UVA CS 4501: Machine Learning

Lecture 3: Linear Regression

Dr. Yanjun Qi

University of Virginia

Department of
Computer Science

Notation

- Inputs
  - X, X_j (jth element of vector X) : random variables written in capital letter
  - p #inputs, N #observations
  - $\mathbf{X}$: matrix written in bold capital
  - Vectors are assumed to be column vectors
  - Discrete inputs often described by characteristic vector (dummy variables)

- Outputs
  - quantitative Y
  - qualitative C (for categorical)

- Observed variables written in lower case
  - The i-th observed value of X is x_i and can be a scalar or a vector

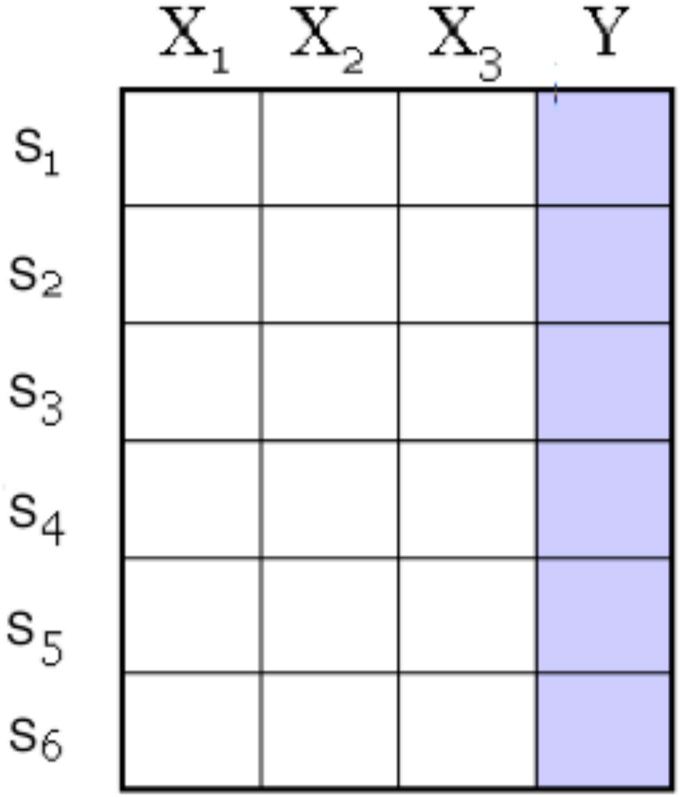
SUPERVISED LEARNING

$$f : X \longrightarrow Y$$

- Find function to map input space X to output space Y

- Generalisation: learn function / hypothesis from past data in order to "explain", "predict", "model" or "control" new data examples

KEY

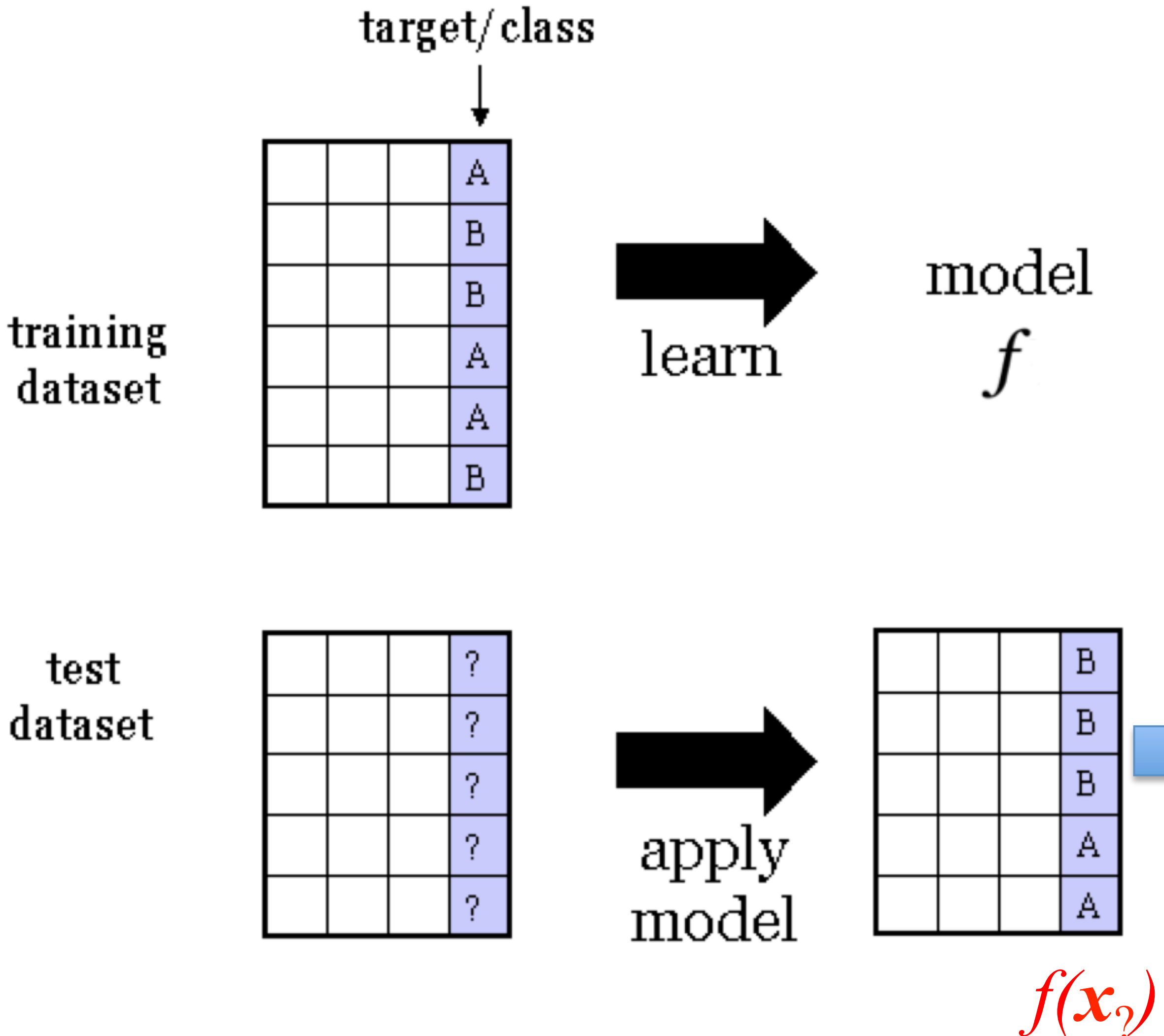


A Dataset

$$f : \boxed{X} \longrightarrow \boxed{Y}$$

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/ predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

SUPERVISED LEARNING



Training dataset consists of **input-output** pairs

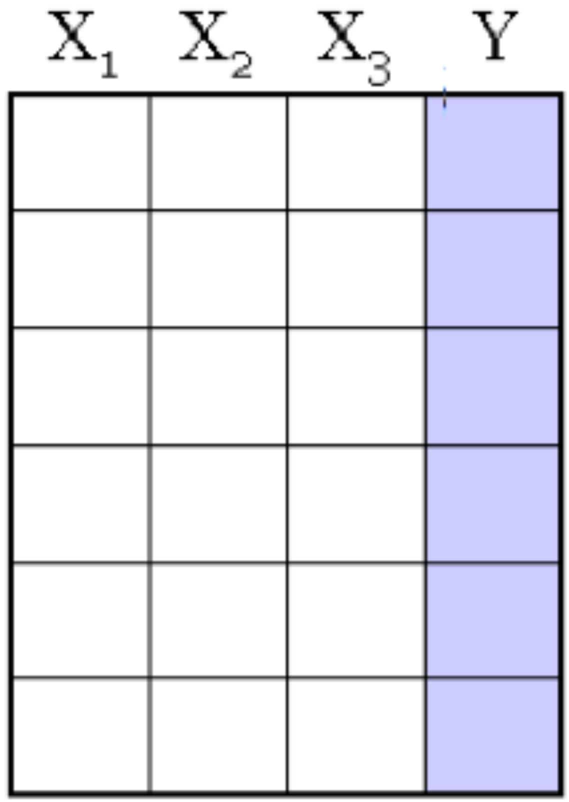
Measure Loss on pair
➔ Error ($f(\mathbf{x}_?)$, $y_?$)

training dataset

$$\mathbf{X}_{train} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y}_{train} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

test dataset

$$\mathbf{X}_{test} = \begin{bmatrix} -- & \mathbf{x}_{n+1}^T & -- \\ -- & \mathbf{x}_{n+2}^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_{n+m}^T & -- \end{bmatrix} \qquad \vec{y}_{test} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+m} \end{bmatrix}$$



