Grao en Enxeñaría Informática

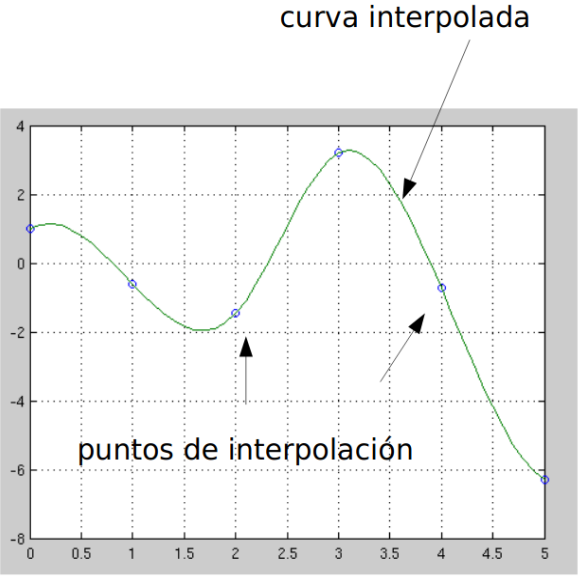
Aprendizaxe automática

Tema 1. Clasificación e regresión

- Aprendizaxe automática: síntese dunha función a partir de valores seleccionados (exemplos de entrenamento).
- Caso simple: axuste por mínimos cadrados
- Colección de puntos $(x_1,y_1),...,(x_N,y_N)$
- Calcular coeficientes de polinomio de grao m que minimiza o erro cadrático medio
- Isto permite calcular o valor do polinomio para calquer $x \neq x_i$: predecir a función

Introducción (I)

- Coeficientes do polinomio: parámetros axustábeis ou aprendíbeis (entrenábeis)
- A maior grao m, máis parámetros
- Do axuste pasamos á interpolación
- Aproximar unha función coñecen do uns valores de exemplo $y_1 \ldots y_N$ para $x_1 \ldots x_N$



- Calquera función, non so polinomio: interpolación linear, spline cúbica: definida por unha serie de parámetros.

Introducción (II)

- Se a dimensión n=1 e son poucos datos, o cálculo dos parámetros entrenábeis admite expresión analítica (fórmulas)
- Que pasa se n>1 ou se os datos son moitos?
- Medran moi rápido:
  - A complexidade do problema
  - O nº de parámetros entrenábeis
- Os métodos analíticos xa non existen ou son moi lentos (exhaustivos)
- Necesítanse técnicas eficientes: aprendizaxe automática

Clasificación e regresión

- Como son os valores da función a aproximar ou predecir?
- Continuos? **Regresión**: valores reais, existe unha orde
- Discretos (categóricos)? **Clasificación**: a etiqueta de clase é un valor enteiro
- Sen ordeamento: clasificación **nominal**: os erros de clasificación son igualmente importantes (agás casos especiais, p.ex. enfermidades)
- Ordeados: clasificación **ordinal**: os erros de clasificación son menos importantes entre clases contiguas

Aprendizaxe supervisada e non supervisada

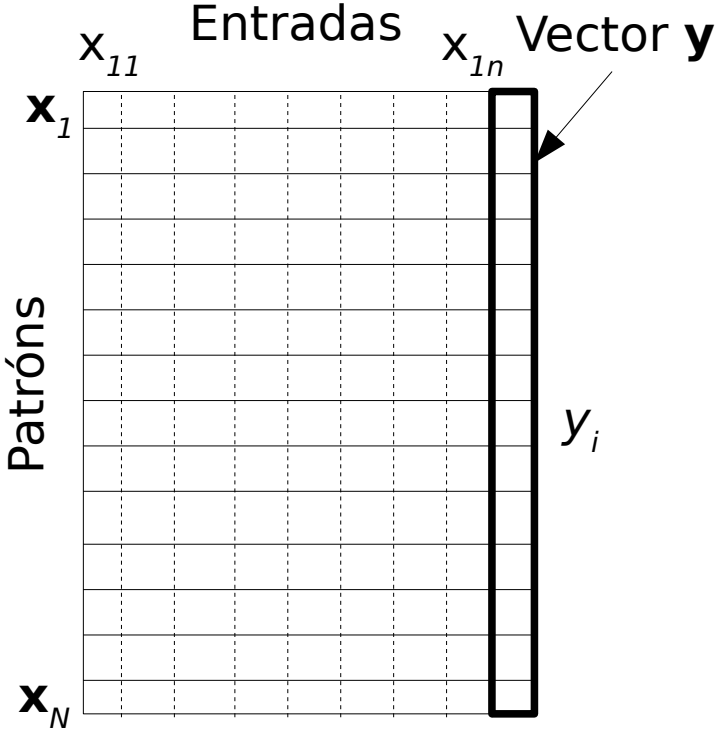
- No axuste, interpolación e aprendizaxe automática coñécese o valor da función a predecir para os patróns de entrenamento: aprendizaxe **supervisada**
  - Clasificación, regresión
  - Para predecir os valores da función usa os valores coñecidos
- Aprendizaxe **non supervisada**: agrupa datos ou extrae información dos datos usando criterios (p.ex. distancias ou similitudes)
  - Agrupamento (clustering), reducción da dimensionalidade
- Aprendizaxe por reforzo

Nomenclatura (I)

