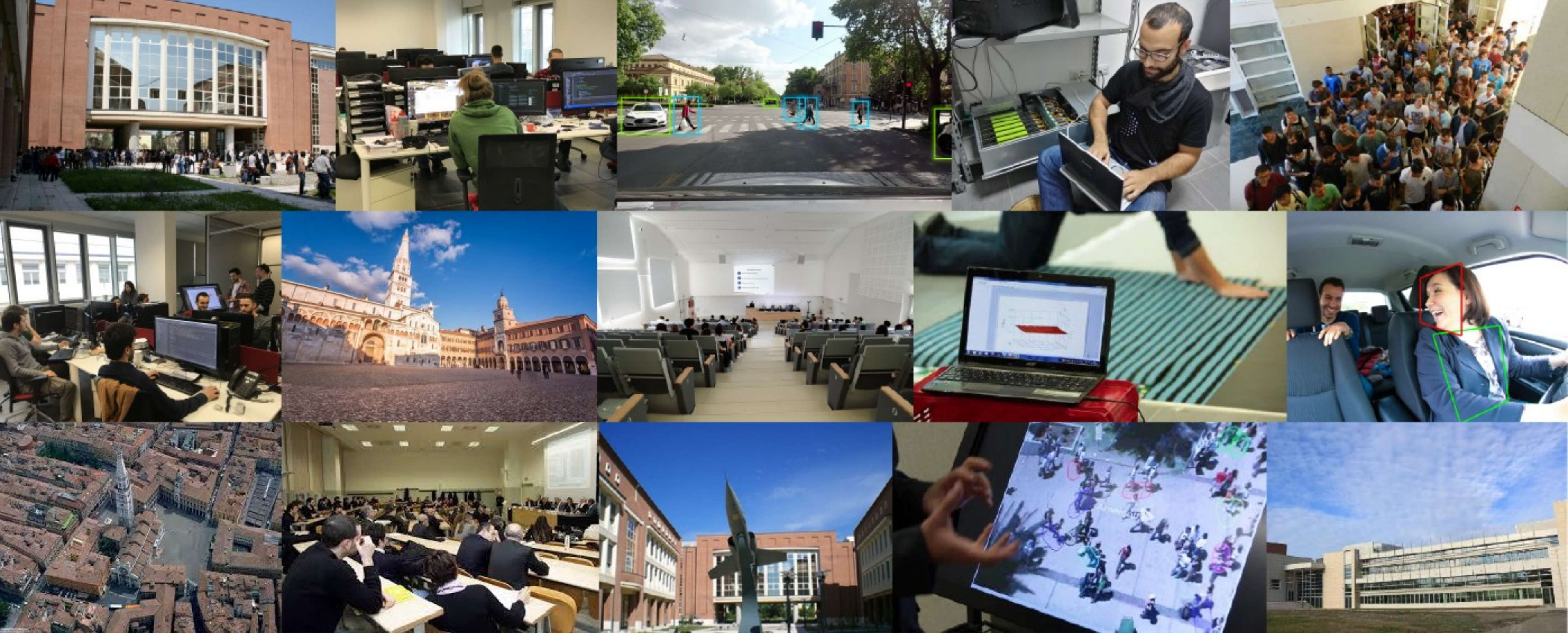


MACHINE LEARNING
Linear Regression -

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Linear regression

In questa lezione vedremo più in dettaglio la linear regression e i metodi di ottimizzazione ad essa corrispondenti

Inoltre, inizieremo a parlare di testing e di performance

Notazione

x_j, j = 1, …, d-1 → variabili indipendenti/features

$\mathbf{x} = [x_1, \ldots, x_{d-1}]$ → feature vector (con *d*-1 feature)