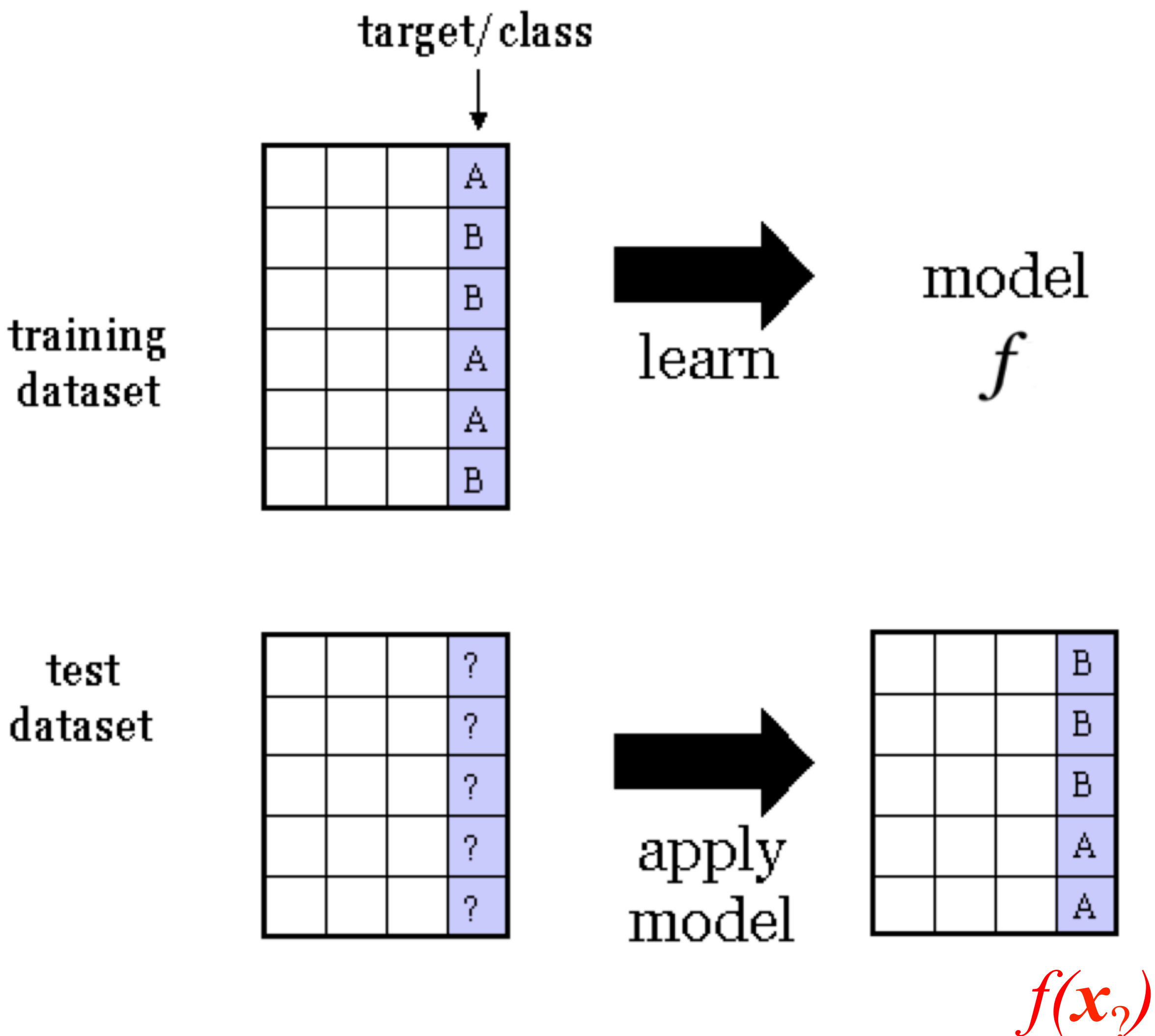
A Dataset for regression

$$f : \boxed{X} \longrightarrow \boxed{Y}$$

continuous valued variable

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/ predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

SUPERVISED Regression



Training dataset consists of **input-output** pairs

- When, target Y is a continuous target variable

e.g. A Practical Application of Regression Model

Movie Reviews and Revenues: An Experiment in Text Regression*

Mahesh Joshi   Dipanjan Das   Kevin Gimpel   Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{maheshj,dipanjan,kgimpel,nasmith}@cs.cmu.edu

Abstract

We consider the problem of predicting a movie's opening weekend revenue. Previous work on this problem has used metadata about a movie—e.g., its genre, MPAA rating, and cast—with very limited work making use of text *about* the movie. In this paper, we use the text of film critics' reviews from several sources to predict opening weekend revenue. We describe a new dataset pairing movie reviews with metadata and revenue data, and show that review text can substitute for metadata, and even improve over it, for prediction.

Proceedings of HLT '2010 Human Language Technologies:

I. The Story in Short

- Use metadata and critics' reviews to predict opening weekend revenues of movies
- Feature analysis shows what aspects of reviews predict box office success

II. Data

- 1718 Movies, released 2005-2009
- Metadata (genre, rating, running time, actors, director, etc.): *www.metacritic.com*
- Critics' reviews (~7K): Austin Chronicle, Boston Globe, Entertainment Weekly, LA Times, NY Times, Variety, Village Voice
- Opening weekend revenues and number of opening screens: *www.the-numbers.com*

Movie Reviews and Revenues: An Experiment in Text Regression, Proceedings of HLT '10 Human Language Technologies:

e.g. counts of a ngram in the text

IV. Features

I	Lexical n-grams (1,2,3)
II	Part-of-speech n-grams (1,2,3)
III	Dependency relations (nsubj,advmod,...)
Meta	U.S. origin, running time, budget (log), # of opening screens, genre, MPAA rating, holiday release (summer, Christmas, Memorial day,...), star power (Oscar winners, high-grossing actors)

Movie Reviews and Revenues: An Experiment in Text Regression, Proceedings of HLT '10 Human Language Technologies:

III. Model

- Linear regression with the elastic net (Zou and Hastie, 2005)

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}=(\beta_0,\boldsymbol{\beta})}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \right)^2 + \lambda P(\boldsymbol{\beta})$$

$$P(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left(\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j| \right)$$

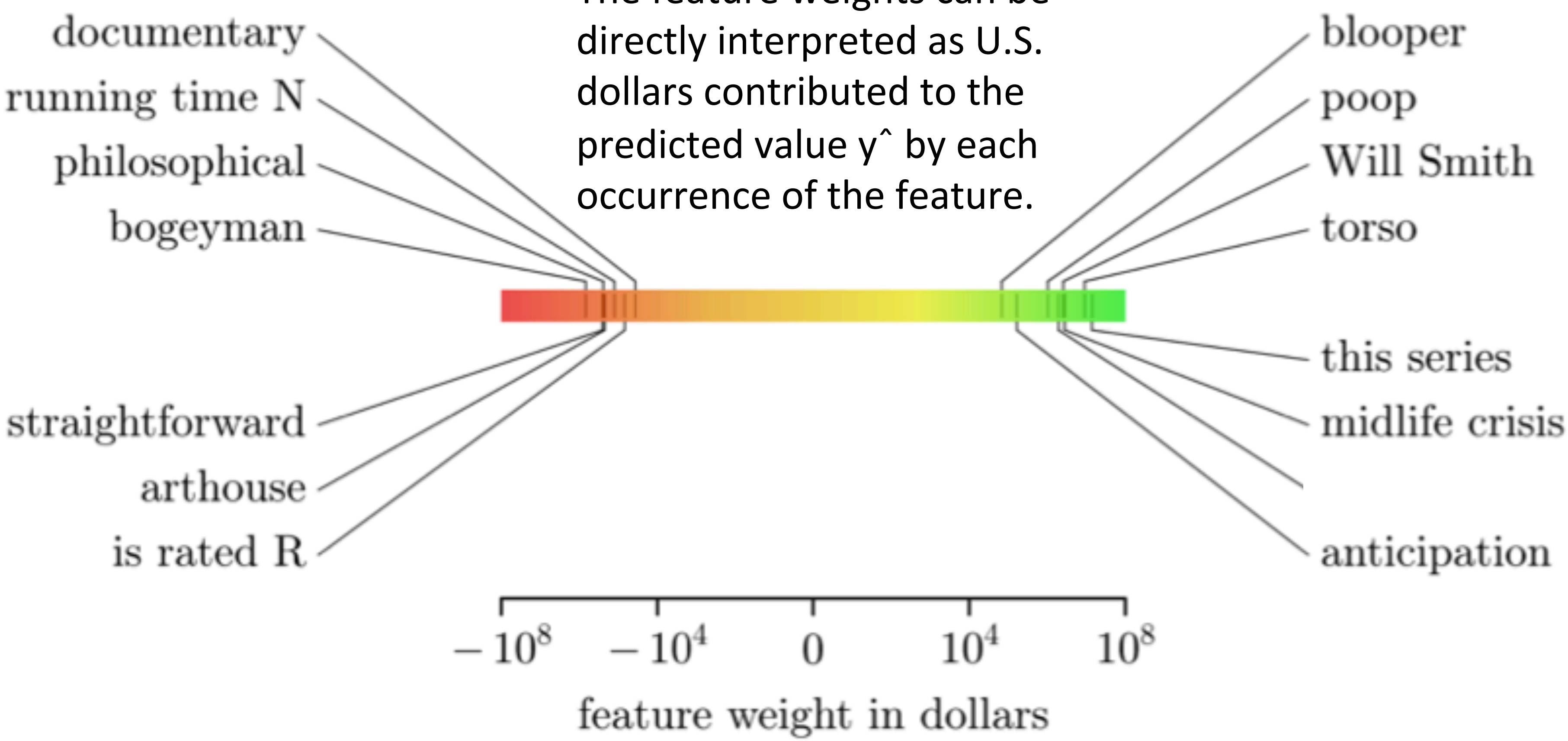
Use linear regression to directly predict the opening weekend gross earnings, denoted y, based on features x extracted from the movie metadata and/or the text of the reviews.

