Machine Learning for Networks: Regression

Andrea Araldo

May 16, 2023

ML methods

	ML task	Linear Regression	Logistic Regression	Tree-based learning	Neural Networks	k-Neirest Neighbors
Supervised	Regression	x			x	
	Classification		x	x	x	
Unsupervised	Clustering				x	x
	Dimensionality reduction				x	
	Anomaly detection			x	x	x
	Recommender Systems				x	

Outline

- Supervised Learning Models
- Linear Regression
- Train / Test / Cross-Validation

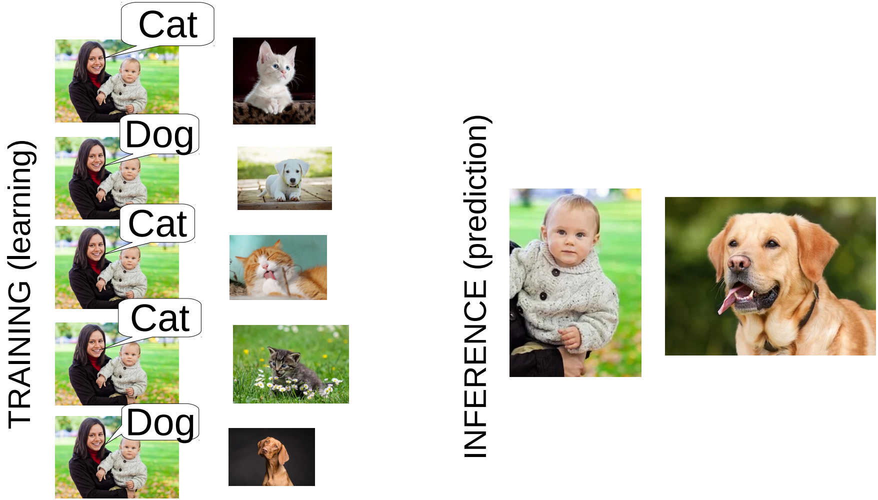
Section 1

Introduction to Supervised Learning

How humans learn

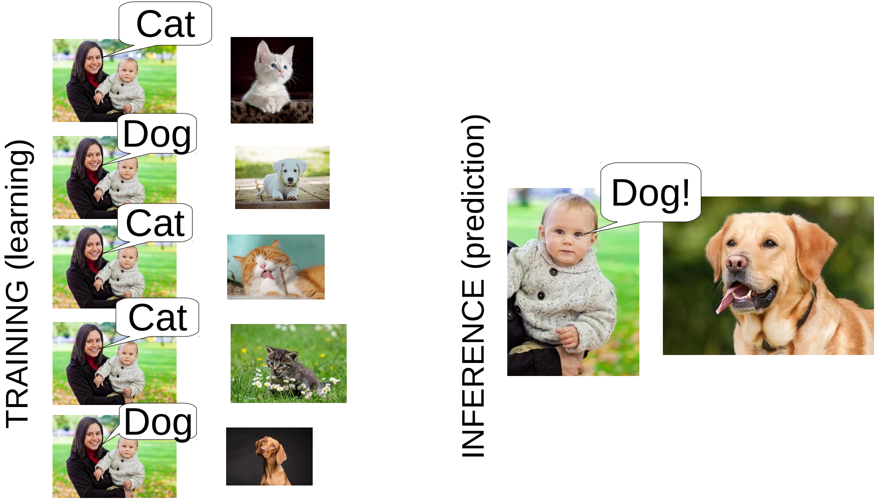
Some think instead that we learn in a different [Tho20]

How humans learn



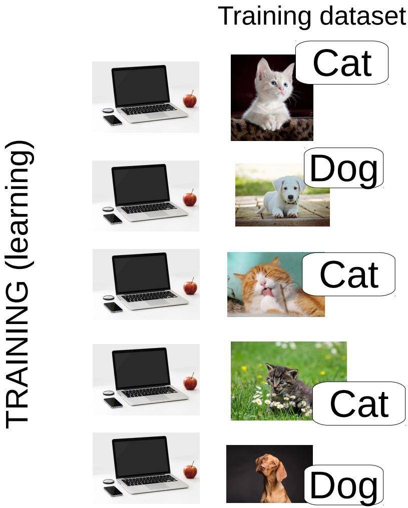
Some think instead that we learn in a different [Tho20]

How humans learn

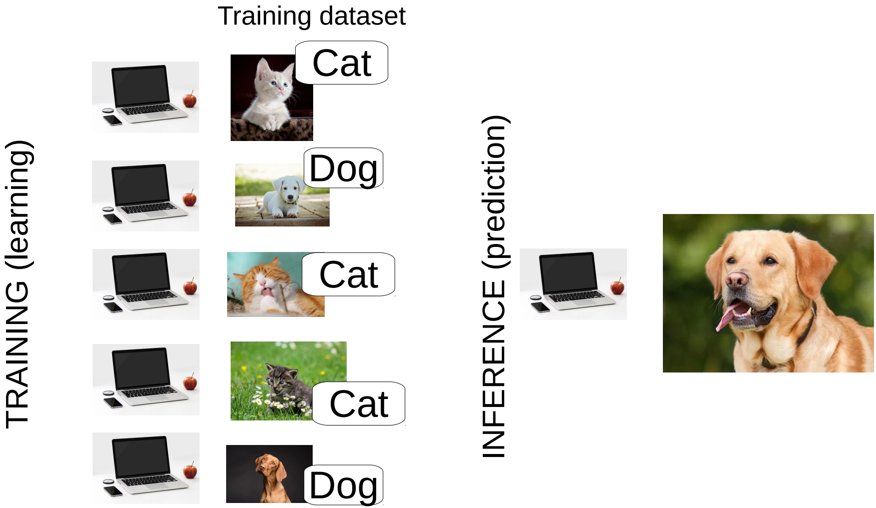


Some think instead that we learn in a different [Tho20]

