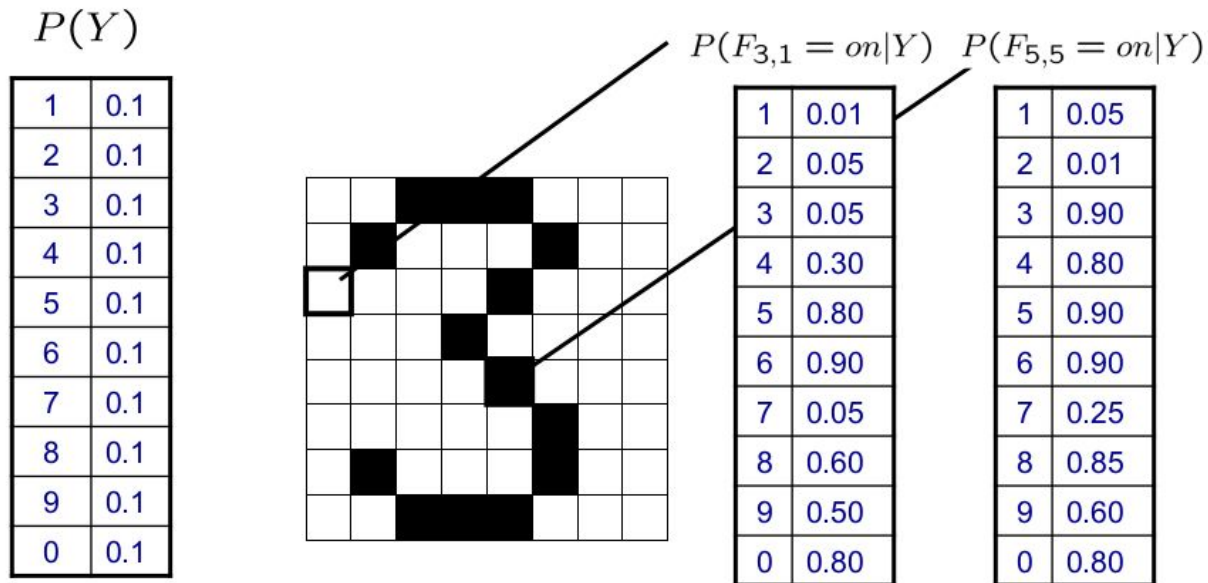


- Output: a digit 0-9



Pregunta: ¿cuán realista es la hipótesis del clasificador naïve Bayes en este ejemplo?

Ejemplo: reconocimiento de dígitos



Estimación de parámetros por MV

- Dado un conjunto de datos, obtener $\text{Count}(A=a, B=b)$, es decir, el número de ejemplos en donde $A=a$ y $B=b$.
- MV para naïve Bayes sobre variables discretas:
 - Prior:

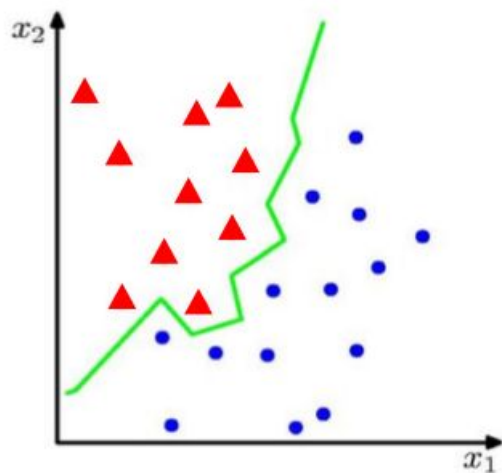
$$P(Y = y) = \frac{\text{Count}(Y = y)}{\sum_{y'} \text{Count}(Y = y')}$$

- Distribución condicionales (observación):

$$P(X_i = x | Y = y) = \frac{\text{Count}(X_i = x, Y = y)}{\sum_{x'} \text{Count}(X_i = x', Y = y)}$$

