CSCI567 Machine Learning (Spring 2018)

Michael Shindler

Lecture on January 17, 2018

Outline

Outline

Administrative stuff

- If you have not already completed the syllabus quiz and git survey, do so soon.

Outline

Multi-class classification

Classify data into one of the multiple categories

- Input (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Output (label): $y \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$
- Learning goal: $y = f(\boldsymbol{x})$

Special case: binary classification

- Number of classes: $\mathsf{C} = 2$
- Labels: $\{0, 1\}$ or $\{-1, +1\}$

Tuning hyperparameter/Model Selection by using a validation dataset

Training data (set)

- N samples/instances: $\mathcal{D}^{\mathrm{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$
- They are used for learning $f(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\mathrm{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{M}}, y_{\mathsf{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\mathrm{TRAIN}}$

Development (or validation) data

- L samples/instances: $\mathcal{D}^{\mathrm{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{L}}, y_{\mathsf{L}})\}$
- They are used to optimize hyperparameter(s).

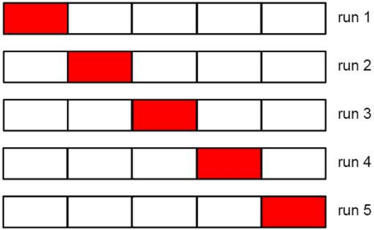
Training data, validation and test data should *not* overlap!

Cross-validation

What if we do not have validation data?

- We split the training data into S equal parts.
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that *on average*, the model performing the best

S $= 5$: 5-fold cross validation



Special case: when S = N, this will be leave-one-out.

Outline

Regression

Predicting a continuous outcome variable

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

Regression

Predicting a continuous outcome variable

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

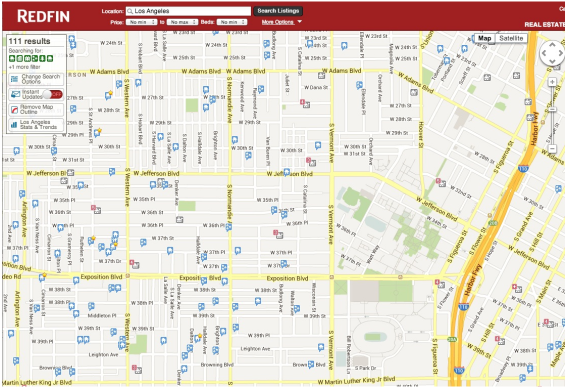
Key difference from classification

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

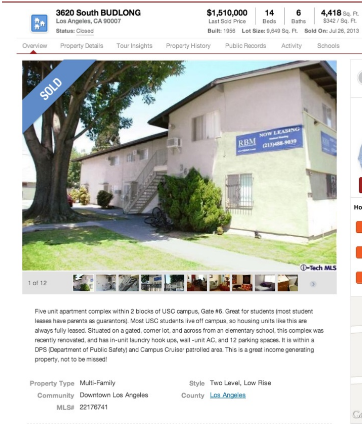
Ex: be a savvy purchaser by predicting the sale price of a house

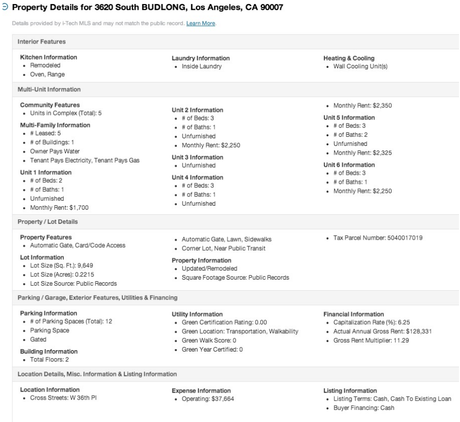
Retrieve historical sales records

(This is 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

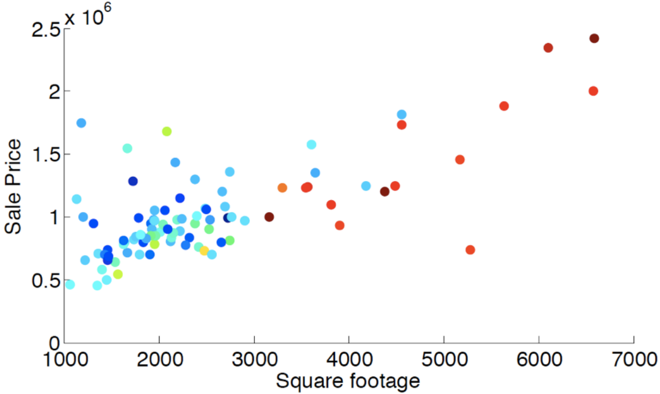
Location Details, Misc. Information & Listing Information

Location Information
- Cross Streets: W 36th Pl

Expense Information
- Operating: $37,664

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

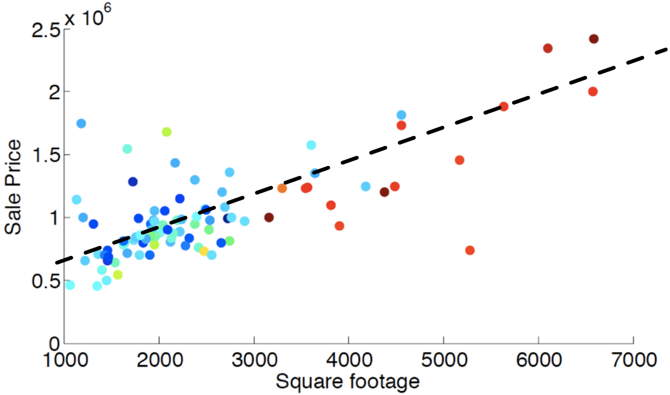
Correlation between square footage and sale price



(Unlike the Fisher's flower classification example, the colors of the dots in this scatterplot do not mean anything.)

Possibly linear relationship

Sale price $\approx$ price_per_sqft $\times$ 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
...	...

Reduce prediction error

How to measure errors?

- The classification error (got it *right* or *wrong*) 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	810K	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	810K	720K	90K	8100
2100	907K	800K	107K	107^2
1100	312K	350K	38K	38^2
5500	2,600K	2,600K	0	0
...	...			
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: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$
  We will use x_{nd} representing the dth dimension of the nth sample $\boldsymbol{x}_n$
- Model: $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$, with ^{T} standing for vector transpose.
  $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ is called *weights*, *parameters*, or *parameter vector*. w_0 is called *bias*.

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

Goal

Minimize prediction error as much as possible

- Residual Sum of Squares (RSS)

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

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

A simple case: x is just one-dimensional

Our errors are

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - f(\boldsymbol{x}_n)]^2 = \sum_n [y_n - (w_0 + w_1 x_n)]^2$$

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

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

Derivation

$$RSS(w_0, w_1) = \sum_n [y_n - (w_0 + w_1 x_n)]^2$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_0} =$$

Solution when x is one-dimensional

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

$$\left(\begin{array}{cc} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{array}\right)\left(\begin{array}{c} w_0 \\ w_1 \end{array}\right) = \left(\begin{array}{c} \sum_n y_n \\ \sum_n x_n y_n \end{array}\right)$$

$$\rightarrow \left(\begin{array}{c} w_0^{LMS} \\ w_1^{LMS} \end{array}\right) = \left(\begin{array}{cc} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{array}\right)^{-1}\left(\begin{array}{c} \sum_n y_n \\ \sum_n x_n y_n \end{array}\right)$$

NB. We sometimes call it least square solutions (LSE) 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

$RSS(\tilde{w})$ 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

$RSS(\tilde{w})$ 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

$RSS(\tilde{w})$ 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

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

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

Design matrix and target vector

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

Compact expression

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

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 $\mathsf{D} = 1$

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

For those who are familiar with this step, you can look up the formula in The Matrix Cookbook

Mini-Summary

- Linear regression is the *linear combination of features*.
  $f : \boldsymbol{x} \to y$, 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*.

Computational complexity

Bottleneck of computing the solution

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

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?

- Roughly, on the order of $O((\mathsf{D}+1)^3)$
- Impractical for very large D
  We will look at some ideas of addressing this issue later

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

Can you think of any reasons why that could happen?

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

Can you think of any reasons why that could happen?

$\mathsf{N} < \mathsf{D} + 1$. **Intuitively, not enough data to estimate all the parameters.**

$\mathsf{D} + 1$ unknown but we have only N training samples

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

Can you think of any reasons why that could happen?

$\mathsf{N} < \mathsf{D} + 1$. **Intuitively, not enough data to estimate all the parameters.**

$\mathsf{D} + 1$ unknown but we have only N training samples
Example: $\mathsf{D} = 1, N = 0,$ **or** 1, ie, the following "empty" training dataset

sqft	sale price	prediction	error	squared error
1000	2000			

How about the following?

$\mathsf{D} = 1, \mathsf{N} = 2$

sqft	sale price	prediction	error	squared error
1000	2000			
1000	2000			

How about the following?

$\mathsf{D} = 1, \mathsf{N} = 2$

sqft	sale price	prediction	error	squared error
1000	2000			
1000	2000			

We still cannot determine the model (uniquely), even now $\mathsf{N} \geq \mathsf{D} + 1$:

- Sale price = sqft x 2
- Sale price = sqft x 1 + 1000
- ...

Namely, we need *informative* training data.

Challenge

Can you summarize those bad scenarios, as illustrated before, with more concise statements about the relationship between training data and the unknown?

This will be left as an exercise to the student.

How to solve this problem?

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 $r < \mathsf{D} + 1$. $\boldsymbol{U}$ is unitary matrix.

Discussion section will talk a bit more about this: this linear algebra step is called eigendecomposition.

How to solve this problem?

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

$$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1^{-1} & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2^{-1} & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \lambda_r^{-1} & 0 \\ 0 & \cdots & \cdots & 0 & \frac{1}{0} \end{bmatrix} \boldsymbol{U}$$

where $\frac{1}{0}$ is the issue.

Discussion section will talk a bit more about this: this linear algebra step is called eigendecomposition.

Fix the problem

Adding something positive

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

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

Later, we will justify why this is a sensible thing for us to do

Fix the problem

Now we can invert

$$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}+\lambda\boldsymbol{I})^{-1}=\boldsymbol{U}^{\mathrm{T}}\begin{bmatrix}(\lambda_1+\lambda)^{-1} & 0 & 0 & \cdots & 0\\ 0 & (\lambda_2+\lambda)^{-1} & 0 & \cdots & 0\\ 0 & \cdots & \cdots & \cdots & 0\\ 0 & \cdots & \cdots & (\lambda_r+\lambda)^{-1} & 0\\ 0 & \cdots & \cdots & 0 & \frac{1}{\lambda}\end{bmatrix}\boldsymbol{U}$$

and the solution is

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

Note that this solution is not the LMS solution to the original problem where the matrix $\tilde{\boldsymbol{X}}^{T}\tilde{\boldsymbol{X}}$ is not invertible.

How to choose λ?

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

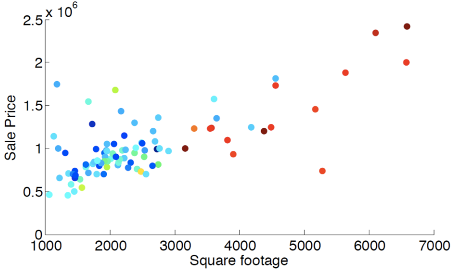
- Use validation or cross-validation
- Other approaches such as Bayesian linear regression — we will describe them briefly later if we have time

Brain teaser for Linear Regression

What if $\mathrm{D} = 0$, ie, not using any predictor/features?

What the model looks like when D $= 0$

Sale price = fixed_expense + unexplainable_stuff, namely $f(x) = w_0$



So this is a horizontal line...But where this line should be vertically (ie, what is w_0)?

Intuition: the average of the all the sale prices in the training data

From $\mathsf{D} = 1$

$$\left(\begin{array}{cc} \sum_n 1 & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{array}\right)\left(\begin{array}{c} w_0 \\ w_1 \end{array}\right) = \left(\begin{array}{c} \sum_n y_n \\ \sum_n x_n y_n \end{array}\right)$$

to $\mathsf{D} = 0$

$$\sum_n 1 \times w_0 = \sum_n y_n \rightarrow w_0 = \frac{1}{N}\sum_n y_n$$

In other words, when we say "The housing in City A is more expensive than it in City B", we are just referring to comparing the bias terms w_0, ie, the average price in each city, without taking into consideration other features of each individual property.

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 N.
  In linear regression, there are $\mathrm{D}+1$ 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.