Regressione

- Lineare
  - Simple linear regression
  - Multiple linear regression
  - Accuratezza di predizione
  - Variabile indipendente non lineare
  - Regression vs Geometrical fitting
- Non lineare
  - Dipendenza nota: ottimizzazione numerica
  - Regressori non lineari
  - Support Vector Regressor
  - Random Forest Regressor
  - Gradient Boosting Regression
  - Quantile regression
  - Riepilogo

Definizioni

- Nei problemi di classificazione supervisionata ad ogni pattern $\mathbf{x}_i$ del training set è associata un'etichetta y_i (classe di appartenenza). Obiettivo del training è l'apprendimento di un mapping dallo spazio dei pattern a quello (discreto) delle etichette.
- Nella regressione i valori y_i sono numerici e continui (in $\Re$), e l'obiettivo del training è l'apprendimento di una funzione $f(\mathbf{x}) \rightarrow y$.
  - f è controllata da un insieme di parametri. Se la dipendenza dai parametri è lineare (es. assenza di elevazioni a potenza dei parametri nella definizione di f), la regressione si definisce lineare.
  - Nella regressione $\mathbf{x}$ è detta variabile indipendente e y variabile dipendente.
  - Si assume che la variabile indipendente sia esatta mentre la variabile dipendente sia affetta da errore (es. imprecisione di misura).
  - Se la variabile indipendente è uno scalare x (i.e., una sola variabile indipendente), parliamo di simple linear regression (retta di regressione)
  - Se la variabile indipendente è un vettore $\mathbf{x}$ (i.e., più variabili indipendenti), parliamo di multiple linear regression (esempio: iperpiano di regressione).

Simple Linear Regression

- Dato un training set TS contenente n pattern $(x_1, y_1) \dots (x_n, y_n)$, e la seguente dipendenza tra variabile indipendente e dipendente:

$$y_i = \alpha + \beta \cdot x_i + \varepsilon_i$$

dove:

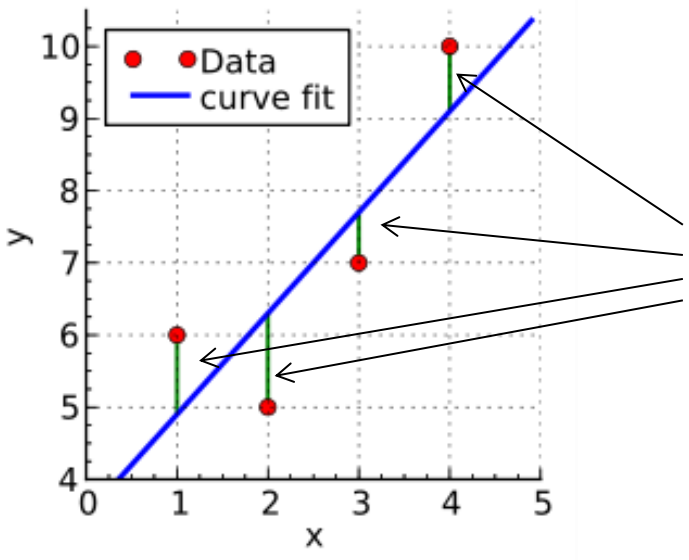
- ε_i è l'errore di misura (incognito) del pattern x_i
- α, β sono parametri da determinare

la soluzione ai minimi quadrati (Least Square) del problema consiste nel determinare l'equazione della retta:

$$f(x) = y = \alpha + \beta \cdot x$$

che minimizza la somma dei quadrati dei residui:

$$\alpha^*, \beta^* = arg \min_{\alpha,\beta} \sum_{i=1\dots n} (f(x) - y_i)^2 = \sum_{i=1\dots n} {\varepsilon_i}^2$$



Simple Linear Regression (2)

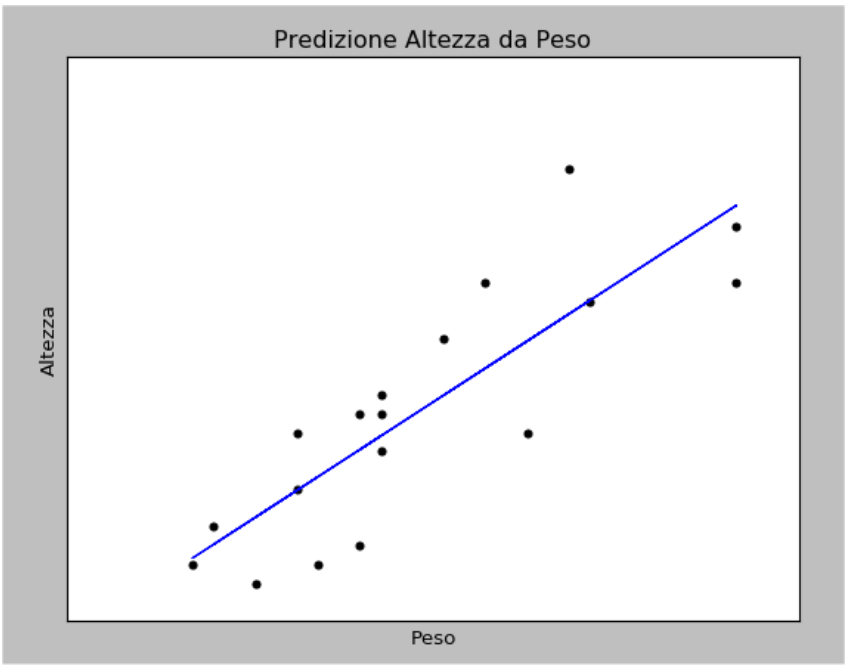
- Il precedente problema di minimizzazione può essere facilmente risolto uguagliando a zero le derivate parziali della funzione obiettivo rispetto ai parametri α, β ottenendo le cosiddette equazioni normali.
  - risolvendo il sistema (due equazioni in due incognite) si ottiene:

$$\beta^* = \frac{\sum_{i=1\ldots n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1\ldots n}(x_i - \bar{x})^2}, \qquad \alpha^* = \bar{y} - \beta^* \bar{x}$$

    dove $\bar{x}$ e $\bar{y}$ sono la media degli x_i e y_i rispettivamente.

    Necessari almeno due punti (con x_i diverse), altrimenti il denominatore si annulla.

- Esempio: predizione dell'altezza a partire dal peso (scikit-learn).



Peso	Altezza
72	173
54	159
65	172
58	170
62	165
72	176
60	173
64	179
55	166
46	158
52	158
47	160
55	167
54	166
55	164
49	157
51	165
51	162

Multiple Linear Regression

- Generalizziamo ora a iperpiani ($\mathbf{x} \in \Re^d$).
- Per semplicità di trattazione sia $\mathbf{X}$ la matrice rettangolare $(n \times (d+1))$ ottenuta inserendo sulle righe gli n pattern del TS, e avendo cura di inserire a destra una colonna con tutti valori 1.

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1d} & 1 \\ x_{21} & & x_{2d} & 1 \\ \vdots & \ddots & \vdots & \\ x_{n1} & \cdots & x_{nd} & 1 \end{bmatrix}$$

pattern $\mathbf{x}_1$

- Siano:

uno scalare per ogni pattern

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \text{e} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{d+1} \end{bmatrix}$$

termine noto

i vettori costruiti a partire dai valori della variabile indipendente (uno per ogni pattern) e dai $d+1$ parametri da determinare.

- Le relazioni tra variabili indipendenti e dipendente possono essere scritte attraverso le equazioni:

$$y_i = \sum_{j=1\ldots d+1} x_{ij} \cdot \beta_j \qquad \text{per} \quad i = 1 \ldots n$$

indici i, j riferiti alla matrice $\mathbf{X}$

o in forma matriciale come:

$$\mathbf{y} = \mathbf{X}\,\boldsymbol{\beta}$$

Per $n > (d+1)$, $\mathbf{X}$ è rettangolare $\rightarrow$ sistema di equazioni sovradeterminato.

Multiple Linear Regression (2)

- In formato matriciale la funzione obiettivo da minimizzare (Least Square) può essere scritta come:

$$\|\mathbf{y} - \mathbf{X}\,\boldsymbol{\beta}\|^2$$

Ancora una volta le equazioni normali possono essere ottenute derivando rispetto ai parametri. Per la derivazione «elegante» in formato matriciale vedi:

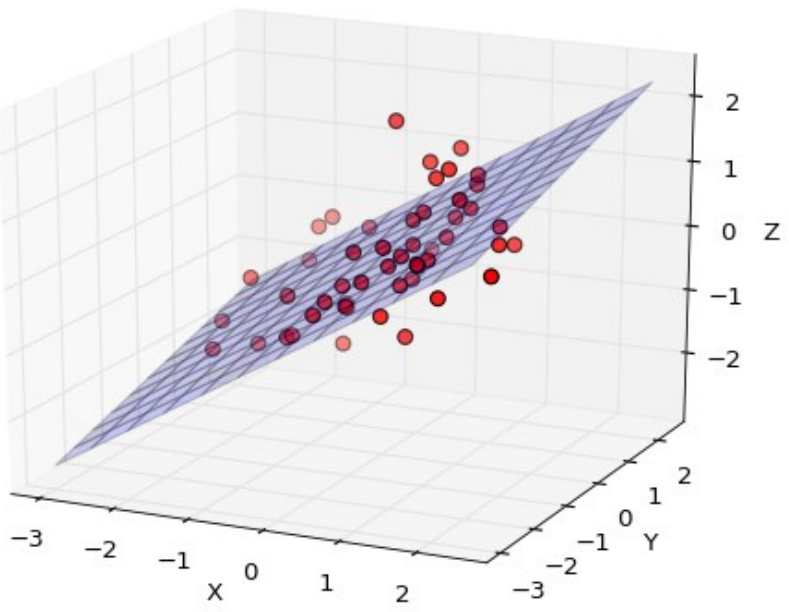
https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics).

$$(\mathbf{X}^t\mathbf{X})\boldsymbol{\beta}^* = \mathbf{X}^t\mathbf{y}$$

Infine, invertendo otteniamo i parametri del modello:

$$\boldsymbol{\beta}^* = (\mathbf{X}^t\mathbf{X})^{-1}\,\mathbf{X}^t\mathbf{y}$$

Esempio d = 2:



Multiple Linear Regression (3)

- Nota bene:
  - Se $n = (d+1)$, la matrice $\mathbf{X}$ è quadrata e l'iperpiano di regressione tocca tutti i punti (interpolazione)
  - Se $n > (d+1)$, il numero di equazioni è superiore al numero di incognite e l'iperpiano approssima i punti.
  - Per essere invertibile la matrice $\mathbf{X}^t\mathbf{X}$ deve essere a rango massimo: problemi in caso di punti collineari (colonne di $\mathbf{X}$ linearmente dipendenti).
  - La matrice $\mathbf{X}^t\mathbf{X}$ è spesso «mal condizionata» e il calcolo dell'inversa può risultare numericamente instabile. Il problema può essere risolto in modo numericamente più stabile attraverso decomposizione SVD (Singular Value Decomposition) di $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Gamma}\mathbf{V}^t$$

Dove $\mathbf{U}$ è una matrice ortogonale $n \times n$, $\mathbf{V}$ è una matrice ortogonale $(d+1) \times (d+1)$ e $\boldsymbol{\Gamma}$ una matrice diagonale rettangolare $n \times (d+1)$ (i.e., possiede elementi non nulli solo quando gli indici di riga e colonna coincidono).

Si dimostra che:

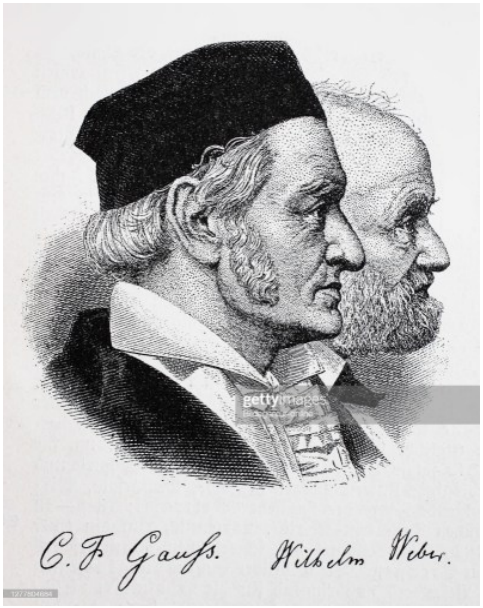
$$\boldsymbol{\beta}^* = \mathbf{V}\boldsymbol{\Gamma}^+\mathbf{U}^t\,\mathbf{y}$$

dove $\boldsymbol{\Gamma}^+$ (denominata pseudoinversa di $\boldsymbol{\Gamma}$), vista la particolare forma di $\boldsymbol{\Gamma}$, si ottiene semplicemente invertendo gli elementi non nulli di $\boldsymbol{\Gamma}$ e calcolando la trasposta.

Chi scoprì la regressione lineare?

- L'approccio di regressione lineare è noto da molto tempo ed è stato usato in molteplici discipline. Curiosa questa storiella sulla sua sulla doppia scoperta (rif. The Batch, #146, Andrew Ng) :

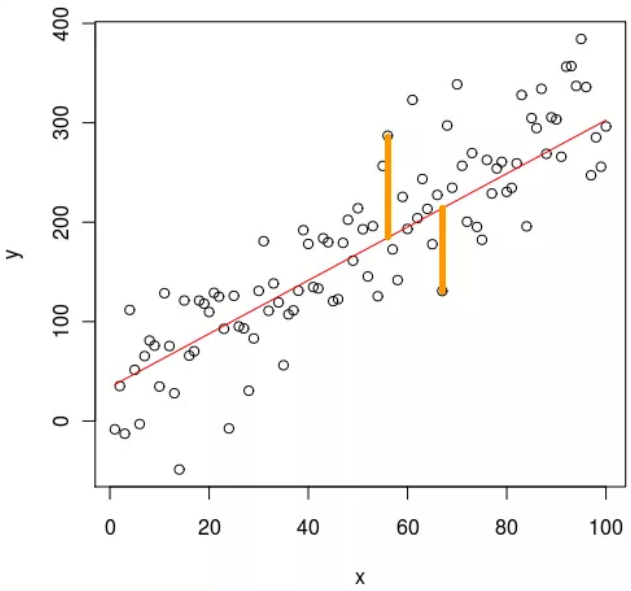
  In 1805, French mathematician Adrien-Marie Legendre (ritratto sotto a destra) published the method of fitting a line to a set of points while trying to predict the location of a comet (celestial navigation being the science most valuable in global commerce at the time, much like AI is today — the new electricity, if you will, two decades before the electric motor). Four years later, the 24-year-old German wunderkind Carl Friedrich Gauss (ritratto sotto a sinistra) insisted that he had been using it since 1795 but had deemed it too trivial to write about. Gauss' claim prompted Legendre to publish an addendum anonymously observing that "a very celebrated geometer has not hesitated to appropriate this method."



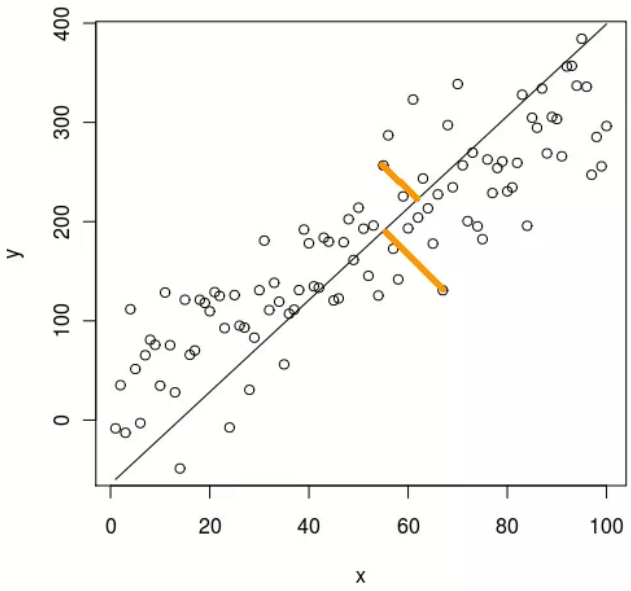
Regression vs Geometrical fitting

- *La regressione lineare (least square estimator) è un buon metodo per il fitting geometrico di nuvole di punti 2D o 3D?*