VIII. Get the Data!

www.ark.cs.cmu.edu/movie$-data

V. What May Have Brought You to movies

The feature weights can be directly interpreted as U.S. dollars contributed to the predicted value y^ by each occurrence of the feature.



Where are we ? ➔ Five major sections of this course

- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

Today →

Regression (supervised)

- ❑ Four ways to train / perform optimization for linear regression models
  - ❑ Normal Equation
  - ❑ Gradient Descent (GD)
  - ❑ Stochastic GD
  - ❑ Newton's method

- ❑ Supervised regression models
  - ❑ Linear regression (LR)
  - ❑ LR with non-linear basis functions
  - ❑ Locally weighted LR
  - ❑ LR with Regularizations

Machine Learning Variations in a Nutshell

Task: y

↓

Representation: x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters : f(), w, b

ML grew out of work in AI

Optimize a performance criterion using example data or past experience,

Aiming to generalize to unseen data

Today

- ❑ Linear regression (aka **least squares**)
- ❑ Learn to derive the least squares estimate by normal equation
- ❑ Evaluation with Cross-validation

For Example,

Machine learning for apartment hunting

- Now you've moved to Charlottesville !!

  And you want to find the **most reasonably priced** apartment satisfying your **needs:**

  square-ft., # of bedroom, distance to campus …

Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

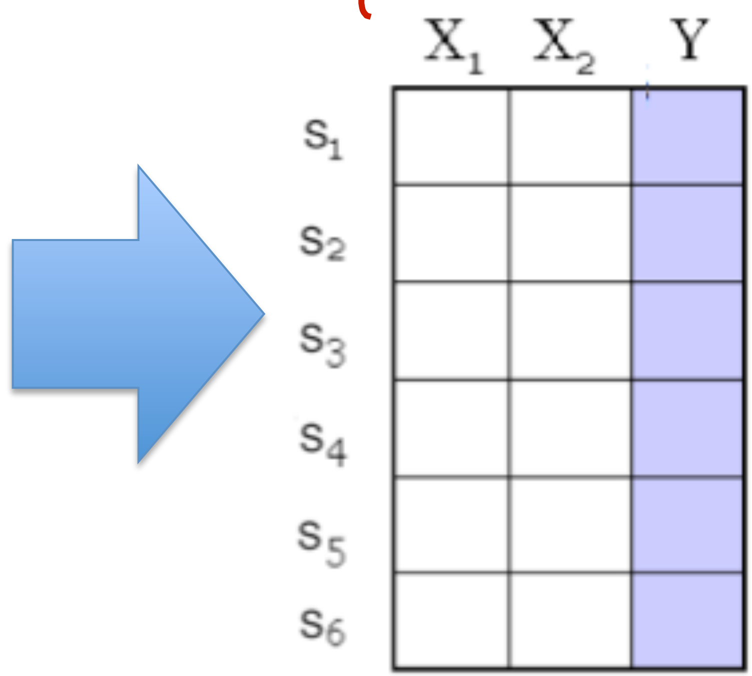
For Example,

Machine learning for apartment hunting

features output

Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

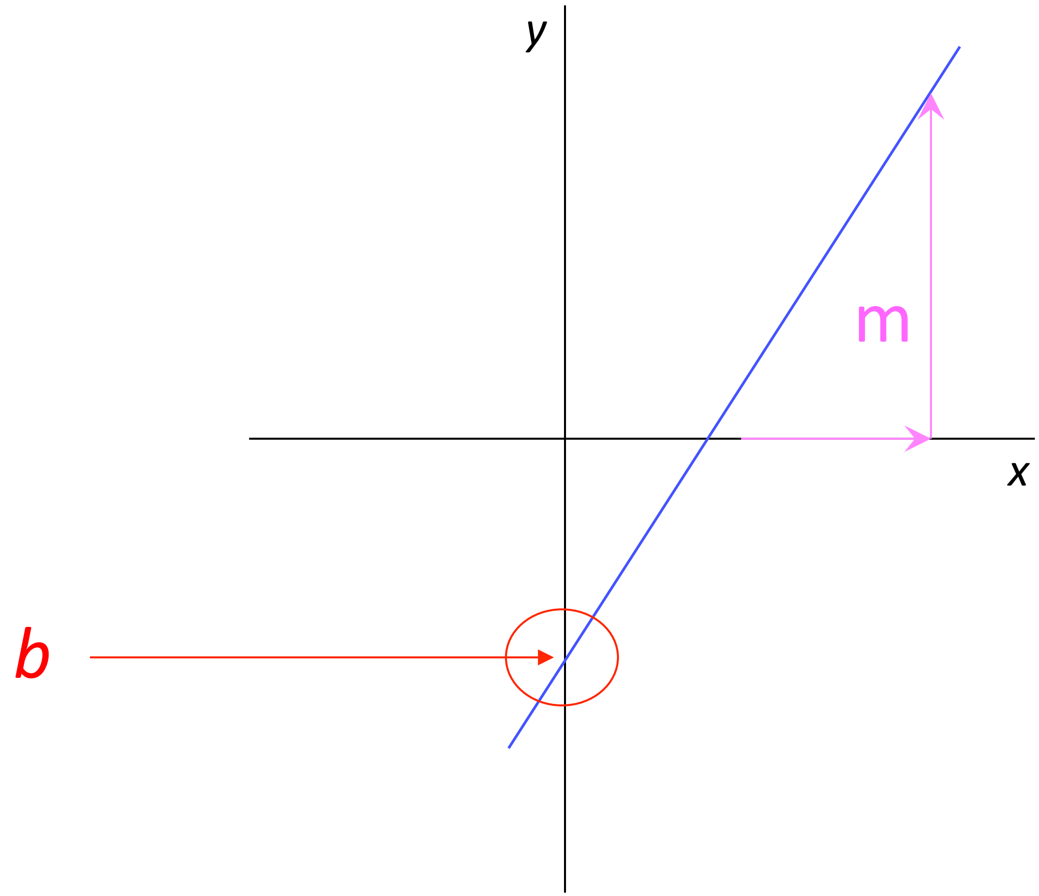
features X output



Review: f(x) is Linear when X is 1D

- $y=f(x)=mx+b$?