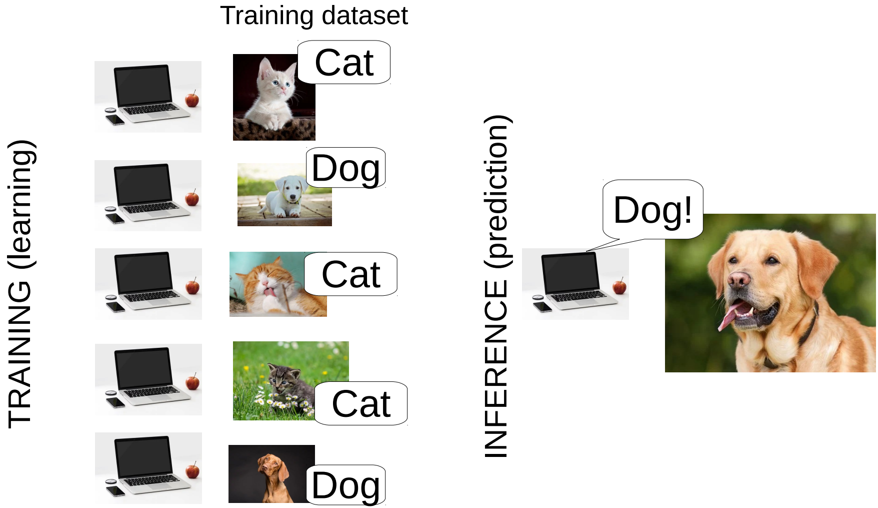
How machines learn



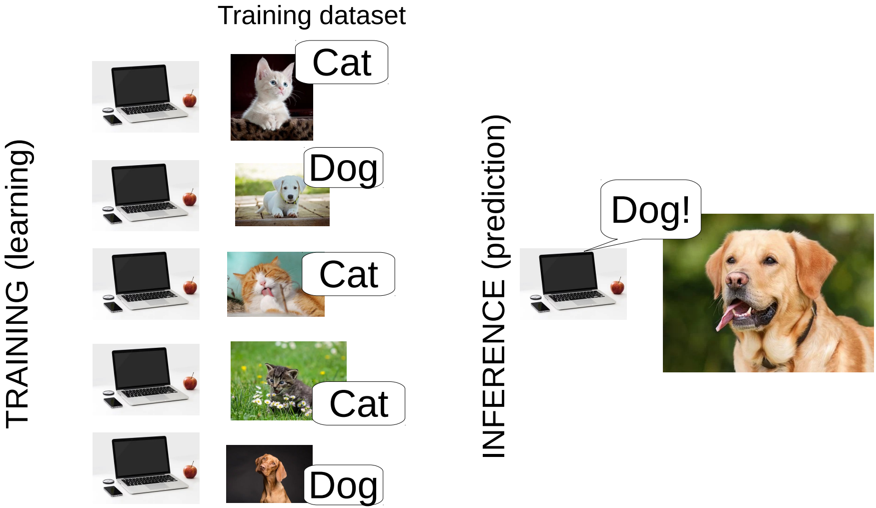
How machines learn



How machines learn



How machines learn



Deep Neural Network better than humans to classify several image datasets.

Definition of learning

Cambridge English Dictionary:

the process of getting an understanding of something by studying it or by ***experience***

Definition of learning

Cambridge English Dictionary:

the process of getting an understanding of something by studying it or by ***experience***

What can we learn in a network, by just observing traffic (encrypted)?

- Anomaly/attack
- Application protocols hidden in encrypted traffic
- Quality of Experience (QoE)
- Malfunctioning of a device
- ...

Problem example

- **Problem** [KEB19]:
  Infer video chunk resolution
  only observing network information, e.g., throughput, packet size, inter-arrival times, etc.
  Traffic is encrypted.
- **Motivation**:
  Network operators can have insights on QoE without *Deep Packet Inspection*

Formalization

Features (or *Columns*):

- Throughput mean (bps)
  Throughput st.dev. (bps)
  Packet size mean (B)
  Packet size st.dev.
  Inter-arrival mean (ms), etc.

Label (or *target*):

- Resolution of downloaded video chunks (360p, 720p)
- (it indicates the number of vertical pixels)

Formalization

Features (or *Columns*):

- Throughput mean (bps)
  Throughput st.dev. (bps)
  Packet size mean (B)
  Packet size st.dev.
  Inter-arrival mean (ms), etc.

Label (or *target*):

- Resolution of downloaded video chunks (360p, 720p)
- (it indicates the number of vertical pixels)

Model:

- A function $h(\text{features}) = \mathit{label}$
- If label $\in \mathbb{R}$: **regression** model
- If label discrete: **classification** model

Example

		Features (or Independent variables)					True Label		
	Sample	TP mean	TP st.dev.	Pkt size mean	Pkt std. dev.	Int-Arriv mean	$y^{(i)}$ (Resolution)		
Our dataset	$\mathbf{x}^{(1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	**475**		
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		
	$\mathbf{x}^{(M)^T} =$	2 Mbps	1.1 Kbps	1.3 KB	400 B	20 ms	**720**		
New samples	$\mathbf{x}^{(M+1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	?		
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		

Example

		Features (or Independent variables)					True Label	Pred Label	
	Sample	TP mean	TP st.dev.	Pkt size mean	Pkt std. dev.	Int-Arriv mean	$y^{(i)}$ (Resolution)	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$	
Our dataset	$\mathbf{x}^{(1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	**475**	482	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	$\mathbf{x}^{(M)^T} =$	2 Mbps	1.1 Kbps	1.3 KB	400 B	20 ms	**720**	693	
New samples	$\mathbf{x}^{(M+1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	?	1078	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

- Model: function h used to predict

Example

	Sample	Features (or Independent variables)					True Label	Pred Label	Pred Label
		TP mean	TP st.dev.	Pkt size mean	Pkt std. dev.	Int-Arriv mean	$y^{(i)}$ (Resolution)	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$
Our dataset	$\mathbf{x}^{(1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	**475**	482	1083
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{x}^{(M)^T} =$	2 Mbps	1.1 Kbps	1.3 KB	400 B	20 ms	**720**	693	323
New samples	$\mathbf{x}^{(M+1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	?	1078	376
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Model: function h used to predict

Which of the two models would you trust more to predict?

Example

		Features (or Independent variables)					True Label	Pred Label	Pred Label
	Sample	TP mean	TP st.dev.	Pkt size mean	Pkt std. dev.	Int-Arriv mean	$y^{(i)}$ (Resolution)	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$
Our dataset	${\mathbf{x}^{(1)}}^T =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	**475**	482	1083
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	${\mathbf{x}^{(M)}}^T =$	2 Mbps	1.1 Kbps	1.3 KB	400 B	20 ms	**720**	693	323
New samples	${\mathbf{x}^{(M+1)}}^T =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	?	1078	376
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Model: function h used to predict

  Which of the two models would you trust more to predict?

- h is "good" if $h(\mathbf{x}^{(i)}) \simeq y^{(i)}$ for the new samples.

Example

		Features (or Independent variables)					True Label	Pred Label	Pred Label
	Sample	TP mean	TP st.dev.	Pkt size mean	Pkt std. dev.	Int-Arriv mean	$y^{(i)}$ (Resolution)	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$	$\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$
Our dataset	$\mathbf{x}^{(1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	**475**	482	1083
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{x}^{(M)^T} =$	2 Mbps	1.1 Kbps	1.3 KB	400 B	20 ms	**720**	693	323
New samples	$\mathbf{x}^{(M+1)^T} =$	2 Mbps	1 Kbps	1 KB	300 B	30 ms	?	1078	376
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

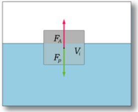
- Model: function h used to predict

Which of the two models would you trust more to predict?

- h is "good" if $h(\mathbf{x}^{(i)}) \simeq y^{(i)}$ for the new samples.
- But we do not know $y^{(i)}$ for the new samples.
- We can just evaluate how $h(\cdot)$ performs in our dataset.