$\mathbf{x} = [1, x_1, \ldots, x_{d-1}]$ → feature vector (con x_0=1)

y → variabile dipendente/target

$(\mathbf{x}^{(i)}, y^{(i)})$, i = 1, …, n → training sample generico

$y = h(\mathbf{x})$ → funzione di predizione/funzione ipotesi

$\theta = [\theta_0, \theta_1, \ldots, \theta_{k-1}]$ weight/parameter vector

$J(\theta)$ → Loss function

Regressione lineare

Come abbiamo visto nella lezione precedente, nella **Linear Regression**, assumiamo che la relazione tra l'input ***x*** e l'output *y* della funzione ipotesi sia *lineare*:

$$h([x_1, x_2]; [\theta_0, \theta_1, \theta_2]) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

o, scritto in forma equivalente:

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Usando la notazione $x_0 = 1$ («intercept ter

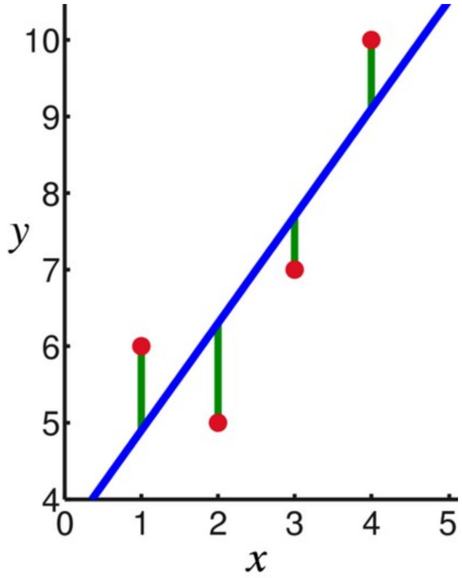
$$\boxed{\begin{aligned} \mathbf{x} &= [x_0 = 1, x_1, ..., x_{d-1}] \\ \theta &= [\theta_0, \theta_1, ..., \theta_{d-1}] \end{aligned}}$$

$$h(\mathbf{x}) = \sum_{i=0}^{d\text{-}1} \theta_i x_i = \theta^T \mathbf{x}$$

Scegliamo la loss function

Per trovare i parametri appropriati, definiamo la seguente funzione di costo (loss function) detta dei «**minimi quadrati**» (Least Squares):

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



Importante: utilizzando la precedente definizione di *h()*, *J()* è continua e differenziabile

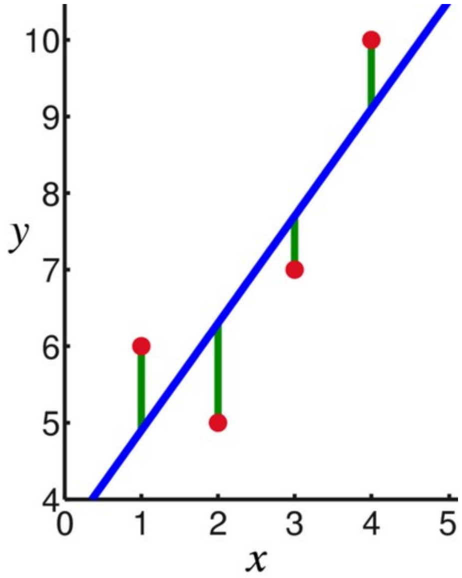
Esempio

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

$h_{[a,b]}(x) = ax + b$

$T = \{(1,6), (2,5), (3,7), (4,10)\}$

$$\begin{aligned} J([a,b]) = \tfrac{1}{2} [\ &((a + b) - 6)^2 + \\ &((2\,a + b) - 5)^2 + \\ &((3\,a + b) - 7)^2 + \\ &((4\,a + b) - 10)^2\ ] \end{aligned}$$



Least Squares loss function

La stessa loss function ai minimi quadrati definita prima può essere usata per trovare i parametri di *h()* utilizzando strategie diverse:

1. Gradient descent (soluzione iterativa)
2. Closed-form solution (soluzione diretta)

Soluzione 1: Gradient Descent

Nel caso specifico della regressione lineare, il Gradient Descent prende anche il nome di Least Squares Algorithm

Calcolo del gradiente

Devo ora calcolare il gradiente di *J()*
Iniziamo, per semplicità, supponendo di avere un unico sample: $T = \{(\mathbf{x}, y)\}$
Devo calcolare le derivate parziali di *J()* rispetto ad ogni parametro:

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

Esempio

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

$h_{[a,b]}(x) = ax + b$

$T = \{(1,6)\}$

$J([a,b]) = ½ [((a + b) – 6)^2] = ½ [a^2 + b^2 + 2 ab – 12 a – 12 b + 36] =$
$= ½ a^2 + ½ b^2 + ab – 6 a – 6 b + 18$

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial a} = a + b - 6 \\ \frac{\partial J}{\partial b} = b + a - 6 \end{bmatrix}$$

Gradiente con un solo sample: derivate parziali

Veniamo al caso generale in cui, come anticipato, assumeremo inizialmente di avere un unico sample: *T = {(**x**, y)}*
Calcolo le derivate parziali di *J()* rispetto al (generico) parametro j-esimo:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(\mathbf{x}) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(\mathbf{x}) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(\mathbf{x}) - y \right) \Leftarrow D[F(x)G(x)] = F'(x)G(x) + G'(x)F(x) \Rightarrow D[F(x)^2] = 2F'(x)F(x) \\
&= \left(h_\theta(\mathbf{x}) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d\text{-}1} \theta_i x_i - y \right) \\
&= \left(h_\theta(\mathbf{x}) - y \right) x_j
\end{aligned}
$$

Gradiente calcolato con tutti i sample: derivate parziali

Nel caso generale, quando *T* è composto da *n* samples, ho (per la proprietà lineare della derivata):

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

$$\frac{\partial}{\partial\theta_j}J(\theta) = \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}]$$

Gradiente con tutti i sample

Quindi, nel caso di *n* samples, il gradiente è:

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta) = \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)}] \\ \dots \\ \frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}] \\ \dots \\ \frac{\partial}{\partial \theta_{d-1}} J(\theta) = \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)}] \end{bmatrix}$$

Batch vs Stochastic Gradient Descent

$L(\boldsymbol{w}_t)$ può essere calcolato usando strategie diverse per "accumulare l'errore":

- **Batch Gradient Descent**: uso tutto il dataset (n esempi) ad ogni passo

$$\nabla J(\theta) = \begin{bmatrix} \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)}] \\ \dots \\ \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}] \\ \dots \\ \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)}] \end{bmatrix}$$

Batch vs Stochastic Gradient Descent

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- **Stochastic Gradient Descent**: uso un sott'insieme piccolo *B* (|*B*| = *m, m>= 1*) del dataset, scelto in maniera random *ad ogni iterazione*

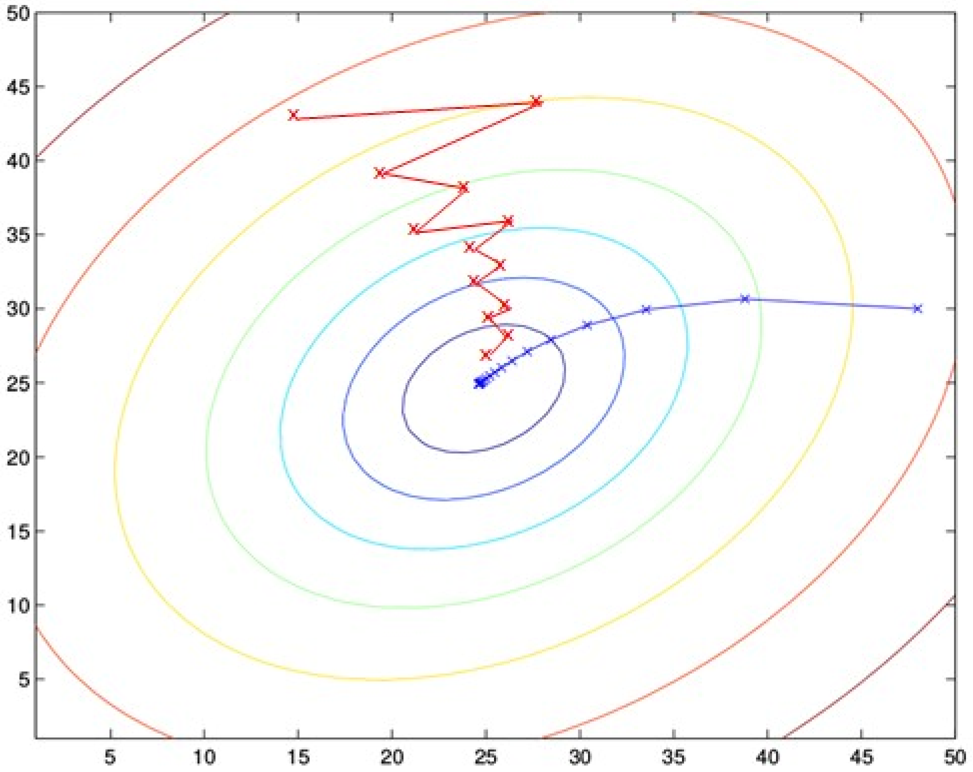
  B (detto "mini-batch") può avere anche cardinalità 1