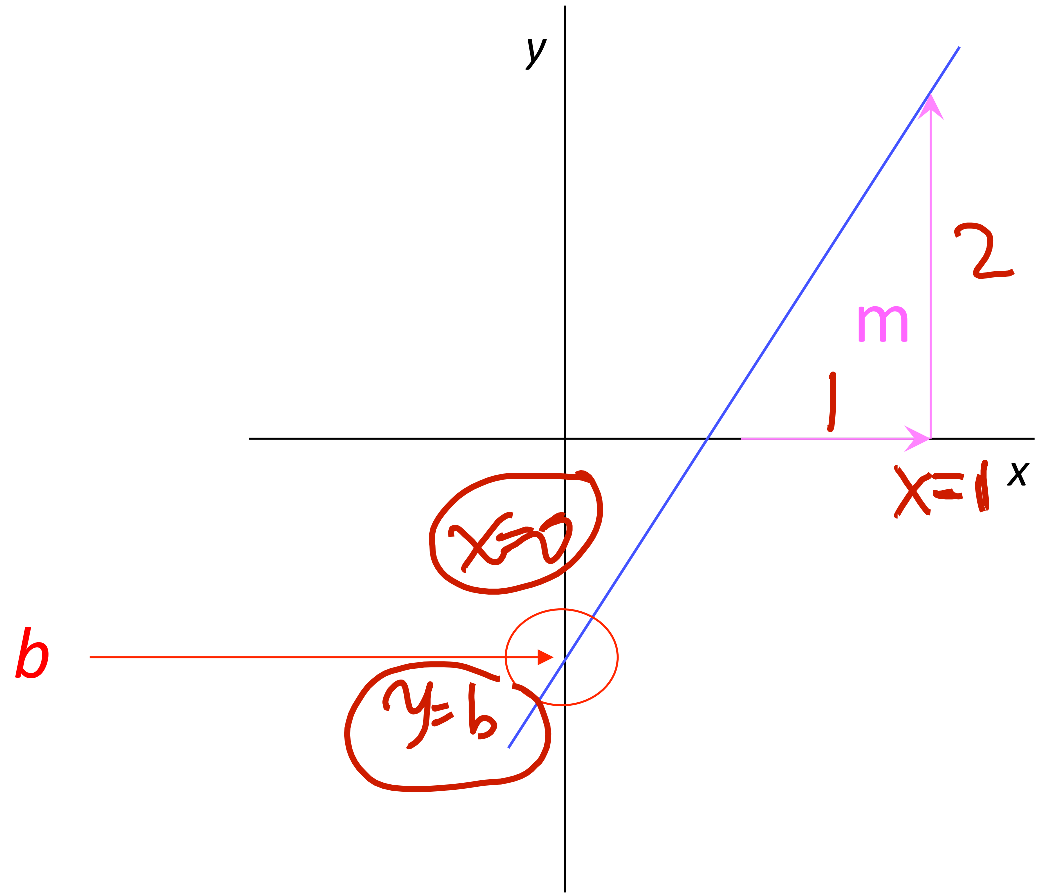
A slope of 2 (i.e. m=2) means that every 1-unit change in X yields a 2-unit change in Y.

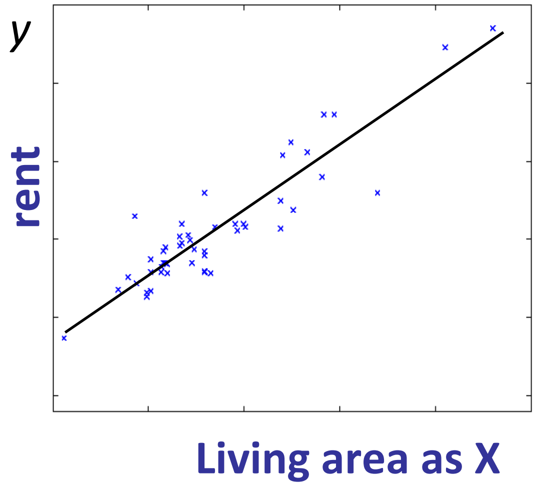


Review: f(x) is Linear when X is 1D

- *y=mx+b?*

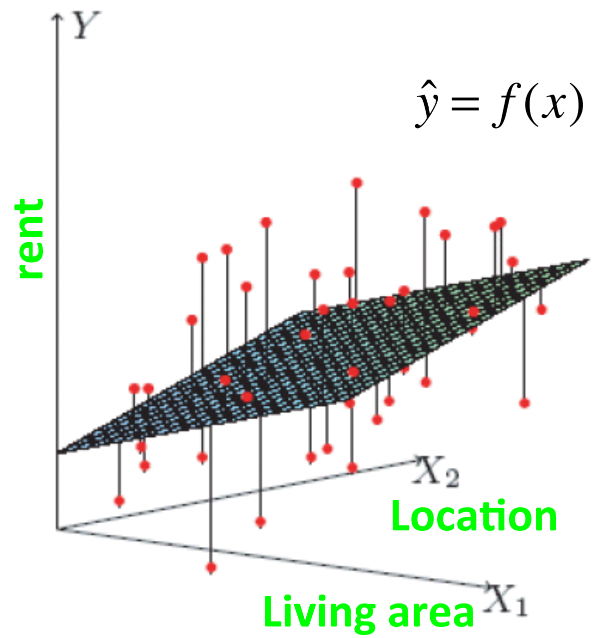
A slope of 2 (i.e. m=2) means that every 1-unit change in X yields a 2-unit change in Y.





y=mx+b

1D case ($\mathcal{X} = \mathbb{R}$): a line



$$\hat{y} = f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$\mathcal{X} = \mathbb{R}^2$: a plane

(Living area, Location) as X

Linear SUPERVISED Regression

$$\boxed{f} : X \longrightarrow Y$$

e.g. Linear Regression Models

$$\hat{y} = f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

=> Features **x**:

Living area, distance to campus, # bedroom …

=> Target y:

Rent → Continuous

Review: Special Uses for Matrix Multiplication

- Dot (or Inner) Product of two Vectors <x, y>

which is the sum of products of elements in similar positions for the two vectors

<x, y> = <y, x> $\quad$ $\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a}$

Where <x, y> = $x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ x_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i.$

A new representation (for each single data sample)

- Assume that each sample **x** is a column vector,
  - Here we assume a pseudo "feature" x_0=1 (this is the intercept term), and RE-define the feature vector to be:

$$\mathbf{x}^T = [x_0=1, x_1, \; x_2, \; \ldots, x_p]$$

  - the parameter vector θ is also a column vector

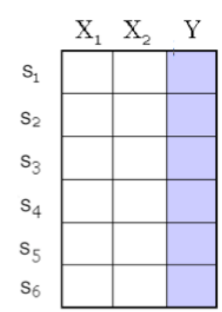
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_p \end{bmatrix}$$

$$\hat{y} = f(\mathbf{x}) = \mathbf{x}^T\theta = \theta^T\mathbf{x}$$

Training Set to Matrix Form

- Now represent the whole Training set (with n samples) as matrix form :

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,p} \\ x_{2,0}^{0} & x_{2,1} & \dots & x_{2,p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,0} & x_{n,1} & \dots & x_{n,p} \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$



Regression Formulation: from a single example to multiple examples in train set

- Matrix-Vector Products (I)

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$, their product is a vector $y = Ax \in \mathbb{R}^m$.

If we write A by rows, then we can express Ax as,

$$y = Ax = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}.$$

$$\begin{bmatrix} — & x_1^T & — \\ — & x_2^T & — \\ & \vdots & \\ — & x_m^T & — \end{bmatrix} \qquad \theta = X\theta$$

Regression Formulation: from a single example to multiple examples in train set

- Represent as matrix form:
  - Predicted output

$$\hat{Y} = \mathbf{X}\theta = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T\theta \\ \mathbf{x}_2^T\theta \\ \vdots \\ \mathbf{x}_n^T\theta \end{bmatrix}$$

n×p p×1

n×1

  - Labels (given output value)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

n×1

Now How to Learn the Regression Model?

- Using matrix form, we get the following general representation of the linear regression function:

$$\hat{Y} = \mathbf{X}\theta$$

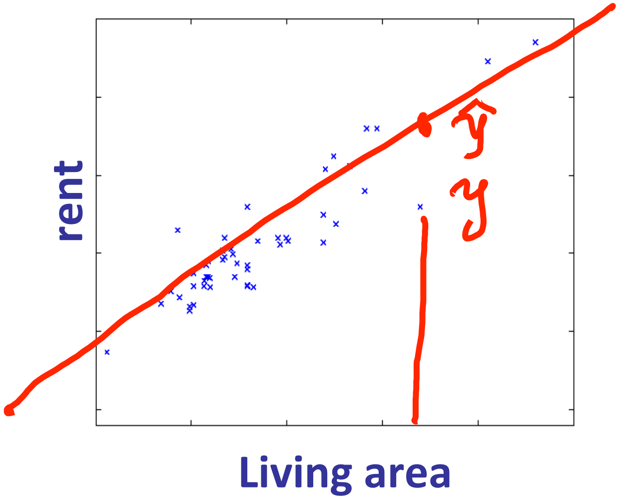
Y

$\Rightarrow \|Y-\hat{Y}\|_2^2$

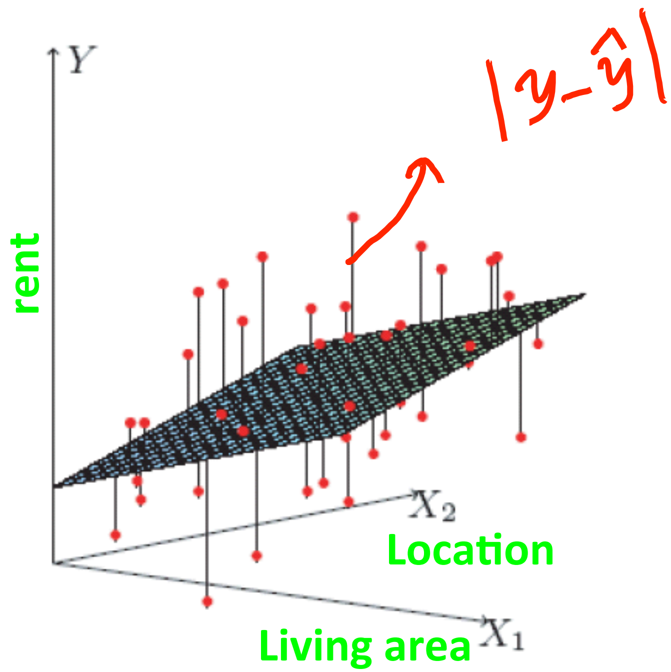
- Our goal is to pick the optimal θ that minimize the following cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2$$