Modelos no paramétricos: vecinos más cercanos

Classification



- Suppose we are given a training set of N observations

(x_1, \dots, x_N) and (y_1, \dots, y_N) , $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$

- Classification problem is to estimate $f(x)$ from this data such that

$$f(x_i) = y_i$$

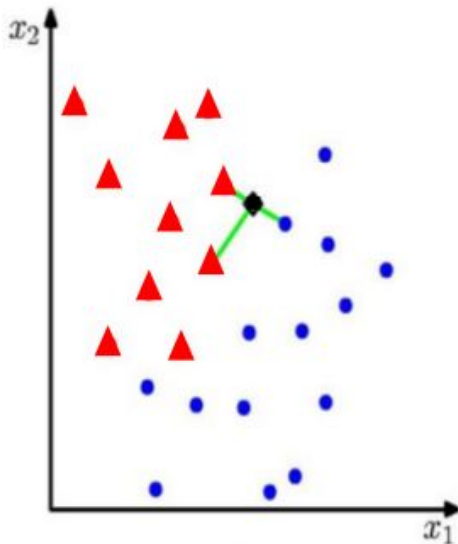
K Nearest Neighbour (K-NN) Classifier

Algorithm

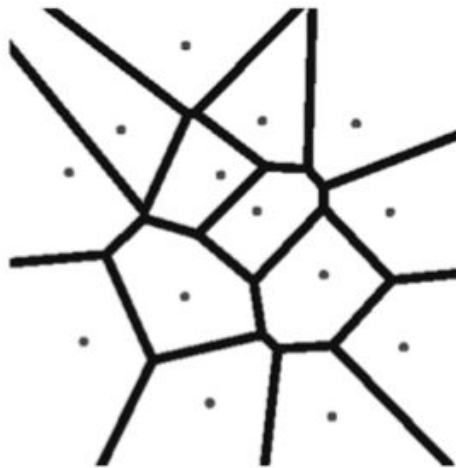
- For each test point, x , to be classified, find the K nearest samples in the training data
- Classify the point, x , according to the majority vote of their class labels

e.g. $K = 3$

- applicable to multi-class case

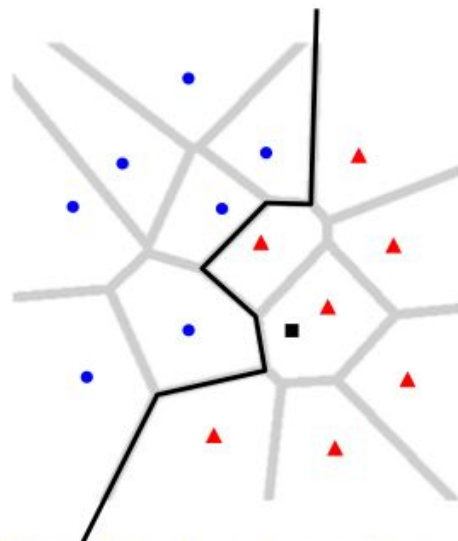


$$K = 1$$



Voronoi diagram:

- partitions the space into regions
- boundaries are equal distance from training points



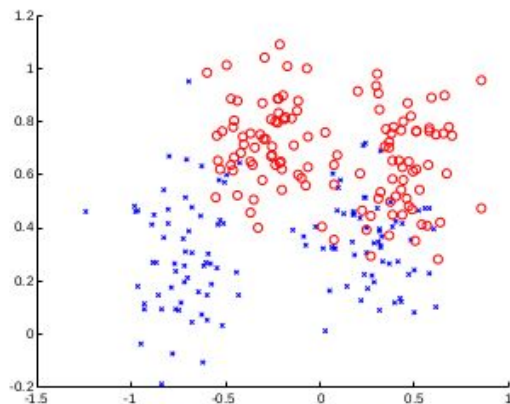
Classification boundary:

- non-linear

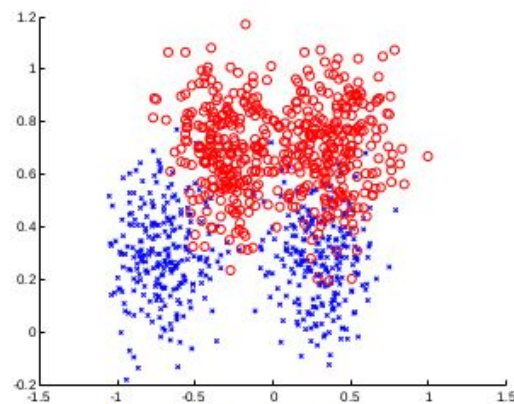
A sampling assumption: training and test data

- Assume that the training examples are drawn independently from the set of all possible examples.
- This makes it very unlikely that a strong regularity in the training data will be absent in the test data.

- Measure classification error as $= \frac{1}{N} \sum_{i=1}^N \underbrace{[y_i \neq f(\mathbf{x}_i)]}_{\text{loss function}}$ The “risk”



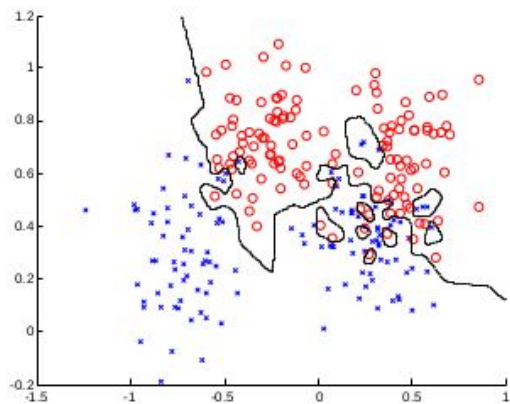
Training data



Testing data

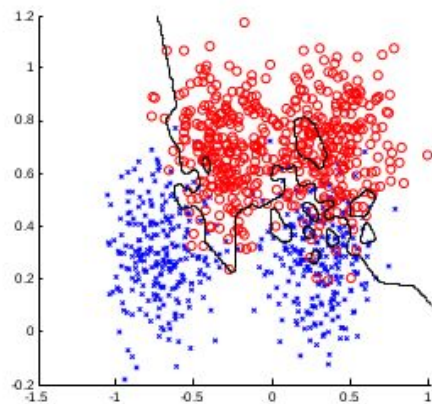
$K = 1$

Training data



error = 0.0

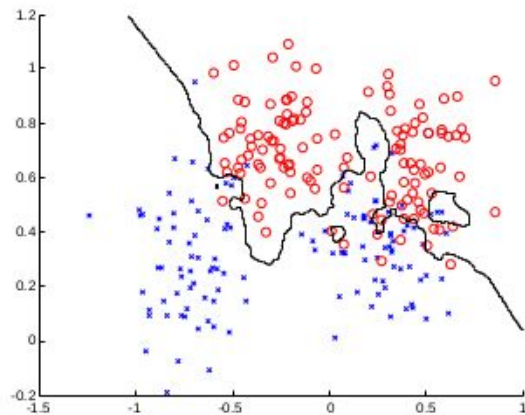
Testing data



error = 0.15

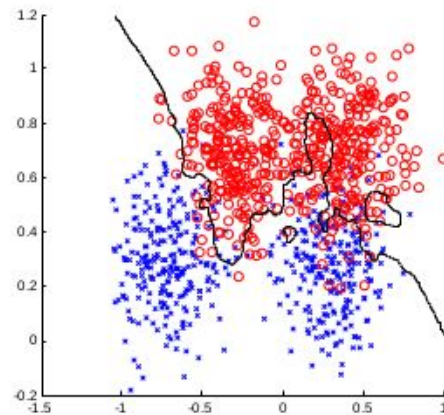
$K = 3$

Training data



error = 0.0760

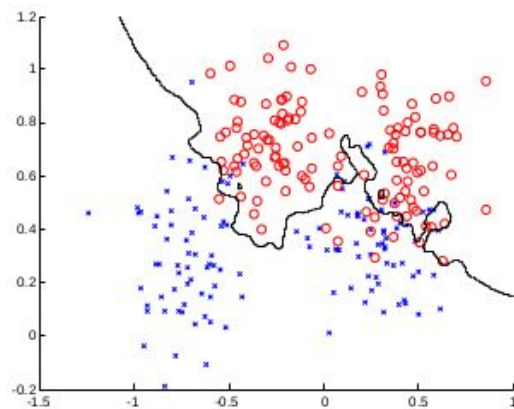
Testing data



error = 0.1340

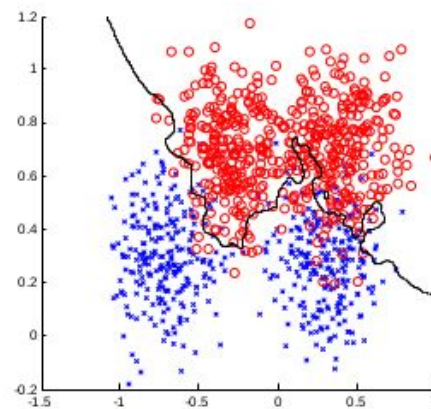
$K = 7$

Training data



error = 0.1320

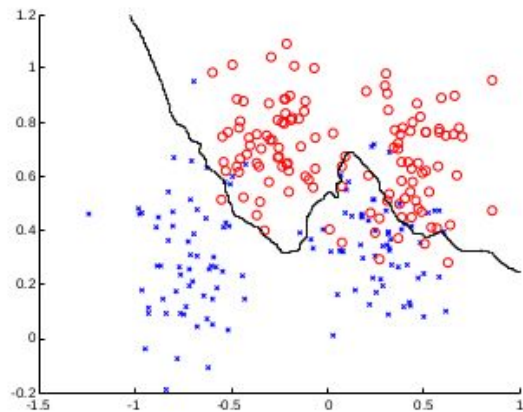
Testing data



error = 0.1110