Linear Regression
(Least Square)



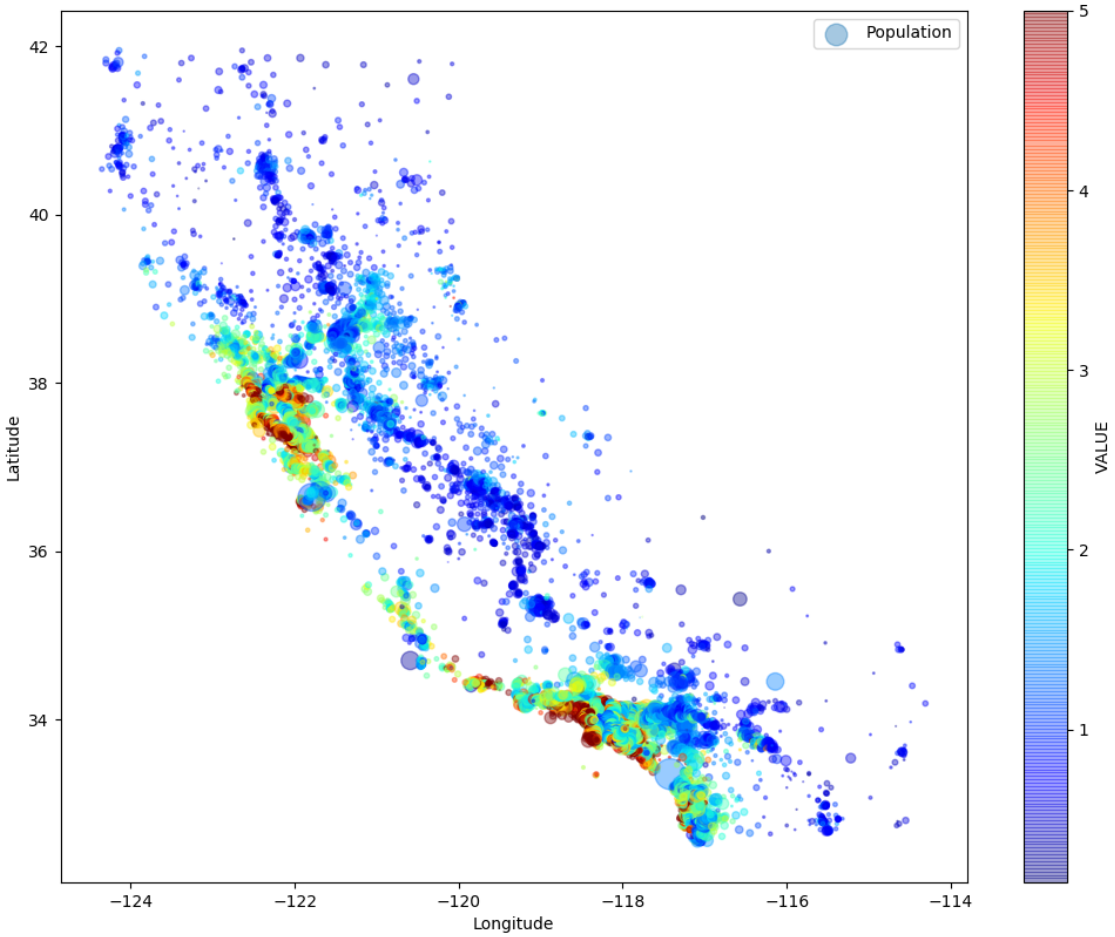
PCA
(Principal Component Analysis)



- La regressione lineare (least square) minimizza le distanze «verticali» dall'iperpiano e non le distanze euclidee (cosa semplicemente ottenibile, come vedremo, attraverso PCA).
- Ciò è conseguenza del fatto che le variabili indipendente e dipendente non hanno ruolo simmetrico (la variabile indipendente si considera esatta, mentre quella dipendente affetta da errore). Nota: scambiando le due variabili il risultato cambia.
- Nel caso di nuvole di punti geometrici le coordinate hanno lo stesso ruolo e PCA fitting è preferibile.

