CSCI567 Machine Learning (Fall 2014)

Drs. Sha & Liu

{feisha,yanliu.cs}@usc.edu

September 23, 2014

Outline

A few announcements

- Homework 1 successfully submitted
- Pls start working on Homework 2

Outline

Regression

Predicting a continuous outcome variable

- Predicting a company's future stock price using its pat and existing financial info
- Predicting the amount of rain fall
- Predicting ...

Regression

Predicting a continuous outcome variable

- Predicting a company's future stock price using its pat and existing financial info
- Predicting the amount of rain fall
- Predicting ...

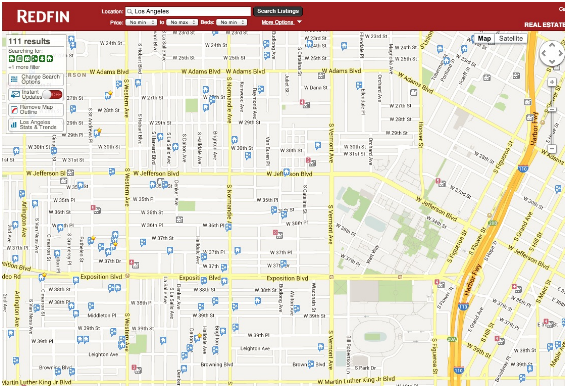
Key difference from classification

- We measure prediction errors differently.
- This will lead to quite different learning models and algorithms.

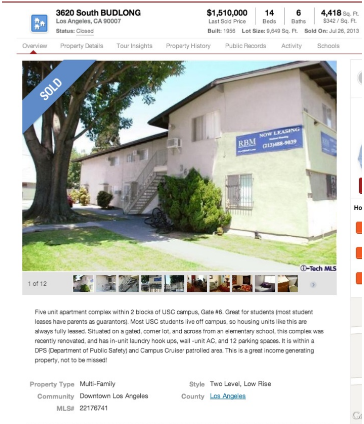
Ex: predicting the sale price of a house

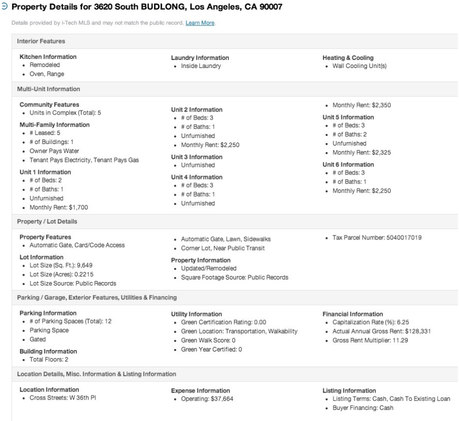
Retrieve historical sales records

(This will be our training data)



Features used to predict





Property Details for 3620 South BUDLONG, Los Angeles, CA 90007

Details provided by i-Tech MLS and may not match the public record. Learn More.

Interior Features

Kitchen Information
- Remodeled
- Oven, Range

Laundry Information
- Inside Laundry

Heating & Cooling
- Wall Cooling Unit(s)

Multi-Unit Information

Community Features
- Units in Complex (Total): 5

Multi-Family Information
- # Leased: 5
- # of Buildings: 1
- Owner Pays Water
- Tenant Pays Electricity, Tenant Pays Gas

Unit 1 Information
- # of Beds: 2
- # of Baths: 1
- Unfurnished
- Monthly Rent: $1,700

Unit 2 Information
- # of Beds: 3
- # of Baths: 1
- Unfurnished
- Monthly Rent: $2,250

Unit 3 Information
- Unfurnished

Unit 4 Information
- # of Beds: 3
- # of Baths: 1
- Unfurnished
- Monthly Rent: $2,350

Unit 5 Information
- # of Beds: 3
- # of Baths: 2
- Unfurnished
- Monthly Rent: $2,325

Unit 6 Information
- # of Beds: 3
- # of Baths: 1
- Monthly Rent: $2,250

Property / Lot Details

Property Features
- Automatic Gate, Card/Code Access
- Automatic Gate, Lawn, Sidewalks
- Corner Lot, Near Public Transit
- Tax Parcel Number: 5040017019

Lot Information
- Lot Size (Sq. Ft.): 9,649
- Lot Size (Acres): 0.2215
- Lot Size Source: Public Records