$$\nabla J(\theta) = \begin{bmatrix} \sum_{i=1}^{m}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)}] \\ \dots \\ \sum_{i=1}^{m}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}] \\ \dots \\ \sum_{i=1}^{m}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)}] \end{bmatrix}$$

Usando un batch *B* invece che tutto *T* ho un risparmio computazionale

Tuttavia, lo Stochastic Gradient Descent è più instabile

Gradient Descent: Esempio



Batch Gradient Descent

Stochastic Gradient Descent

Batch Gradient Descent

Nel caso del Least Squares, ovvero quando *J(Θ)* è definita come somma di quadrati, il Gradient Descent diventa:

1. Inizializzo *Θ* in maniera random, *t:= 0*

2. Iterazione t-esima:

   a) Per og

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} [(y^{(i)} - h_\theta(\boldsymbol{x}^{(i)}))\boldsymbol{x}_j^{(i)}]$$

$$\nabla J(\theta) = \begin{bmatrix} \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)}] \\ \ldots \\ \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}] \\ \ldots \\ \sum_{i=1}^{n} [(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)}] \end{bmatrix}$$

   b) If Condizione-di-Terminazione = true

      then return current *Θ*

      Dove la "Condizione-di-Terminazione" può essere, e.g., uno dei seguenti casi:

- *J(Θ)* è sufficientemente piccolo
- Il modulo del gradiente $|\nabla J|$ è abbastanza vicino allo 0
- Il numero di iterazioni *t* è abbastanza grande

Stochastic Gradient Descent

Il Least Squares con un mini-batch *di un singolo sample* ($B = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, $m = 1$) può essere scritto come:

1. Inizializzo Θ in maniera random

   a) Per ogni sample $(x^{(i)}, y^{(i)})$ in T:

      - Per ogni $j = 0, \ldots, d-1$:

        $$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(\boldsymbol{x}^{(i)}))\boldsymbol{x}_j^{(i)}$$

      - If Condizione-di-Terminazione = true,

        then return current Θ

   b) Torna ad a)

$$\nabla J(\theta) = \begin{bmatrix} (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)} \\ \ldots \\ (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)} \\ \ldots \\ (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)} \end{bmatrix}$$

Soluzione 2: closed form solution

Nel caso specifico della regressione lineare, si può dimostrare che la funzione di costo precedentemente definita (minimi quadrati, Least Squares) è convessa.