Esempio: stima prezzo case

- California housing dataset contiene 20640 pattern, ciascuno riferito a un quartiere (gruppo di case) in California. I pattern sono 9 dimensionali, per ciascuno di essi:
  - la variabile indipendente è costituita dal vettore 8-dimensionale [MedInc, HouseAge, AveRooms, AveBedrms, Population, AveOccup, Latitude, Longitude]
  - la variabile dipendente (valore target) dal costo medio delle case nel quartiere [Value] (1 ≡100K$).
- Pandas è un tool Python utile per esplorare e manipolare dati tabellari:



Stima con Multiple Linear Regression

- Split dei dati in training e test (80 - 20%):

```
X_train, X_test, y_train, y_test = train_test_split(housing.data,
                                   housing.target,test_size = 0.2)
```

- Con la funzione Scikit-Learn di regressione lineare:

```
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_train_predicted = lin_reg.predict(X_train)
rmse = np.sqrt(mean_squared_error(y_train, y_train_predicted))
print('Train RMSE: ', rmse)
y_test_predicted = lin_reg.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_test_predicted))
print('Test RMSE: ', rmse)
```

- Con implementazione esplicita ($\boldsymbol{\beta}^* = (\mathbf{X}^t\mathbf{X})^{-1}\,\mathbf{X}^t\mathbf{y}$):

```
X_train_b = np.c_[np.ones((X_train.shape[0],1)), X_train]
X_test_b = np.c_[np.ones((X_test.shape[0],1)), X_test]
beta_star = np.linalg.inv(X_train_b.T.dot(X_train_b)).dot(
X_train_b.T).dot(y_train)
y_train_predicted = X_train_b.dot(beta_star)
…
```

Stesso risultato (ultimi decimali a parte)

Train RMSE: 0.7233

Test RMSE: 0.7274 (72K$ di scostamento medio)

Accuratezza di predizione

Fino a questo momento abbiamo considerato sempre RMSE (Root Mean Square Error) come indicatore di accuratezza di predizione.

- RMSE è la radice quadrata del criterio MSE (Mean Square Error) $= \frac{1}{N}\sum_{i=1..N}(pred_i - true_i)^2$ implicito nell'ottimizzazione least square del modello lineare.

Altri modi di valutare l'accuratezza di predizione (*non direttamente utilizzabili come loss per modello lineare*) sono:

- MAE (Mean Absolute Error) $= \frac{1}{N}\sum_{i=1..N}|pred_i - true_i|$:
  - è più robusta di MSE rispetto a outliers (dati non affidabili con ampi scostamenti).
  - mantiene l'unità di misura (es. scostamento in metri)
- MAPE (Mean Absolute Percentage Error) $= \frac{1}{N}\sum_{i=1..N}\left|\frac{pred_i - true_i}{true_i}\right|$

  è una variante del MAE dove la penalizzazione dipende dallo scostamento percentuale rispetto alla grandezza dei pattern.
- R2 (R Squared o coefficiente di determinazione)
  - Indicatore statistico (con valori tra 0 e 1) che indica quanto il modello lineare «fitta» i dati (ovvero spiega la varianza della variabile dipendente).
  - Non valido per modelli non lineari
  - https://en.wikipedia.org/wiki/Coefficient_of_determination

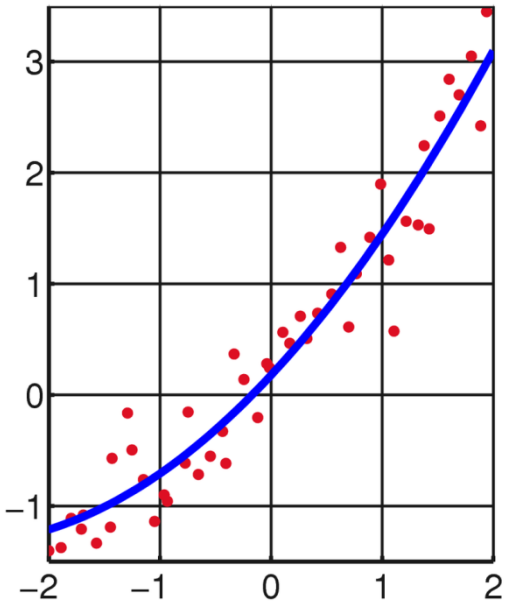
Variabile indipendente non lineare

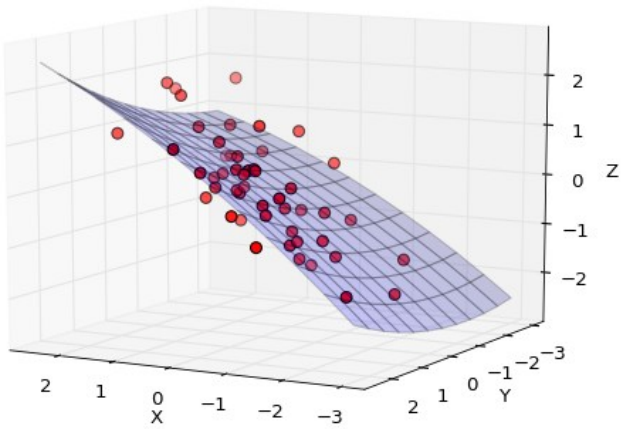
- Con un semplice accorgimento è possibile continuare ad utilizzare la regressione lineare per approssimare i dati con curve e superfici:
  - Nella teoria introdotta infatti l'importante è che la funzione di regressione sia lineare rispetto ai parametri e non rispetto alla variabile indipendente, i cui elementi possono comparire elevati a potenza, combinati tra loro, come argomento di funzioni trascendenti, ecc.

- Esempio: per $d = 1$ e modello quadratico $y = \beta_1 x^2 + \beta_2 x + \beta_3$ scriviamo $\mathbf{X}$ come:

$$\mathbf{X} = \begin{bmatrix} {x_1}^2 & x_1 & 1 \\ {x_2}^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ {x_n}^2 & x_n & 1 \end{bmatrix}$$

ottenendo →





← Esempio $d = 2$

Regressione non lineare
(dipendenza nota)

- Nel caso di dipendenza non lineare del modello dai parametri non esiste soluzione least square in «forma-chiusa».

  Esempio dalla biologia (enzyme-mediated reaction)

$$y = \frac{\beta_1 x}{\beta_2 + x}$$

- Se la dipendenza è nota (come nell'esempio sopra), per la risoluzione si possono applicare tecniche iterative di ottimizzazione numerica:

  - È possibile utilizzare il classico algoritmo di Gradient Descent (metodo del primo ordine che esegue passi in direzione contraria al gradiente).

  - L'algoritmo più noto e utilizzato è quello di Gauss-Newton (metodo del secondo ordine che richiede stima Hessiana).

    https://en.wikipedia.org/wiki/Gauss–Newton_algorithm

Regressori non lineari
(dipendenza sconosciuta)

- Quando la dipendenza tra la variabile indipendente e dipendente non è nota (potrebbe essere lineare o non lineare) il problema può essere approcciato provando e validando sia il modello lineare (baseline), sia modelli non lineari via via più complessi.

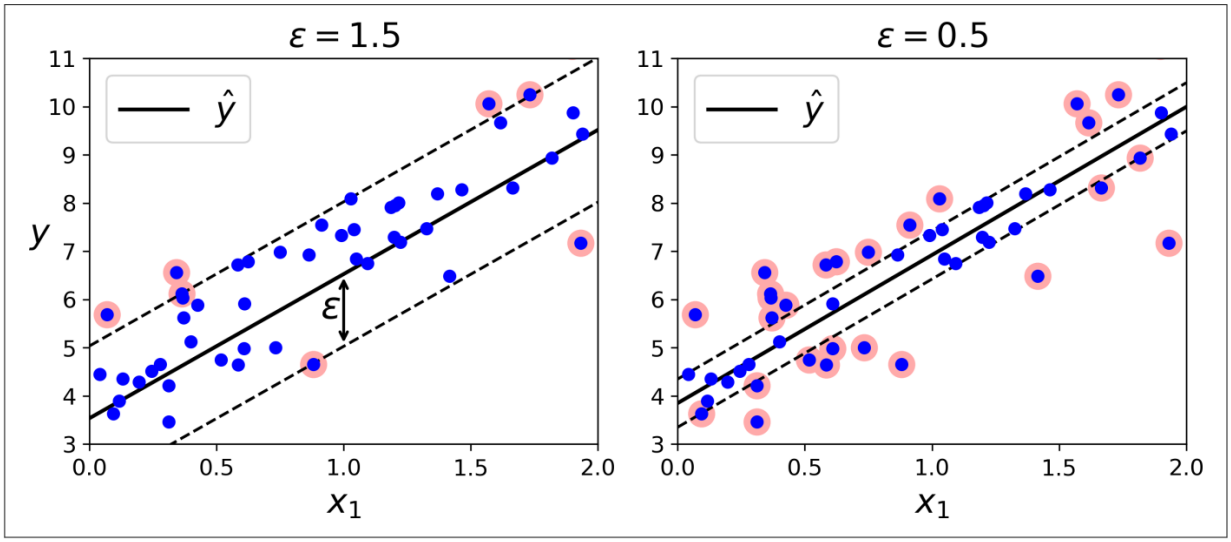
- Per alcune tecniche di classificazione studiate esiste una variante efficace per problemi di regressione. Casi notevoli (disponibili in scikit-learn) sono:
  - Support Vector Regressor (SVR)
  - Random Forest Regressor
  - Gradient Boosting Regressor Trees

- Come vedremo anche le Reti Neurali addestrate con classico error backpropagation consentono di risolvere efficacemente problemi di regressione.

- I vantaggi di un modello non lineare sono:
  - Maggiore flessibilità (più gradi di libertà del modello).
  - Controllo della complessità della soluzione (gradi di libertà del modello, regolarizzazione, ecc.)
  - Possibilità di utilizzare loss alternative rispetto a MSE (che è implicita nella risoluzione least square) quali MAE, MAPE, ecc.

Support Vector Regressor (SVR)

- Rispetto alla classificazione, dove si volevano spingere i pattern delle due classi oltre al margine, qui si inverte la logica: vogliamo che nella fascia del margine ricada il maggior numero di pattern del training set.



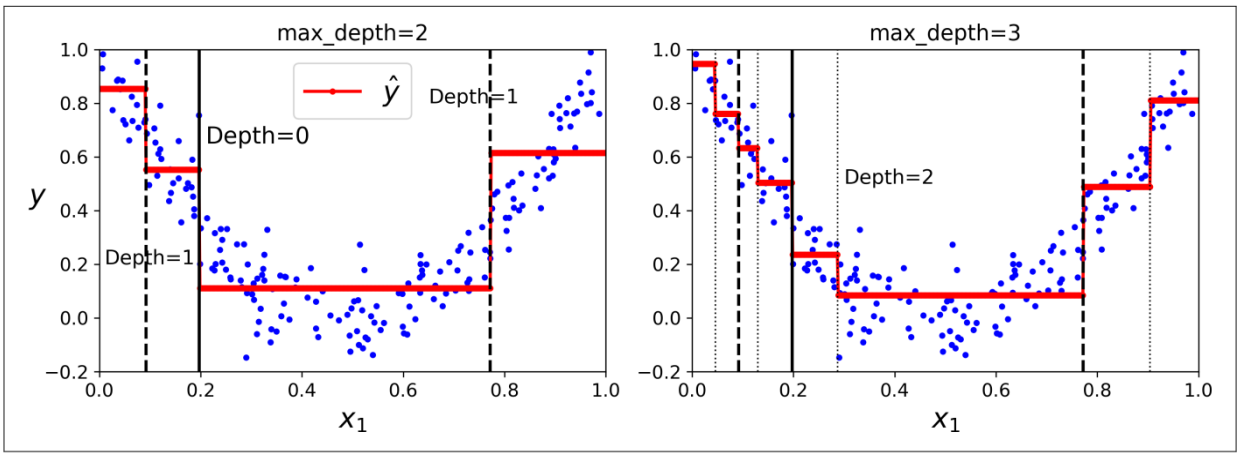
[A. Géron]

- L'approssimazione può essere lineare o non–lineare (es. RBF).

