Machine learning

Linear regression

Corso di Laurea Magistrale in Informatica

Università di Roma Tor Vergata

Prof. Giorgio Gambosi

a.a. 2021-2022

- Linear combination of input features

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$$

  with $\mathbf{x} = (x_1, \ldots, x_d)$
- Linear function of parameters $\mathbf{w}$
- Linear function of features $\mathbf{x}$

Possiamo vederla o come funzione di w fissato x o viceversa, è sempre lineare.

More compactly,

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \overline{\mathbf{x}}$$

where $\overline{\mathbf{x}} = (1, x_1, \ldots, x_d)$

Stiamo contando il w_0 che è solo, questo è un modo compatto di rappresentare la stessa cosa.

Base functions

- Extension to linear combination of base functions $\phi_1, \dots, \phi_m$ defined on $\mathbb{R}^d$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{m} w_j \phi_j(\mathbf{x})$$

- Each vector $\mathbf{x}$ in $\mathbb{R}^d$ is mapped to a new vector in $\mathbb{R}^m$, $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_m(\mathbf{x}))$
- the problem is mapped from a d-dimensional to an m-dimensional space (usually with $m > d$)

Invece di fare la combinazione lineare delle feature:
- definiamo un certo numero di funzioni da poter applciare sull'insieme delle feature, ad esempio

$\phi_1(x_1, \ldots, x_n) = \phi_1$ (lo stesso per le altre ϕ_2 , etc ...)

Un esempio banale può essere quello di avere una sola fueature x, prendere come insieme di funzioni che trasformano i valori delle feature. A questo punto, se abbiamo k funzoni la regressione diventa su k variabili questo è simile a ciò che avviene nell'ingegneria delle feature.

In generale, abbiamo un insieme di feature, magari vogliamo trasformarle in qualcos'altro, quindi definiamo un insieme di funzioni di cui ognuna restitusice un valore.

Se abbiamo un elemento $x_1, \ldots, x_d$ definendo m funzioni otteniamo m valori.

Quindi, dato un punto a d dimensioni la nostra predizione verrà effettuata in uno spazio a diversa dimensione, non sappiamo quali funzioni usare, che sono dette funzioni di base o come queste effettuano la trasformazione.

Supponiamo di conisderare solo funzioni polinomiali: $\phi_1(x) = x, \phi_2(x) = x^2, \phi_3(x) = x^3$ quindi se x=1,

avremo il vettore (1,1,1 ...), mentre se x=2 avremo (2,4,8 ...)

Base functions

- Many types:
  - Polynomial (global functions)

$$\phi_j(x) = x^j$$

  - Gaussian (local)

$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$

  - Sigmoid (local)

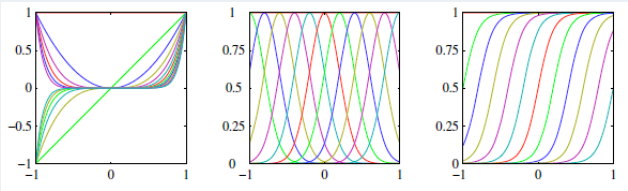
$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right) = \frac{1}{1+e^{-\frac{x-\mu_j}{s}}}$$

  - Hyperbolic tangent (local)

$$\phi_j(x) = \tanh(x) = 2\sigma(x) - 1 = \frac{1-e^{-\frac{x-\mu_j}{s}}}{1+e^{-\frac{x-\mu_j}{s}}}$$

Ha la caratteristica di avere un valore significativamente $\neq 0$ se nell'intorno di μ_j. Se quindi la feature è in un certo intervallo allora il valore prodotto è qualcosa, altrimenti è 0.

L'intervallo è più o meno determinato da s.



Observe that a set of items (extended by 1 values)

$$\overline{\mathbf{X}} = \begin{pmatrix} - & \overline{\mathbf{x}}_1 & - \\ & \vdots & \\ - & \overline{\mathbf{x}}_n & - \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{pmatrix} \quad n \times d \ (n \times d+1)$$

is transformed into

$$\Phi = \begin{pmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_m(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_n) & \phi_2(\mathbf{x}_n) & \cdots & \phi_m(\mathbf{x}_n) \end{pmatrix} \quad n \times m$$

Avendo le funzioni base: partiamo da un insieme di dati, dove ogni riga è un elemento e questa è trasformata in una diversa matrice dove ogni riga è una trasformazione ottenuta applicando tutte le funzioni ad ogni elemento

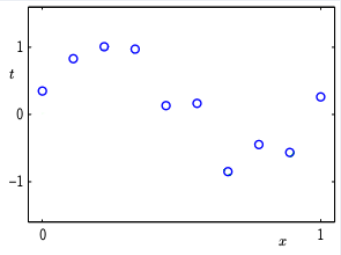
Example: polynomial regression

Problem

- A set of n observations of two variables $x, t \in \mathbb{R}$: $(x_1, t_1), \ldots, (x_n, t_n))$ is available. We wish to exploit these observations to predict, for any value $\tilde{x}$ of x, the corresponding unknown value of the target variable t
- The training set is a pair of vectors $\mathbf{x} = (x_1, \ldots, x_n)^T$ and $\mathbf{t} = (t_1, \ldots, t_n)^T$, related through an unknown rule (function)

Example of a training set.

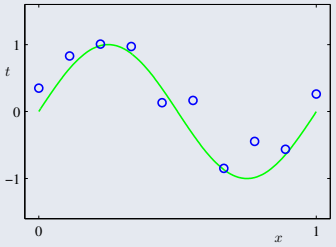
Vogliamo fare qualcosa del tipo: $y = w_0 + w_1 x$. Applichiamo prima delle funzioni di base, quindi la predizione sarà $y = w_0 + w_1 x + \ldots + w_K x^K$ se ad esempio usiamo funzioni polinomiali.



Example: polynomial regression

Training set

In this case, we assume that the (unknown) relation between x and t in the training set is provided by the function $t = \sin(2\pi x)$, with an additional gaussian noise with mean 0 and given variance σ^2. Hence, $t_i = \sin(2\pi x_i) + \varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$.



Purpose

Guessing, or approximating as well as possible, the deterministic relation $t = \sin(2\pi x)$, on the basis of the analysis of data in the training set.

Example: polynomial regression

Approach

Let us approximate the unknown function through a suitable polynomial of given degree $m > 0$

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

whose coefficients $\mathbf{w} = (w_0, w_1, \ldots, w_m)^T$ are to be computed.

Lineare nei coefficienti w. Vogliamo trovare il valore di w che minimizza la funzione di costo.

Linear models

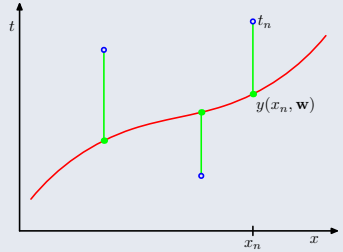
$y(x, \mathbf{w})$ is a nonlinear function of x, but is a linear function (model) of $\mathbf{w}$.

Parameter estimation

The values assigned to coefficients should minimize some error function (a.k.a. cost function), when applied to data in the training set (then, to $\mathbf{x}$, $\mathbf{t}$ and $\mathbf{w}$).

Least squares

A most widely adopted error function is least quares, i.e. the sum, for all items in the training set, of the (squared) difference between the value returned by the model and the target value.



Sarebbe il Zio chio empirico.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(y(x_i, \mathbf{w}) - t_i\right)^2 = \frac{1}{2}\sum_{i=1}^{n}\left(w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_m x_i^m - t_i\right)^2$$

↳ non $\frac{1}{n}$ perché conta poco se massimizziamo/minimizziamo.

↳ la funzione di costo è $(\text{predizione} - \text{target})^2$

Error minimization

- To minimize $E(\mathbf{w})$, set its derivative w.r.t. $\mathbf{w}$ to $\mathbf{0}$
- $E(\mathbf{w})$ quadratic implies that its derivative is linear, hence that it is zero in one point w^*
- The resulting function is $y(x, \mathbf{w}^*)$

$= w_0^* + w_1^* x + \dots + w_M^* x^M$

Derivative with respect to $\mathbf{w}$

The derivative w.r.t. $\mathbf{w}$ is indeed a collection of derivatives. A linear system is then obtained:

$$\frac{\partial E(\mathbf{w})}{\partial w_0} = \sum_{i=1}^{n} \left(y(x_i, \mathbf{w}) - t_i\right) = 0$$

$$\frac{\partial E(\mathbf{w})}{\partial w_1} = \sum_{i=1}^{n} x_i \left(y(x_i, \mathbf{w}) - t_i\right) = 0$$

$$\dots$$

$$\frac{\partial E(\mathbf{w})}{\partial w_m} = \sum_{i=1}^{n} x_i^m \left(y(x_i, \mathbf{w}) - t_i\right) = 0$$

Each of the $m+1$ equations is linear w.r.t. each coefficient in $\mathbf{w}$. A linear system results, with $m+1$ equations and $m+1$ unknowns, which, in general and with the exceptions of degenerate cases, has precisely one solution.

FAI I CONTI BENE: DENOMINATORE CHE NON VA MAI IN GRADO, COGLIONE!!

Gradient descent

- The minimum of $E(\mathbf{w})$ can be computed numerically, by means of gradient descent methods
- Initial assignment $\mathbf{w}^{(0)} = (w_1^{(0)}, w_2^{(0)}, \dots, w_m^{(0)})$, with a corresponding error value

$$E(\mathbf{w}^{(0)}) = \frac{1}{2} \sum_{i=1}^{N} \left(t_i - (\mathbf{w}^{(0)})^T \boldsymbol{\phi}(\mathbf{x}_i) \right)^2$$

- Iteratively, the current value $\mathbf{w}^{(i-1)}$ is modified in the direction of steepest descent of $E(\mathbf{w})$, that is the one corresponding to the negative of the gradient evaluated at $\mathbf{w}^{(i-1)}$
- At step i, $w_j^{(i-1)}$ is updated as follows:

$$w_j^{(i)} := w_j^{(i-1)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_j} \right|_{\mathbf{w}^{(i-1)}}$$

- In matrix notation:

$$\mathbf{w}^{(i)} := \mathbf{w}^{(i-1)} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(i-1)}}$$

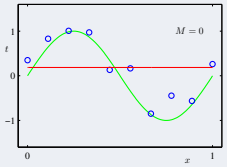
- By definition of $E(\mathbf{w})$:

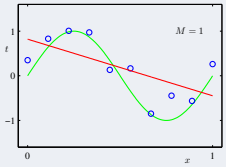
$$\mathbf{w}^{(i)} := \mathbf{w}^{(i-1)} - \eta(t_i - \mathbf{w}^{(i-1)}\boldsymbol{\phi}(\mathbf{x}_i))\boldsymbol{\phi}(\mathbf{x}_i)$$

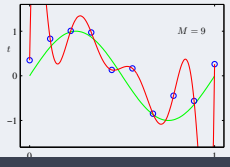
Example: fitting of polynomials

Polynomial degree

- Example of model selection: assigning a value to M determines the model to be used, the choice of M implies the number of coefficients to be estimated
- increasing M allows to better approximate the training set items, decreasing the error
- if $M + 1 = n$ the model allows to obtain a null error (overfitting)







C'è un discorso di bias induttivo: possiamo definire la classe dei polinomi, ma poi possiamo farla variare, ad esempio cercando solo fra le rette, solo i polinomi di grado 2 etc ...
Quindi, pur fissando la classe, c'è un iper-parametro che ha a che fare con la definizione stessa del modello che è il grado in questo caso.

In overfitting.ipynb:
- scegliamo una funzione e generiamo un dataset intorno a questa funzione. Definiamo 30 ascisse:
    - calcoliamo il valore della funzione
    - a parte, calcoliamo una componente di rumore da una Gaussiana
    - abbiamo poi 30 elementi di test set: facciamo regressione sul train set e poi vediamo che accade sul test set.

Example: fitting of polynomials

Overfitting

- The function $y(x, \mathbf{w})$ is derived from items in the training set, but should provide good predictions for other items.
- It should provide a suitable generalization to all items in the whole domain.
- If $y(x, \mathbf{w})$ is derived as a too much accurate depiction of the training set, it results into an unsuitable generalization to items not in the training set

Evaluation of the generalization

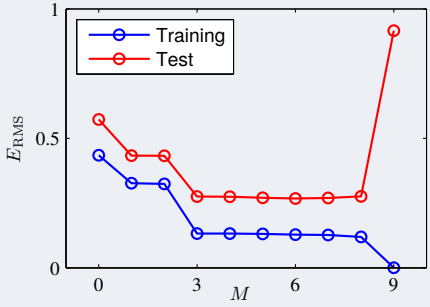
- Test set $\mathbf{X}_{test}$ of 100 new items, generated by uniformly sampling x in $[0, 1,]$ and ε from $N(0, \sigma^2)$, and computing $t = \sin 2\pi x + \varepsilon$
- For each M:
  - derives $\mathbf{w}^*$ from the training set $\mathbf{X}_{train}$
  - compute the error $E(\mathbf{w}^*, \mathbf{X}_{test})$ on the test set, or the square root of its mean

$$E_{RMS}(\mathbf{w}^*, \mathbf{X}_{test}) = \sqrt{\frac{E(\mathbf{w}^*, \mathbf{X}_{test})}{|\mathbf{X}_{test}|}} = \sqrt{\frac{1}{2|\mathbf{X}_{test}|} \sum_{x \in \mathbf{X}_{test}} (y(x, \mathbf{w}) - t)^2}$$

- a lower value of $E_{RMS}(\mathbf{w}^*, \mathbf{X}_{test})$ denotes a good generalization

Example: fitting of polynomials

Plot of E_{RMS} w.r.t. M, on the training set and on the test set.



- As M increases, the error on the training set tends to 0.
- On the test set, the error initially decreases, since the higher complexity of the model allows to better represent the characteristics of the data set. Next, the error increases, since the model becomes too dependent from the training set: the noise component in t is too represented.

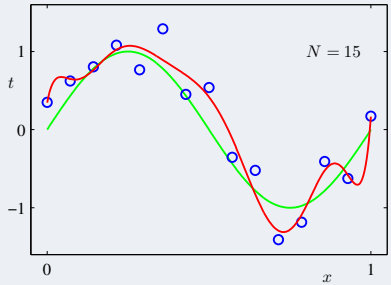
Per il train set ci possiamo aspettare che se aumentiamo il grado, la migliore di quel grado andrà non peggio di quella precedente. Quindi al limite si andrà uguale, ma non peggio.

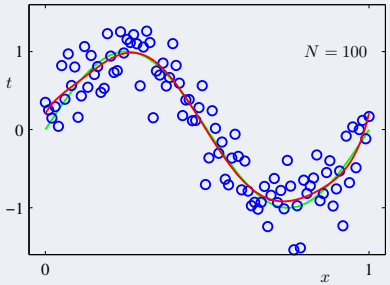
Sul test set, inizialmente le cose vanno meglio, ma ci sarà un punto in cui il modello ha una complessità tale che per il test set il costo comincia ad aumentare. Vediamo quindi bene le due zone di under ed overfitting (sx e dx del grafico): occorre qui usare dei metodi un po' più euristici.

C'è un punto nel mezzo dove le cose vanno bene per entrambi, la cosa più semplice è quindi provare cosa succede per i vari gradi e prendere quello per cui le cose vanno meglio per il test set. Questa è una delle fasi della model selection.

Example: fitting of polynomials

For a given model complexity (such as the degree in our example), overfitting decreases as the dimension of the dataset increases.





The larger the dataset, the higher the acceptable complexity of the model.

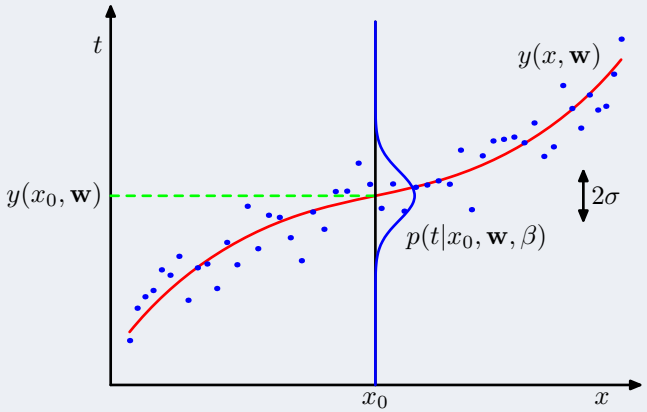
Se il training set è più grande, il modello può avere una complessità maggiore e quindi magari non va in overfitting perché la grandezza del train set non è più piccolo rispetto alla complessità del modello.

Per questo, per supervised learning servono molti dati, in quanto una rete neurale ha milioni di coefficienti altrimenti si rischia che il modello vada in overfitting. Di conseguenza, abbiamo dei tempi di apprendimento lunghi.

Probabilistic model for regression

Assume that, given an item $\mathbf{x}$, the corresponding unknown target t is normally distributed around the value returned by the model $\mathbf{w}^T\overline{\mathbf{x}}$, with a given variance $\sigma^2 = \beta^{-1}$:

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



Abbiamo quindi visto l'approccio lineare, se seguiamo un approccio probabilistico. Qui, ci interessa la distrbuzione di probabilità condizionata $p(t|x)$, quindi vogliamo un valore di probabilità per ogni possibile t. Questa deve essere parametrica, quindi cerchiamo la migliore fra le possibili: nel caso della regressione lineare, assumiamo che le nostre coppie x,t siano state generate come:

- p(x) uniforme
- $p(t|x)$ è stato generato così: c'è un insieme di coefficienti $w_0, w_1, ..., w_k$ che sono i coefficienti del modello. È arrivato quindi x, calcoliamo sempre $w_0 + w_1 x + w_2 x^2 + ...$ che ci da un valore che è x_0 .

  Il valore del target è estratto casualmente da una distr. Gaussiana centrata su y(x_0 , w).

Quindi la probabilità $p(t|x, w, \beta)$ è Gaussiana, con media il valore predetto $t|y(x, w)$ e varianza β^{-1} .

N.B: x e t sono noti. La variabile vera è w, se questa cambia può accadere che cambi la curva rossa, ma quindi può accadere che venga spostata la Gaussiana e che quindi la probabilità di t sia maggiore o minore (a seconda di dove viene spostata la Gaussiana).
La probabilità sarà il prodotto delle probabilità di tutti i punti e quindi cerchiamo il w che massimizza la probabilità congiunta.

An estimate of both β_{ML} and the coefficients $\mathbf{w}_{ML}$ can be performed on the basis of the likelihood w.r.t. the assumed normal distribution:

$$L(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{n} N(t_i|y(\mathbf{x}_i, \mathbf{w}), \beta^{-1})$$

Parameters $\mathbf{w}$ and β can be estimated as the values which maximize the data likelihood, or its logarithm

$$l(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \log p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \sum_{i=1}^{n} \log N(t_i|y(\mathbf{x}_i, \mathbf{w}), \beta^{-1})$$

which results into

$$\begin{aligned} l(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) &= \sum_{i=1}^{n} \log \left(\frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_i - y(\mathbf{x}_i, \mathbf{w}))^2} \right) \\ &= -\sum_{i=1}^{n} \frac{\beta}{2} \left(t_i - y(\mathbf{x}_i, \mathbf{w})\right)^2 + \frac{n}{2} \log \beta - \frac{n}{2} \log(2\pi) \\ &= -\frac{\beta}{2} \sum_{i=1}^{n} \left(t_i - y(\mathbf{x}_i, \mathbf{w})\right)^2 + \frac{n}{2} \log \beta + \text{cost} \end{aligned}$$

È possibile anche stimare quanto è il valore della varianza, ha solo effetto in termini statistici: possiamo stimare quanto devono essere "larghe" le Gaussiane per far si che la verosimiglianza sia la più alta possibile.

Nel caso della regressione lineare, ci arriva un punto e facciamo un predizione (una volta determinato w)

Nel caso del modello probabilistico, abbiamo che dato un punto, se arriva un valore x non abbiamo y bensì una distribuzione Gaussiana, centrata intorno ad y, ma che assegna un valore di probabilità a tutti valori del target.

How to limit the complexity of the model?

Limitiamo la f. di costo: aggiungiamo un termine che ostacoli i coefficienti nell'assumere valori molto grandi.

N.B! attenzione a log reg pts

- Regularization term in the cost function

funzione di $\mathbf{w}, X, t$

sola funzione di $\mathbf{w}$

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

$E_D(\mathbf{w})$ dependent from the dataset (and the parameters), $E_W(\mathbf{w})$ dependent from the parameters alone.

- The regularization coefficient controls the relative importance of the two terms.

Cerco di minimizzare tutta la funzione $E_D(\mathbf{w})$, che non è detto dipenda solo dai valori di $\mathbf{w}$ perché c'è un addendo in più.

$E_D(\underline{w})$: fanno in modo di predire bene i dati, questa dependent.

$\delta_W(\underline{w})$: ostacola i $\underline{w}$ affinché predicano i dati al meglio

λ: quanto la funzione 2 pesa rispetto alla 1. È il coefficiente di regolarizzazione.

- Simple form

norma quadratica del vettore dei coefficienti.

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\sum_{i=1}^{m} w_i^2$$

- Sum-of-squares cost function: ridge regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i))^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Il min. della f. di costo è ancora rappresentabile in forma chiusa.

with solution

$$\mathbf{w} = (\lambda\mathbf{I} + \Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$$

Vogliamo minimizzare: vogliamo $\underline{w}$ che rendano vicini i valori predetti a quelli reali, ma se sono troppo grandi domina il peso aggiunto $\Rightarrow$ limita i valori di $\underline{w}$

Abbiamo una funzione di costo:
  - se lambda conta poco, allora la funzione di costo farà quanto meglio per predirre i valori target
  - ma se lambda è grande, l'esigenza di tenere bassi i valori di w, la solzuione è indipendente però dai dati
  e questo comoprta che se cambio train set mi verrà la stessa cosa perché il risultato è indipendente dai dati.

Abbiamo quiindi bassa varianza, perché anche cambiando train set i risultati sono uguali. Avrò però un bias alto oerché teniamo poco conto dei dati, quindi prediciamo male e sempre allo stesso modo, siamo quindi nel caso bias alto e varianza bassa.

Introduciamo quindi un effetto di limitazione nel modellare il training set

Forma generale del caso precedente

- A more general form

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i))^2 + \frac{\lambda}{2}\sum_{j=1}^{m}|w_j|^q$$

- The case $q = 1$ is denoted as lasso: sparse models are favored

Caso generale, si considera la somma dei moduli dei coefficienti. L'effetto di regolarizzazione è che tende a dare un modello più sparso.

Example: polynomial regression

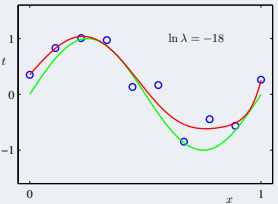
Use of regularization to limit complexity and overfitting.

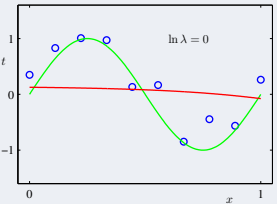
- inclusion of a penalty term in the error function
- purpose: limiting the possible values of coefficients
- usually: limiting the absolute value of the coefficients

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(y(x_i,\mathbf{w}) - t_i\right)^2 + \frac{\lambda}{2}\sum_{k=0}^{M} w_k^2 = \frac{1}{2}\sum_{i=1}^{n}\left(y(x_i,\mathbf{w}) - t_i\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Dependance from the value of the hyperparameter λ. → va fatto su dataset con model selection

λ piccolo (e^{-18}) prende meglio i dati.

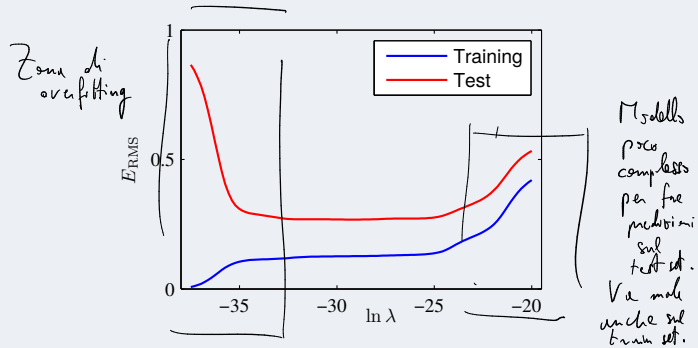




→ λ grande: modello troppo poco conto dei dati. Bias è elevato

Example: polynomial regression

Plot of the error w.r.t λ, ridge regression.



- Small λ: overfitting. Small error on the training set, large error on the test set.
- Large λ: the effect of data values decreases. Large error on both test and training sets.
- Intermediate λ. Intermediate error on training set, small error on test set.

Example: polynomial regression

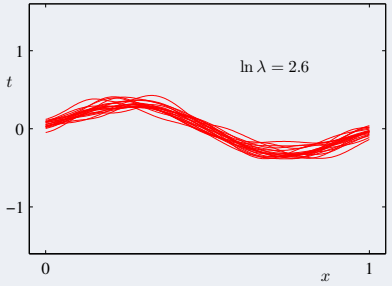
- Consider the case of function $y = \sin 2\pi x$ and assume $L = 100$ training sets $T_1, \dots, T_L$ are available, each of size $n = 25$.
- Given $m = 24$ gaussian basis functions $\phi_1(x), \dots, \phi_m(x)$, from each training set T_i a prediction function $y_i(x)$ is derived by minimizing the regularized cost function

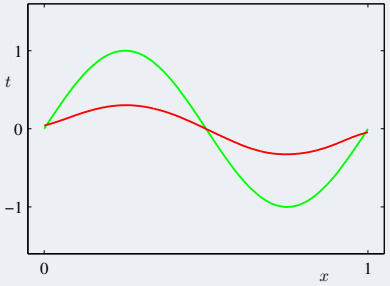
$$E(\mathbf{w}) = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{t})^T(\Phi\mathbf{w} - \mathbf{t}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Usiamo la ridge regression. Troviamo fuori un vettore di dim 24 e determiniamo i migliori coefficienti fissato λ.

Example: polynomial regression

Le nuove curve di predizione dei train set. → Alle x sono applicate le funzioni base.





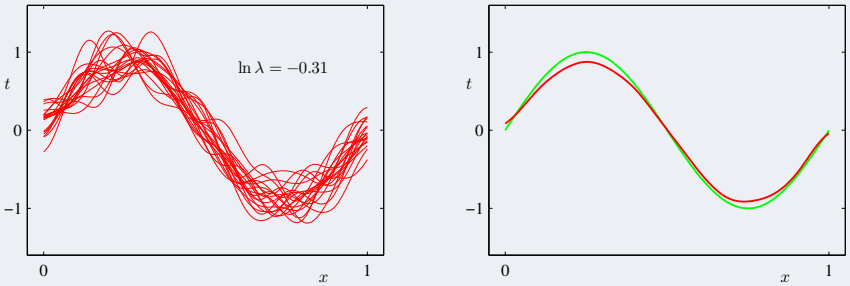
← media delle curve precedenti. La verde è la funzione da predire: molto bias.

Left, a possible plot of prediction functions $y_i(\mathbf{x})$ ($i = 1, \dots, 100$), as derived, respectively, by training sets $T_i, i = 1, \dots, 100$ setting $\ln \lambda = 2.6$. Right, their expectation, with the unknown function $y = \sin 2\pi x$.

The prediction functions $y_i(\mathbf{x})$ do not differ much between them (small variance), but their expectation is a bad approximation of the unknown function (large bias).

In ognuno dei casi vengono diverse funzioni ottenute da diversi train set. Vediamo cosa accade variando λ.

Example: polynomial regression

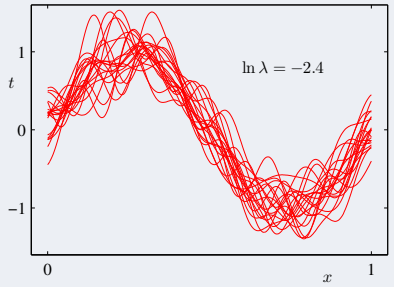


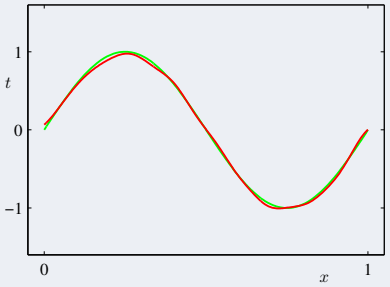
Plot of the prediction functions obtained with $\ln \lambda = -0.31$.

λ più alto $\Rightarrow$ c'è più varianza, più dipendenza dal training set ma bias minore.

Example: polynomial regression

In media meno bene, ma in considero una delle curve => non è detto che sia la migliore. O' la media poi che in bene.





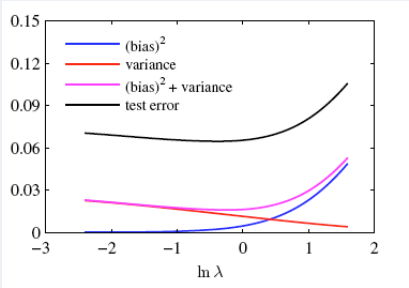
Plot of the prediction functions obtained with $\ln \lambda = -2.4$. As λ decreases, the variance increases (prediction functions $y_i(\mathbf{x})$ are more different each other), while bias decreases (their expectation is a better approximation of $y = \sin 2\pi x$).

Varianza elevata anche non in bene.

Example: polynomial regression

Occorre trovare un valore di λ a livello di model selection per bilanciare gli effetti.

Per scegliere λ: fisso, validation set e cerco la λ migliore in base al MSE migliore.



- Plot of $(\text{bias})^2$, variance and their sum as functions of λ: las λ increases, bias increases and varinace decreases. Their sum has a minimum in correspondance to the optimal value of λ.
- The term $E_{\mathbf{x}}[\sigma^2_{y|\mathbf{x}}]$ shows an inherent limit to the approximability of $y = \sin 2\pi x$.

Con regressione lasso o λ? non ho una forma chiusa, quindi posso ricorrere applicare la discesa del gradiente.

Nella regolarizzazione quindi, lo spazio di ricerca è lo stesso, ma cambiamo la funzione di costo tentando di ostacolare valori che non "ci piacciono".

È una limitazione più smooth (soft) in quanto farla da un punto di vista matematico sarebbe più complesso. Matematicamente si gestisce meglio un problema di ottimizzazione senza vincoli, per questo motivo andiamo a modificare opprtunamente la funzione di costo.

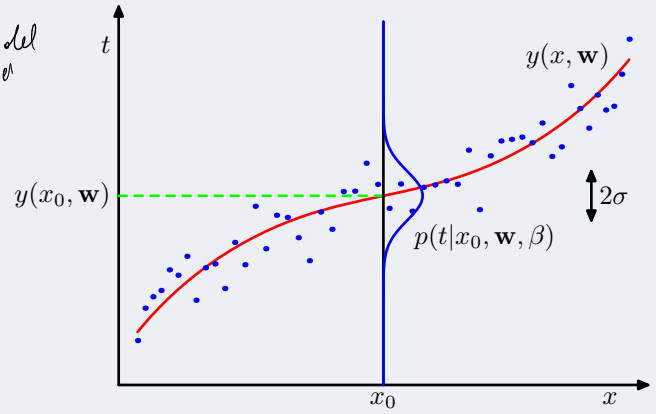
Probabilistic model for regression

Assume that, given an item $\mathbf{x}$, the corresponding unknown target t is normally distributed around the value returned by the model $\mathbf{w}^T\overline{\mathbf{x}}$, with a given variance $\sigma^2 = \beta^{-1}$:

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

Usiamo la max. verosimiglianza: fissato $\mathbf{w}$, fissato $\mathbf{x}$ posizione verticale della Gaussiana e la prob. del target sono diversi $\rightarrow$ ci la max likelihood.

TAGLIA DA QUI IN GIÙ?



An estimate of both β_{ML} and the coefficients $\mathbf{w}_{ML}$ can be performed on the basis of the likelihood w.r.t. the assumed normal distribution:

ipotesi di indipendenza.

$$L(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{n} N(t_i|y(\mathbf{x}_i, \mathbf{w}), \beta^{-1})$$

Parameters $\mathbf{w}$ and β can be estimated as the values which maximize the data likelihood, or its logarithm

$$l(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \log p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \sum_{i=1}^{n} \log N(t_i|y(\mathbf{x}_i, \mathbf{w}), \beta^{-1})$$

which results into

$$\begin{aligned} l(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) &= \sum_{i=1}^{n} \log \left(\frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_i - y(\mathbf{x}_i, \mathbf{w}))^2} \right) \\ &= -\sum_{i=1}^{n} \frac{\beta}{2} (t_i - y(\mathbf{x}_i, \mathbf{w}))^2 + \frac{n}{2} \log \beta - \frac{n}{2} \log(2\pi) \\ &= -\frac{\beta}{2} \sum_{i=1}^{n} (t_i - y(\mathbf{x}_i, \mathbf{w}))^2 + \frac{n}{2} \log \beta + \text{cost} \end{aligned}$$

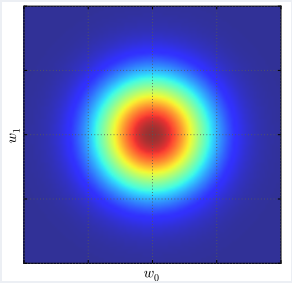
frontiera di costo dei minimi quadrati.

Voglio i valori di $\mathbf{w}$ tale che la prob. del target sia la più alta.

Probabilistic model for regression

Prior distribution of parameters: gaussian with mean $\mathbf{0}$ and diagonal covariance matrix with variance equal to the inverse of hyperparameter α

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



Why a gaussian prior?

Posterior proportional to prior times likelihood: likelihood is gaussian (gaussian noise).

$$p(\mathbf{t}|\Phi, \mathbf{w}, \beta) = \prod_{i=1}^{n} \mathcal{N}(t_i|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1}) = \prod_{i=1}^{n} e^{-\frac{\beta}{2}(t_i - \mathbf{w}^T\boldsymbol{\phi}(x_i))^2}$$

Given the prior $p(\mathbf{w}|\alpha)$, the posterior distribution for $\mathbf{w}$ derives from Bayes' rule

$$p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \sigma) = \frac{p(\mathbf{t}|\Phi, \mathbf{w}, \sigma)p(\mathbf{w}|\alpha)}{p(\mathbf{t}|\Phi, \alpha, \sigma)} \propto p(\mathbf{t}|\Phi, \mathbf{w}, \sigma)p(\mathbf{w}|\alpha)$$

Why a gaussian prior?

In general, conjugate of gaussian is gaussian: choosing a gaussian prior distribution of $\mathbf{w}$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

results into a gaussian posterior distribution

$$p(\mathbf{w}|\mathbf{t}, \Phi) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

where

$$\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\Phi^T\mathbf{t})$$
$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\Phi^T\Phi$$

Why a gaussian prior?

Here, we have

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \qquad p(\mathbf{t}|\mathbf{w}, \Phi) = \mathcal{N}(\mathbf{t}|\mathbf{w}^T\Phi, \beta^{-1}\mathbf{I})$$

and the posterior distribution is gaussian

$$p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \sigma) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

with

$$\mathbf{S}_N = (\alpha\mathbf{I} + \beta\Phi^T\Phi)^{-1} \qquad \mathbf{m}_N = \beta\mathbf{S}_N\Phi^T\mathbf{t}$$

Why a gaussian prior?

Se ci sbilanciamo poco sul prior, andiamo vicino la verosimiglianza.

↳ conta poco

Note that as $\alpha \to 0$ the prior tends to have infinite variance, and we have minimum information on $\mathbf{w}$ before the training set is considered. In this case,

$$\mathbf{m}_N \to (\Phi^T \beta \mathbf{I} \Phi)^{-1}(\Phi^T \beta \mathbf{I} \mathbf{t}) = (\Phi^T \Phi)^{-1}(\Phi^T \mathbf{t})$$

espressione della media che massimizza la verosimiglianza, dove il prior pesa poco.

that is $\mathbf{w}_{ML}$, the ML estimation of $\mathbf{w}$.

- Given the posterior distribution $p(\mathbf{w}|\Phi, \mathbf{t}, \alpha, \beta)$, we may derive the value of $\mathbf{w}_{MAP}$ which makes it maximum (the mode of the distribution)
- This is equivalent to maximizing its logarithm

$$\log p(\mathbf{w}|\Phi, \mathbf{t}, \alpha, \beta) = \log p(\mathbf{t}|\mathbf{w}, \Phi, \beta) + \log p(\mathbf{w}|\alpha) - \log p(\mathbf{t}|\Phi, \beta)$$

non conta. è l'evidenza.

and, since $p(\mathbf{t}|\Phi, \beta)$ is a constant wrt $\mathbf{w}$

$$\mathbf{w}_{MAP} = \underset{\mathbf{w}}{\operatorname{argmax}} \log p(\mathbf{w}|\Phi, \mathbf{t}, \alpha, \beta) = \underset{\mathbf{w}}{\operatorname{argmax}} \left(\log p(\mathbf{t}|\mathbf{w}, \Phi, \beta) + \log p(\mathbf{w}|\alpha)\right)$$

that is,

$$\mathbf{w}_{MAP} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(-\log p(\mathbf{t}|\Phi, \mathbf{w}, \beta) - \log p(\mathbf{w}|\alpha)\right)$$

In this case

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}; \alpha, \beta) \propto p(\mathbf{t}|\mathbf{X}, \mathbf{w}; \beta)p(\mathbf{w}|\alpha)$$
$$= \prod_{i=1}^{n} \left(\frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_i - y(\mathbf{x}_i, \mathbf{w}))^2} \right) \left(\frac{\alpha}{2\pi} \right)^{\frac{M+1}{2}} e^{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}}$$

The maximization of the posterior distribution (MAP) is equivalent to the maximization of the corresponding logarithm

$$-\frac{\beta}{2} \sum_{i=1}^{n} (t_i - y(\mathbf{x}_i, \mathbf{w}))^2 + \frac{n}{2} \log \beta - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \frac{m+1}{2} \log \frac{\alpha}{2\pi} + \text{cost}$$

The value $\mathbf{w}_{MAP}$ which maximize the probability (mode of the distribution) also minimizes

$$\frac{\beta}{2} \sum_{i=1}^{n} (t_i - y(\mathbf{x}_i, \mathbf{w}))^2 + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} = \beta \left(\frac{1}{2} \sum_{i=1}^{n} (t_i - y(\mathbf{x}_i, \mathbf{w}))^2 + \frac{\alpha}{2\beta} \|\mathbf{w}\|^2 \right)$$

The ratio $\frac{\alpha}{\beta}$ corresponds to a regularization hyperparameter.

Avere introdotto la MAP ci ha permesso di inserire la regolarizzazione

Con la MAP, non vogliamo che i dati contino troppo ovvero non vogliamo modellarli troppo. La presenza del prior sta a bilanciare la verosimiglianza: massimizzare la verosimilgianza vuol dire trovare i valori di w che meglio permettono di predire i dati. Con il prior, non vogliamo i migliori w che perdicano i dati, ma siano quelli che vengono dal prior.

È il target condizionato alle feature (l'elemento) dato il modello.

The same considerations of ML appy here for what concerns deriving the predictive distribution of t given $\mathbf{x}$, which results now

$$p(t|\mathbf{x}; \mathbf{w}, \beta_{MAP}) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta_{MAP}^{-1}) = \sqrt{\frac{\beta_{MAP}}{2\pi}} e^{-\frac{\beta_{MAP}}{2}(t-y(\mathbf{x}, \mathbf{w}_{MAP}))^2}$$

where, as it is easy to see, $\beta_{MAP} = \beta_{ML}$

Ha la stessa forma della distrib. precedente, ma al posto di $\mathbf{w}_{ML}$ c'è $\mathbf{w}_{MAP}$

Possiamo assumere di stimare la varianza per max. verosimiglianza.

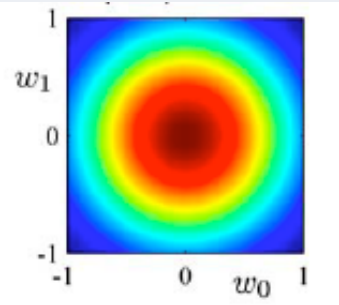
- The posterior after observing T_1 can be used as a prior for the next training set acquired.
- In general, for a sequence $T_1, \dots, T_n$ of training sets,

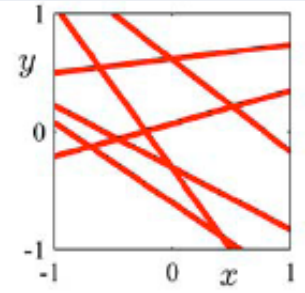
- Parto da una ipotesi di prior
- Acquisisco sempre nuovi punti del training set
- Ottengo il nuovo prior.

$$
\begin{aligned}
&p(\mathbf{w}|T_1, \dots T_n) \propto p(T_n|\mathbf{w}) p(\mathbf{w}|T_1, \dots T_{n-1}) \\
&p(\mathbf{w}|T_1, \dots T_{n-1}) \propto p(T_{n-1}|\mathbf{w}) p(\mathbf{w}|T_1, \dots T_{n-2}) \\
&\dots \\
&p(\mathbf{w}|T_1) \propto p(T_1|\mathbf{w}) p(\mathbf{w})
\end{aligned}
$$

Example

- Input variable x, target variable t, linear regression $y(x, w_0, w_1) = w_0 + w_1 x$.
- Dataset generated by applying function $y = a_0 + a_1 x$ (with $a_0 = -0.3$, $a_1 = 0.5$) to values uniformly sampled in $[-1, 1]$, with added gaussian noise ($\mu = 0$, $\sigma = 0.2$).
- Assume the prior distribution $p(w_0, w_1)$ is a bivariate gaussian with $\boldsymbol{\mu} = \mathbf{0}$ and $\Sigma = \sigma^2 \mathbf{I} = 0.04\mathbf{I}$

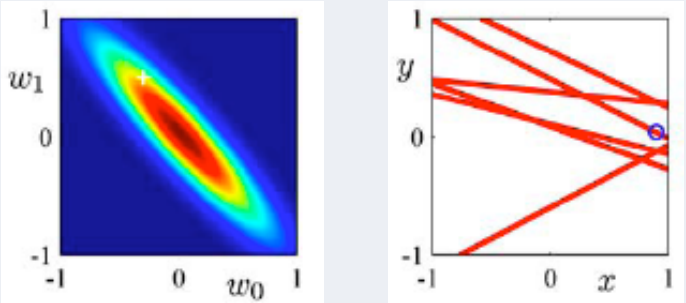




Left, prior distribution of w_0, w_1; right, 6 lines sampled from the distribution.

Example

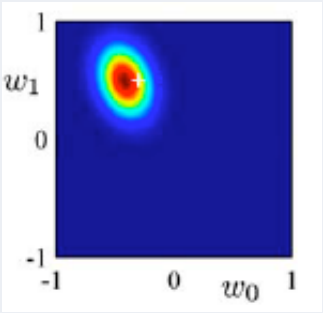
After observing item (x_1, y_1) (circle in right figure).

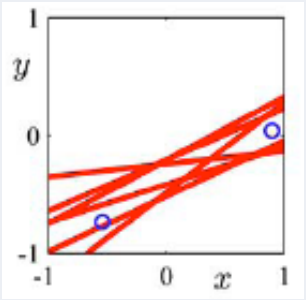


Left, posterior distribution $p(w_0, w_1|x_1, y_1)$; right, 6 lines sampled from the distribution.

Esempio

After observing items $(x_1, y_1), (x_2, y_2)$ (circles in right figure).

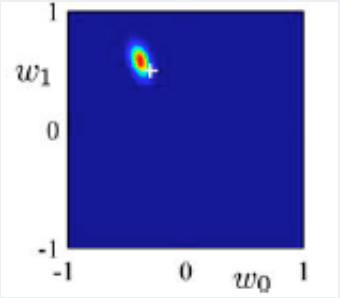


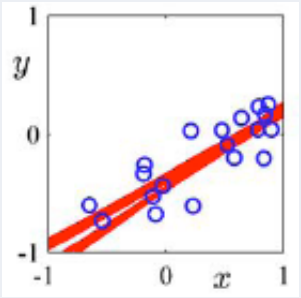


Left, posterior distribution $p(w_0, w_1|x_1, y_1, x_2, y_2)$; right, 6 lines sampled from the distribution.

Example

After observing a set of n items $(x_1, y_1), \dots, (x_n, y_n)$ (circles in right figure).





Left, posterior distribution $p(w_0, w_1 | x_i, y_i, i = 1, \dots, n)$; right, 6 lines sampled from the distribution.

Example

- As the number of observed items increases, the distribution of parameters w_0, w_1 tends to concentrate (variance decreases to 0) around a mean point a_0, a_1.
- As a consequence, sampled lines are concentrated around $y = a_0 + a_1 x$.

Approaches to prediction in linear regression

Classical

- A value $\mathbf{w}_{LS}$ for $\mathbf{w}$ is learned through a point estimate, performed by minimizing a quadratic cost function, or equivalently by maximizing likelihood (ML) under the hypothesis of gaussian noise; regularization can be applied to modify the cost function to limit overfitting
- Given any $\mathbf{x}$, the obtained value $\mathbf{w}_{LS}$ is used to predict the corresponding t as $y = \overline{\mathbf{x}}^T \mathbf{w}_{LS}$, where $\overline{\mathbf{x}}^T = (1, \mathbf{x})^T$, or, in general, as $y = \boldsymbol{\phi}(\mathbf{x})^T \mathbf{w}_{LS}$

Bayesian point estimation

- The posterior distribution $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta)$ is derived and a point estimate is performed from it, computing the mode $\mathbf{w}_{MAP}$ of the distribution (MAP)
- Equivalent to the classical approach, as $\mathbf{w}_{MAP}$ corresponds to $\mathbf{w}_{LS}$ if $\lambda = \frac{\alpha}{\beta}$
- The prediction, for a value $\mathbf{x}$, is a gaussian distribution $p(y|\boldsymbol{\phi}(\mathbf{x})^T\mathbf{w}_{MAP}, \beta)$ for y, with mean $\boldsymbol{\phi}(\mathbf{x})^T\mathbf{w}_{MAP}$ and variance β^{-1}
- The distribution is not derived directly from the posterior $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta)$: it is built, instead, as a gaussian with mean depending from the expectation of the posterior, and variance given by the assumed noise.

Abbiamo quindi due approcci, che condividono: abbiamo la distribuzione di w_0 e w_1, tutto il processo porta alla stima di un vettore di (w_1,w_0) che descrive una retta di regressione che usiamo.

Lo possiamo fare per massima verosimiglianza o con la MAP.

Fully bayesian

- The real interest is not in estimating $\mathbf{w}$ or its distribution $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta)$, but in deriving the predictive distribution $p(y|\mathbf{x})$. This can be done through expectation of the probability $p(y|\mathbf{x}, \mathbf{w}, \beta)$ predicted by a model instance wrt model instance distribution $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta)$, that is

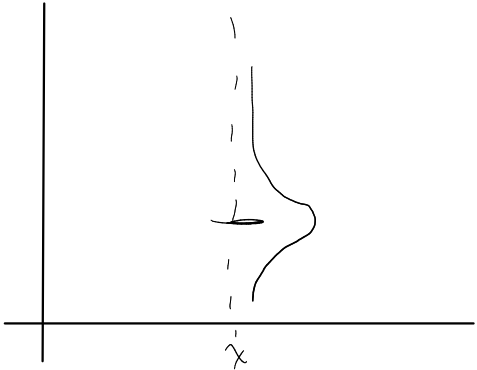
$$p(y|\mathbf{x}, \mathbf{t}, \Phi, \alpha, \beta) = \int p(y|\mathbf{x}, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta) d\mathbf{w}$$

- $p(y|\mathbf{x}, \mathbf{w}, \beta)$ is assumed gaussian, and $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta)$ is gaussian by the assumption that the likelihood $p(\mathbf{t}|\mathbf{w}, \Phi, \beta)$ and the prior $p(\mathbf{w}|\alpha)$ are gaussian themselves and by their being conjugate

$$p(y|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(y|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \beta)$$
$$p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\beta \mathbf{S}_N \Phi^T \mathbf{t}, \mathbf{S}_N)$$

where $\mathbf{S}_N = (\alpha \mathbf{I} + \beta \Phi^T \Phi)^{-1}$

Abbiamo $p(w \mid t, X)$ vogliamo considerare le predizioni di tutte le rette di regressione pesate con le prob. di quelle rette di regressione.



Pesiamo la predizione di ogni retta di regressione, pesando con la probabilità che il modello sia quello con quei valori di w e facciamo la media, ovvero il valor medio della Gaussiana.

Nel caso MAP, prendiamo la moda della distribuzione, qui consideriamo tutti i valori attesi. Otteniamo una distribuzione del valore target associato all'elemento che abbiamo derivato a partire dai dati.

L'integrale non è sempre risolvibile analiticamente, è possibile in casi semplici. Semplici nel senso che se possiamo dire che entrambe le distribuzioni del prodotto nell'integrale sono Gaussiane, allora anche la distribuzione predittiva lo è.

Fully bayesian

Under such hypothesis, $p(y|\mathbf{x})$ is gaussian

$$p(y|\mathbf{x}, \mathbf{y}, \Phi, \alpha, \beta) = \mathcal{N}(y|m(\mathbf{x}), \sigma^2(\mathbf{x}))$$

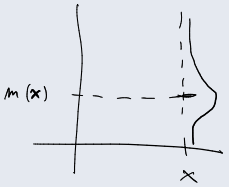
with mean

and variance

dipendono da $\mathbf{x}$

$$m(\mathbf{x}) = \beta \boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \Phi^T \mathbf{t}$$

$$\sigma^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$



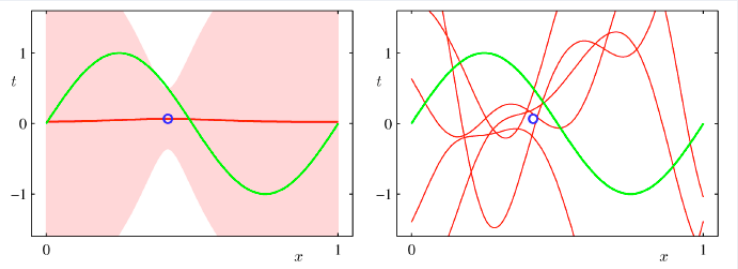
- $\frac{1}{\beta}$ is a measure of the uncertainty intrinsic to observed data (noise)
- $\boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$ is the uncertainty wrt the values derived for the parameters $\mathbf{w}$
- as the noise distribution and the distribution of $\mathbf{w}$ are independent gaussians, their variances add
- $\boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}) \to 0$ as $n \to \infty$, and the only uncertainty remaining is the one intrinsic into data observation

Example

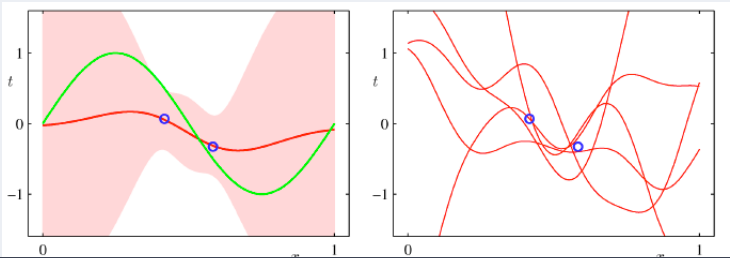
- predictive distribution for $y = \sin 2\pi x$, applying a model with 9 gaussian base functions and training sets of 1, 2, 4, 25 items, respectively
- left: items in training sets (sampled uniformly, with added gaussian noise); expectation of the predictive distribution (red), as function of x; variance of such distribution (pink shade within 1 standard deviation from mean), as a function of x
- right: items in training sets, 5 possible curves approximating $y = \sin 2\pi x$, derived through sampling from the posterior distribution $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta)$

Example

$n = 1$

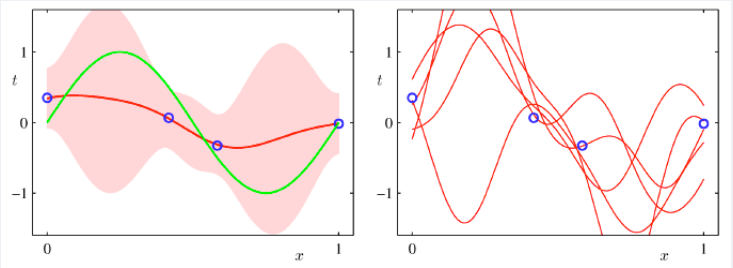


$n = 2$

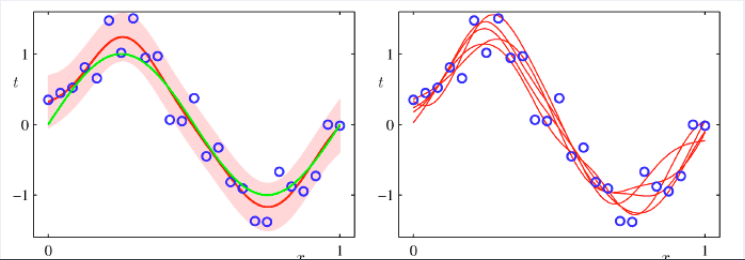


Example

$n = 4$



$n = 25$



Fully bayesian regression and hyperparameter marginalization

α e β sono assunti fissati nel modello. Posso avere un modello più complesso con α e β che hanno una distrib. che uso a stimare.

- In a fully bayesian approach, also the hyper-parameters α, β are marginalized

$$p(t|\mathbf{x}, \mathbf{t}, \Phi) = \int p(t|\mathbf{x}, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta) p(\alpha, \beta|\mathbf{t}, \Phi) d\mathbf{w} d\alpha d\beta$$

  where, as seen before,
  - $p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \beta)$
  - $p(\mathbf{w}|\mathbf{t}, \Phi, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$, with $\mathbf{S}_N = (\alpha \mathbf{I} + \beta \Phi^T \Phi)^{-1}$ e $\mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{t}$

  this marginalization wrt $\mathbf{w}, \alpha, \beta$ is analytically intractable
- we may consider an approximation where point estimation is applied to derive hyper-parameter values by maximizing the posterior distribution $p(\alpha, \beta|\mathbf{t}, \Phi)$

Considero tutti i valori di α, β, per ognuno considero una retta di regressione, per ognuno considero un valore del target $\Rightarrow$ complesso, ma approssimato.

Sono i modelli Bayesiani gerarchici.

- since $p(\alpha, \beta|\mathbf{t}, \Phi) \propto p(\mathbf{t}|\Phi, \alpha, \beta)p(\alpha, \beta)$, if we assume that $p(\alpha, \beta)$ is relatively flat, then

$$\underset{\alpha,\beta}{\operatorname{argmax}}\ p(\alpha, \beta|\mathbf{t}, \Phi) \simeq \underset{\alpha,\beta}{\operatorname{argmax}}\ p(\mathbf{t}|\Phi, \alpha, \beta)$$

  and we may consider the maximization of the marginal likelihood (marginal wrt to coefficients $\mathbf{w}$)

$$p(\mathbf{t}|\Phi, \alpha, \beta) = \int p(\mathbf{t}|\mathbf{w}, \Phi, \beta)p(\mathbf{w}|\alpha)d\mathbf{w}$$

- if we assume that $p(\Phi)$ is constant this is equivalent to maximize the evidence

$$p(\Phi, \mathbf{t}|\alpha, \beta) = p(\mathbf{t}|\Phi, \alpha, \beta)p(\Phi|\alpha, \beta) \propto p(\mathbf{t}|\Phi, \alpha, \beta)$$

No?

Equivalent kernel

- The expectation of the predictive distribution can be written as

$$y(\mathbf{x}) = \beta\boldsymbol{\phi}(\mathbf{x})^T\mathbf{S}_N\Phi^T\mathbf{t} = \sum_{i=1}^{n}\beta\boldsymbol{\phi}(\mathbf{x})^T\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}_i)t_i$$

Algebra! prodotto fra matrici come somma di elem. di questa espressione.

- The prediction can be seen as a linear combination of the target values t_i of items in the training set, with weights dependent from the item values $\mathbf{x}_i$ (and from $\mathbf{x}$)

$$y(\mathbf{x}) = \sum_{i=1}^{n}\kappa(\mathbf{x}, \mathbf{x}_i)t_i$$

The weight function $\kappa(\mathbf{x}, \mathbf{x}') = \beta\boldsymbol{\phi}(\mathbf{x})^T\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}')$ is said equivalent kernel

è una rappresentazione compatta dei dati, qui non ce l'ho. Applico una certa $g(x, X, t)$ e faccio la predizione, i dati stessi sono i parametri.

ϕ(x) è il valore rispetto al quale vogliamo effettuare la predizione

Il valore predetto dato x è dato da una cosa che considera un elemento alla volta del train set, per ognuno fa un'operazione dove compaiono:
- feature dell'elemento da predirre
- feature dell'elemento considerato è come avere: $\sum_{i=1}^{n} f(x, x_i) t_i$

la prima parte è beta ... che è una funzione di x e del punto i-esimo che sto considerando. Questa funzione moltiplica il target del punto del train set. Questo vuol dire che la predizzione effettuata è calcolato mediante una combinazione lineare di tutti i punti del train set, ognuno pesato dalla f(x, x_i) che mette in relazione l'elemento x con x_i.

f misura la similitudine: è più alta tanto più x è vicino ad x_i, ci dice che il target di x si può prevedere considerando i target di tutti i punti che conosco, combinandoli insieme ma pesandoli per la vicinanza, i punti più vicini peseranno di più.

la f sarebbe la k, il target associato ad un punto più vicino ad x ha un effetto maggiore nel determinare la predizione. Una funzione di questo tipo prende il nome di kernel, questo è il kernel equivalente.

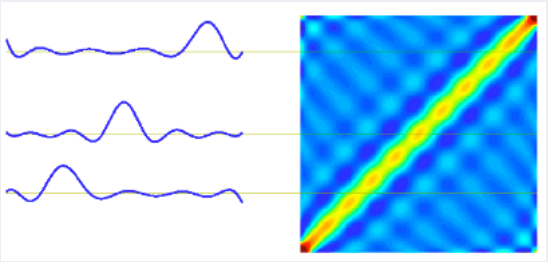
I coefficienti w non ci sono più: la predizione viene effettuata in modo sostanzialmente diverso, detto duale la predizione del target viene effettuata combinando insieme i valori target degli elementi nel train set pesandoli mediante la funzione kernel. Non è più un approccio parametrico: prima, tiravamo fuori i parametri e noti i valori, facevamo la predizione mentre qui no. Ogni volta che si fa la predizione, si considerano i dati, è un esempio di modello non parametrico, o meglio i singoli punti del train set sono i parametri.

La funzione k c'è sempre, mi da il peso di quanto l'elemento per cui effetto la predizione è vicino (simile) a quello del training set.

Equivalent kernel

Right: plot on the plane (x, x_i) of a sample equivalent kernel, in the case of gaussian basis functions.

Left: plot as a function of x_i for three different values of x



In deriving y, the equivalent kernel tends to assign greater relevance to the target values t_i corresponding to items x_i near to x.