Property Information
- Updated/Remodeled
- Square Footage Source: Public Records

Parking / Garage, Exterior Features, Utilities & Financing

Parking Information
- # of Parking Spaces (Total): 12
- Parking Space
- Gated

Building Information
- Total Floors: 2

Utility Information
- Green Certification Rating: 0.00
- Green Location: Transportation, Walkability
- Green Walk Score: 0
- Green Year Certified: 0

Financial Information
- Capitalization Rate (%): 6.25
- Actual Annual Gross Rent: $128,331
- Gross Rent Multiplier: 11.29

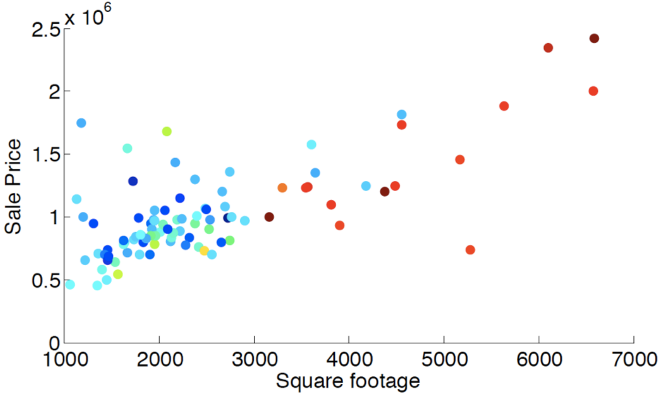
Location Details, Misc. Information & Listing Information

Location Information
- Cross Streets: W 36th Pl

Expense Information
- Operating: $37,664

Listing Information
- Listing Terms: Cash, Cash To Existing Loan
- Buyer Financing: Cash

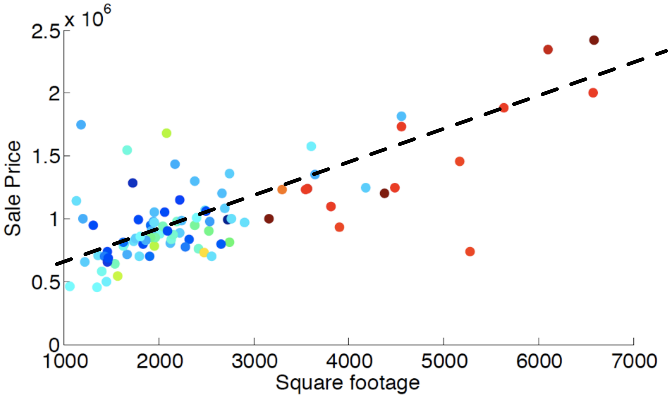
Correlation between square footage and sale price



(Unlike classification, the colors of the dots in this scatterplot do not mean anything.)

Possibly linear relationship

Sale price ≈ price_per_sqft × square_footage + fixed_expense



How to learn the unknown parameters?

training data (past sales record)

sqft	sale price
2000	800K
2100	907K
1100	312K
5500	2,600K
$\cdots$	$\cdots$

Reduce prediction error

How to measure errors?

- The classification error(*hit* or *miss*) is not appropriate for continuous outcomes.
- We can look at the *absolute* difference: | prediction - sale price|

  However, for simplicity, we look at the *squared* errors: (prediction - sale price)^2

sqft	sale price	prediction	error	squared error
2000	800K	720K	90K	8100
2100	907K	800K	107K	107^2
1100	312K	350K	38K	38^2
5500	2,600K	2,600K	0	0
...	...			

Minimize squared errors

Our model

Sale price = price_per_sqft × square_footage + fixed_expense + unexplainable_stuff

Training data

sqft	sale price	prediction	error	squared error
2000	800K	720K	90K	8100
2100	907K	800K	107K	107^2
1100	312K	350K	38K	38^2
5500	2,600K	2,600K	0	0
$\cdots$	$\cdots$			
Total				$8100 + 107^2 + 38^2 + 0 + \cdots$

Aim

Adjust price_per_sqft and fixed_expense such that the sum of the squared error is minimized — i.e., the residual/remaining unexplainable_stuff is minimized.

Linear regression

Setup

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$
- Model: $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

  $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ is called *weights*, *parameters*, or *parameter vector*

  w_0 is called *bias*.

  We also sometimes call $\tilde{\boldsymbol{w}} = [w_0 \; w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ parameters too! *So please pay attention to contexts when you read papers, textbooks, or assigned reading material.*

Goal

Minimize prediction error as much as possible

- Residual sum of squares

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - f(\boldsymbol{x}_n)]^2 = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2$$

- Other definitions of errors are also possible
  We will see an example very soon.

A simple case: x is just one-dimensional

Identify stationary points, by taking derivative with respect to parameters, and setting to zeroes

$$\left\{ \begin{array}{l} \frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \\ \frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \end{array} \right. \Rightarrow \left(\begin{array}{cc} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{array} \right) \left(\begin{array}{c} w_0 \\ w_1 \end{array} \right) = \left(\begin{array}{c} \sum_n y_n \\ \sum_n x_n y_n \end{array} \right)$$

Why minimizing RSS is a sensible thing?

Probabilistic interpretation

- Noisy observation model

$$Y = w_0 + w_1 X + \eta$$

where $\eta \sim N(0, \sigma^2)$ is a Gaussian random variable

- Likelihood of one training sample (x_n, y_n)

$$p(y_n|x_n) = N(w_0 + w_1 x, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y_n - (w_0 + w_1 x_n)]^2}{2\sigma^2}}$$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$\log P(\mathcal{D}) = \log \prod_{n=1}^{\mathsf{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n)$$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$\log P(\mathcal{D}) = \log \prod_{n=1}^{\mathsf{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n)$$

$$= \sum_n \left\{ -\frac{[y_n - (w_0 + w_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\}$$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$
\begin{aligned}
\log P(\mathcal{D}) &= \log \prod_{n=1}^{\mathsf{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n) \\
&= \sum_n \left\{ -\frac{[y_n - (w_0 + w_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\} \\
&= -\frac{1}{2\sigma^2} \sum_n [y_n - (w_0 + w_1 x_n)]^2 - \frac{\mathsf{N}}{2} \log \sigma^2 - \mathsf{N} \log \sqrt{2\pi}
\end{aligned}
$$

Probabilistic interpretation (cont'd)

Log-likelihood of the training data $\mathcal{D}$ (assuming i.i.d)

$$\begin{aligned}
\log P(\mathcal{D}) &= \log \prod_{n=1}^{\mathsf{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n) \\
&= \sum_n \left\{ -\frac{[y_n - (w_0 + w_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\} \\
&= -\frac{1}{2\sigma^2} \sum_n [y_n - (w_0 + w_1 x_n)]^2 - \frac{\mathsf{N}}{2} \log \sigma^2 - \mathsf{N} \log \sqrt{2\pi} \\
&= -\frac{1}{2} \left\{ \frac{1}{\sigma^2} \sum_n [y_n - (w_0 + w_1 x_n)]^2 + \mathsf{N} \log \sigma^2 \right\} + \text{const}
\end{aligned}$$

i.i.d stands for independently and identically distributed.

Maximum likelihood estimation

Estimating σ, w_0 and w_1 are decoupled

- Maximize over w_0 and w_1

$$\max \ \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (w_0 + w_1 x_n)]^2 \leftarrow \text{That is RSS}(\tilde{\boldsymbol{w}})!$$

Maximum likelihood estimation

Estimating σ, w_0 and w_1 are decoupled

- Maximize over w_0 and w_1

$$\max \ \log P(\mathcal{D}) \Leftrightarrow \min \ \sum_n [y_n - (w_0 + w_1 x_n)]^2 \leftarrow \text{That is RSS}(\tilde{\boldsymbol{w}})!$$

- Maximize over $s = \sigma^2$ (we could estimate σ directly)

$$\frac{\partial \log P(\mathcal{D})}{\partial s} = -\frac{1}{2}\left\{-\frac{1}{s^2}\sum_n [y_n - (w_0 + w_1 x_n)]^2 + \mathsf{N}\frac{1}{s}\right\} = 0$$

Maximum likelihood estimation

Estimating σ, w_0 and w_1 are decoupled

- Maximize over w_0 and w_1

$$\max \ \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (w_0 + w_1 x_n)]^2 \leftarrow \text{That is RSS}(\tilde{\boldsymbol{w}})!$$

- Maximize over $s = \sigma^2$ (we could estimate σ directly)

$$\frac{\partial \log P(\mathcal{D})}{\partial s} = -\frac{1}{2}\left\{-\frac{1}{s^2}\sum_n [y_n - (w_0 + w_1 x_n)]^2 + \mathsf{N}\frac{1}{s}\right\} = 0$$

$$\rightarrow \sigma^2 = s = \frac{1}{\mathsf{N}}\sum_n [y_n - (w_0 + w_1 x_n)]^2$$

Interpretation

Least mean square (LMS) solution (minimizing residual sum of errors)

$$\begin{pmatrix} w_0^{LMS} \\ w_1^{LMS} \end{pmatrix} = \begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_n y_n \\ \sum_n x_n y_n \end{pmatrix}$$

Interpretation

Least mean square (LMS) solution (minimizing residual sum of errors)

$$\begin{pmatrix} w_0^{LMS} \\ w_1^{LMS} \end{pmatrix} = \begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_n y_n \\ \sum_n x_n y_n \end{pmatrix}$$

Additionally

$$\sigma^2 = \frac{1}{\mathsf{N}} \sum_n [y_n - (w_0^{LMS} + w_1^{LMS} x_n)]^2$$

Interpretation

Least mean square (LMS) solution (minimizing residual sum of errors)

$$\begin{pmatrix} w_0^{LMS} \\ w_1^{LMS} \end{pmatrix} = \begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_n y_n \\ \sum_n x_n y_n \end{pmatrix}$$

Additionally

$$\sigma^2 = \frac{1}{\mathsf{N}} \sum_n [y_n - (w_0^{LMS} + w_1^{LMS} x_n)]^2$$

Remarks

- LMS is the same as the maximum likelihood estimation when the noise is assumed to be *Gaussian*.
- The remaining residuals provide a maximum likelihood estimate of the noise's *variance.*

NB. We sometimes call it least square solutions too.

Solution when x is one-dimensional

Least mean square (LMS) solution (minimizing residual sum of errors)

$$\begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} \sum_n y_n \\ \sum_n x_n y_n \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} w_0^{LMS} \\ w_1^{LMS} \end{pmatrix} = \begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_n y_n \\ \sum_n x_n y_n \end{pmatrix}$$

NB. We sometimes call it least square solutions too.

LMS when $\boldsymbol{x}$ is D-dimensional

$RSS(\tilde{w})$ in matrix form

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\boldsymbol{x}} \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_{\mathsf{D}}]^{\mathrm{T}}, \quad \tilde{\boldsymbol{w}} \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_{\mathsf{D}}]^{\mathrm{T}}$$

LMS when $\boldsymbol{x}$ is D-dimensional

$RSS(\tilde{\boldsymbol{w}})$ in matrix form

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\boldsymbol{x}} \leftarrow [1 \; x_1 \; x_2 \; \ldots \; x_{\mathsf{D}}]^{\mathrm{T}}, \quad \tilde{\boldsymbol{w}} \leftarrow [w_0 \; w_1 \; w_2 \; \ldots \; w_{\mathsf{D}}]^{\mathrm{T}}$$

which leads to

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n (y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n)(y_n - \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}})$$

LMS when $\boldsymbol{x}$ is D-dimensional

$RSS(\tilde{\boldsymbol{w}})$ in matrix form

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\boldsymbol{x}} \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_{\mathsf{D}}]^{\mathrm{T}}, \quad \tilde{\boldsymbol{w}} \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_{\mathsf{D}}]^{\mathrm{T}}$$

which leads to

$$\begin{aligned} RSS(\tilde{\boldsymbol{w}}) &= \sum_n (y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n)(y_n - \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}}) \\ &= \sum_n \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - 2 y_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} + \text{const.} \end{aligned}$$

LMS when $\boldsymbol{x}$ is D-dimensional

$RSS(\tilde{\boldsymbol{w}})$ in matrix form

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\boldsymbol{x}} \leftarrow [1 \ x_1 \ x_2 \ \ldots \ x_{\mathsf{D}}]^{\mathrm{T}}, \quad \tilde{\boldsymbol{w}} \leftarrow [w_0 \ w_1 \ w_2 \ \ldots \ w_{\mathsf{D}}]^{\mathrm{T}}$$

which leads to

$$\begin{aligned} RSS(\tilde{\boldsymbol{w}}) &= \sum_n (y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n)(y_n - \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}}) \\ &= \sum_n \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - 2 y_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} + \text{const.} \\ &= \left\{ \tilde{\boldsymbol{w}}^{\mathrm{T}} \left(\sum_n \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} - 2 \left(\sum_n y_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} \right\} + \text{const.} \end{aligned}$$

$RSS(\tilde{\boldsymbol{w}})$ in new notations

Design matrix and target vector

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{\mathsf{N} \times D}, \quad \tilde{\boldsymbol{X}} = \begin{pmatrix} \mathbf{1} & \boldsymbol{X} \end{pmatrix} \in \mathbb{R}^{\mathsf{N} \times (D+1)}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

$RSS(\tilde{\boldsymbol{w}})$ in new notations

Design matrix and target vector

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{\mathsf{N}\times D}, \quad \tilde{\boldsymbol{X}} = (\boldsymbol{1} \quad \boldsymbol{X}) \in \mathbb{R}^{\mathsf{N}\times(D+1)}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

Compact expression

$$RSS(\tilde{\boldsymbol{w}}) = \left\{ \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - 2 \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} \right)^{\mathrm{T}} \tilde{\boldsymbol{w}} \right\} + \text{const}$$

Solution in matrix form

Normal equation

Take derivative with respect to $\tilde{\boldsymbol{w}}$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial \tilde{\boldsymbol{w}}} \propto \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\boldsymbol{w} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = 0$$

This leads to the least-mean-square (LMS) solution

$$\tilde{\boldsymbol{w}}^{LMS} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

Solution in matrix form

Normal equation

Take derivative with respect to $\tilde{\boldsymbol{w}}$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial \tilde{\boldsymbol{w}}} \propto \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\boldsymbol{w} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = 0$$

This leads to the least-mean-square (LMS) solution

$$\tilde{\boldsymbol{w}}^{LMS} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

Verify the solution when $\mathsf{D} = 1$

$$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_{\mathsf{N}} \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdots & \cdots \\ 1 & x_{\mathsf{N}} \end{pmatrix} = \begin{pmatrix} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}$$

Mini-Summary

- Linear regression is the linear combination of features.
  $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
- If we minimize residual sum squares as our learning objective, we get a closed-form solution of parameters.
- Probabilistic interpretation: maximum likelihood if assuming residual is Gaussian distributed
- Other interpretations exist: if interested, please consult the slides from last year's lectures.

Computational complexity

Bottleneck of computing the solution

$$\boldsymbol{w} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}\boldsymbol{y}$$

is to invert the matrix $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} \in \mathbb{R}^{(\mathsf{D}+1)\times(\mathsf{D}+1)}$

How many operations do we need?

- On the order of $O((\mathsf{D}+1)^3)$ (using Gauss-Jordan elimination) or $O((\mathsf{D}+1)^{2.373})$ (recent advances in computing)
- Impractical for very large D

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. Compute the gradient (ignoring the constant factor)
     $\nabla RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}}^{(t)} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. Compute the gradient (ignoring the constant factor) $\nabla RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}}^{(t)} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$
  2. Update the parameters $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta\nabla RSS(\tilde{\boldsymbol{w}})$

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
    1. Compute the gradient (ignoring the constant factor) $\nabla RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}}^{(t)} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$
    2. Update the parameters $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta\nabla RSS(\tilde{\boldsymbol{w}})$
    3. $t \leftarrow t + 1$

Alternative method: an example of using numerical optimization

(Batch) Gradient descent

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. Compute the gradient (ignoring the constant factor) $\nabla RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}}^{(t)} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$
  2. Update the parameters $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta\nabla RSS(\tilde{\boldsymbol{w}})$
  3. $t \leftarrow t + 1$

What is the complexity here?

Why would this work?

If gradient descent converges, it will converge to the same solution using matrix inversion.

This is because $RSS(\tilde{\boldsymbol{w}})$ is a convex function in its parameters $\boldsymbol{w}$

$$RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - 2\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\tilde{\boldsymbol{w}}$$

Why would this work?

If gradient descent converges, it will converge to the same solution using matrix inversion.

This is because $RSS(\tilde{\boldsymbol{w}})$ is a convex function in its parameters $\boldsymbol{w}$

$$RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - 2\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\tilde{\boldsymbol{w}}$$

$$\Rightarrow \frac{\partial^2 RSS(\tilde{\boldsymbol{w}})}{\partial\tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^{\mathrm{T}}} = 2\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$$

Why would this work?

If gradient descent converges, it will converge to the same solution using matrix inversion.

This is because $RSS(\tilde{\boldsymbol{w}})$ is a convex function in its parameters $\boldsymbol{w}$

$$RSS(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - 2\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\tilde{\boldsymbol{w}}$$

$$\Rightarrow \frac{\partial^2 RSS(\tilde{\boldsymbol{w}})}{\partial\tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^{\mathrm{T}}} = 2\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$$

as $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is positive semidefinite, because for any $\boldsymbol{v}$

$$\boldsymbol{v}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\boldsymbol{v} = \|\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{v}\|_2^2 \geq 0$$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

$$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

$$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$

  3. Update the parameters
     $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta \boldsymbol{g}_t$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)

$$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$

  3. Update the parameters
     $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta \boldsymbol{g}_t$
  4. $t \leftarrow t + 1$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient (ignoring the constant factor)
  $$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$
  3. Update the parameters
  $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta \boldsymbol{g}_t$
  4. $t \leftarrow t + 1$

What is the complexity here?

Mini-summary

- Batch gradient descent computes the exact gradient.
- Stochastic gradient descent computes the gradient pretending only one instance.
  Its expectation equals to the true gradient.
- Other forms can be used.
  Mini-batch: trade-off between accuracy of estimating gradient and computational cost
- Similar ideas extend to other optimization problems in machine learning.
  For large-scale problems, stochastic gradient descent often works well.

What if $\tilde{X}^{\mathrm{T}}\tilde{X}$ is not invertible

Can you think of any reasons why that could happen?

What if $\tilde{X}^{\mathrm{T}}\tilde{X}$ is not invertible

Can you think of any reasons why that could happen?

Answer 1: $\mathrm{N} < \mathrm{D}$. Intuitively, not enough data to estimate all the parameters.

What if $\tilde{X}^{\mathrm{T}}\tilde{X}$ is not invertible

Can you think of any reasons why that could happen?

Answer 1: $\mathrm{N} < \mathrm{D}$. Intuitively, not enough data to estimate all the parameters.

Answer 2: $\boldsymbol{X}$ columns are not linearly independent. Intuitively, there are two features that are perfectly correlated. In this case, solution is not unique.

Ridge regression

Intuition: what does a non-invertible $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ mean?

$$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \lambda_r & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \boldsymbol{U}$$

where $\lambda 1 \geq \lambda_2 \geq \cdots \lambda_r > 0$ and $r <$ D.

Ridge regression

Intuition: what does a non-invertible $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ mean?

$$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \lambda_r & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \boldsymbol{U}$$

where $\lambda 1 \geq \lambda_2 \geq \cdots \lambda_r > 0$ and $r <$ D.

Fix the problem by adding something positive

$$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda \boldsymbol{I} = \boldsymbol{U}^{\mathrm{T}}\text{diag}(\lambda_1 + \lambda, \lambda_2 + \lambda, \cdots, \lambda)\boldsymbol{U}$$

where $\lambda > 0$ and $\boldsymbol{I}$ is the identity matrix

Regularized least square (ridge regression)

Solution

$$\tilde{w} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda\boldsymbol{I}\right)^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

Regularized least square (ridge regression)

Solution

$$\tilde{\boldsymbol{w}} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda \boldsymbol{I}\right)^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

This is equivalent to adding an extra term to $RSS(\tilde{\boldsymbol{w}})$

$$\overbrace{\frac{1}{2}\left\{\tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - 2\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\tilde{\boldsymbol{w}}\right\}}^{RSS(\tilde{\boldsymbol{w}})} + \underbrace{\frac{1}{2}\lambda\|\tilde{w}\|_2^2}_{\text{regularization}}$$

Regularized least square (ridge regression)

Solution

$$\tilde{\boldsymbol{w}} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda \boldsymbol{I}\right)^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

This is equivalent to adding an extra term to $RSS(\tilde{\boldsymbol{w}})$

$$\overbrace{\frac{1}{2}\left\{\tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - 2\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\tilde{\boldsymbol{w}}\right\}}^{RSS(\tilde{\boldsymbol{w}})} + \underbrace{\frac{1}{2}\lambda\|\tilde{w}\|_2^2}_{\text{regularization}}$$

Benefits

- Numerically more stable, invertible matrix
- Prevent overfitting — more on this later

How to choose λ?

Again, λ is referred as *hyperparameter*, to be distinguished from $\boldsymbol{w}$.

- Use validation or cross-validation
- Other approaches such as Bayesian linear regression — we will describe them briefly later

linear regression versus nearest neighbors

Parametric versus non-parametric

- Parametric
  The size of the model does not grow with respect to the size of the training dataset.
  In linear regression, there are $\mathsf{D}+1$ parameters, irrelevant to how many training instances we have.
- Non-parametric
  The size of the model grows with respect to the size of the training dataset.
  In nearest neighbor classification, the training dataset itself needs to be kept in order to make prediction. Thus, the size of the model is the size of the training dataset.

Non-parametric does not mean *parameter-less*. It just means the number of parameters is a function of the training dataset.