- Dato=patrón (instancia, pattern) de entrenamento, vector: $\mathbf{x}_i=(x_{i1}...x_{in})$: n entradas (atributos, características, features)
- Nº de patróns de entrenamento (N): $\mathbf{x}_1...\mathbf{x}_N$
- Patrón de teste: $\mathbf{x}$ (distinto aos de entrenamento)
- Saída desexada (predicción): y_i para o patrón de entrenamento $\mathbf{x}_i$
- Clasificación: C clases: $y_i \in \{1...C\}$: o clasificador aprende a asignar $\mathbf{x}_i$ á clase y_i
- Regresión: $y_i \in \mathbb{R}$: o regresor aprende a predecir y_i para $\mathbf{x}_i$. Normalmente, y_i está nun certo entorno de 0

Nomenclatura (II)

- Temos unha matriz de datos **X** con N filas (patróns) e n columnas (entradas)
- A fila i-ésima é $\mathbf{x}_i$, e o seu valor na columna j-ésima é x_{ij}
- Os valores verdadeiros da función a predecir forman un vector columna **y** con N valores
- Esta columna ten y_i na fila i-ésima, con i=1..N



Metodoloxía de validación (I)

- Conxunto de entrenamento: exemplos+función: $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$. O nº de patróns de entrenamento (datos) é N
- $\mathbf{x}_i$: patrón n-dimensional; y_i: valor a predecir (verdadeiro): pode ser real (regresión) ou discreto (0-1, clasificación binaria, ou 1-2-3-4: clasificación multiclase)
- z_i: saída predita polo algoritmo de aprendizaxe para $\mathbf{x}_i$
- Entrenamento: cálculo de parámetros entrenábeis, da como resultado un modelo entrenado (clasificador ou regresor)
- Validación ou teste: o modelo entrenado emprégase para predecir a función sobre datos distintos aos de entrenamento
- O teste serve para avaliar a calidade da predicción
- Distintas medidas de calidade para clasificación e regresión

Metodoloxía de validación (II)

- Fiabilidade da calidade medida?
- Modelo optimizado para datos de entrenamento
- Non se pode avaliar a súa calidade usando os datos de entrenamento
- Non sería realista, senón falsamente elevada (sesgada ou polarizada de forma optimista)
- Necesítase un protocolo ou metodoloxía de validación obxectivo, realista, fiábel
- O conxunto de entrenamento debe ser grande, é decir, representativo do problema a aprender

Validación cruzada K-fold (I)

- Cross validation (CV): K é o nº de probas (K=4, 5, 10)
- Divide datos dispoñibles en K particións
- Entrena con K-1 particións
- Valida coa parte restante
- Repite o proceso K veces, en cada vez valida cunha partición distinta
- As particións débense xerar de modo aleatorio
- O resultado (medida de calidade) promédiase sobre as K probas

4-fold CV	Proba 1	Proba 2	Proba 3	Proba 4
Entrena	1 2 3	2 3 4	3 4 1	4 1 2
Valida	4	1	2	3

Validación cruzada (II)

1) Xeras un ordeamento aleatorio dos N patróns
2) Divides os patróns aleatoriamente ordeados en K particións
3) Asignas cada partición ao entrenamento ou á validación en cada proba

- Clasificación: cada partición de entrenamento ou validación debe conter patróns de tódalas clases, conservando as súas poboacións relativas no conxunto completo
- Regresión: ordea os patróns por orde crecente da saída antes de facer o reparto aleatorio, para que cada partición teña patróns con saídas y_i en todo o rango $[\min(y_i),\max(y_i)]$
- Inicializa o xerador de números aleatorios sempre do mesmo xeito para que o experimento sexa reproducíbel

Validación cruzada (III)

- Cada proba: entrenamento+validación
- Canto máis grande K, máis veces hai que repetir o ciclo: máis lento
- Con moitos patróns (large-scale), usa K baixo
- Con poucos patróns (small-sample), K=N: deixar un fóra (LOOCV, leave-one-out cross-validation):

1) En cada proba, deixa un patrón fóra e entreno con tódolos demais
2) Valido co patrón que quedou fóra
3) Repito o proceso cos N patróns e promedio a calidade sobre as N probas.

- Con problemas grandes é moi lento

Preprocesamento dos datos

- Estandarización: media cero e desviación 1:

$$x_{ij}' = \frac{x_{ij} - m_j}{d_j} \qquad m_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \qquad d_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_{ij} - m_j)^2}$$

- Centra e iguala as varianzas de tódalas entradas
- As medias e desviacións débense calcular usando so os patróns de entrenamento
- Os patróns de teste débense preprocesar usando estas medias e desviacións de entrenamento
- Na regresión, pode ser necesario estandarizar a saída
- Se a entrada x_i é discreta con M valores: convirte a M variables indicadoras binarias (dummy variables) $\{x'_j\}_{j=1}^{M}$: se x_i toma o valor k-ésimo, entón $x'_k=1$ e $x'_m=0$ para $m \neq k$

Medidas de calidade para problemas de clasificación (I)

- Matriz de confusión: C_{ij}=nº de patróns da clase i asignados polo clasificador á clase j
- Erros de clasificación: fóra da diagonal
- Acerto (accuracy):

		Preditas	
		Clase 1	Clase 2
Verdadeiras	Clase 1	**C_{11}**	C_{12}
	Clase 2	C_{21}	**C_{22}**

$$Acc(\%)=\frac{100\sum_{i=1}^{C} C_{ii}}{\sum_{i=1}^{C}\sum_{j=1}^{C} C_{ij}}$$

Moi sensíbel ao desbalanceo entre clases

- Kappa: $\kappa=100\frac{a-e}{s-e}$, $a=\sum_{i=1}^{C} C_{ii}$, $e=\frac{1}{s}\sum_{i=1}^{C}\left(\sum_{j=1}^{C} C_{ij}\right)\left(\sum_{k=1}^{C} C_{ki}\right)$
- Ambas valen para 2 ou máis clases

$$s=\sum_{i=1}^{C}\sum_{j=1}^{C} C_{ij}$$

Medidas de calidade para problemas de clasificación (II)

- Mellor medida: kappa. Os seus valores pódense interpretar como:

1) Kappa≤0%: concordancia malísima entre clase verdadeira e predita
2) 1%≤Kappa≤20%: concordancia baixa
3) 21%≤Kappa<40%: lixeira
4) 41%≤Kappa≤60%: moderada
5) 61%≤Kappa≤80%: sustancial
6) 81%≤Kappa≤100%: case perfecta

Fonte: “The Measurement of Observer Agreement for Categorical Data”, J. Landis and H. G. Koch, *Biometrics*, No. 1, pp. 159-174 (1977)

Clasificación con 2 clases (binaria)

- Considérase un problema de detección da clase 2
- A clase 1 é negativa (N) e a clase 2 é positiva (P)
- V=verdadero, F=falso,

	C1	C2
C1	**VN**	FP
C2	FN	**VP**

- Sensibilidade (Se) ou *recall* (Rc) ou *true positive rate* (TPR): probabilidade de que un patrón da clase 2 sexa clasificado como 2

$$Se = Rc = TPR = \frac{VP}{VP + FN}$$

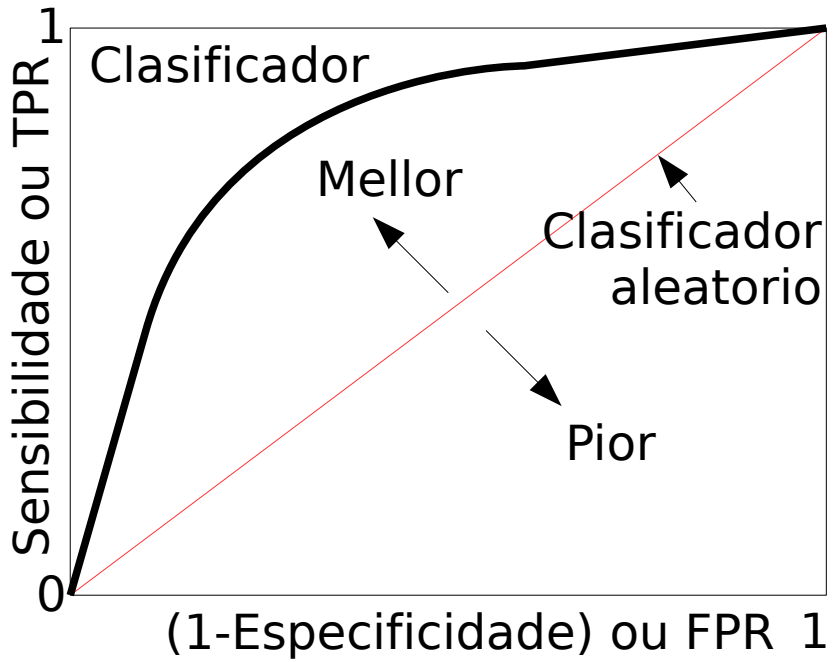
- Especificidade (Es) ou *true negative rate* (TNR): probabilidade de que un patrón da clase 1 sexa clasificado como 1

$$Es = TNR = \frac{VN}{VN + FP}$$

- Curva ROC (*Receiver Operating Characteristic*): é unha representación da sensibilidade (ou de VP) frente a 1-especificidade (ou frente a FP)

Curva ROC para problemas de 2 clases

- Os puntos da curva negra obtéñense executando o clasificador con varios valores dun umbral para elexir unha das dúas clases.
- Canto máis á esquerda e arriba estea a curva, mellor clasificador
- Os puntos inferior esquerdo e superior dereito corresponden aos valores extremos do umbral, onde se asignan tódolos patróns á clase 1 (FP=0, esquerda) ou 2 (FP=1,VN=0, dereita)



- Con mais de 2 clases: unha curva ROC para cada clase (Positiva) frente ás demais (Negativa)

Outras medidas en clasificación binaria (I)

	C1	C2
C1	**VN**	FP
C2	FN	**VP**

- Área baixo a curva ROC (area under curve, AUC) mide a calidade do clasificador en problemas de 2 clases
- Predictividade positiva (PPV) ou precisión: $$PPV = Pr = \frac{VP}{VP + FP}$$
- F-score: $\beta \in [0, +\infty)$ é un factor de ponderación da PP (Pr) e Se (Rc): $\beta = 0$ pondera so PP, $\beta = \infty$ pondera so sensibilidade (Rc)

$$F - score = (1 + \beta^2) \frac{Pr \cdot Rc}{\beta^2 Pr + Rc} = \frac{(1 + \beta^2) VP}{(1 + \beta^2) VP + \beta^2 FN + FP}$$

- $\beta = 2$ (>1) pondera máis a Se, $\beta = 0.5$ (<1) pondera máis a PP. Normalmente, $\beta = 1$, e entón chámase F1:

$$F1 = \frac{2 VP}{2 VP + FN + FP} = \frac{2 \cdot Pr \cdot Rc}{Pr + Rc}$$

Outras medidas en clasificación binaria (II)

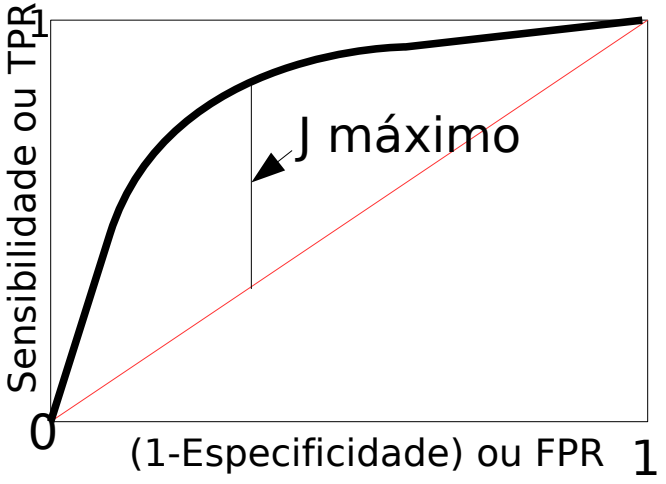
- Índice de Fowlkes-Mallows: $FM = \sqrt{Se \cdot PP} = \frac{VP}{\sqrt{(VP+FP)(VP+FN)}}$
- Acerto balanceado: $Bacc = \frac{Se+Es}{2} = \frac{1}{2}\left(\frac{VP}{VP+FN} + \frac{VN}{VN+FP}\right)$
- Coeficiente de correlación de Matthews (ϕ):

$$MCC = \phi = \frac{VP \cdot VN - FP \cdot FN}{\sqrt{(VP+FP)(VP+FN)(VN+FP)(VN+FN)}}$$

- Índice de Youden: J=Se+Es-1

$$J = \frac{VP}{VP+FP} + \frac{VN}{VN+FP} - 1$$

- False positive rate (FPR): $FPR = \frac{FP}{FP+VN}$
- False negative rate (FNR): $FNR = \frac{FN}{FN+VP}$



Medidas de calidade para problemas de regresión (I)

- Típica: erro cuadrático medio (RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - z_i)^2}$$

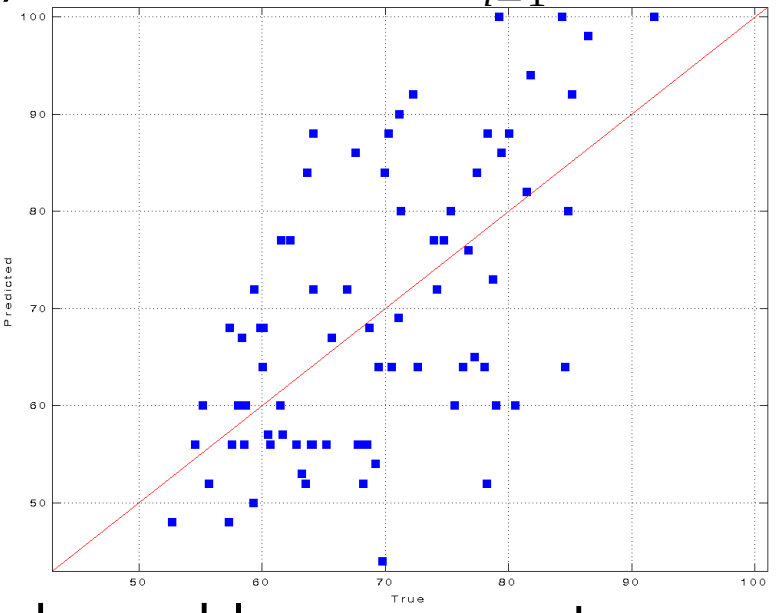
- Erro absoluto medio (MAE)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - z_i|$$

- Correlación (R) e squared correlation (R^2):
  $\overline{z}$ é o valor medio de $\{z_i\}_{i=1}^{N}$

$$R = \frac{\sum_{i=1}^{N}(y_i - \overline{y})(z_i - \overline{z})}{\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2}\sqrt{\sum_{i=1}^{N}(z_i - \overline{z})^2}}$$

- Diagrama de dispersión (scatterplot)



- A medida a empregar pode depender do problema concreto: os valores de RMSE, MAE ou R poden non ser representativos dependendo da tolerancia aceptábel na aproximación

Medidas de calidade para problemas de regresión (II)

- Representación dos valores verdadeiros preditos (ordeados):
- Significado da correlación:

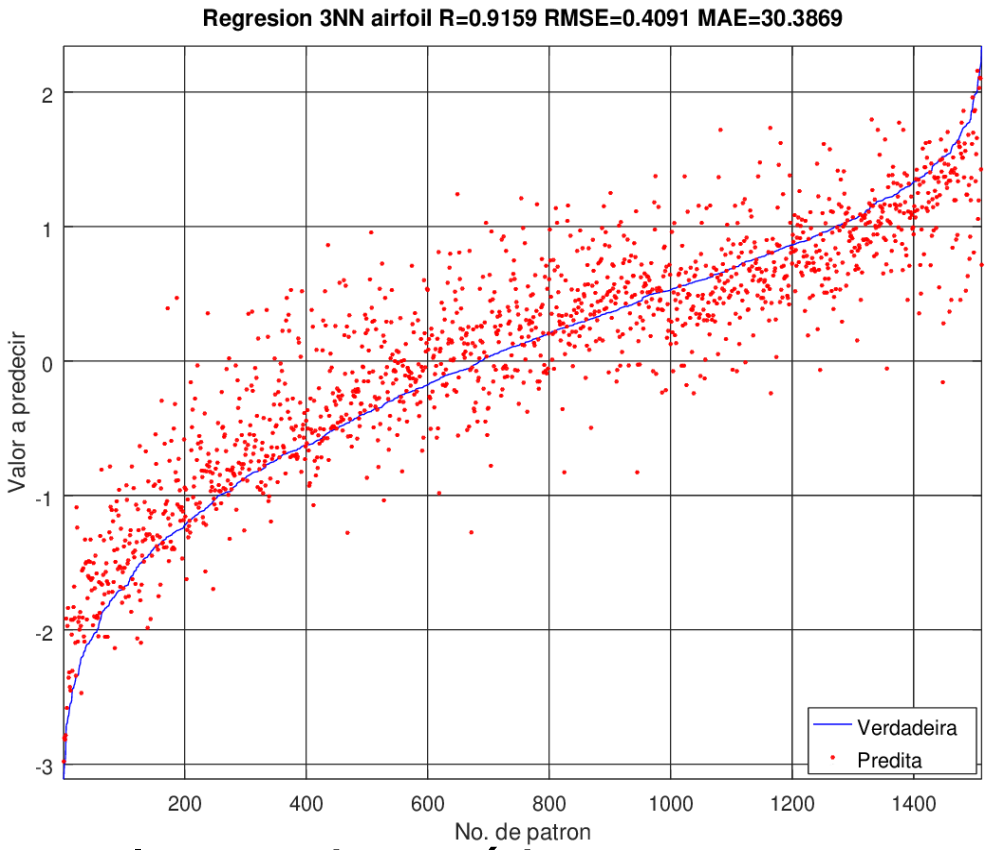
1) $R<0.15$: baixo

2) $0.15<R<0.5$: moderado

3) $0.5<R<0.75$: aceptábel

4) $0.75<R<1$: excelente

Fonte: T. Colton, “Statistical in medicine”, Little Brown and Co, 1974.



- Os valores de RMSE e MAE son relativos e dependen dos valores da saída

Métodos de veciños máis cercanos

- Exemplo simple de clasificador/regresor (NN: nearest neighbors)
- **Clasificación**: a clase dun patrón decídese votando entre os patróns de entrenamento máis cercanos
- 1NN: clasificador de veciño máis cercano (**x**: teste): $$y(\boldsymbol{x}) = y_j, j = \underset{i=1..N}{argmin} \{ |\boldsymbol{x} - \boldsymbol{x}_i| \}$$
- Distancia euclídea, outras (L1, Mahalanobis, ...)
- Non ten entrenamento nin parámetros entrenábeis
- Esixe almacenar todo o conxunto de entrenamento: ineficiente
- Número V de veciños V>1: votación, máis robusta: v_i=nº (impar) de patróns da clase i entre os V veciños mais cercanos $$y(\boldsymbol{x}) = y_j, j = \underset{i=1..C}{argmax} \{ v_i \}$$ $$v_i = \sum_{k \in v(\boldsymbol{x})} \delta(i, y_k), \delta(i, j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$
- **Regresión**: a votación substitúese por un promedio para V>1: $$y(\boldsymbol{x}) = \frac{1}{V} \sum_{i \in v(\boldsymbol{x})} y_i$$

  onde v(**x**) é o conxunto dos V veciños (patróns de entrenamento) máis cercanos a **x** (patrón de teste)

Selección de modelos (I)

- Que nº V de veciños usamos?
- V: é un hiper-parámetro (os hiper-parámetros non se calculan durante o entrenamento), debe coñecerse para que o clasificador KNN funcione
- Solución: probar con varios V (impares para evitar empates) e elexir o valor que mellor funciona: grid-search
- V: hiper-parámetro sintonizable. Case tódolos modelos de aprendizaxe automática teñen algún hiper-parámetro
- Escoller o V que da mellor resultado sobre o conxunto de teste daría resultados optimistas: non vale
- Avaliar como funciona o clasificador para cada V require un conxunto de proba distinto do conxunto de teste
- Imposible usando so particións de entrenamento e teste
- Solución: entrenamento + validación + teste

Selección de modelos (II)

1) Usa os conxuntos de **entrenamento** para entrenar o modelo cunha determinada combinación de valores dos hiper-parámetros

2) Usa os conxuntos de **validación** para avaliar a calidade do modelo entrenado cunha combinación de valores

   - Repite o bucle entrenamento-validación: repítese para tódalas combinacións de valores dos hiper-parámetros.
   - Selecciona a combinación coa mellor calidade promedio sobre os conxuntos de validación

3) Usa os conxuntos de **teste** para avaliar a calidade do modelo entrenado cos valores seleccionados dos hiper-parámetros sobre os conxuntos de entrenamento e validación

K=4	Proba 1	Proba 2	Proba 3	Proba 4
Entrena	1 2	2 3	3 4	4 1
Valida	3	4	1	2
Test	4	1	2	3

Selección de modelos (III)

Xuntando todo o anterior, temos dúas etapas:

1) **Sintonización** de hiper-parámetros: para cada combinación de valores dos hiper-parámetros sintonizábeis:
   a) Entrénase o algoritmo sobre os conxuntos de entrenamento
   b) Testéase sobre os conxuntos de validación e avalíase a súa calidade
   c) Selecciónase a combinación de valores dos hiper-parámetros que acada mellor calidade promedio sobre os conxuntos de validación
2) **Teste**:
   a) Entrénase sobre os conxuntos de entrenamento e validación
   b) Testéase sobre os conxuntos de teste e avalíase a súa calidade
   c) A calidade final do algoritmo é o promedio sobre os conxuntos de teste

Dilema nesgo-varianza e sobreaprendixaze (I)