SSE: Sum of squared error



1D case ($\mathcal{X} = \mathbb{R}$): a line



$\mathcal{X} = \mathbb{R}^2$: a plane

Today

- ❑ Linear regression (aka **least squares**)
- ❑ Learn to derive the least squares estimate by Normal Equation
- ❑ Evaluation with Cross-validation

Method I: normal equations

- Write the cost function in matrix form:

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^{T}\theta-y_i)^2$$
$$=\frac{1}{2}\left(X\theta-\vec{y}\right)^T\left(X\theta-\vec{y}\right)$$
$$=\frac{1}{2}\left(\theta^T X^T X\theta-\theta^T X^T\vec{y}-\vec{y}^T X\theta+\vec{y}^T\vec{y}\right)$$

$$\mathbf{X}=\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

To minimize $J(\theta)$, take derivative and set to zero:

$$\Rightarrow \boxed{X^T X\theta = X^T\vec{y}}$$

The normal equations

WHY ??

$$\Downarrow$$

$$\theta^* = \left(X^T X\right)^{-1} X^T\vec{y}$$

Review: Special Uses for Matrix Multiplication

- Sum the Squared Elements of a Vector

$$\mathbf{a} = \begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix}$$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i{}^T\theta - y_i)^2$$

$$a^T = \begin{bmatrix} 5 & 2 & 8 \end{bmatrix}$$

$$a^T\mathbf{a} = \begin{bmatrix} 5 & 2 & 8 \end{bmatrix}\begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix} = 5^2 + 2^2 + 8^2 = 93$$

Next : see White Board

$$a = \begin{bmatrix} \mathbf{x}_1^T\theta - y_1 \\ \mathbf{x}_2^T\theta - y_2 \\ \vdots \\ \mathbf{x}_n^T\theta - y_n \end{bmatrix} = X\theta - \vec{y}$$

$$\mathbf{a}^T\mathbf{a} = 2J(\theta) = \sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta-y_i)^2$$

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{n}(x_i^T\theta-y_i)^2$$

$$=\frac{1}{2}(\underset{n\times p}{X}\underset{p\times 1}{\theta}-\underset{n\times 1}{Y})^T(X\theta-Y)$$

$$=\frac{1}{2}(\theta^TX^T-y^T)(X\theta-Y)$$

$$=\frac{1}{2}(\theta^TX^TX\theta+y^Ty-\theta^TX^TY-Y^TX\theta)$$

$$a^T b = b^T a$$

$$\Rightarrow \theta^T X^T y = y^T X \theta$$

$$\Rightarrow J(\theta) = \frac{1}{2}\left(\theta^T X^T X \theta - 2\theta^T X^T y + y^T y\right)$$

A convex function is minimized @ point whose

✓ derivative (slope) is zero

✓ gradient is zero vector (multivariate case)

Review

$$\frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$$

$$\frac{\partial \mathbf{x}^T \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{B} + \mathbf{B}^T)\mathbf{x}$$

$$\Rightarrow \frac{\partial J(\theta)}{\partial \theta} = \frac{1}{\cancel{2}}\left(\cancel{2}X^T X\theta - \cancel{2}X^T y\right) \overset{\text{Set}}{\underset{\text{to}}{=}} 0$$

Review (I): a simple example

$$f(w) = w^T a = \left[w_1, w_2, w_3 \right] \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = w_1 + 2w_2 + 3w_3$$

$$\frac{\partial f}{\partial w_1} = 1$$
$$\frac{\partial f}{\partial w_2} = 2$$
$$\frac{\partial f}{\partial w_3} = 3$$

→ *Denominator layout*

$$\frac{\partial f}{\partial w} = \frac{\partial w^T a}{\partial w} = a = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\frac{\partial(\theta^T X^T y)}{\partial \theta} = X^T y$$

Review (II): Gradient of Quadratic Func

- See L2-note Page 17, Page 23-24
- See white board

$$\frac{\partial(\theta^T X^T X\theta)}{\partial\theta} = \frac{\partial(\theta^T G\theta)}{\partial\theta} = 2G\theta = 2X^T X\theta$$

Taking Gradient of the J()

$$\Rightarrow \frac{\partial J(\theta)}{\partial \theta} = \frac{1}{\cancel{2}}(\cancel{2}X^TX\theta - \cancel{2}X^Ty) \overset{\text{Set}}{\underset{\text{to}}{=}} 0$$

$$\Rightarrow \boxed{X^TX\theta = X^T\vec{y}}$$

The normal equations

$$\Downarrow$$

$$\theta^* = \left(X^TX\right)^{-1} X^T\vec{y}$$

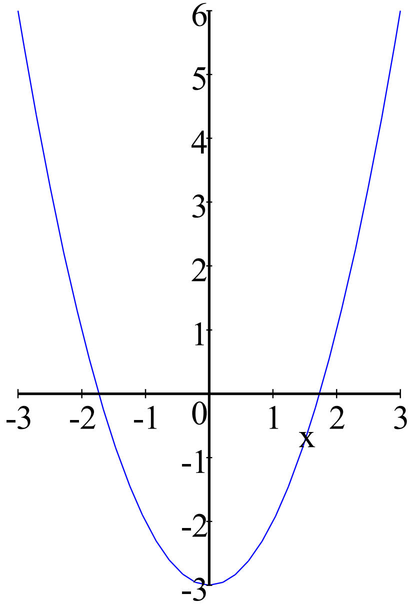
$$\Rightarrow \theta = \underbrace{(\underset{p\times n}{X^T}\ \underset{n\times p}{X})^{-1}}_{p\times p}\ \underbrace{\underset{p\times n}{X^T}\ \underset{n\times 1}{y}}_{p\times 1} \Rightarrow p\times 1$$

Extra: Convex function

- Intuitively, a convex function (1D case) has a single point at which the derivative goes to zero, and this point is a minimum.
- Intuitively, a function f (1D case) is convex on the range [a,b] if a function's second derivative is positive every-where in that range.

- Intuitively, if a multivariate function's Hessians is psd (positive semi-definite!), this (multivariate) function is Convex
  - Intuitively, we can think "Positive definite" matrices as analogy to positive numbers in matrix case

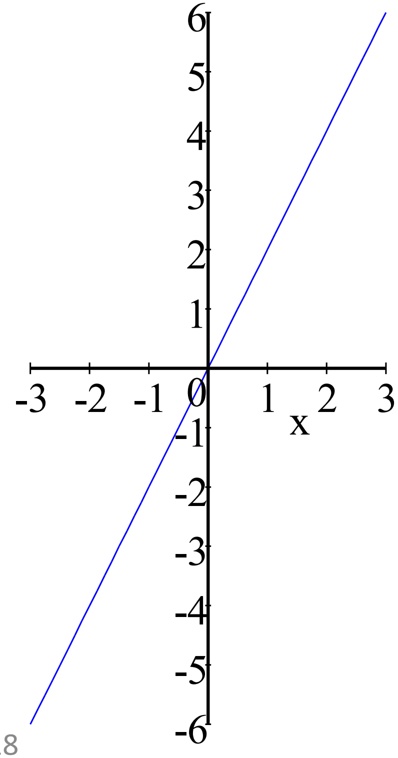
Review: Derivative of a Quadratic Function



$$y = x^2 - 3$$

$$y' = 2x$$

$$y'' = 2$$



This convex function is minimized @ the unique point whose derivative (slope) is zero.
➔ If finding zeros of the derivative of this function, we can also find minima (or maxima) of that function.

Extra: Loss J() is Convex

$$\Rightarrow J(\theta) = \frac{1}{2}\left(\theta^T X^T X \theta - 2\theta^T X^T y + y^T y\right)$$

$$\Rightarrow \text{Hessian}(J(\theta)) = X^T X$$

Gram matrix

$\Downarrow$

[PSD]

$\Downarrow$

$J(\theta)$ is convex

$\Downarrow$

If $\nabla J(\theta^*) = 0$, $J(\theta)$ is minimized @ θ^*

Extra: positive semi-definite!

L2-Note: Page 17

$A \in \mathbb{R}^{n \times n}$, $\forall x \in \mathbb{R}^n$

If $\underset{1 \times n}{x^T} \underset{n \times n}{A} \underset{n \times 1}{x} \geq 0$

$\Rightarrow$ A is positive semi-definite (PSD)

If $X^T A X > 0$

$\Rightarrow$ A is PD $\Rightarrow$ full rank / invertible

See proof on L2-Note: Page 18

Extra: Gram Matrix $G = X^T X$ is always positive semi-definite!

Because for any vector a

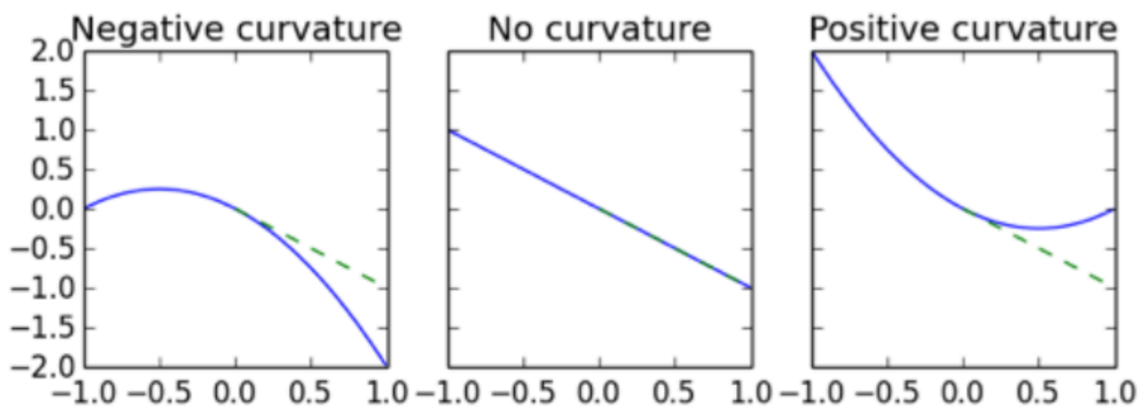
$$a^T X^T X a = \| X a \|_2^2 \geq 0$$

Besides, when X is full rank, G is invertible

Extra: Hessian

Derivatives and Second Derivatives

Cost function	Gradient	Hessian
$J(\boldsymbol{\theta})$	$\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$	$\boldsymbol{H}$
	$g_i = \frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta})$	$H_{i,j} = \frac{\partial}{\partial \theta_j} g_i$



$\underbrace{H \text{ PD}}$ for positive curvature

$$J(\theta) = (X\theta - y)^T (X\theta - y)\frac{1}{2}$$

$$= ((X\theta)^T - y^T)(X\theta - y)\frac{1}{2}$$

$$= (\theta^T X^T - y^T)(X\theta - y)\frac{1}{2}$$

$$= \left(\theta^T X^T X \theta - \underbrace{\theta^T X^T y - y^T X \theta} + y^T y\right)\frac{1}{2}$$

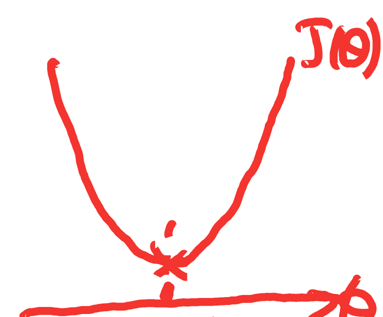
Since $\theta^T X^T y = y^T X \theta$

$\langle X\theta, y\rangle \quad \langle y, X\theta\rangle$

$$= \left(\theta^T X^T X \theta - 2\,\theta^T X^T y + y^T y\right)\frac{1}{2}$$

$\Rightarrow J(\theta)$ quadratic func of θ;

1d case



$$\frac{\partial J(\theta)}{\partial \theta} = 0$$

See handout 4.1 + 4.3 $\Rightarrow$ matrix calculus, partial deri $\Rightarrow$ Gradient

$$\nabla_\theta(\theta^T X^T X \theta) = 2X^T X \theta \qquad (P24)$$

$$\nabla_\theta(-2\theta^T X^T y) = -2X^T Y \qquad (P24)$$

$$\nabla_\theta(y^T y) = 0$$

$$\Rightarrow \nabla_\theta J(\theta) = \boxed{X^T X \theta - X^T Y}$$

gram matrix is PSD

if X full rank, $X^T X$ PD $\Rightarrow$ Invert

$$\Rightarrow \theta = \underbrace{(\underset{p \times n}{X^T} \underset{n \times p}{X})^{-1}}_{p \times p} \underbrace{\underset{p \times n}{X^T} \underset{n \times 1}{y}}_{p \times 1} \Rightarrow p \times 1$$

This loss function's Hessian is Positive Semi-definite

Comments on the normal equation

- In most situations of practical interest, the number of data points n is larger than the dimensionality p of the input space and the matrix $\mathbf{X}$ is of full column rank. If this condition holds, then it is easy to verify that $X^T X$ is necessarily invertible.

  $n >> p$

- The assumption that $X^T X$ is invertible implies that it is positive definite, thus the critical point we have found is a minimum.

- What if $\mathbf{X}$ has less than full column rank? → regularization (later).

The following complexity figures assume that arithmetic with individual elements has complexity $O(1)$, as is the case with fixed-preci
operations on a finite field.

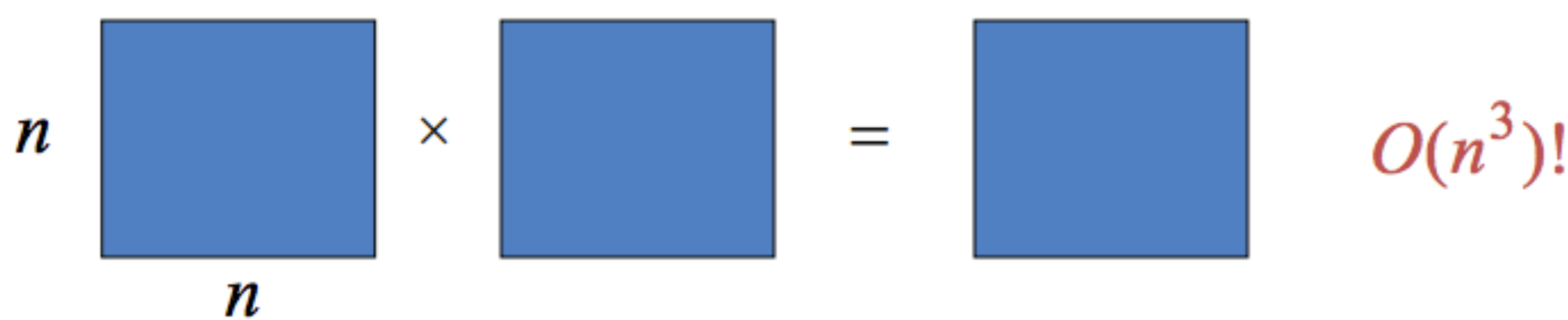
From Wiki

Operation	Input	Output	Algorithm	Complexity
Matrix multiplication	Two $n \times n$ matrices	One $n \times n$ matrix	Schoolbook matrix multiplication	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms[14][15][16]	$O(n^{2.373})$
Matrix multiplication	One $n \times m$ matrix & one $m \times p$ matrix	One $n \times p$ matrix	Schoolbook matrix multiplication	$O(nmp)$
Matrix inversion*	One $n \times n$ matrix	One $n \times n$ matrix	Gauss–Jordan elimination	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms	$O(n^{2.373})$
Singular value decomposition	One $m \times n$ matrix	One $m \times m$ matrix, one $m \times n$ matrix, & one $n \times n$ matrix		$O(mn^2)$ $(m \le n)$
		One $m \times r$ matrix, one $r \times r$ matrix, & one $n \times r$ matrix		
Determinant	One $n \times n$ matrix	One number	Laplace expansion	$O(n!)$
			Division-free algorithm[17]	$O(n^4)$
			LU decomposition	$O(n^3)$
			Bareiss algorithm	$O(n^3)$
			Fast matrix multiplication[18]	$O(n^{2.373})$
Back substitution	Triangular matrix	n solutions	Back substitution[19]	$O(n^2)$

Scalability to big data?

