Introducción al aprendizaje automático

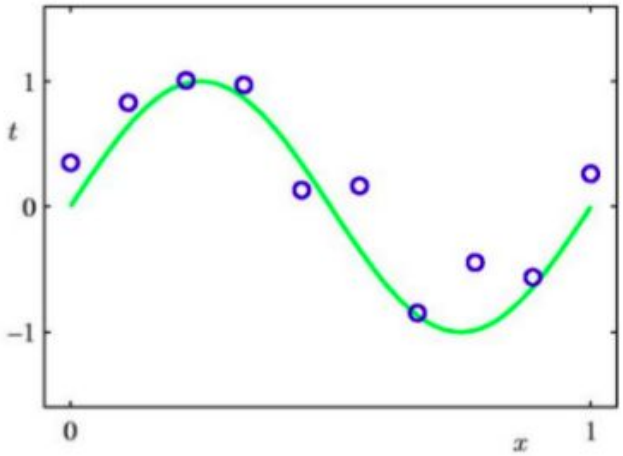
#2. Modelos probabilísticos y no paramétricos

Regresión

- Disponemos de N pares de entrenamiento (observaciones)

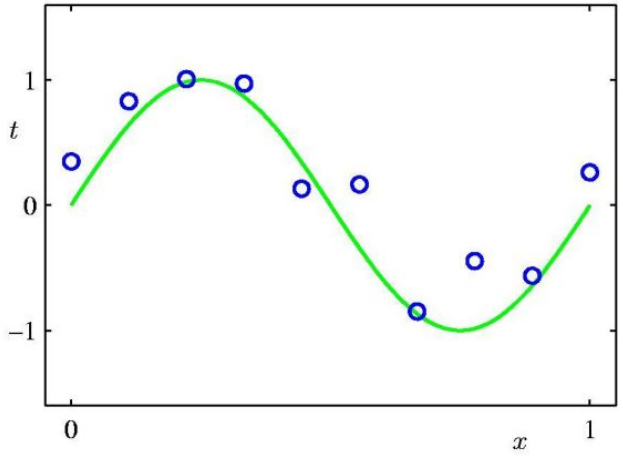
$$\{(x_i, y_i)\}_{i=1}^{N} = \{(x_1, y_1), \cdots, (x_N, y_N)\}$$

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



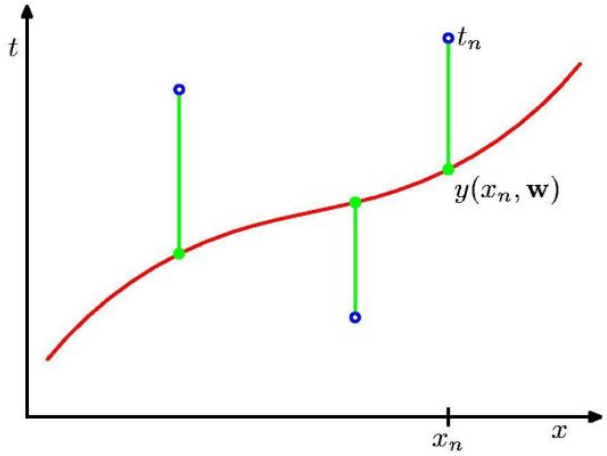
Regresión polinomial

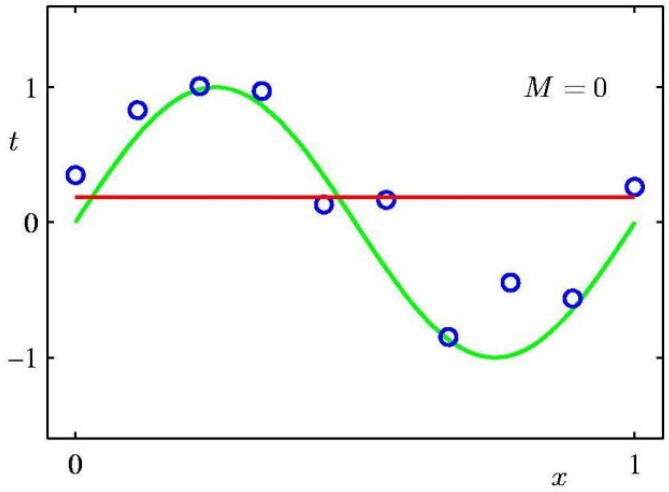
- En verde se ilustra la función "verdadera" (inaccesible)
- Las muestras son uniformes en x y poseen ruido en y
- Utilizaremos una **<u>función de costo</u>** (error cuadrático) que mida el error en la predicción de y mediante $f(x)$

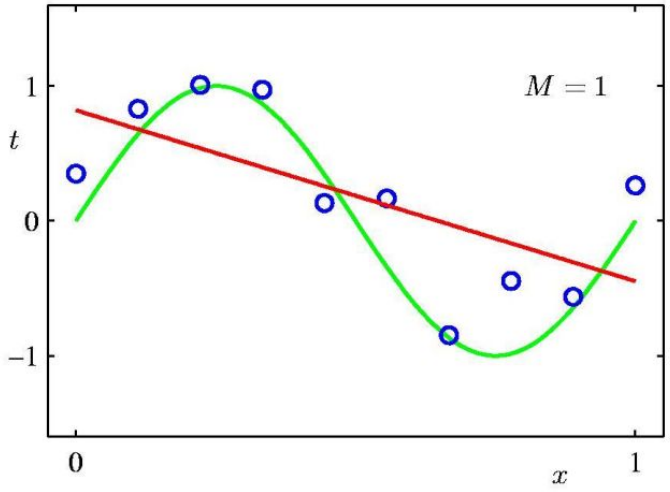


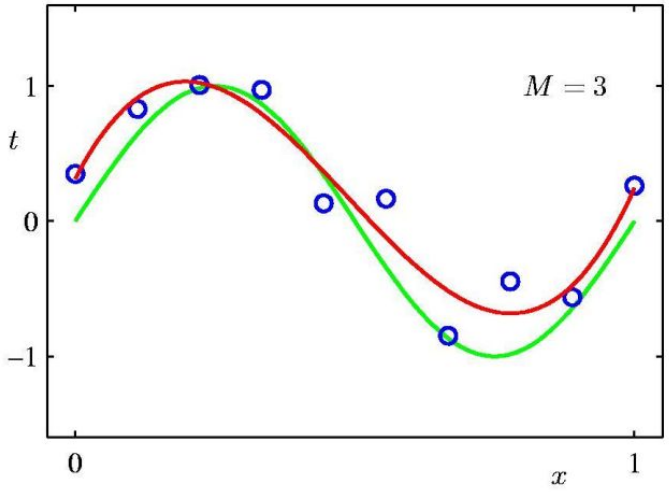
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