- L'iperparametro C e i regolarizzatori del kernel usato (nel caso non lineare) permettono il controllo dell'overfitting, e di mantenere il regressore sufficientemente smoothed.

Random Forest Regressor

- Stessa logica di creazione della foresta con doppio bagging su feature e pattern, ma con gli alberi CART usati come regressori.
- In questo caso ogni nodo (foglie comprese) predice un valore pari alla media delle y dei pattern ad esso appartenenti, e la regola di suddivisione indotta dalla coppia (feature, soglia) minimizza l'MSE delle y nei due nodi figli.
- Nella figura un esempio con X = x_1 (monodimensionale):

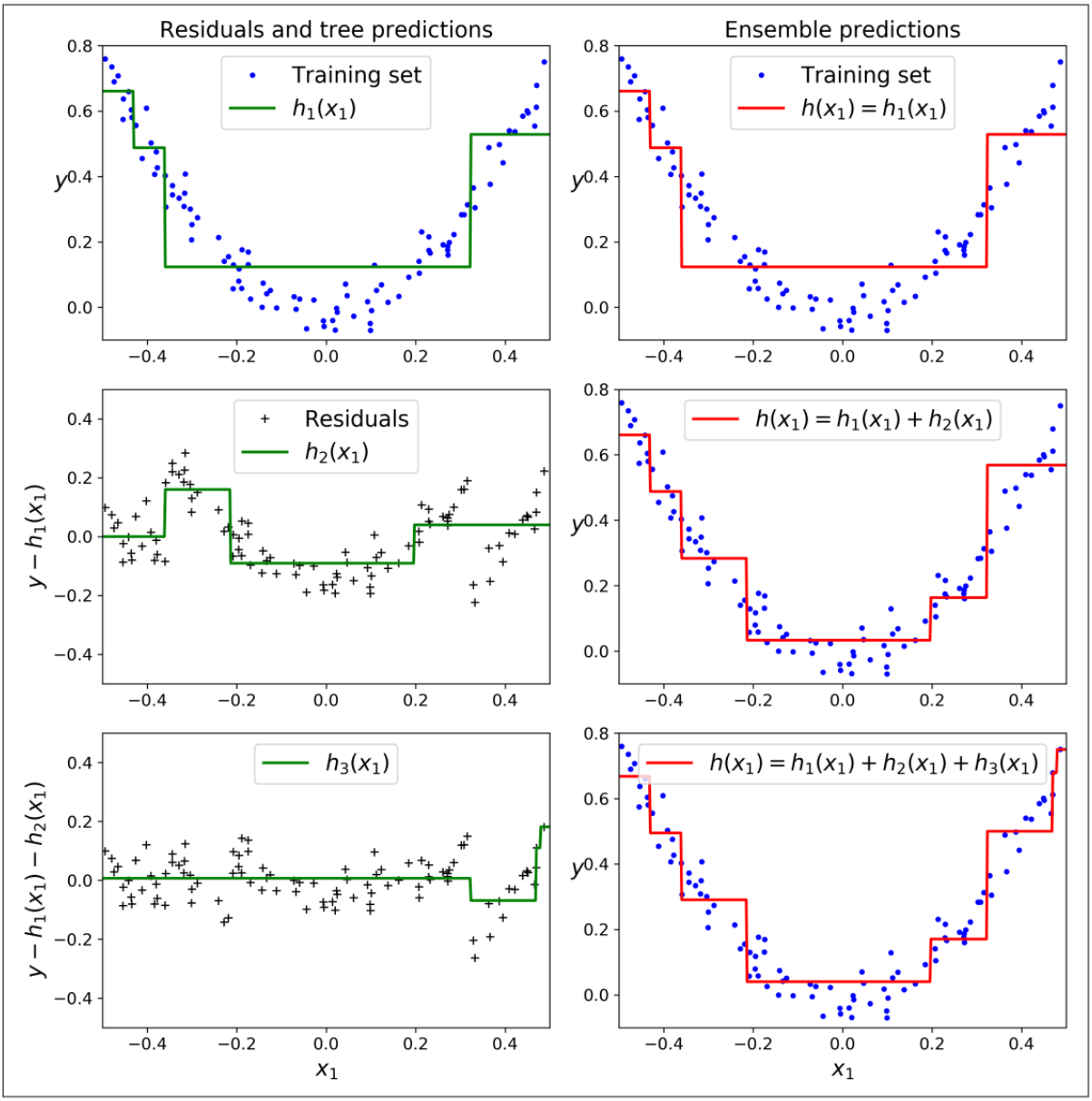


[A. Géron]

- Questo regressore produce dunque un'approssimazione costante a tratti (che può essere discontinua e con «salti» repentini).
- L'iperparametro (massima profondità degli alberi) è il principale regolarizzante per il controllo dell'overfitting.

Gradient Boosting Regressor Trees

- È una tecnica di boosting che aggiunge iterativamente nuovi regression trees addestrati sui residuali (ovvero differenze tra le y e le predizioni prodotte dai predecessori). Il multi-classificatore si ottiene sommando i contributi dei singoli regressori.
- Esempio di regressione con loss MSE:



[A. Géron]

Gradient Boosting Regressor Trees (2)

- La formulazione generale (che rende possibile utilizzare anche altre loss function oltre a MSE) è matematicamente complessa (vedi link) e mostra come i residuali corrispondano ai gradienti dei dati (da qui il nome gradient boosting).
- Per l'applicazione di gradient boosting alla classificazione i modelli (intermedi) sono comunque dei regressori e solo alla fine si applica una funzione che mappa il valore continuo ottenuto in una classe.

- L'implementazione ottimizzata XGBoost ha interfaccia fit/predict standard di scikit-learn è può essere facilmente aggiunta al toolkit dei modelli da considerare per la risoluzione di un problema. Vedi link.
  - XGBoost è molto più veloce in training rispetto alla versione classica (fino a 10 volte) e include varie ottimizzazioni tra cui un termine regolarizzante sulla loss per meglio controllare overfitting.
  - In molte challenge Kaggle (su dati tabulari) XGboost ha ottenuto le prestazioni migliori ed è pertanto considerato uno dei modelli più performati.

Prezzi case: modelli non lineari

- Random Forest Regressor (da Scikit-Learn):

```
forest_reg = RandomForestRegressor(n_estimators = 30)
forest_reg.fit(X_train, y_train)
y_train_predicted = forest_reg.predict(X_train)
…
```

Train RMSE: 0.1970

Test RMSE: 0.5080 (il modello lineare era troppo semplice …)

- Rete Neurale 4 livelli + Gradient Descent (da TensorFlow):

```
…
hidden1 = tf.layers.dense(X, 100, name = "hidden1", activation =
                          tf.nn.relu)
hidden2 = tf.layers.dense(hidden1, 100, name = "hidden2",
                          activation = tf.nn.relu)
y_pred = tf.layers.dense(hidden2, 1, name = "logits")
…
```

Train RMSE: 0.4647

Test RMSE: 0.5198 (sembra non si possa fare molto meglio …)

Quantile Regression

- Il modello lineare risolto ai minimi quadrati è uno stimatore della media condizionata. Significa che la retta di regressione è determinata per approssimare la media dei dati, e quindi l'entità degli errori di previsione positivi e negativi è la stessa.

- Nel caso in cui si voglia dare un peso diverso ai positivi e negativi (es. le conseguenze di una sottostima potrebbero essere più gravi di quelle di una sovrastima) si introduce la regressione ai quantili (o percentili).
  - Con percentile 50% si passa da media a mediana (la metà dei punti nel training set sono sopra la retta di regressione individuata e la metà sono sotto).
  - Con percentile p, la retta è individuata in modo da avere il p% di punti sotto e (1-p)% sopra.

- Nella formulazione si utilizza una loss MAE con penalità diversa per predizioni in eccesso e difetto.

- L'implementazione con modello lineare richiede ottimizzazione numerica. Esiste però un algoritmo iterativo che richiama la lineare «pesata»:
  - Quantile Linear Regression per scikit-learn (estensione):

    http://www.xavierdupre.fr/app/mlinsights/helpsphinx/mlinsights/mlmodel/quantile_regression.html

- Quantile Regression con modelli non lineari:

  https://towardsdatascience.com/quantile-regression-from-linear-models-to-trees-to-deep-learning-af3738b527c3

Riepilogando

- Per problemi di regressione di piccole-medie dimensioni:
  - Il modello lineare (rispetto ai parametri) risolto tramite least square (con fattorizzazione SVD) è l'approccio più semplice che costituisce una sorta di baseline.
  - Un modello non-lineare rispetto alla variabile indipendente (ma lineare rispetto ai parametri) può essere un'ottima seconda scelta perché coniuga semplicità e flessibilità.
  - Quando sono necessari modelli più potenti, SVR, Random Forest e Gradient Boosting sono valide alternative. In particolare, i metodi basati su alberi sono la prima scelta per dati tabellari eterogenei, mentre SVR meglio modella fenomeni fisici continui.
  - Se la dipendenza tra variabile dipendente e indipendente è nota (ma non lineare rispetto ai parametri) l'ottimizzazione numerica con Gauss-Newton è la prima scelta.
- Per problemi di regressione di grandi dimensioni (specie su dati non strutturati) l'approccio iterativo Gradient Descent utilizzato in combinazione con Reti Neurali è spesso preferibile per motivi di trattabilità numerica e complessità computazionale.

  Nell'ambito del deep learning, reti neurali addestrate con gradient descent sono state utilizzate con successo per risolvere problemi di regressione di enormi dimensioni come la stima della posizione di un oggetto in un'immagine a partire da milioni di esempi (vedi Faster R-CNN)
  - la variabile indipendente è un'immagine (pixel)
  - La variabile dipendente la bounding box dell'oggetto (trattasi in questo caso di un vettore $\rightarrow$ multivariate multiple regression).