- Traditional CS view: Polynomial time algorithm, Wow!
- Large-scale learning: Sometimes even O(n) is bad! => Many state-of-the-art solutions (e.g., low rank, sparse, hardware, sampling, randomized…)

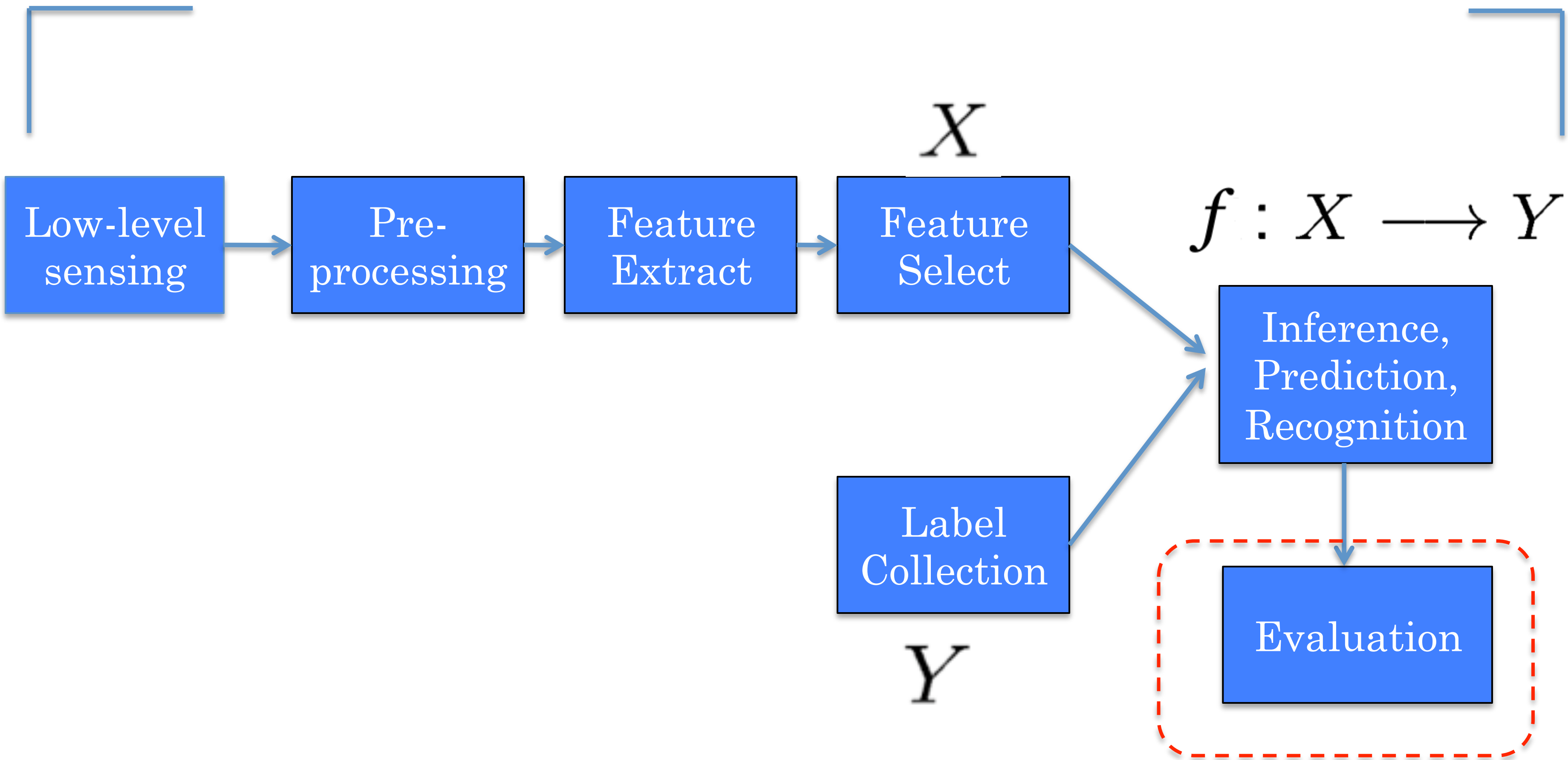
Simple example: Matrix multiplication



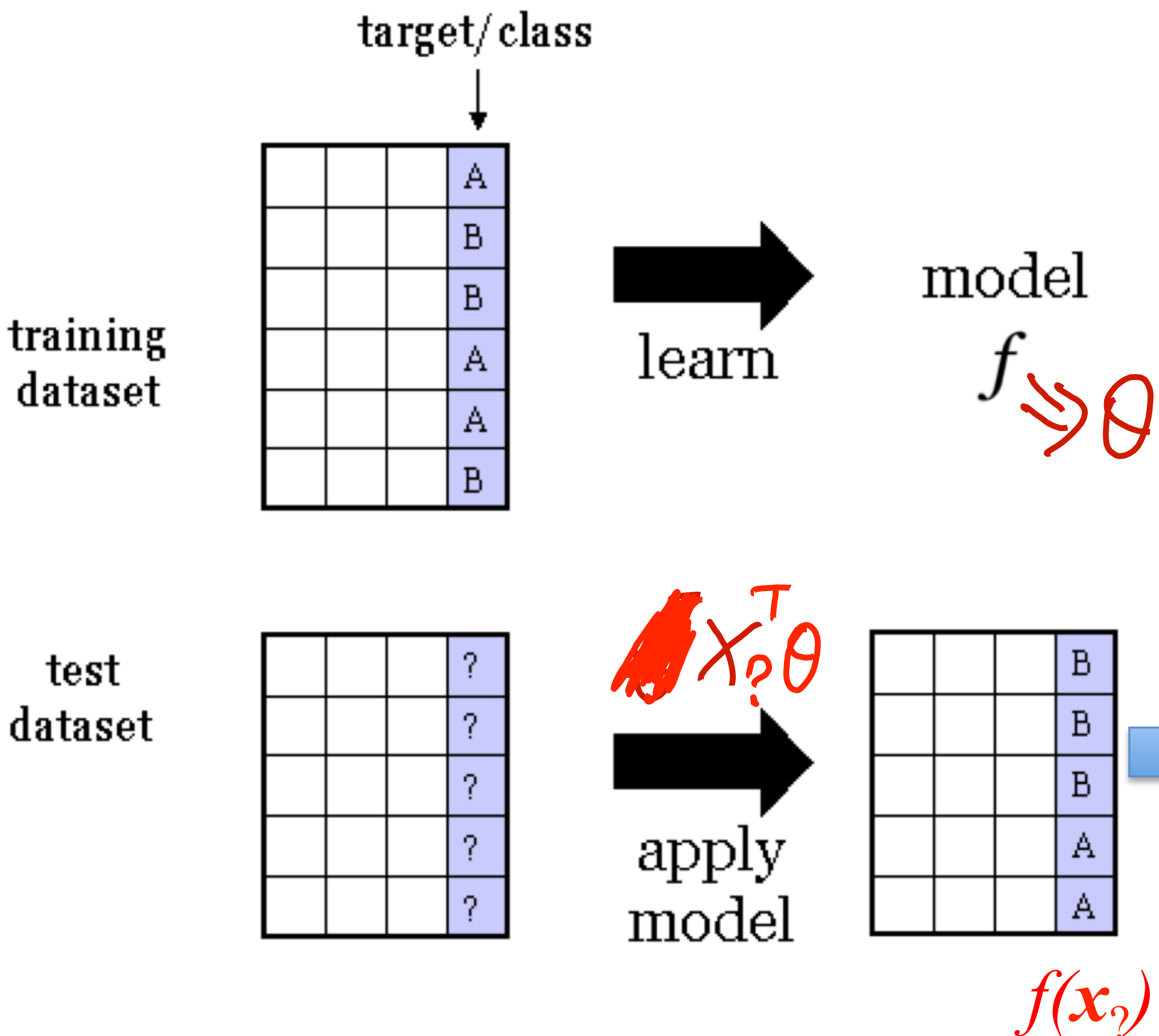
Today

- ❑ Linear regression (aka **least squares**)
- ❑ Learn to derive the least squares estimate by optimization
- ❑ Evaluation with Train/Test OR k-folds Cross-validation

TYPICAL MACHINE LEARNING SYSTEM



Evaluation Choice-I: Train and Test



Training dataset consists of **input-output** pairs

Measure Loss on pair
➔ $(f(x_?),\ y_?)$

Evaluation Choice-I: e.g. for linear regression models

training dataset

$$\mathbf{X}_{train} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y}_{train} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

test dataset

$$\mathbf{X}_{test} = \begin{bmatrix} -- & \mathbf{x}_{n+1}^T & -- \\ -- & \mathbf{x}_{n+2}^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_{n+m}^T & -- \end{bmatrix} \qquad \vec{y}_{test} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+m} \end{bmatrix}$$