Famous examples of models “by hand”

Archimedes (III cent. BC)



weight of displaced fluid = weight of object in vacuum − weight of object in fluid

Newton (XVII cent.)

$$\mathbf{F} = m\mathbf{a}$$

Maxwell (XVII cent.)

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \left(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right)$$

They created a model $h(\cdot)$ relating a target to some features, **based on observation**.

Generalization

For **any** object with m and $\mathbf{a}$, we know the force $\mathbf{F}$ it produces.

Pictures: Wikipedia

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (\quad x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

- Linear Model

$$\begin{aligned} h_\theta(\mathbf{x}^{(i)}) = \theta^T \cdot \mathbf{x}^{(i)} = \sum_{j=0}^{N} \theta_j \cdot x_j^{(i)} \\ = \theta_0 + \theta_{\text{TP mean}} \cdot x_{\text{TP mean}}^{(i)} \\ + \theta_{\text{TP st.dev.}} \cdot x_{\text{TP st.dev.}}^{(i)} + \ldots \end{aligned}$$

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

- Linear Model

$$\begin{aligned} h_\theta(\mathbf{x}^{(i)}) &= \theta^T \cdot \mathbf{x}^{(i)} = \sum_{j=0}^{N} \theta_j \cdot x_j^{(i)} \\ &= \theta_0 + \theta_{\text{TP mean}} \cdot x_{\text{TP mean}}^{(i)} \\ &+ \theta_{\text{TP st.dev.}} \cdot x_{\text{TP st.dev.}}^{(i)} + \ldots \end{aligned}$$

- Quadratic Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_N x_N^{(i)}$$

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

- Linear Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta^T \cdot \mathbf{x}^{(i)} = \sum_{j=0}^{N} \theta_j \cdot x_j^{(i)}$$
$$= \theta_0 + \theta_{\text{TP mean}} \cdot x_{\text{TP mean}}^{(i)}$$
$$+ \theta_{\text{TP st.dev.}} \cdot x_{\text{TP st.dev.}}^{(i)} + \ldots$$

- Quadratic Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_N x_N^{(i)}$$
$$+ \theta_{N+1} \cdot x_1^{(i)^2} + \cdots + \theta_N \cdot x_{N+N}^{(i)^{\;\;2}}$$

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

- Linear Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta^T \cdot \mathbf{x}^{(i)} = \sum_{j=0}^{N} \theta_j \cdot x_j^{(i)}$$
$$= \theta_0 + \theta_{\text{TP mean}} \cdot x_{\text{TP mean}}^{(i)}$$
$$+ \theta_{\text{TP st.dev.}} \cdot x_{\text{TP st.dev.}}^{(i)} + \ldots$$

- Quadratic Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_N x_N^{(i)}$$
$$+ \theta_{N+1} \cdot x_1^{(i)^2} + \cdots + \theta_N \cdot x_{N+N}^{(i)^{\;2}}$$
$$+ \theta_{2N+1} \cdot x_1^{(i)} \cdot x_2^{(i)} + \theta_{2N+1} \cdot x_1^{(i)} \cdot x_N^{(i)}$$
$$+ \ldots$$

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

- Linear Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta^T \cdot \mathbf{x}^{(i)} = \sum_{j=0}^{N} \theta_j \cdot x_j^{(i)}$$
$$= \theta_0 + \theta_{\text{TP mean}} \cdot x_{\text{TP mean}}^{(i)}$$
$$+ \theta_{\text{TP st.dev.}} \cdot x_{\text{TP st.dev.}}^{(i)} + \ldots$$

- Quadratic Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_N x_N^{(i)}$$
$$+ \theta_{N+1} \cdot x_1^{(i)^2} + \cdots + \theta_N \cdot x_{N+N}^{(i)^{\;2}}$$
$$+ \theta_{2N+1} \cdot x_1^{(i)} \cdot x_2^{(i)} + \theta_{2N+1} \cdot x_1^{(i)} \cdot x_N^{(i)}$$
$$+ \ldots$$

- Neural Network (NN):

  $h_\theta(\cdot)$ is the output of a NN with weights θ.

Parametric models

- Convention

$$\mathbf{x}^{(i)^T} = (1, x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbb{R}^{N+1}$$

- Linear Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta^T \cdot \mathbf{x}^{(i)} = \sum_{j=0}^{N} \theta_j \cdot x_j^{(i)}$$
$$= \theta_0 + \theta_{\text{TP mean}} \cdot x_{\text{TP mean}}^{(i)}$$
$$+ \theta_{\text{TP st.dev.}} \cdot x_{\text{TP st.dev.}}^{(i)} + \ldots$$

- Quadratic Model

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_N x_N^{(i)}$$
$$+ \theta_{N+1} \cdot x_1^{(i)^2} + \cdots + \theta_N \cdot x_{N+N}^{(i)\ \ 2}$$
$$+ \theta_{2N+1} \cdot x_1^{(i)} \cdot x_2^{(i)} + \theta_{2N+1} \cdot x_1^{(i)} \cdot x_N^{(i)}$$
$$+ \ldots$$

- Neural Network (NN):

  $h_\theta(\cdot)$ is the output of a NN with weights θ.

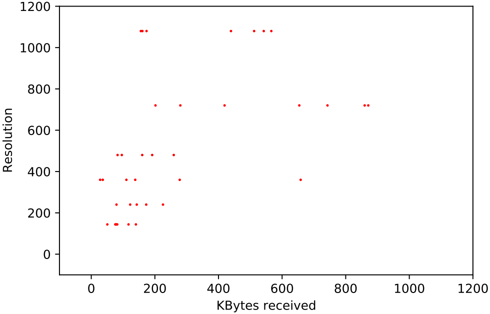
Loss: $J = \frac{1}{M} \sum_{i=1}^{M} \left(\underbrace{y^{(i)} - h_\theta(\mathbf{x}^{(i)})}_{\textbf{Residual } \varepsilon^{(i)}} \right)^2$

Training

Find $\theta^* \triangleq \arg\min_\theta J(\theta, \mathbf{X}, \mathbf{y})$

Example of univariate model

Predict the video resolution just based on KB received in 100ms time slot.
Are you able to find a simple model $h(\text{KBytesReceived})$ “by hand”?

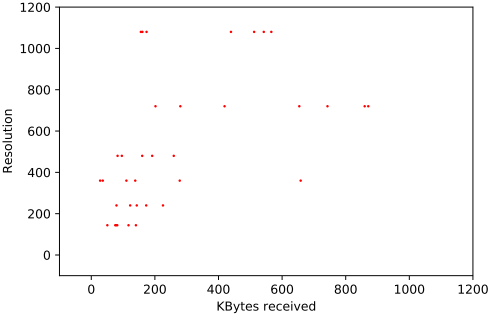


KBytesReceived	avg_qual
10.086	0
16.218	0
16.187	0
13.187	0
36.199	360
16.832	0
17.040	0
16.522	0
77.605	144
16.475	0
16.344	0
75.432	144
11.361	0
11.507	0
17.902	0
278.065	360
19.808	0
27.840	360

[Source of data [GGA+19]]

Example of univariate model

Predict the video resolution just based on KB received in 100ms time slot. Are you able to find a simple model $h(\text{KBytesReceived})$ "by hand"?