Queremos un modelo que funcione ben:

a) sobre os datos de entrenamento

b) sobre datos novos (de validación ou teste)

- A existencia de ruido nos datos provoca que as dúas cousas se opoñan (*bias-variance dilemma*)
- O erro pódese descompor na suma de 3 termos.

$$E_X[g]=\frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}_i)$$

- Sexan: f= función real descoñecida en patrón de teste **x**, g=función aprendida, X=conxunto de entrenamento, $E_X[g]$=valor medio de *g* en *X*:

a) Nesgo (*bias*) en **x**: $B(\mathbf{x})=[E_X[g]-f(\mathbf{x})]^2$: diferencia entre o valor medio da función aprendida e a función real sobre **x**

b) Varianza (*variance*) sobre *X*: $V=E_X[g^2]-E_X[g]^2$: desviación da función aprendida g a respeito da súa media sobre o conxunto de entrenamento. Mide se g varía moito (varianza alta) ou non (varianza baixa, g$\simeq$constante)

c) Erro irreducible σ^2 : varianza do ruido en $\{y_i\}_{i=1}^{N}$

Dilema nesgo-varianza e sobreaprendixaze (II)

- O nesgo $B(\mathbf{x})$ nun patrón de teste $\mathbf{x}$ pódese calcular usando que $g(\mathbf{x}_i)=z(\mathbf{x}_i)$ para $i=1..N$ e que $f(\mathbf{x})=y(\mathbf{x})$:

$$B(\boldsymbol{x})=[E_X[g]-f(\boldsymbol{x})]^2=\left[\frac{1}{N}\sum_{i=1}^{N} z(\boldsymbol{x}_i)-y(\boldsymbol{x})\right]^2$$

- O nesgo B_t calculado sobre o conxunto de entrenamento:

$$B_t=\sum_{i=1}^{N} B(\boldsymbol{x}_i)=\sum_{j=1}^{N}\left[\frac{1}{N}\sum_{i=1}^{N} z(\boldsymbol{x}_i)-y_j\right]^2$$

- A varianza V pódese calcular como:

$$V=E_X[g^2]-E_X[g]^2=\frac{1}{N}\sum_{i=1}^{N} z(\boldsymbol{x}_i)^2-\left[\frac{1}{N}\sum_{i=1}^{N} z(\boldsymbol{x}_i)\right]^2$$

Dilema nesgo-varianza e sobreaprendixaze (III)

- Nesgo B_t alto: a función aprendida g non se axusta ben á función f a aprender: o modelo aprende mal o conxunto de entrenamento
- Nesgo B_t baixo: o modelo axústase ben ao conxunto de entrenamento
- Varianza V alta: a función g aprendida desvíase moito a respeito da súa media: axuste excesivo aos datos de entrenamento, malos resultados sobre datos novos: **sobreaprendizaxe**.
- Varianza V baixa: pouca desviación a respeito da media
- Como nesgo e varianza son ambos erros, ambos deberían ser o máis baixos posibles, pero están relacionados: reducir o nesgo leva a aumentar a varianza.

Dilema nesgo-varianza e sobreaprendixaze (IV)

- É mellor ter máis nesgo B_t se isto evita ter alta varianza (axuste excesivo aos datos de entrenamento pero funciona mal nos datos de teste): sobreaprendizaxe.
- Un modelo sobreaprendido:

1) É demasiado complexo e excesivamente adaptado aos datos de entrenamento.
2) Ten unha baixa capacidade de xeralización a datos novos

- Debe haber un equilibrio entre nesgo e varianza: ningún deles debe ser alto, xa que ambos son erros.
- Tamén se produce sobreaprendizaxe cando o nº de parámetros entrenábeis é moi elevado comparado co nº de patróns (N)

Maldición da dimensionalidade

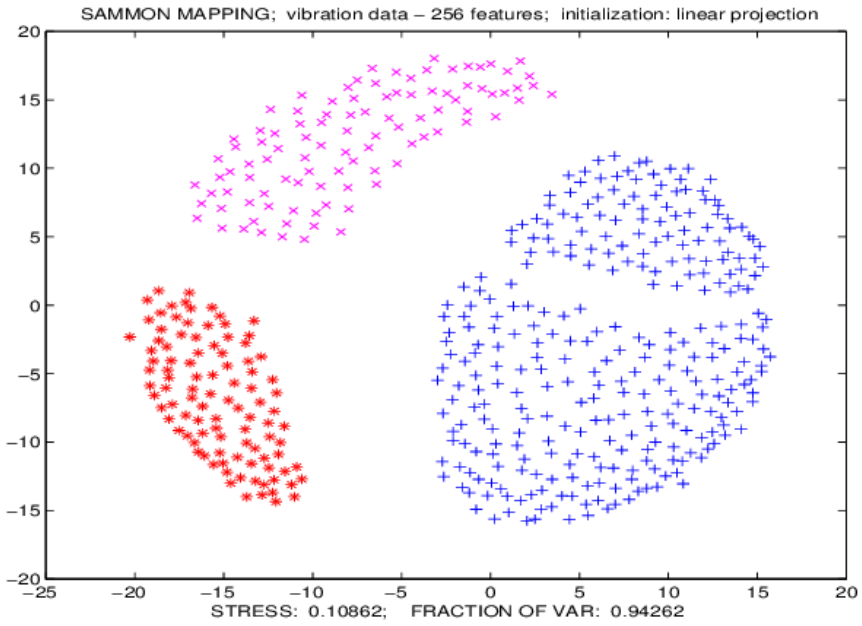
- Curse of dimensionality: cando n medra, o volume do espazo de entrada medra moi rápido
- Requírense moitos datos (N elevado) para cubrir o espazo: os datos vólvense dispersos
- Demostrouse que para manter a densidade de datos necesaria para aprender un problema, o nº de patróns necesarios debe aumentar exponencialmente con n
- Os algoritmos funcionan mal pola baixa densidade de datos: a información é escasa para a elevada dimensión dos datos

Representación gráfica de problemas de clasificación

- Plano ou mapa con patróns proxectados a 2D, cor=clase
- Mapeo de Sammon: método de reducción da dimensionalidade: visualiza problemas de clasificación
- Parte de patróns en $\mathbb{R}^2$ aleatorios, e vainos actualizando de modo que as distancias entre patróns $\mathbf{x}_i$ en $\mathbb{R}^2$ e $\mathbf{y}_i$ en $\mathbb{R}^n$ sexan parecidas
- Minimiza o estrés φ de Kruskall ou Sammon:

$$\varphi = \frac{\sum_{i<j} \frac{\left(d\,\boldsymbol{x}_{ij} - d\,\boldsymbol{y}_{ij}\right)^2}{d\,\boldsymbol{x}_{ij}}}{\sum_{i<j} d\,\boldsymbol{x}_{ij}} \qquad d\,\boldsymbol{x}_{ij} = |\boldsymbol{x}_i - \boldsymbol{x}_j|$$

- Visualiza o problema de clasificación e mostra a superposición entre clases



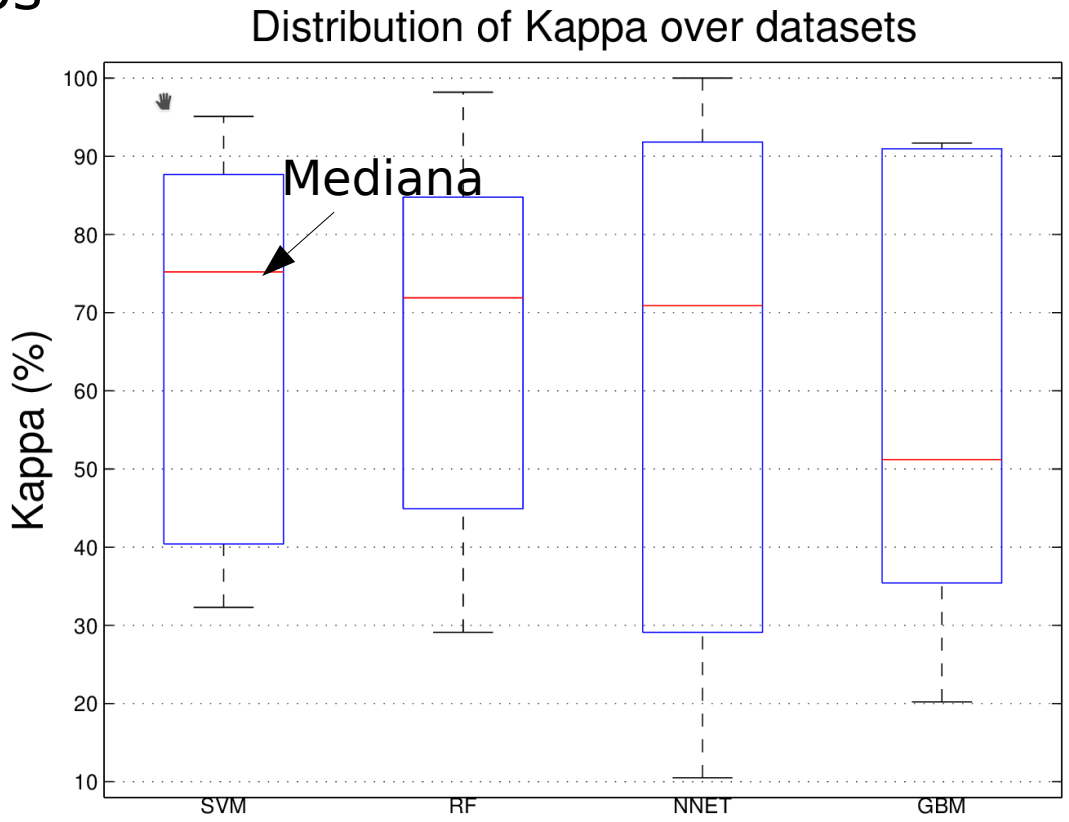
https://www.researchgate.net/figure/Sammon-mapping-of-vibration-data_fig3_2704140

Comparación de algoritmos (I)

- Sobre un problema: +calidade (maior Kappa para clasificación, menor RMSE para regresión)→mellor algoritmo
- Que un algoritmo A (clasificador ou regresor) sexa mellor que outro B sobre un problema non significa que A sexa mellor que B para tódolos problemas
- Uns algoritmos funcionan mellor sobre uns problemas, e outros algoritmos funcionan mellor sobre outros
- É imposíbel que en tódolos problemas o mellor algoritmo sexa o mesmo: no-free-lunch theorem (Wolpert & Macready, 1997): “calquera algoritmo de optimización é equivalente cando se promedia sobre tódolos posíbeis problemas”
- Para comparar algoritmos: promediamos a medida de calidade sobre unha colección ampla de problemas

Comparación de algoritmos (II)

- Cando máis ampla sexa a colección, máis fiábel será a comparativa. O tamano da colección é fundamental
- Se calculamos a calidade media, os problemas con calidade alta pesan máis que os outros
- Podemos comparar gráficamente as distribucións da medida de calidade (p.ex., kappa): boxplot
- Tamén podemos usar tests estatísticos para evaluar a significancia estatística da diferencia entre dous algoritmos



Comparación de algoritmos (III)

- Comparación entre 2 algoritmos: entre outros, o teste de suma de rangos de **Wilcoxon** (Mann–Whitney U-test)
- Función ranksum(**x**,**y**) de Octave, sendo **x** e **y** os vectores coa medida de calidade dos dous clasificadores sobre tódolos problemas
- Testea a **hipótese nula** de que a medida de calidade (p.ex. Kappa ou R) dos clasificadores ou regresores X e Y (p.ex., SVM e RF, ou SVR e LR) sobre a colección de problemas pertencen a distribucións con igual media
- Resumindo: testea se X e Y son igual de bos. O teste pode decir que si ou que non
- A función ranksum retorna o **p-value** do teste (p): un valor alto significa si, un valor baixo significa non

Comparación de algoritmos (IV)

1) Se $p<0.05$ (o 5%), rexéitase a hipótese nula, é decir:

- A diferencia é estatísticamente significativa en favor do algoritmo de maior calidade media
- A diferencia é alta dabondo para consideralo mellor que o outro

2) Se $p\geq0.05$, a diferencia non é estatísticamente significativa (non é alta dabondo): non se pode considerar que sobre a actual colección de problemas un algoritmo sexa mellor que o outro

- Podes ampliar a colección: canto máis datos, menor ten que ser a diferencia para que $p<0.05$
- Ten en conta que a diferencia entre dous algoritmos normalmente diminúe cando o nº de problemas aumenta

Comparación de algoritmos (V)

- Isto vale para comparar dous algoritmos
- Se queres elaborar un ranking (lista) de (máis de dous) algoritmos ordenados por calidade decrecente, debes usar o **ranking de Friedman**
- Para cada problema da colección, ordea os algoritmos por calidade descendente
- O rango (rank) de cada algoritmo é a súa posición promedio sobre todos los problemas da colección
- Supoñamos os seguintes clasificadores e problemas (kappa):

Algoritmo	P1	P2	P3	P4	P5
SVM	95.1	32.3	85.2	75.2	43.1
RF	98.2	29.1	80.3	71.9	50.2
NNET	100	35.3	89.1	70.9	10.5
GBM	91.7	40.5	90.7	20.2	51.2

Comparación de algoritmos (VI)

- Elaboración do ranking de Friedman:

Ordeamento	1º	2º	3º	4º
P1	NNET	RF	SVM	GBM
P2	GBM	NNET	SVM	RF
P3	GBM	NNET	SVM	RF
P4	SVM	RF	NNET	GBM
P5	GBM	RF	SVM	NNET

Posición	P1	P2	P3	P4	P5	Media
SVM	3	3	3	1	3	2.6
RF	2	4	4	2	2	2.8
NNET	1	2	2	3	4	2.4
GBM	4	1	1	4	1	2.2

Posición	Pos.	Rango
GBM	1ª	2.2
NNET	2ª	2.4
SVM	3ª	2.6
RF	4ª	2.8

A diferencia entre os rangos de dous algoritmos indica a “distancia” entre eles

Comparación de algoritmos (VII)

- Con medidas de erro (RMSE), hai que ordear por valores de RMSE crecentes

```
% calculation of Friedman rank results
% perf: matrix with performance measure
%   with models by rows and datasets by columns
% order: 'descend' for performance measurements,
%  'ascend' for error measurements
function fr=friedman_rank(perf,order)
[nmodel ndata]=size(perf);pos=zeros(nmodel,ndata);
for i=1:ndata
    [~,ind]=sort(perf(:,i),order);
    for j=1:nmodel
        pos(ind(j),i)=j;
    end
end
fr=mean(pos,2);
end
```

Código en Octave para calcular o ranking de Friedman dunha colección de algoritmos sobre unha colección de problemas

Linguaxes de programación

- Octave/Matlab: linguaxe moi intuitiva
- Python: paquete scikit-learn
- R: moitos paquetes para clasificación
- Weka/Java:
  - Interfaz de usuario para execución manual
  - Programación en Java para execución automatizada
  - Inclusión en desenvolvemento de sotfware

Algoritmos, paquetes e linguaxes

	Octave/Matlab	Python/sklearn	R	Java/Weka
Nearest neighbors	Programación directa	neighbors/KNeighborsClassifier-Regressor	class/knn	weka.classifiers.lazy.IBk
LDA	Programación directa	discriminant_analysis/ LinearDiscriminantAnalysis	MASS/lda	weka.classifiers.functions. LDA
Ridge	Programación directa	linear_model/Ridge	ridge/ linearRidge	weka.classifiers.functions. LinearRegression
Trees	Programación directa, fitctree/fitrtree	tree/DecisionTreeClassifier-Regressor	rpart/rpart	weka.classifiers.trees.J48
MLP	newff	neural_network/MLPClassifier-Regressor	nnet/nnet	weka.classifiers.functions. MultilayerPerceptron
Deep learning	Deep learning toolbox	keras/Sequential	deepnet/dnn	weka.classifiers.functions. Dl4jMlpClassifier
SVM	Libsvm	SVC-SVR, Libsvm	kernlab/ksvm	weka.classifiers.functions. LibSVM
Adaboost	cboost/rboost	AdaboostClassifier-Regressor	adabag/boosting	weka.classifiers.meta. AdaboostM1
Random forest	TreeBagger	RandomForestClassifier-Regressor	randomForest/ randomForest	weka.classifiers.trees. RandomForest