Evaluation Choice-I: e.g. for linear regression models

- Training SSE (sum of squared error):

$$J_{train}(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

- Minimize $J_{train}(\theta)$ ➔ *Normal Equation to get*

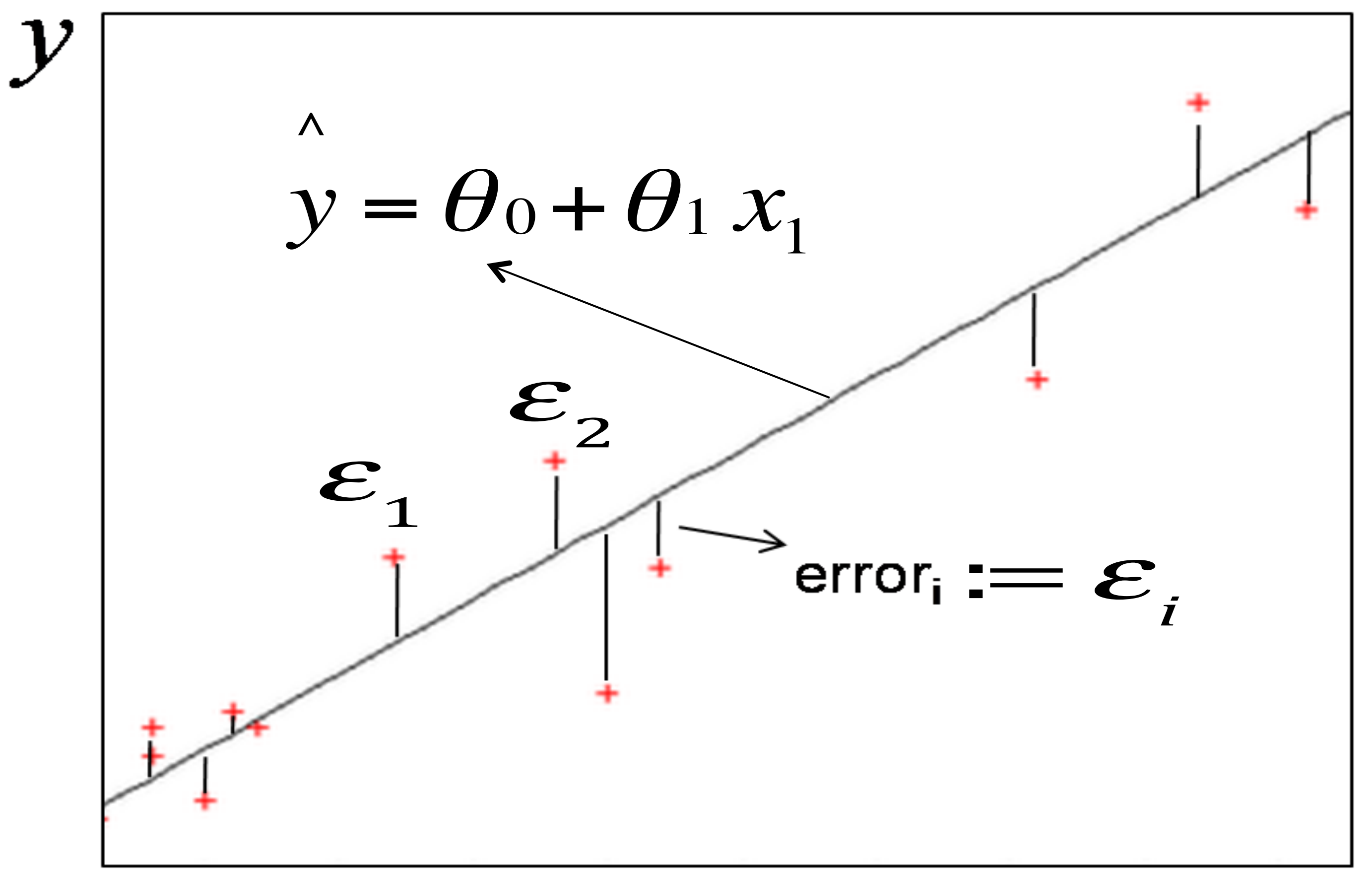
$$\theta^* = \operatorname{argmin} J_{train}(\theta) = \left(X_{train}^T X_{train}\right)^{-1} X_{train}^T \vec{y}_{train}$$

Evaluation Choice-I: e.g. for Regression Models

- Testing MSE Error to report:

$$J_{test} = \frac{1}{m} \sum_{i=n+1}^{n+m} (\mathbf{x}_i^T \theta^* - y_i)^2 = \frac{1}{m} \sum_{i=n+1}^{n+m} \varepsilon_i^2$$

Linear regression (1D example)



$$\theta^* = \left(X^T X\right)^{-1} X^T \vec{y}$$

Evaluation Choice-II: Cross Validation

- Problem: don't have enough data to set aside a test set
- Solution: Each data point is used both as train and test
- Common types:
    - -K-fold cross-validation (e.g. K=5, K=10)
    - -2-fold cross-validation
    - -Leave-one-out cross-validation (LOOCV, i.e., k=n_reference)

K-fold Cross Validation

- Basic idea:

  -Split the whole data to N pieces;

  -N-1 pieces for fit/train model; 1 for test;

  -Cycle through all N folds;

  -K=10 “folds” a common rule of thumb.

- The advantage:

  - all pieces are used for both training and validation;
  - each observation is used for validation exactly once.

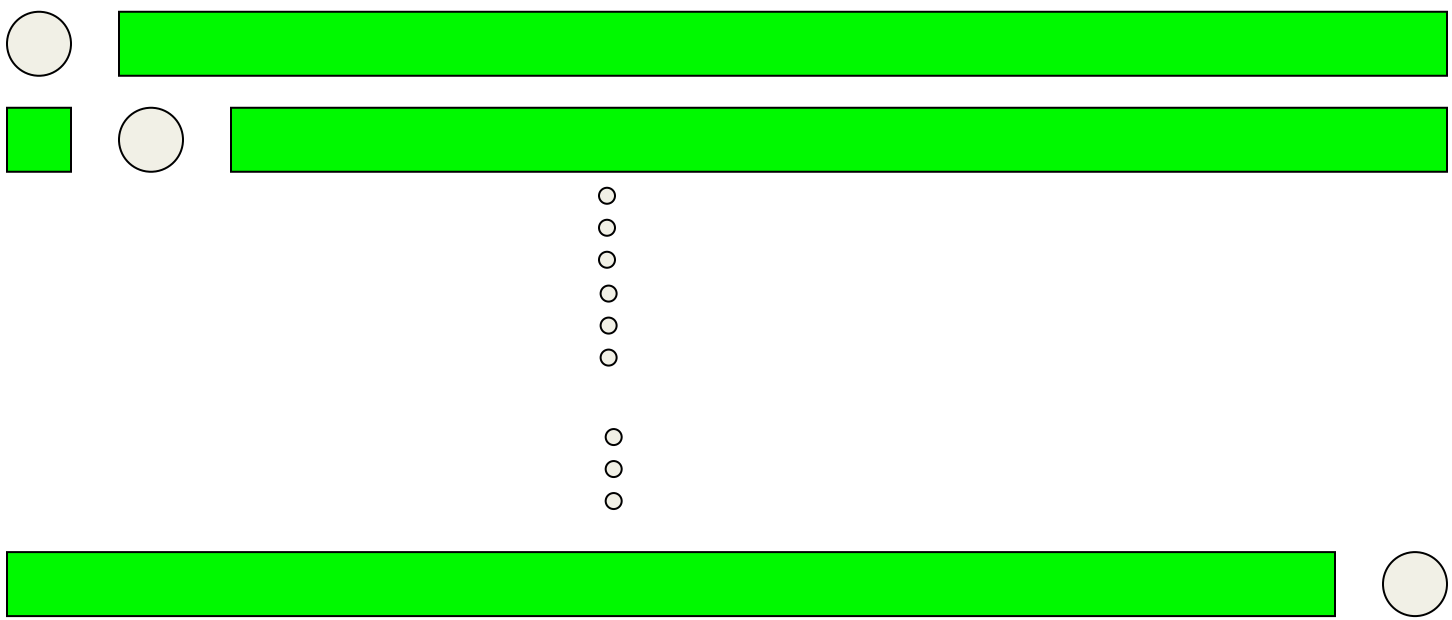
e.g. 10 fold Cross Validation

- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as test set
- Collect the scores from the diagonal
- We normally use the mean of the scores

model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	train	train	train	train	train	train	train	train	train	test
2	train	train	train	train	train	train	train	train	test	train
3	train	train	train	train	train	train	train	test	train	train
4	train	train	train	train	train	train	test	train	train	train
5	train	train	train	train	train	test	train	train	train	train
6	train	train	train	train	test	train	train	train	train	train
7	train	train	train	test	train	train	train	train	train	train
8	train	train	test	train	train	train	train	train	train	train
9	train	test	train	train	train	train	train	train	train	train
10	test	train	train	train	train	train	train	train	train	train

e.g. Leave-one-out / LOOCV (n-fold cross validation)

n is num. of data samples



Today Recap

- ❑ Linear regression (aka **least squares**)
- ❑ Learn to derive the least squares estimate by normal equation
- ❑ Evaluation with Train/Test OR k-folds Cross-validation

(1) Multivariate Linear Regression

Task	Regression
↓	↓
Representation	Y = Weighted linear sum of X's
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra
↓	↓
Models, Parameters	Regression coefficients

$$\hat{y} = f(x) = \theta^T x$$

References

- Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides

❑ http://www.cs.cmu.edu/~zkolter/course/15-884/linalg-review.pdf (please read)

❑ Prof. Alexander Gray's slides