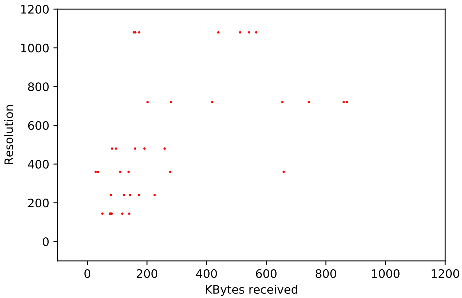
KBytesReceived	avg_qual
10.086	0
16.218	0
16.187	0
13.187	0
36.199	360
16.832	0
17.040	0
16.522	0
77.605	144
16.475	0
16.344	0
75.432	144
11.361	0
11.507	0
17.902	0
278.065	360
19.808	0
27.840	360

Generalization

If we see that a new connection with 500 KB / 100ms, what is the predicted video resolution?

[Source of data [GGA+19]]

Training a Linear Model: Example



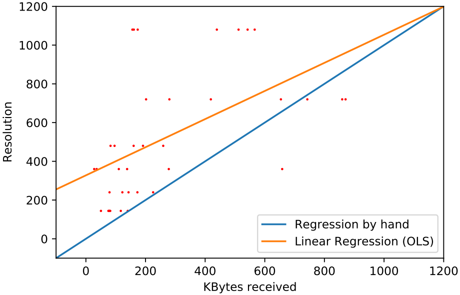
Find θ_0, θ_1 such that $h_\theta(\mathbf{x}^{(i)}) \simeq \theta_0 + \theta_1 \cdot x_1^{(i)}$.

- Regression "by hand"

KBytesReceived	avg_qual
10.086	0
16.218	0
16.187	0
13.187	0
36.199	360
16.832	0
17.040	0
16.522	0
77.605	144
16.475	0
16.344	0
75.432	144
11.361	0
11.507	0
17.902	0
278.065	360
19.808	0
27.840	360

Source of data [GGA+19]

Training a Linear Model: Example



Find θ_0, θ_1 such that $h_\theta(\mathbf{x}^{(i)}) \simeq \theta_0 + \theta_1 \cdot x_1^{(i)}$.

- Regression "by hand"
- Ordinary Least Square (OLS) regression

$$\theta^* = \arg\min_\theta J \qquad \theta_0^* = 351.8 \qquad \theta_1^* = 0.7.$$

KBytesReceived	avg_qual
10.086	0
16.218	0
16.187	0
13.187	0
36.199	360
16.832	0
17.040	0
16.522	0
77.605	144
16.475	0
16.344	0
75.432	144
11.361	0
11.507	0
17.902	0
278.065	360
19.808	0
27.840	360

Source of data [GGA+19]

Bi-variate linear model

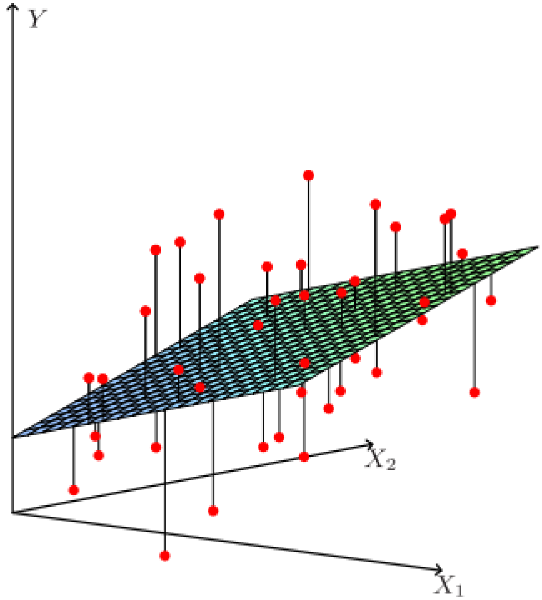


Figure 3.1 of [HTF09]

Matricial form

- Matrix of samples: $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)^T} \\ \vdots \\ \mathbf{x}^{(M)^T} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_N^{(1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_1^{(M)} & x_2^{(M)} & \dots & x_N^{(M)} \end{bmatrix}$
- Vector of true labels: $\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(M)} \end{bmatrix}$
- Vector of model parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_N \end{bmatrix}$
- Vector of predicted labels: $\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(M)} \end{bmatrix} = \begin{bmatrix} h_\theta(\mathbf{x}^{(1)}) \\ \vdots \\ h_\theta(\mathbf{x}^{(M)}) \end{bmatrix}$

Examples of loss function

For regression

- Mean Square Error (MSE)

$$J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M} \sum_{i=1}^{M} \left(\underbrace{y^{(i)} - h_\theta(\mathbf{x}^{(i)})}_{\textbf{Residual } \varepsilon^{(i)}} \right)^2$$

- Root Mean Square Error (RMSE)

$$J(\theta, \mathbf{X}, \mathbf{y}) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)}) \right)^2}$$

- ...

For classification:

- Misclassification Rate

$$J(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{M} \sum_{i=1}^{M} I_{y^{(i)} \neq h_\theta(\mathbf{x}^{(i)})}$$

- ...

Training

Find $\theta^* \triangleq \arg\min_\theta J(\theta, \mathbf{X}, \mathbf{y})$

Solution algorithm:

- Linear regression: matrix inversion
- Neural network: backpropagation
- ...

Note that

- θ^* depends on the observed data $(\mathbf{X}, \mathbf{y})$.
- If we observed other data $(\mathbf{X}', \mathbf{y}')$ we would get another θ^*.
- ...

Section 2

Ordinary Least Squares

Training a Linear Regression Model

For any sample $\mathbf{x}^{(i)}$, a linear model is:

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 \cdot 1 + \theta_1 \cdot x_1^{(i)} + \cdots + \theta_N \cdot x_N^{(i)} = \mathbf{x}^{(i)^T} \cdot \theta$$

Assume the loss function below:

$$\text{MSE} = J(\theta, \mathbf{X}, \hat{\mathbf{y}}) = \frac{1}{M} \sum_{i=1}^{M} \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)}) \right)^2$$

The model h_{θ^*} that minimizes MSE is called **Ordinary Least Squares** (OLS) model.

Theorem: Normal equation

The optimal parameter vector is

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

(provided that $(\mathbf{X}^T \cdot \mathbf{X})^{-1}$ is invertible)

NB: Given a certain training set $(\mathbf{X}, \mathbf{y})$, we can immediately find the optimal θ^*.