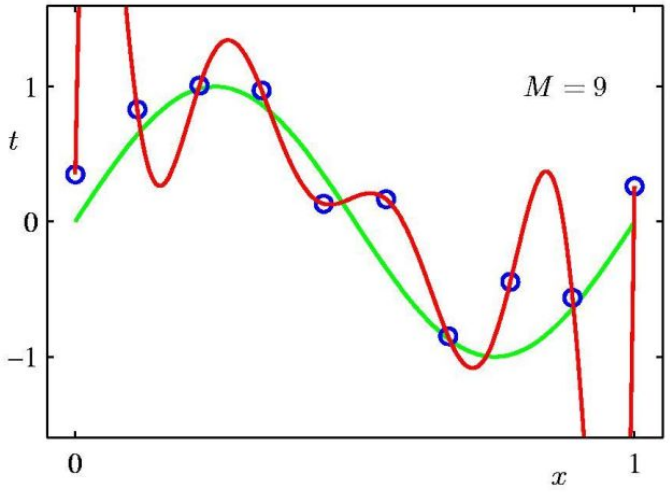
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$




M = 0
t
x
1
0
−1
0
1

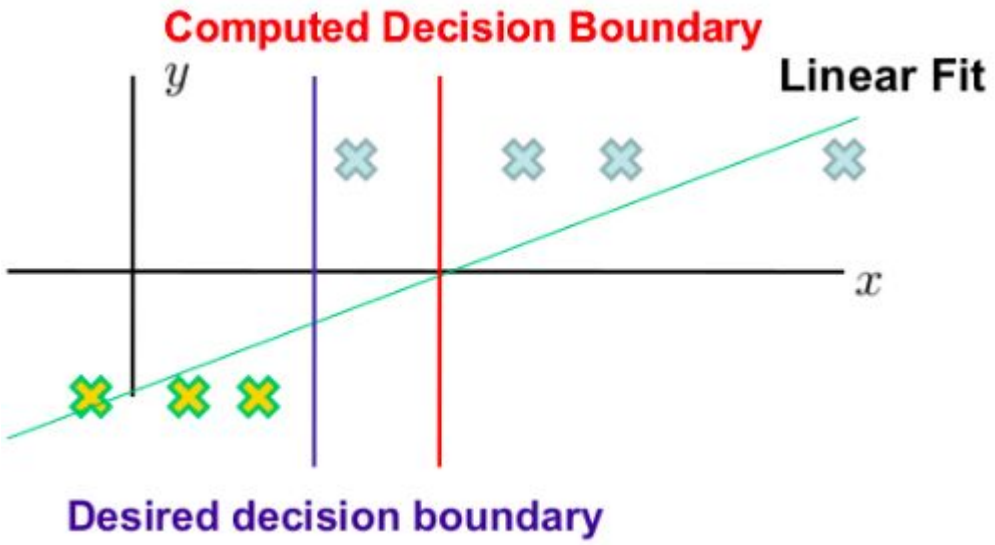

M = 1
t
x
1
0
−1
0
1


M = 3
t
x
1
0
−1
0
1


M = 9
t
x
1
0
−1
0
1

Error cuadrático en clasificación

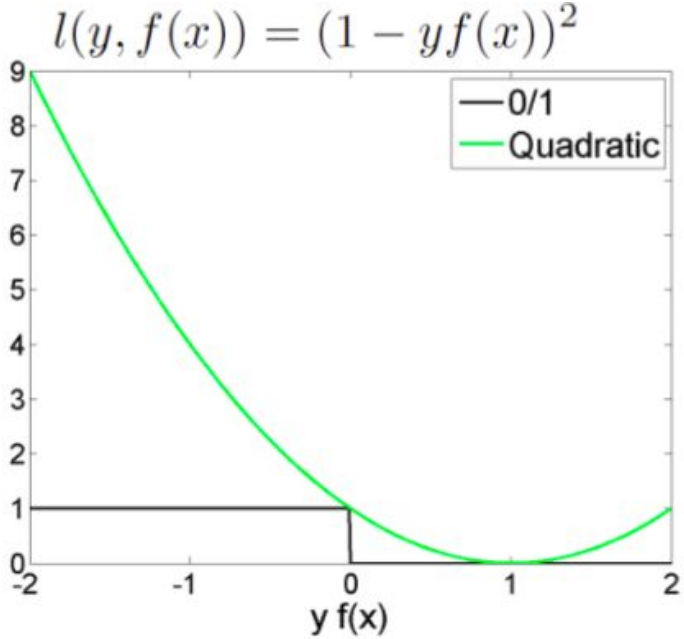
- Mínimo global único y solución en forma cerrada
- Pero, ¿es una medida del error de clasificación? ¿es adecuada?



Error cuadrático en clasificación

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

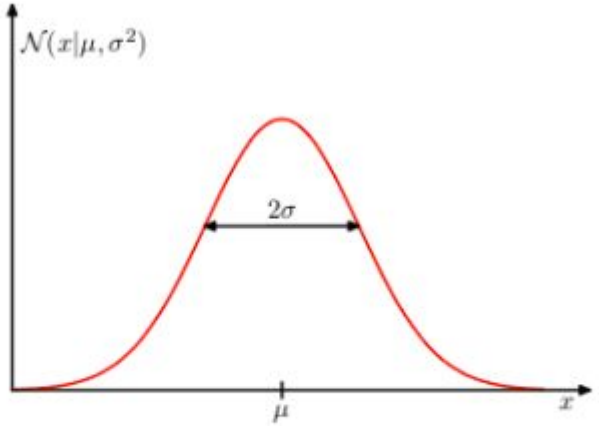
$$\begin{aligned} l(y, f(x)) &= (y - f(x))^2 \\ &\overset{y^2=1}{=} y^2(y - f(x))^2 \\ &= (y^2 - yf(x))^2 \\ &\overset{y^2=1}{=} (1 - yf(x))^2 \end{aligned}$$



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

Regresión polinomial. Solución por MV

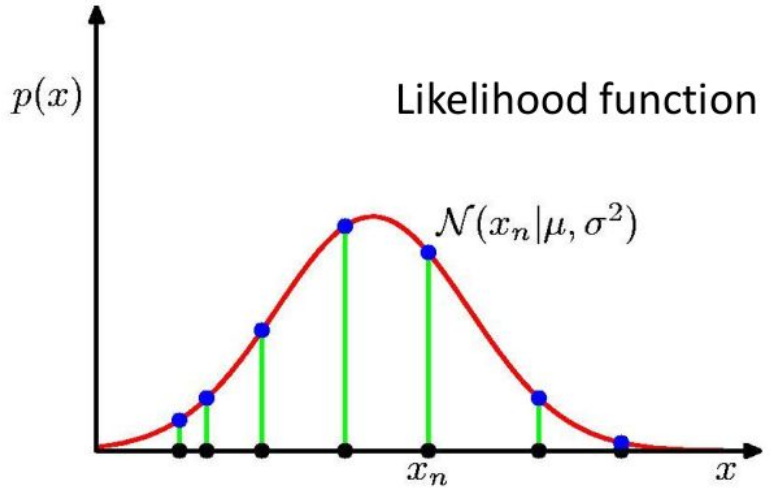
Distribución gaussiana



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp\Big\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \Big\}$$

- Siempre positiva, integra a 1
- precisión $\beta = 1/\sigma^2$
- valor esperado $\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu,\sigma^2) x \, \mathrm{d}x = \mu$
- varianza $\mathrm{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2$

Máxima verosimilitud (MV)



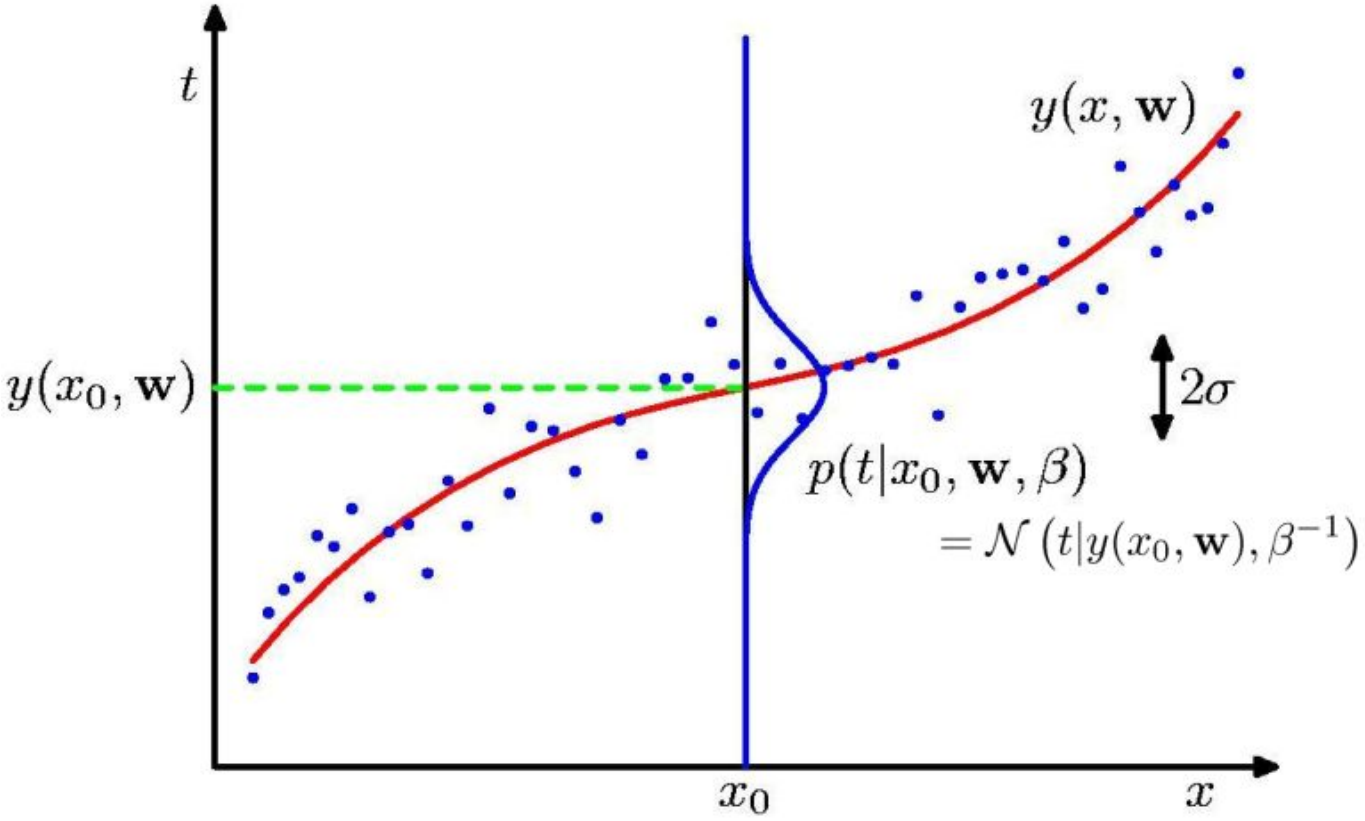
- Muestras iid
- Función de verosimilitud $p(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu,\sigma^2)$
- Logaritmo de la función de verosimilitud $\ln p(\mathbf{x}|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\sigma)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$
- Media muestral por MV $\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n$
- Varianza muestral por MV $\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{ML})^2$

Revisando el ajuste de curvas

- Objetivo: predecir valores de salida t para nuevas entradas x, en base a un conjunto de pares de entrenamiento (x_1,t_1), … , (x_N,t_N).

- Para capturar la incertidumbre sobre los valores de salida, podemos asumir que, dado un x, el valor de t se genera a partir de una gaussiana de media $y(x; w)$ (la curva polinomial)

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right)$$


t
y(x, w)
y(x0, w)
2σ
p(t|x0, w, β)
= N (t|y(x0, w), β−1)
x0
x

Probabilidades bayesianas

- Conocimiento "a priori" sobre los parámetros en $\mathrm{p}(w)$ (*prior*)
- Efecto de las observaciones $D=\{t_1, \ldots t_N\}$ en el proceso de inferencia sobre w se expresa mediante $\mathrm{p}(w|D)$ (*likelihood*)
- La incertidumbre sobre w <u>después</u> de observar D (*posterior*)

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

- El denominador $\mathrm{p}(D)$ es un factor de normalización

Revisando el ajuste de curvas

- Entrenamiento por MV, asumiendo muestras iid y distribución $\mathrm{p}(t|x,w,\beta)$:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|y(x_n, \mathbf{w}, \beta^{-1}\right)$$

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)$$

- La solución por MV, después de notar que los últimos dos términos no dependen de w y que β es un factor de escala, se obtiene de forma equivalente minimizando el error cuadrático medio:

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

Revisando el ajuste de curvas

- También podemos utilizar MV para estimar β:

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

- Con w y β podemos hacer predicciones sobre x mediante la "distribución predictiva"

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1}\right)$$

- Si consideramos un *prior* Gaussiano sobre w

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2}\exp\left\{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\}$$

Máximo a posteriori (MAP)

- Posterior ∝ likelihood × prior

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

- Tomando el logaritmo de la función de verosimilitud de $\mathrm{p}(w|x, t, \alpha, \beta)$ y considerando como antes sólo los términos que dependen de w

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

resulta en error cuadrático con regularización L_2 de parámetro $\lambda=\alpha/\beta$

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:

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

- ¿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 $X_1, \ldots, X_n$ e Y

- Podemos definir una función de predicción de la forma:

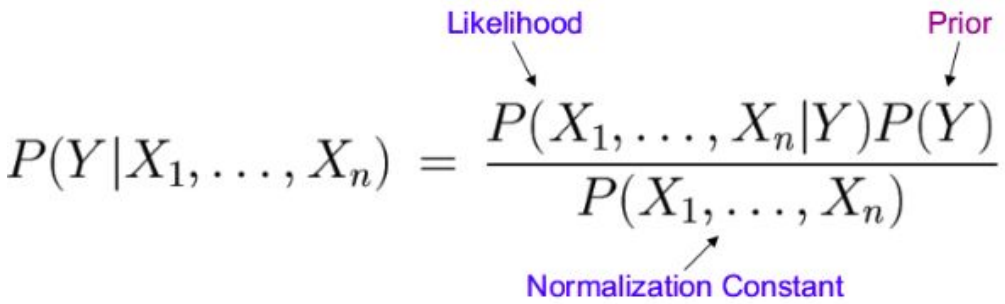
$$\arg\max_Y P(Y|X_1, \ldots, X_n)$$

  - por ejemplo: ¿cuál es la probabilidad de que una imagen represente un "5" dado el valor de sus píxeles?

- Problema: ¿cómo computamos $\mathrm{P}(Y|X_1, \ldots, X_n)$? …

El clasificador de Bayes

- … ¡Usando regla de Bayes!



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

Naïve Bayes

- Hipótesis: los X_i son independientes dado Y

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$
$$= P(X_1|Y)P(X_2|Y)$$

- O en forma más general:

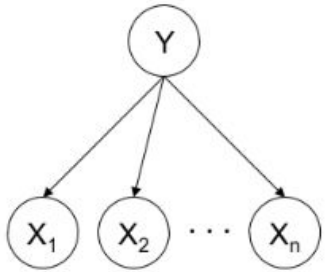
$$P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)$$

- 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 $\mathrm{P}(Y)$
  - n features X_{i} condicionalmente independientes dada la clase Y

- Para cada X_{i}, especificar $\mathrm{P}(X_{\mathrm{i}} \mid Y)$

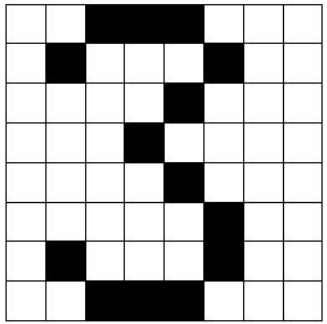


- Función de decisión:

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

Ejemplo: reconocimiento de dígitos

- Input: pixel grids



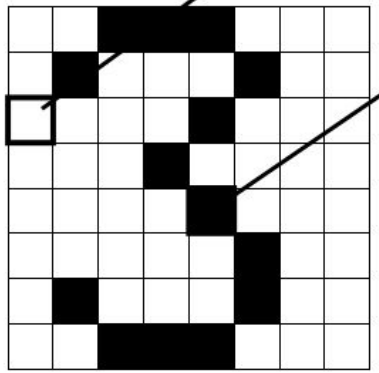
- Output: a digit 0-9

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

Ejemplo: reconocimiento de dígitos

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Estimación de parámetros por MV

- Dado un conjunto de datos, obtener 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:

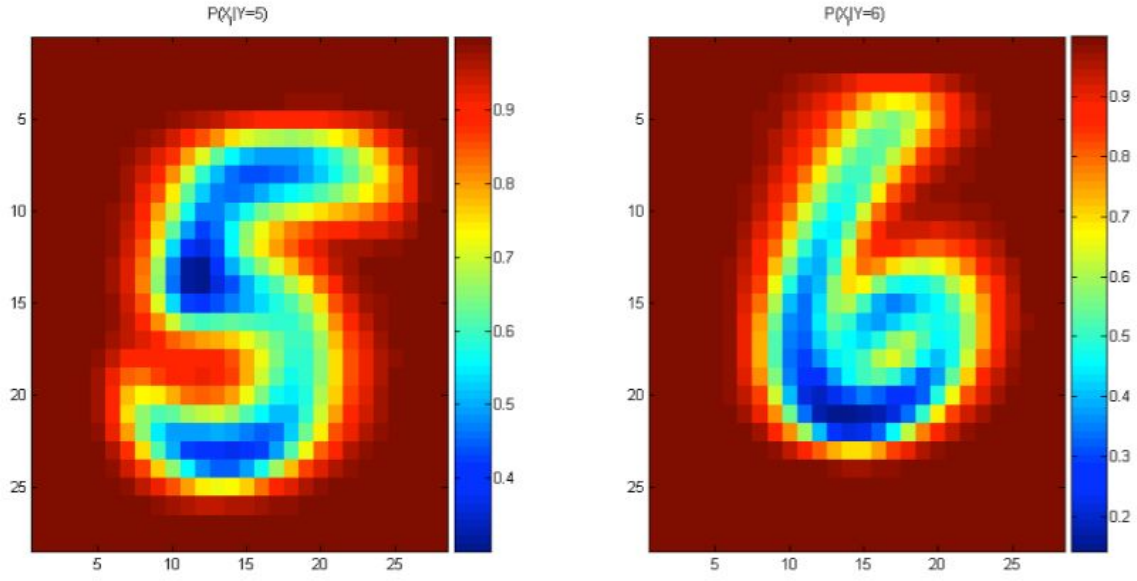
$$P(Y = y) = \frac{Count(Y = y)}{\sum_{y'} Count(Y = y')}$$

  - Distribución condicionales (observación):

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

Ejemplo: estimación de parámetros por MV

- El entrenamiento consiste en promediar los ejemplos para cada clase



MAP estimation for NB

- Given dataset
  - Count(A=a,B=b) ← number of examples where A=a and B=b
- MAP estimation for discrete NB, simply:
  - Prior:

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

  - Observation distribution:

$$P(X_i = x | Y = y) = \frac{Count(X_i = x, Y = y) \ \textbf{+ a}}{\sum_{x'} Count(X_i = x', Y = y) \ \textbf{+ |X_i|*a}}$$

- Called “smoothing”. Corresponds to Dirichlet prior!

Estimación de parámetros por MAP

- Dado un conjunto de datos, obtener 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:

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

  - Distribución condicionales (observación):

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

Estimación de parámetros por MAP

- Dado un conjunto de datos, obtener 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:

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

  - Distribución condicionales (observación):

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

smoothing (Dirichlet prior)

Regresión logística

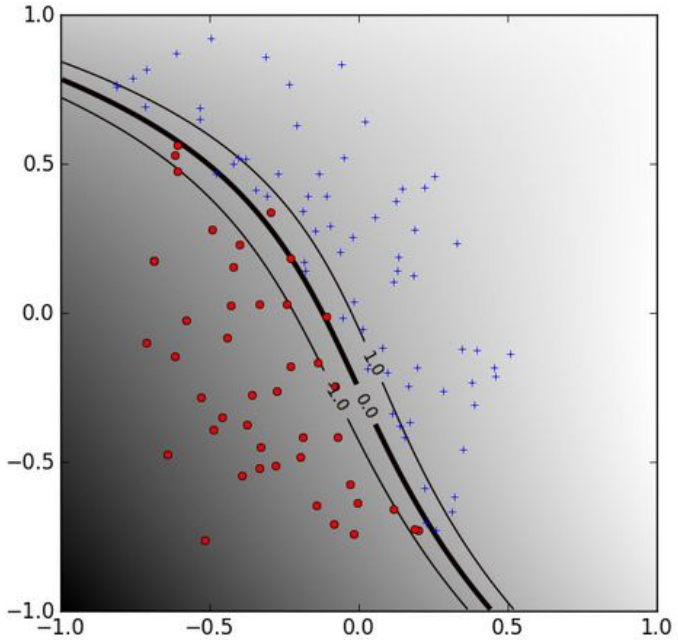
Clasificación basada en probabilidades

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

- Recordar:

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

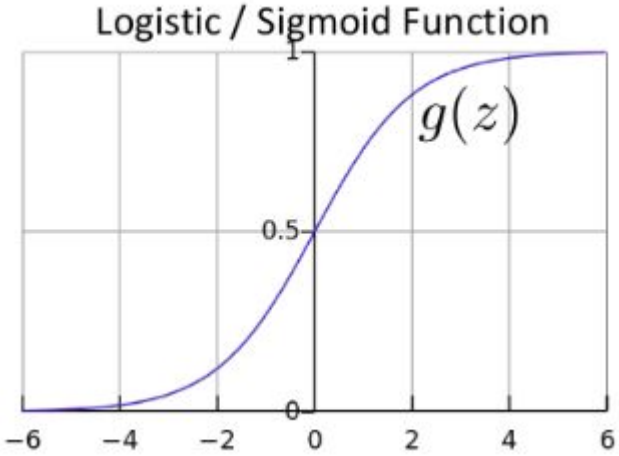
$$\mathrm{p}(evento) + \mathrm{p}(\neg evento) = 1$$



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 $\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}$

  where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d, \; y^{(i)} \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$

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

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

Regresión logística. Función de costo

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

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

  - Using the logistic regression model

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}$$

    results in a non-convex optimization

Regresión logística. Función de costo

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

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 w that maximizes the likelihood of labels in the training set

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

Intuition Behind the Objective

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

- Cost of a single instance:

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

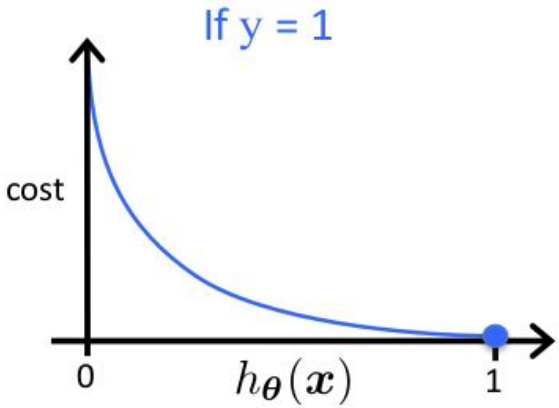
- Can re-write objective function as

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

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

Intuition Behind the Objective

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

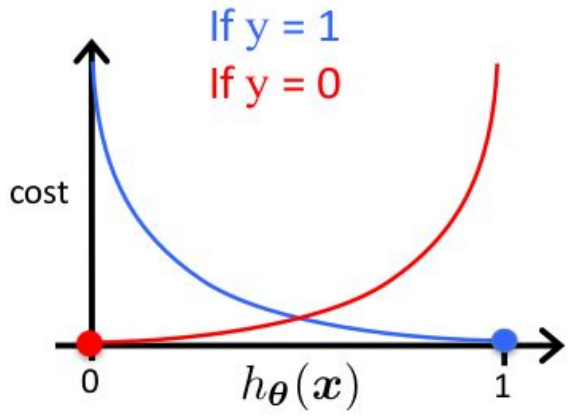


If y = 1

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

Intuition Behind the Objective

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



If y = 0

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

Regularized Logistic Regression

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

- We can regularize logistic regression exactly as before:

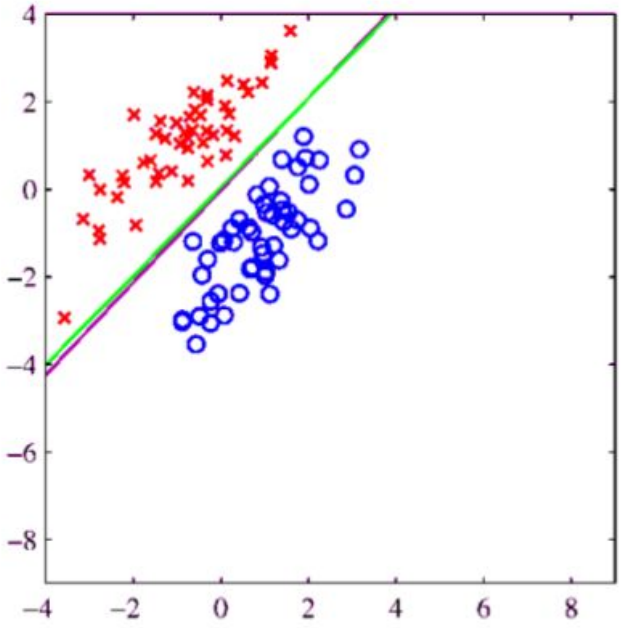
$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda\sum_{j=1}^{d}\theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

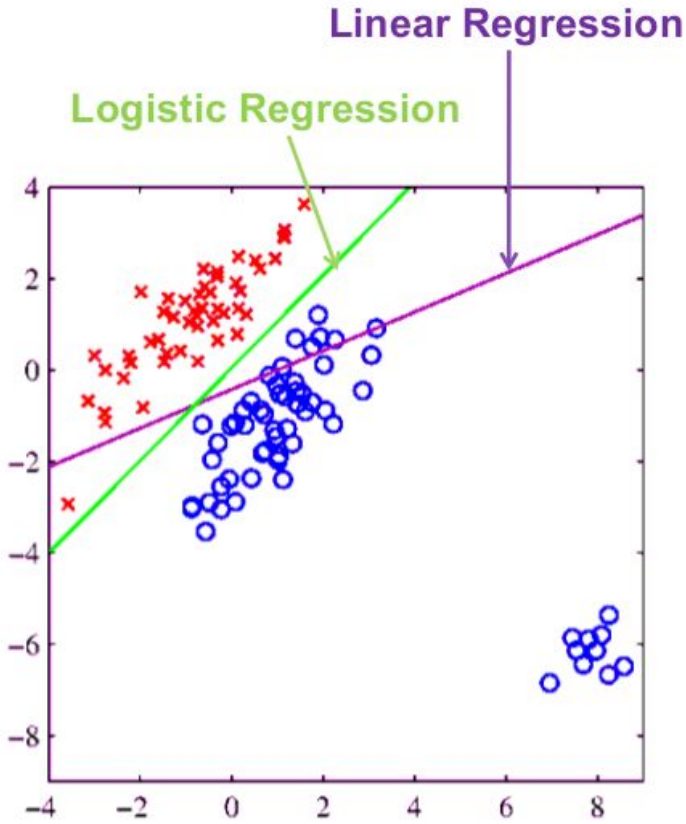
[1:d] => exclude the bias!

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

Logistic vs Linear Regression

Logistic regression is more robust





Multi-Class Logistic Regression

- For 2 classes:

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

weight assigned to y = 0

weight assigned to y = 1

- For C classes {1, ..., C}:

$$p(y = c \mid \boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C) = \frac{\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}$$

  - Called the **softmax** function

Implementing Multi-Class Logistic Regression

- Use $h_c(\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$ as the model for class c

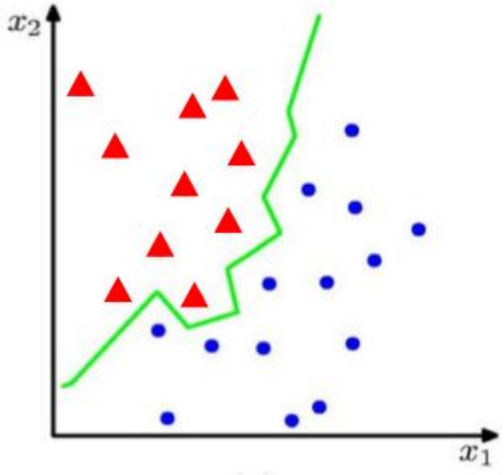
- Gradient descent simultaneously updates all parameters for all models
  - Same derivative as before, just with the above $\mathrm{h_c(x)}$

- Predict class label as the most probable label

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

Modelos no paramétricos: vecinos más cercanos

Classification



- Suppose we are given a training set of N observations

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

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

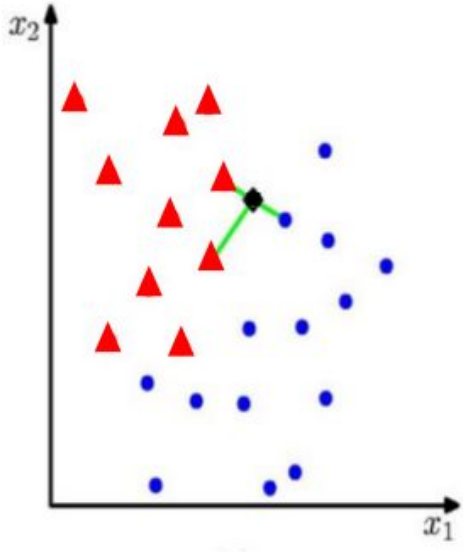
$$f(x_i) = y_i$$

K Nearest Neighbour (K-NN) Classifier

Algorithm

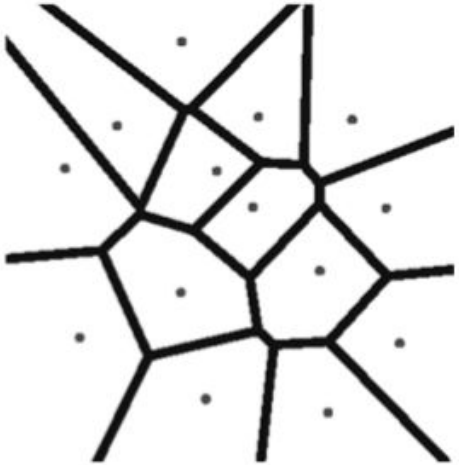
- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

e.g. K = 3



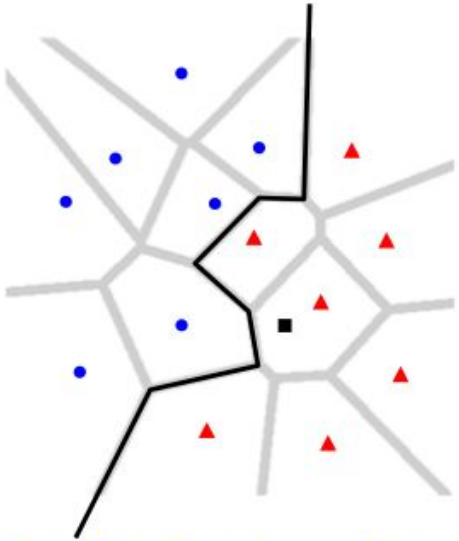
• applicable to multi-class case

K = 1



Voronoi diagram:

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



Classification boundary:

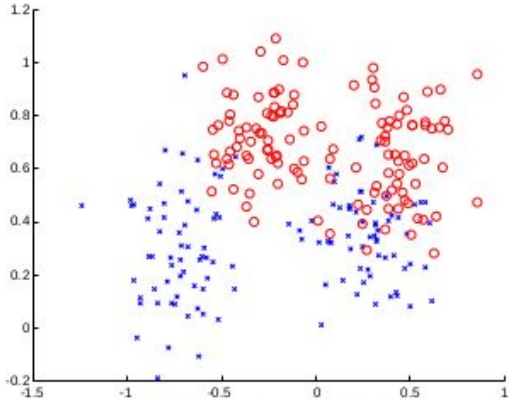
- non-linear

A sampling assumption: training and test data

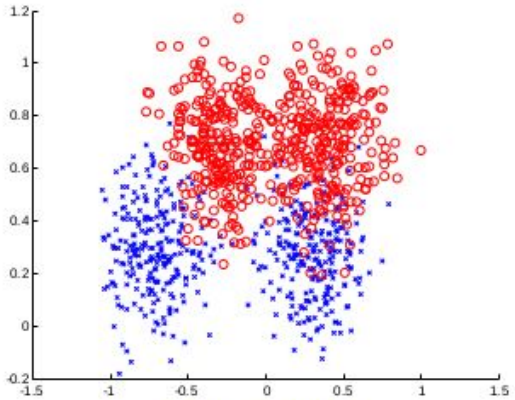
• Assume that the training examples are drawn independently from the set of all possible examples.

• This makes it very unlikely that a strong regularity in the training data will be absent in the test data.

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



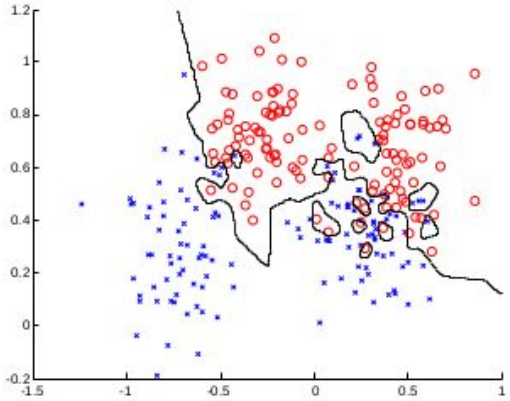
Training data



Testing data

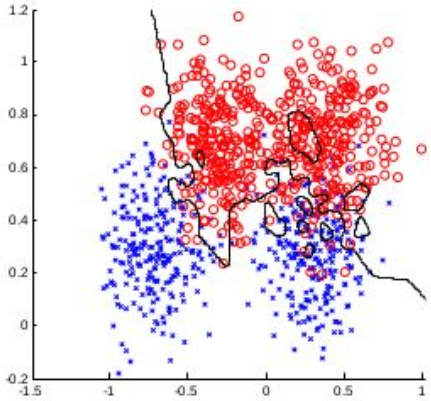
K = 1

Training data



error = 0.0

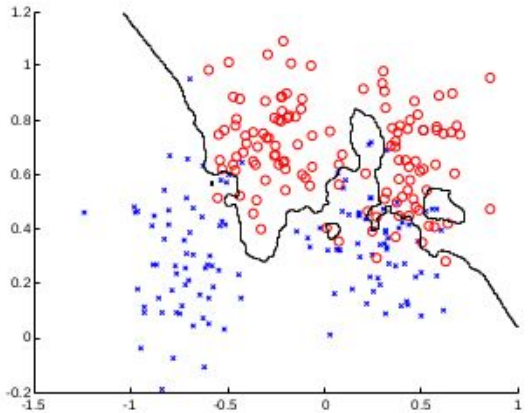
Testing data



error = 0.15

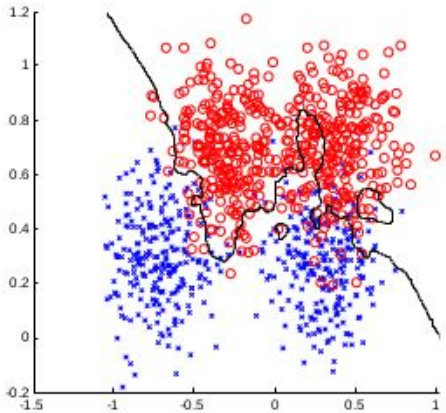
K = 3

Training data



error = 0.0760

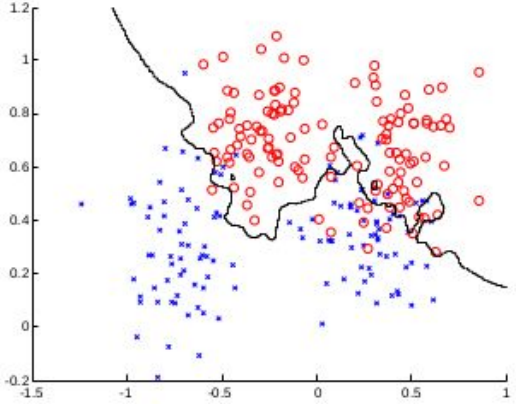
Testing data



error = 0.1340

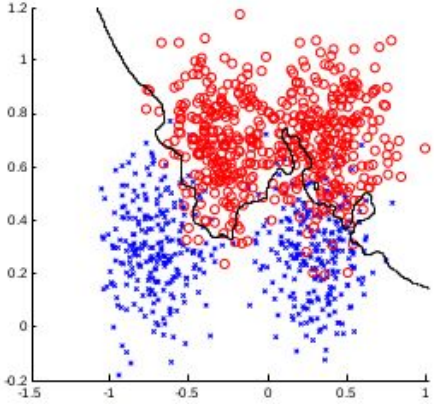
K = 7

Training data



error = 0.1320

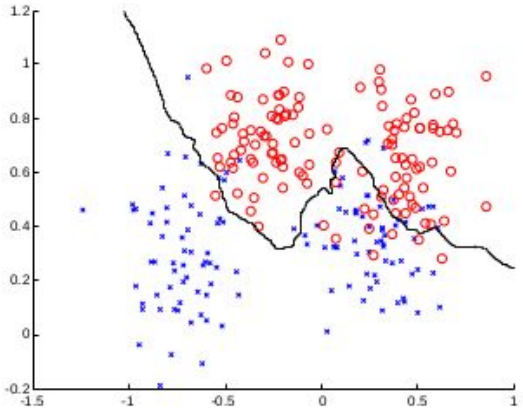
Testing data



error = 0.1110

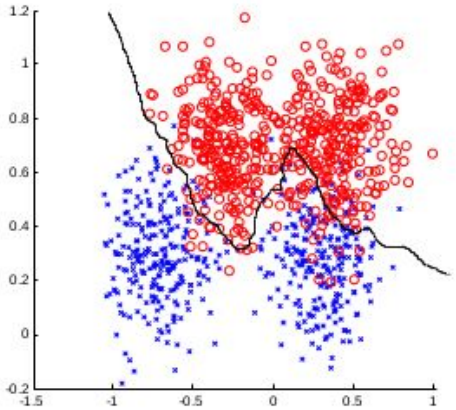
K = 21

Training data



error = 0.1120

Testing data



error = 0.0920

Properties and training

As K increases:

- Classification boundary becomes smoother
- Training error can increase

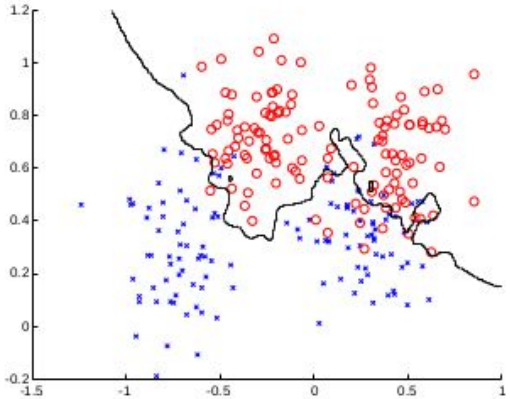
Choose (learn) K by cross-validation

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

Summary

Advantages:

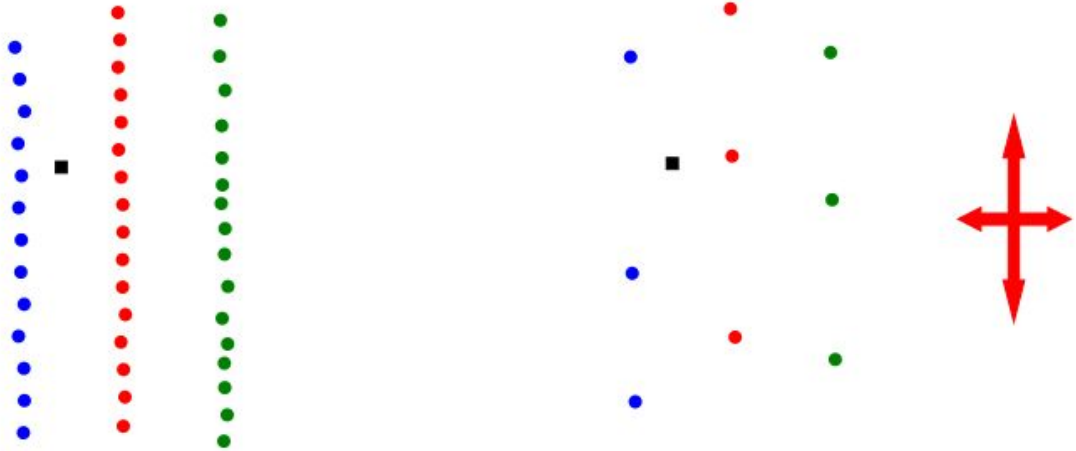
- K-NN is a simple but effective classification procedure
- Applies to multi-class classification
- Decision surfaces are non-linear
- Quality of predictions automatically improves with more training data
- Only a single parameter, K; easily tuned by cross-validation



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.



Multi-Class Logistic Regression

- For 2 classes:

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

weight assigned to y = 0

weight assigned to y = 1

- For C classes {1, ..., C}:

$$p(y = c \mid \boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C) = \frac{\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}$$

  - Called the **softmax** function

Implementing Multi-Class Logistic Regression

- Use $h_c(\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$ as the model for class c

- Gradient descent simultaneously updates all parameters for all models
  - Same derivative as before, just with the above $\mathrm{h_c(x)}$

- Predict class label as the most probable label

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