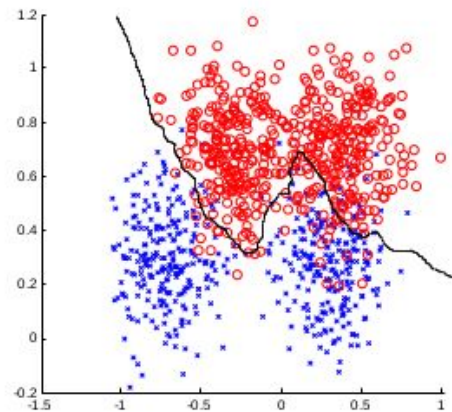
$K = 21$

Training data



error = 0.1120

Testing data



error = 0.0920

Properties and training

As K increases:

- Classification boundary becomes smoother
- Training error can increase

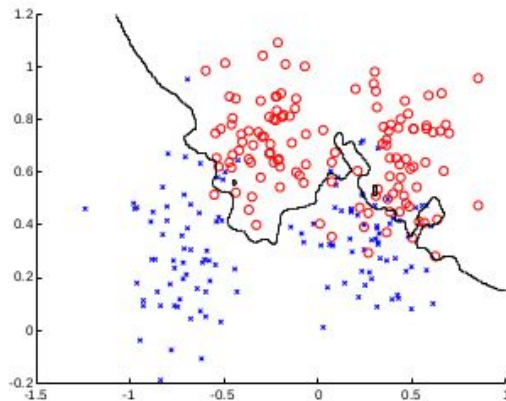
Choose (learn) K by cross-validation

- Split training data into training and validation
- Hold out validation data and measure error on this

Summary

Advantages:

- K-NN is a simple but effective classification procedure
- Applies to multi-class classification
- Decision surfaces are non-linear
- Quality of predictions automatically improves with more training data
- Only a single parameter, K ; easily tuned by cross-validation



Summary

Disadvantages:

- What does nearest mean? Need to specify a distance metric.
- Computational cost: must **store** and **search** through the entire training set at test time. Can alleviate this problem by thinning, and use of efficient data structures like KD trees.