Training a Linear Regression Model (I)

For any sample $\mathbf{x}^{(i)}$, the prediction is (This proof is similar to §3.2 of [HTF09].):

$$h_\theta(\mathbf{x}^{(i)}) = \theta_0 \cdot 1 + \theta_1 \cdot x_1^{(i)} + \cdots + \theta_N \cdot x_N^{(i)} = \mathbf{x}^{(i)^T} \cdot \theta$$

The loss function is:

$$J(\theta, \mathbf{X}, \hat{\mathbf{y}}) = \frac{1}{M} \sum_{i=1}^{M} \left(y^{(i)} - h_\theta(\mathbf{x}^{(i)}) \right)^2 = \frac{1}{M} \sum_{i=1}^{M} \left(y^{(i)} - \mathbf{x}^{(i)^T} \cdot \theta \right)^2$$

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

To minimize the function, we set the gradient to 0:

$$\nabla_\theta J(\theta, \mathbf{X}, \hat{\mathbf{y}}) = \mathbf{0}$$

Chain rule of derivation:

$$\frac{2}{M} (\mathbf{y} - \mathbf{X} \cdot \theta) \cdot \nabla_\theta (\mathbf{y} - \mathbf{X} \cdot \theta) = \mathbf{0} \tag{2}$$

$$\frac{2}{M} (\mathbf{y} - \mathbf{X} \cdot \theta) \cdot (-\mathbf{X}) = \mathbf{0} \tag{3}$$

$$-\mathbf{y} \cdot \mathbf{X} + \mathbf{X} \cdot \theta \cdot \mathbf{X} = \mathbf{0} \tag{4}$$

Training a Linear Regression Model (II)

Let us isolate θ:

$$\mathbf{X} \cdot \theta \cdot \mathbf{X} = \mathbf{y} \cdot \mathbf{X} \tag{5}$$

$$\mathbf{X}^T \cdot \mathbf{X} \cdot \theta \cdot \mathbf{X} = \mathbf{X}^T \cdot \mathbf{y} \cdot \mathbf{X} \tag{6}$$

Multiply on the left by $(\mathbf{X}^T \cdot \mathbf{X})^{-1}$ (Assuming it exists):

$$\theta \cdot \mathbf{X} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y} \cdot \mathbf{X} \tag{7}$$

Multiply on the right by $\mathbf{X}^T$

$$\theta \cdot \mathbf{X} \cdot \mathbf{X}^T = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y} \cdot \mathbf{X} \cdot \mathbf{X}^T \tag{8}$$

Multiply on the right by[a] $(\mathbf{X} \cdot \mathbf{X}^T)^{-1}$:

$$\theta = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y} \tag{9}$$

[a] By the property of transpose matrices, if $(\mathbf{X}^T \cdot \mathbf{X})^{-1}$ exists, then $(\mathbf{X} \cdot \mathbf{X}^T)^{-1}$ exists as well.

Interpreting regression results: sign

You want to predict `Resolution` based on `KBytesReceived`:

$$\hat{y}^{(i)} = \theta_0 + \theta_{\text{KBytesReceived}} \cdot x^{(i)}_{\text{KBytesReceived}}$$

KBytesReceived	Resolution
36.199	360
77.605	144
75.432	144
278.065	360

Running the OLS model we get:

	coef	P>|t|
const	351.7606	0.000
KBytesReceived	0.7115	0.001

The model is

$$\hat{y}^{(i)} = 351.76 + 0.71 \cdot x^{(i)}_{\text{KBytesReceived}}$$

Does the **sign** makes sense?
Positive/negative dependency.

Interpreting regression results: p-value

p-values: *significance* of a coefficient.

	coef	P>|t|
const	351.7606	0.000
KBytesReceived	0.7115	0.001

Hypothesis testing:

- **Null hypothesis**: No dependency between `KBytesReceived` and the target.

Interpreting regression results: p-value

p-values: *significance* of a coefficient.

	coef	P>|t|
const	351.7606	0.000
KBytesReceived	0.7115	0.001

Hypothesis testing:

- **Null hypothesis**: No dependency between `KBytesReceived` and the target.

- Performing OLS, under the null hp

$$\mathbb{P}\left[|\mathring{\theta}^*_{\texttt{KBytesReceived}}| \geq 0.71\right] = 0.001$$

- Would you *accept* or *reject* the null hp?

Interpreting regression results: p-value

p-values: *significance* of a coefficient.

	coef	P>|t|
const	351.7606	0.000
KBytesReceived	0.7115	0.001

Hypothesis testing:

- **Null hypothesis**: No dependency between `KBytesReceived` and the target.

- Performing OLS, under the null hp

$$\mathbb{P}\left[|\mathring{\theta}^*_{\texttt{KBytesReceived}}| \geq 0.71\right] = 0.001$$

- Would you *accept* or *reject* the null hp?
  - Accept $\Rightarrow$ 0.71 is not significant
  - Reject $\Rightarrow$ 0.71 is significant.

Interpreting regression results: p-value

p-values: *significance* of a coefficient.

	coef	P>|t|
const	351.7606	0.000
KBytesReceived	0.7115	0.001

Hypothesis testing:

- **Null hypothesis**: No dependency between `KBytesReceived` and the target.

- Performing OLS, under the null hp

$$\mathbb{P}\left[|\mathring{\theta}^*_{\texttt{KBytesReceived}}| \geq 0.71 \right] = 0.001$$

- Would you *accept* or *reject* the null hp?
  - Accept $\Rightarrow$ 0.71 is not significant
  - Reject $\Rightarrow$ 0.71 is significant.

Usually:

- p-value$\leq 5\%$ $\Rightarrow$ reject null hp $\Rightarrow$ Coeff significant
- o.w. Coeff not significant

Interpreting regression results

Predict `Resolution` based on `PacketsSent`, `KBytesReceived` and `BufferHealth`:

$$\hat{y}^{(i)} = \theta_0 + \theta_1 \cdot x_1^{(i)} + \theta_2 \cdot x_2^{(i)} + \theta_3 \cdot x_3^{(i)}$$

PacketsSent	KBytesReceived	BufferHealth	Resolution
6	36.199	10.241165	360
10	77.605	4.446780	144
14	75.432	3.989780	144
33	278.065	3.700462	360
6	27.840	4.512780	360
58	658.375	9.454706	360
14	77.429	4.606780	144
33	201.903	5.301853	720
18	172.740	3.638107	240
23	181.476	5.314732	240

Training the OLS model:

	coef	P>|t|
const	282.8794	0.017
PacketsSent	-0.5551	0.925
KBytesReceived	0.5986	0.194
BufferHealth	18.3779	0.343

The model is

$$\hat{y}^{(i)} = 282.9 - 0.56 \cdot x_1^{(i)} + 0.60 \cdot x_2^{(i)} + 18.34 \cdot x_3^{(i)}$$

Interpretation:

- Fixing all features
- 1KB more received (in the 100 ms window) $\Rightarrow$ resolution $\nearrow$ by 0.60p
- $\iff$ increase of 1MB $\Rightarrow$ resolution $\nearrow$ of 600p.

Interpreting regression results

	coef	P>|t|
const	282.8794	0.017
PacketsSent	-0.5551	0.925
KBytesReceived	0.5986	0.194
BufferHealth	18.3779	0.343

$$\hat{y}^{(i)} = 282.9 - 0.56 \cdot x_1^{(i)} + 0.60 \cdot x_2^{(i)} + 18.34 \cdot x_3^{(i)}$$

Do the **signs** make sense?

Interpreting regression results

	coef	P>|t|
const	282.8794	0.017
PacketsSent	-0.5551	0.925
KBytesReceived	0.5986	0.194
BufferHealth	18.3779	0.343

$$\hat{y}^{(i)} = 282.9 - 0.56 \cdot x_1^{(i)} + 0.60 \cdot x_2^{(i)} + 18.34 \cdot x_3^{(i)}$$

Do the **signs** make sense?

Are coefficients **significant**?

How is the p-value computed (I)

Theorem

Assume that

- *for any sample* $\mathbf{x}$*, the target is a random variable (r.v.):*

$$\mathring{y} = \mathbf{x}^T \cdot \boldsymbol{\beta} + \epsilon \tag{10}$$

  - $\boldsymbol{\beta}$*: some "true" parameter vector*
  - ϵ*: noise, Gaussian r.v.* $\epsilon \sim N(0, \sigma^2)$
  - σ^2*: variance of the residuals*

- *The true label* $y^{(i)}$ *is a realization of* $\mathring{y}$*.*

Then

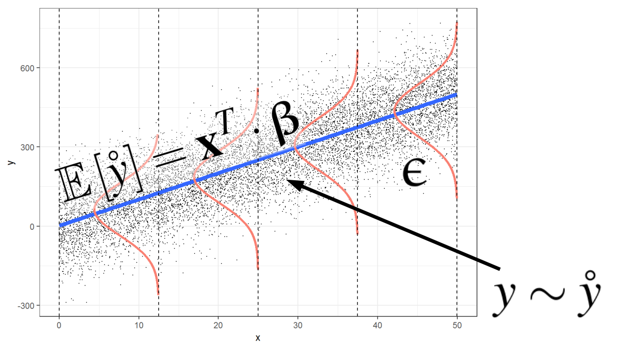
- *the coefficients* $\mathring{\theta}_j^*$ *we get from linear regression are also Gaussian r.v. They are* ***unbiased****, i.e.,*

$$\mathbb{E}\left[\mathring{\theta}_j^*\right] = \beta_j$$

How is the p-value computed (II)

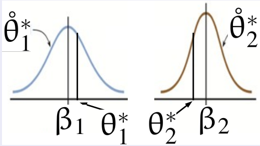
Theorem

Assume that



Picture from [LR19], Ch.1

Then



How is the p-value computed (III)

Proof of the previous claim. [1]

- Let's write (10) in vectorial form. Given a dataset $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)^T} \\ \vdots \\ \mathbf{x}^{(M)^T} \end{bmatrix}$, the target vector is:

$$\mathring{\boldsymbol{y}} = \mathbf{X} \cdot \boldsymbol{\beta} + \boldsymbol{\epsilon}$$

  It is a Gaussian r.v., since it is a constant matrix $\mathbf{X} \cdot \boldsymbol{\beta}$ plus a random vector $\boldsymbol{\epsilon}$.
- The parameter vector we get from OLS regression is:

$$\mathring{\theta}^* = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathring{\boldsymbol{y}}$$

  It is a Gauss.r.v., since it is a constant matrix $(\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T$ multiplied by the Gauss.r.v. $\mathring{\boldsymbol{y}}$).
- The mean is

$$\mathbb{E}\left[\mathring{\theta}^*\right] = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbb{E}\left[\mathring{\boldsymbol{y}}\right] = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{X} \cdot \boldsymbol{\beta} = \boldsymbol{\beta}$$

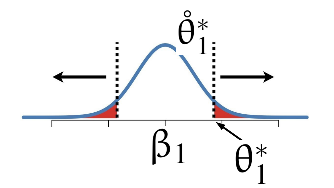
  and thus $\mathbb{E}\left[\mathring{\theta}_j^*\right] = \beta_j$
- We can also compute the variance of $\mathring{\theta}^*$.[2]

[1] To know more, check Sec. 3.8 of [HTF09], these videos and this video.

How is the p-value computed (IV)

- Compute
$$\theta^* = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$
- θ^* is a realization of the Gauss.r.v.
$$\mathring{\theta}^* = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathring{y}$$

- For any feature j:
  - θ^*_j realization of $\mathring{\theta}^*_j$, such that $\mathbb{E}\left[\mathring{\theta}^*_j\right] = \beta_j$
  - Assume true parameter $\beta_j = 0$.
  - Compute the variance $\mathrm{Var}(\mathring{\theta}^*_j)$.
  - Draw the probability density of $\mathring{\theta}^*_j$.



p-value

$\mathbb{P}\left[|\mathring{\theta}^*_j| \geq \theta^*_j\right]$

Significance degradation

Adding features can degrade significance

	coef	P>|t|
const	351.7606	0.000
KBytesReceived	0.7115	0.001

	coef	P>|t|
const	282.8794	0.017
PacketsSent	-0.5551	0.925
KBytesReceived	0.5986	0.194
BufferHealth	18.3779	0.343

Average of OLS predictions

Theorem: prediction and true value averages

In an OLS regression, the average of the predictions equals the average of the true values of the target:

$$\bar{\hat{y}} = \bar{y}$$

where:

$$\bar{\hat{y}} \triangleq \frac{1}{M}\sum_{i=1}^{M} \hat{y}^{(i)} \qquad \bar{y} \triangleq \frac{1}{M}\sum_{i=1}^{M} y^{(i)}$$

Proof

Normal equation:

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

Theorem: prediction and true value averages

In an OLS regression, the average of the predictions equals the average of the true values of the target:

$$\bar{\hat{y}} = \bar{y}$$

where:

$$\bar{\hat{y}} \triangleq \frac{1}{M}\sum_{i=1}^{M}\hat{y}^{(i)} \qquad \bar{y} \triangleq \frac{1}{M}\sum_{i=1}^{M}y^{(i)}$$

Proof

Normal equation:

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

Multiplying on the left by $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\underbrace{\mathbf{X}\cdot\theta^*} = \mathbf{X}^T\mathbf{y}$$

Average of OLS predictions

Theorem: prediction and true value averages

In an OLS regression, the average of the predictions equals the average of the true values of the target:

$$\bar{\hat{y}} = \bar{y}$$

where:

$$\bar{\hat{y}} \triangleq \frac{1}{M}\sum_{i=1}^{M} \hat{y}^{(i)} \qquad \bar{y} \triangleq \frac{1}{M}\sum_{i=1}^{M} y^{(i)}$$

Proof

Normal equation:

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

Multiplying on the left by $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\underbrace{\mathbf{X}\cdot\theta^*}_{\hat{\mathbf{y}}} = \mathbf{X}^T\mathbf{y}$$

Average of OLS predictions

Theorem: prediction and true value averages

In an OLS regression, the average of the predictions equals the average of the true values of the target:

$$\bar{\hat{y}} = \bar{y}$$

where:

$$\bar{\hat{y}} \triangleq \frac{1}{M}\sum_{i=1}^{M} \hat{y}^{(i)} \qquad \bar{y} \triangleq \frac{1}{M}\sum_{i=1}^{M} y^{(i)}$$

Proof

Normal equation:

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

Multiplying on the left by $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\underbrace{\mathbf{X}\cdot\theta^*}_{\hat{\mathbf{y}}} = \mathbf{X}^T\mathbf{y} \Longrightarrow \mathbf{X}^T\hat{\mathbf{y}} = \mathbf{X}^T\mathbf{y}$$

Theorem: prediction and true value averages

In an OLS regression, the average of the predictions equals the average of the true values of the target:

$$\bar{\hat{y}} = \bar{y}$$

where:

$$\bar{\hat{y}} \triangleq \frac{1}{M}\sum_{i=1}^{M} \hat{y}^{(i)} \qquad \bar{y} \triangleq \frac{1}{M}\sum_{i=1}^{M} y^{(i)}$$

Proof

Normal equation:

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

Multiplying on the left by $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\underbrace{\mathbf{X}\cdot\theta^*}_{\hat{\mathbf{y}}} = \mathbf{X}^T\mathbf{y} \Longrightarrow \mathbf{X}^T\hat{\mathbf{y}} = \mathbf{X}^T\mathbf{y}$$

The first element is:

$$[1\ldots 1]\cdot\hat{\mathbf{y}} = [1\ldots 1]\cdot\mathbf{y}$$

Role of the bias

$$\hat{y} = \theta_0 + \underbrace{\theta_1 x_1 + \cdots + \theta_N x_N}_{\text{dependency}}$$

- θ_0 does not capture $(\mathbf{x}, y)$ dependency
- θ_0 just "aligns" predictions to meet the label average.

Section 3

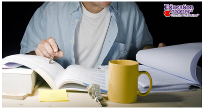
Validation of a model

Train and Test a model

- Supervised learning: we construct a model $h(\mathbf{x})$ based on observed $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots)$ for the purpose of approximating y for new samples $\mathbf{x}$.
- **Generalization:**[a] when the constructed model $h(\mathbf{x})$ is good at approximating labels y of **new** samples $\mathbf{x}$:

$$h(\mathbf{x}) \simeq y$$

- How do we test if the model *generalizes*.



$\Longrightarrow$

- Student analogy
  - He/she checks the answers during **training**
  - Answers are hidden during the **test**

---

[a] Sec. 3.1 of [GBD92].

Training and test sets

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	[illegible]	true	q240p	[illegible]
8.554780	[illegible]	true	q480p	[illegible]
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

$\mathbf{X}$ $\mathbf{y}$

Training and test sets

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
[illegible]	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	[illegible]
[illegible]	0.007990	true	q720p	720
3.638107	[illegible]	[illegible]	[illegible]	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	[illegible]
4.940168	0.008851	true	q1080p	[illegible]
4.940168	[illegible]	[illegible]	q1080p	[illegible]
9.239532	[illegible]	[illegible]	q720p	720

$\mathbf{X}_{\mathbf{train}}$ $\mathbf{Y}_{\mathbf{train}}$

$\mathbf{X}_{\mathbf{test}}$ $\mathbf{Y}_{\mathbf{test}}$

Training and test sets

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
[illegible]	[illegible]	true	q360p	[illegible]
4.606780	0.008046	true	q144p	[illegible]
[illegible]	[illegible]	[illegible]	q720p	[illegible]
3.638107	[illegible]	[illegible]	[illegible]	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	
5.590378	0.009113	true	q480p	
4.940168	0.008851	true	q1080p	
4.940168	[illegible]	[illegible]	q1080p	
9.239532	[illegible]	[illegible]	q720p	[illegible]

$\mathbf{X}_{\mathbf{train}}$ $\mathbf{Y}_{\mathbf{train}}$

$\mathbf{X}_{\mathbf{test}}$ $\mathbf{Y}_{\mathbf{test}}$

Training and test sets

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
[illegible]	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	[illegible]
[illegible]	0.007990	true	q720p	720
3.638107	[illegible]	true	[illegible]	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	
5.590378	0.009113	true	q480p	
4.940168	0.008851	true	q1080p	
4.940168	[illegible]	true	q1080p	
9.239532	[illegible]	true	q720p	[illegible]

$\mathbf{X}_{\text{train}}$ $\mathbf{Y}_{\text{train}}$

$\mathbf{X}_{\text{test}}$ $\mathbf{Y}_{\text{test}}$

Train using only the **training set** $(\mathbf{X}, \hat{\mathbf{y}})$:

$$\theta^* \triangleq \arg\min_{\theta} J(\theta, \mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$$

The trained model is

$$h_{\theta^*}(\cdot)$$

Evaluate the quality of the trained model via:

$$J(\theta^*, \mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$$

(*Test error* or *generalization error*[a] or *out-of-sample error*)

The J used during training and test may be different.

Training and test sets

Which method is better?

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	
5.590378	0.009113	true	q480p	
4.940168	0.008851	true	q1080p	
4.940168	0.008851	true	q1080p	
9.239532	0.016335	true	q720p	

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	
3.638107	0.005493	true	q240p	
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	
1.495841	0.002473	true	q720p	
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	

What if the data provider first experimented with all low resolutions and finally with all the highest?

Let's code ...

Go to notebook 02.regression/a.regression.ipynb

Section 4

Instability of a model (Variance)

Zero-variance features

$$\mathrm{Var}(j) = \frac{1}{M-1}\sum_{i=1}^{M}(x_j^{(i)} - \mu_j)^2$$

- Suppose $x_j^{(i)} = c$ for all samples i
- $\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & c & \dots & x_N^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(M)} & \dots & c & \dots & x_N^{(M)} \end{bmatrix}$
- Not full rank
- $\Rightarrow (\mathbf{X}^T\mathbf{X})^{-1}$ does not exist
- $\theta^* = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$ impossible

Collinearity

- Two features j_1, j_1 (columns of $\mathbf{X}$) are **collinear** if they are proportional

$$\begin{bmatrix} x_{j_1}^{(1)} \\ \vdots \\ x_{j_1}^{(N)} \end{bmatrix} = \alpha \begin{bmatrix} x_{j_2}^{(1)} \\ \vdots \\ x_{j_2}^{(N)} \end{bmatrix}$$

  In this case, column j_2 **adds no information** about prediction.

- $\Longrightarrow$ $\mathbf{X}$ has no full rank
- $\Longrightarrow$ $(\mathbf{X}^T \cdot \mathbf{X})$ is not invertible
- $\Longrightarrow$ not unique.
- $\Longrightarrow$ The normal equation

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

  cannot be computed.

Quasi-collinearity

- In real world, perfect collinearity is rare, but

$$\begin{bmatrix} x_{j_1}^{(1)} \\ \vdots \\ x_{j_1}^{(N)} \end{bmatrix} \simeq \alpha \begin{bmatrix} x_{j_2}^{(1)} \\ \vdots \\ x_{j_2}^{(N)} \end{bmatrix}$$

  and the computer is able to compute the normal equation, but ...

- For any model h_θ:

$$\begin{aligned} h_\theta(\mathbf{x}) &= \theta_{j_1}x_{j_1} + \theta_{j_2}x_{j_2} + \sum_{j \notin \{j_1, j_2\}} \theta_j x_j \\ &\simeq (\alpha\theta_{j_1} + \theta_{j_2})x_{j_2} + \sum_{j \notin \{j_1, j_2\}} \theta_j x_j \end{aligned}$$

- Infinite pairs of $(\theta_{j_1}, \theta_{j_2})$ would be almost equivalent
  - Ex.

- How does OLS choose among such pairs? Randomly
- Small differences of $\mathbf{X} \Rightarrow$ big differences in $(\theta^*_{j_1}, \theta^*_{j_2})$
- $\Rightarrow$ Variance

Multi-collinearity

- If j-th feature is a linear combination of others

$$\mathbf{X}_j = \alpha_1 \mathbf{X}_{j_1} + \alpha_2 \mathbf{X}_{j_2} + \dots$$

- $\mathbf{X} = [\mathbf{1}|\mathbf{X}_1|\dots|\mathbf{X}_N]$ no full rank
- The normal equation

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

  cannot be computed.
- How to discover Multi-collinearity? ...

Variance of a model

- Humans are good at generalizing knowledge ...
- ... because their perception models have low variance

Variance

A model suffers **high variance** if, by perturbing a bit the training dataset, the model changes completely. Suppose $\tilde{\mathbf{X}}, \tilde{\mathbf{y}}$ is a slightly perturbed version of $\mathbf{X}, \mathbf{y}$. If a model has high variance:

$$\tilde{\theta}^* = \arg\min_{\theta} J(\theta, \tilde{\mathbf{X}}, \tilde{\mathbf{y}})$$

completely different than

$$\theta^* = \arg\min_{\theta} J(\theta, \mathbf{X}, \mathbf{y})$$

- **Variance** is the contrary of **Stability**
- High variance $\Rightarrow$ high sensitivity to training data.

(§5 of [BK11])

Let’s code ...

Go to notebook 02regression/.a.regression.ipynb

Section 5

Cross-validation

Cross validation

How can we be sure that, if we change train/test split the error does not change?
$\Longrightarrow$ **Cross-validation**.

Algorithm 1 K-fold validation

1: Divide the dataset in K subsets
2: **for** $i = 1$ to K **do**
3: Keep subset i for test
4: Train on all the others
5: Compute test error
6: **end for**
7: Error = average of test errors

BufferHealth	BufferProgress	BufferValid	label	label_num
10.241165	0.015357	true	q360p	360
4.446780	0.007103	true	q144p	144
3.989780	0.006509	true	q144p	144
3.700462	0.005897	true	q360p	360
4.512780	0.007156	true	q360p	360
9.454706	0.016805	true	q360p	360
4.606780	0.008046	true	q144p	144
5.301853	0.007990	true	q720p	720
3.638107	0.005493	true	q240p	240
5.314732	0.009400	true	q240p	240
8.554780	0.011688	true	q480p	480
4.189780	0.007516	true	q360p	360
3.633641	0.005897	true	q480p	480
1.495841	0.002473	true	q720p	720
8.802211	0.014076	true	q1080p	1080
4.611142	0.009263	true	q144p	144
5.590378	0.009113	true	q480p	480
4.940168	0.008851	true	q1080p	1080
4.940168	0.008851	true	q1080p	1080
9.239532	0.016335	true	q720p	720

TRAINING SET

TEST SET

Let's code ...

Go to notebook 02.regression/a.regression.ipynb

In this lesson

- Supervised Learning
- First Model in Python
  (Linear Regression)
- Feature selection
- Validation: Train/Test; Cross-validation

Regression (continued)

- Polynomial Regression
- Model Variance / Complexity
- Regularization
- Scaling
- Feature Selection

Classification

- Logistic Regression
- Classification Performance
- Class imbalance

To know more

- Dealing with Multi-collinearity [DEB+13]
- Variance Inflation Factor for testing Multi-collinearity: pag. 101-102 of [JWHT13]
- Cross-validation from machinelearningmastery

References I

[BK11] Eric Bauer and Ron Kohavi, *An Empirical Comparison of Voting Classification Algorithms : Bagging , Boosting , and Variants*, Machine Learning **38** (2011), no. 1998, 1–38.

[DEB+13] Carsten F. Dormann, Jane Elith, Sven Bacher, Carsten Buchmann, Gudrun Carl, Gabriel Carré, Jaime R.García Marquéz, Bernd Gruber, Bruno Lafourcade, Pedro J. Leitão, Tamara Münkemüller, Colin Mcclean, Patrick E. Osborne, Björn Reineking, Boris Schröder, Andrew K. Skidmore, Damaris Zurell, and Sven Lautenbach, *Collinearity: A review of methods to deal with it and a simulation study evaluating their performance*, Ecography **36** (2013), no. 1, 027–046.

[GBD92] Stuart Geman, Elie Bienenstock, and René Doursat, *Neural Networks and the Bias/Variance Dilemma*, Neural Computation **4** (1992), no. 1, 1–58.

References II

[GGA+19] Craig Gutterman, Katherine Guo, Sarthak Arora, Xiaoyang Wang, Les Wu, Ethan Katz-Bassett, and Gil Zussman, *Requet: Real-time QoE detection for encrypted YouTube traffic*, ACM MMSys, 2019.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, 2nd ed., vol. 1, Springer, 2009.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An introduction to Statistical Learning*, vol. 7, 2013.

[KEB19] Muhammad Jawad Khokhar, Thibaut Ehlinger, and Chadi Barakat, *From Network Traffic Measurements to QoE for Internet Video*, IFIP Networking, 2019, pp. 1–9.

[LR19] Julie Legler and Paul Roback, *Broadening Your Statistical Horizons*, online, 2019.

References III

[Tho20] Simon Thorpe, *Intelligence artificielle et intelligence naturelle – vers l'IA bio-inspiré*, https://www.college-de-france.fr/site/stephane-mallat/seminar-2020-02-26-11h00.htm, 2020.