Per cui possiamo trovare il minimo globale direttamente (in maniera diretta) ponendo a zero il gradiente

Vedremo che ciò porta alla formulazione di un sistema di equazioni lineari che può essere facilmente risolto con tecniche standard

Soluzione 2: closed form solution

Obiettivo: trovare il minimo formulando il problema come segue:

$$\nabla_\theta J(\theta) = 0$$

Soluzione 2: closed form solution

$$\nabla J(\theta) = \begin{bmatrix} \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)}] \\ \dots \\ \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}] \\ \dots \\ \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)}] \end{bmatrix} = \begin{bmatrix} 0 \\ \dots \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

E’ un sistema lineare di *d* equazioni e *d* incognite

Le *d* incognite sono: $\Theta_0, \dots, \Theta_{d-1}$

Forma matriciale del sistema

Per capire meglio che si tratta di equazioni lineari e per risolvere il sistema utilizzando i metodi noti di algebra lineare, lo riformulo utilizzando la forma matriciale

Forma matriciale del training set

Raggruppiamo i feature vector nella **design matrix**:

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(n)})^T & — \end{bmatrix}$$

Raggruppiamo le label nel **target vector**:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Forma matriciale del gradiente

Con un pò di algebra, posso riscrivere questo:

$$\nabla J(\theta) = \begin{bmatrix} \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_0^{(i)}] \\ \cdots \\ \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_j^{(i)}] \\ \cdots \\ \sum_{i=1}^{n}[(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\boldsymbol{x}_{d-1}^{(i)}] \end{bmatrix}$$

Così:

$$\nabla_\theta J(\theta) \quad = \quad X^T X\theta - X^T \mathbf{Y}$$

Sistema di equazioni lineari (forma matriciale)

$$\nabla_\theta J(\theta) = X^T X\theta - X^T\mathbf{y} \longrightarrow X^T X\theta - X^T\mathbf{y} = 0$$

Pongo a 0 il gradiente

$$\nabla_\theta J(\theta) = 0$$

$$\longrightarrow \underbrace{X^T X}_{\text{"A"}}\underbrace{\theta}_{\text{"x"}} = \underbrace{X^T\mathbf{y}}_{\text{"b"}} \xrightarrow{\text{Soluzione}}$$

vettore
dei termini noti

vettore delle incognite

matrice dei coefficienti

Normal Equation:

$$\theta = (X^T X)^{-1} X^T\mathbf{y}$$

Closed form solution

Non è detto che usare la Normal Equation sia più vantaggioso che usare il Gradient Descent

Questo perché inverire $X^T X$ può essere computazionalmente oneroso se X è grande

In generale, mentre la closed form solution può essere usata con la regressione lineare e la Least Squares loss function, come abbiamo detto più volte, per altri task di ML/loss function, tipicamente NON riesco ad usare una closed form solution

Viceversa, il Gradient Descent è una soluzione abbastanza generale


Now let's discuss
your performance
evaluation
ICANHASCHEEZBURGER.COM

Notazione

n → numero di esempi di testing

$(\mathbf{x}^{(i)}, y^{(i)})$ $i = 1, \ldots, n$ → testing sample generico

$\hat{y}^{(i)} = h_\theta(\boldsymbol{x}^{(i)})$ $\ldots, n$ → prediction (rispetto al testing sample i-esimo)

Residual ____ ____

- **Residual Sum of Squares**:

$$RSS = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

- **Residual Standard Error** – La radice di RSS normalizzato rispetto al numero degli esempi e delle feature:

$$RSE = \sqrt{\frac{1}{n-d-1} RSS}$$

[_____] Mean _____ Error

- **Mean Squared Error**:

$$MSE = \frac{1}{n} RSS$$

- **Root Mean Square Error**:

$$RMSE = \sqrt{MSE}$$

- **Mean Absolute Error**:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} \left| y^{(i)} - \hat{y}^{(i)} \right|$$

Riferimenti

- **A tu per tu col Machine Learning. L'incredibile viaggio di un developer nel favoloso mondo della Data Science**, Alessandro Cucci, The Dot Company, 2017 [cap.4]
- **Pattern Classification, second edition,** R. O. Duda, P. E. Hart, D. G. Stork, Wiley-Interscience, 2000
- **The elements of statistical learning**, Trevor Hastie, Robert Tibshirani, Jerome H. Friedman, New York: Springer series in statistics, 2001 [cap.3]
- **An introduction to statistical learning**, Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, New York: Springer, 2013 [cap. 3]
- https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf