CSCI567 Machine Learning (Spring 2021)

Sirisha Rambhatla

University of Southern California

Jan 22, 2021

Outline

Outline

Logistics

- HW 0 is due today.
- HW 1 will be released today.
- Will be releasing the schedule of lectures.

Outline

Multi-class classification

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$
- Each $\boldsymbol{x_n} \in \mathbb{R}^{\mathsf{D}}$ is called a feature vector.
- Each $y_n \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$ is called a label/class/category.
- They are used to learn $f : \mathbb{R}^{\mathsf{D}} \to [\mathsf{C}]$ for future prediction.

Special case: binary classification

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

K-NNC: predict the majority label within the K-nearest neighbor set

Datasets

Training data

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

Test data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{M}}, y_{\mathsf{M}})\}$
- They are used to evaluate how well $f(\cdot)$ will do.

Development/Validation data

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

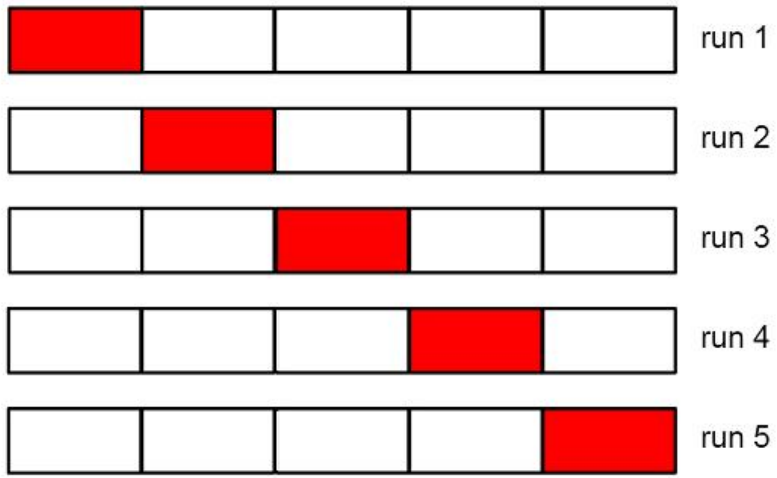
These three sets should *not* overlap!

S-fold Cross-validation

What if we do not have a development set?

- Split the training data into S equal parts.
- Use each part *in turn* as a development dataset and use the others as a training dataset.
- Choose the hyper-parameter leading to best *average* performance.

S = 5: 5-fold cross validation



Special case: S = N, called leave-one-out.

High level picture

Typical steps of developing a machine learning system:

- Collect data, split into training, development, and test sets.
- *Train a model with a machine learning algorithm.* Most often we apply cross-validation to tune hyper-parameters.
- Evaluate using the test data and report performance.
- Use the model to predict future/make decisions.

High level picture

Typical steps of developing a machine learning system:

- Collect data, split into training, development, and test sets.
- *Train a model with a machine learning algorithm.* Most often we apply cross-validation to tune hyper-parameters.
- Evaluate using the test data and report performance.
- Use the model to predict future/make decisions.

How to do the *red part* exactly?

Outline

Regression

Predicting a continuous outcome variable using past observations

- Predicting future temperature (lecture 1)
- Predicting the amount of rainfall
- Predicting the demand of a product
- Predicting the sale price of a house
- ...

Regression

Predicting a continuous outcome variable using past observations

- Predicting future temperature (lecture 1)
- Predicting the amount of rainfall
- Predicting the demand of a product
- Predicting the sale price of a house
- ...

Key difference from classification

- continuous vs discrete
- measure *prediction errors* differently.
- lead to quite different learning algorithms.

Regression

Predicting a continuous outcome variable using past observations

- Predicting future temperature (lecture 1)
- Predicting the amount of rainfall
- Predicting the demand of a product
- Predicting the sale price of a house
- ...

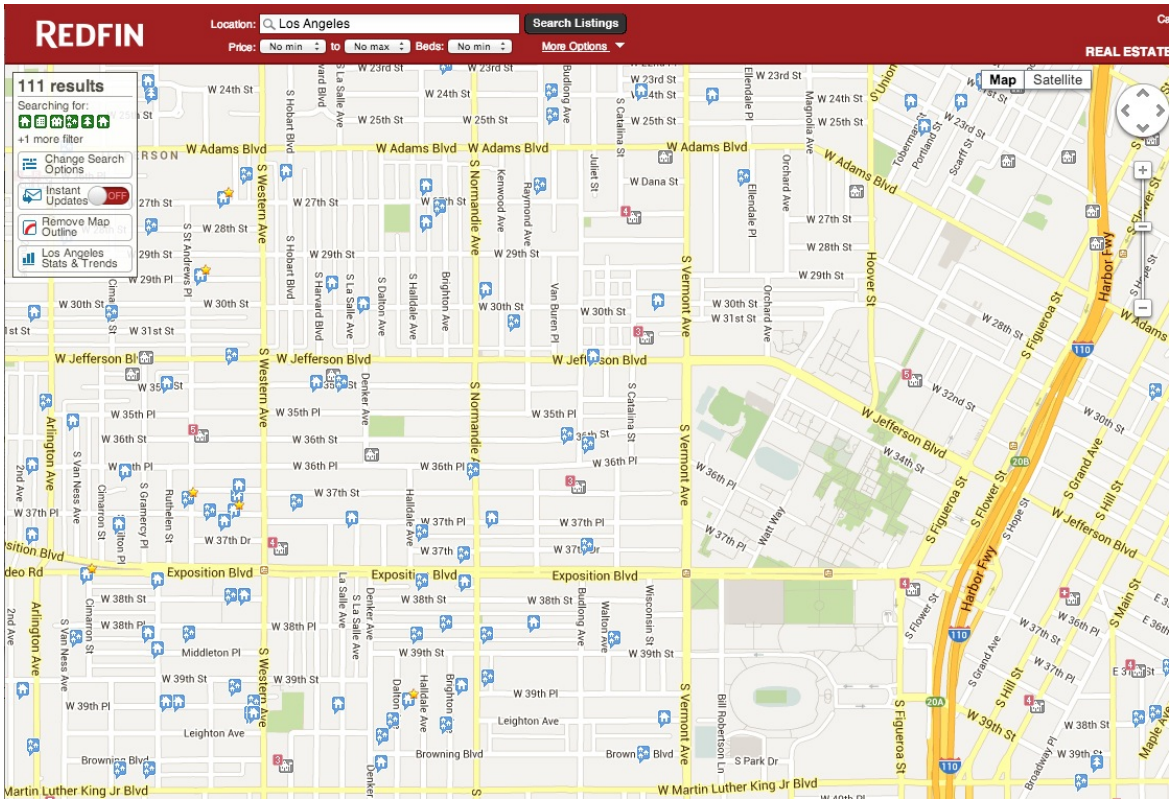
Key difference from classification

- continuous vs discrete
- measure *prediction errors* differently.
- lead to quite different learning algorithms.

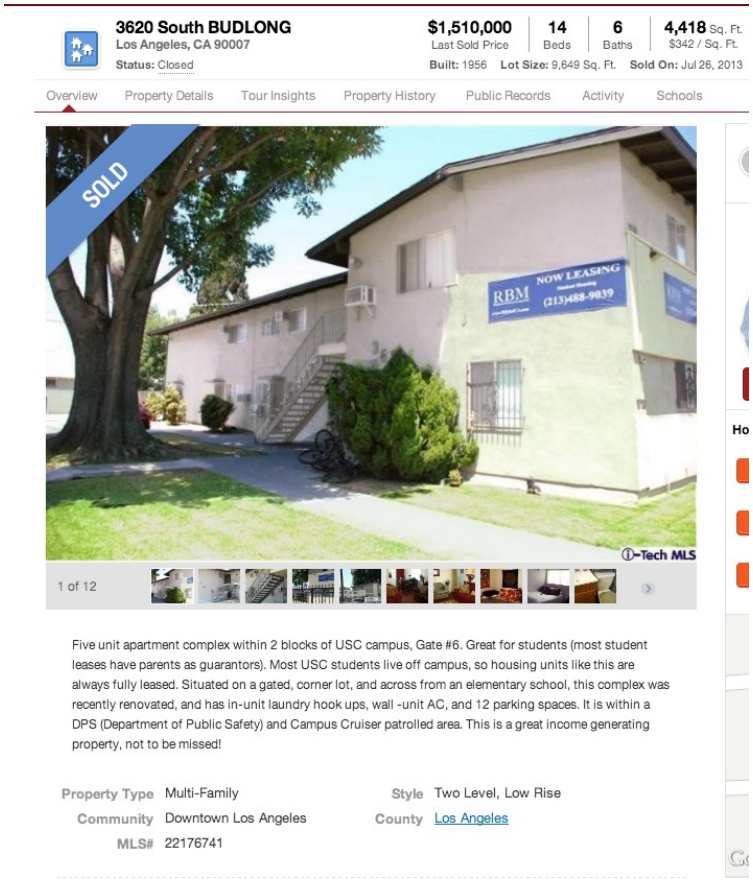
Linear Regression: regression with linear models

Ex: Predicting the sale price of a house

Retrieve historical sales records (training data)



Features used to predict



3620 South BUDLONG
Los Angeles, CA 90007
Status: Closed

$1,510,000 Last Sold Price | 14 Beds | 6 Baths | 4,418 Sq. Ft. $342 / Sq. Ft.
Built: 1956 Lot Size: 9,649 Sq. Ft. Sold On: Jul 26, 2013

Overview | Property Details | Tour Insights | Property History | Public Records | Activity | Schools

1 of 12

Five unit apartment complex within 2 blocks of USC campus, Gate #6. Great for students (most student leases have parents as guarantors). Most USC students live off campus, so housing units like this are always fully leased. Situated on a gated, corner lot, and across from an elementary school, this complex was recently renovated, and has in-unit laundry hook ups, wall -unit AC, and 12 parking spaces. It is within a DPS (Department of Public Safety) and Campus Cruiser patrolled area. This is a great income generating property, not to be missed!

Property Type	Multi-Family	Style	Two Level, Low Rise
Community	Downtown Los Angeles	County	Los Angeles
MLS#	22176741		

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

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

- Automatic Gate, Lawn, Sidewalks
- Corner Lot, Near Public Transit

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

- Tax Parcel Number: 5040017019

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

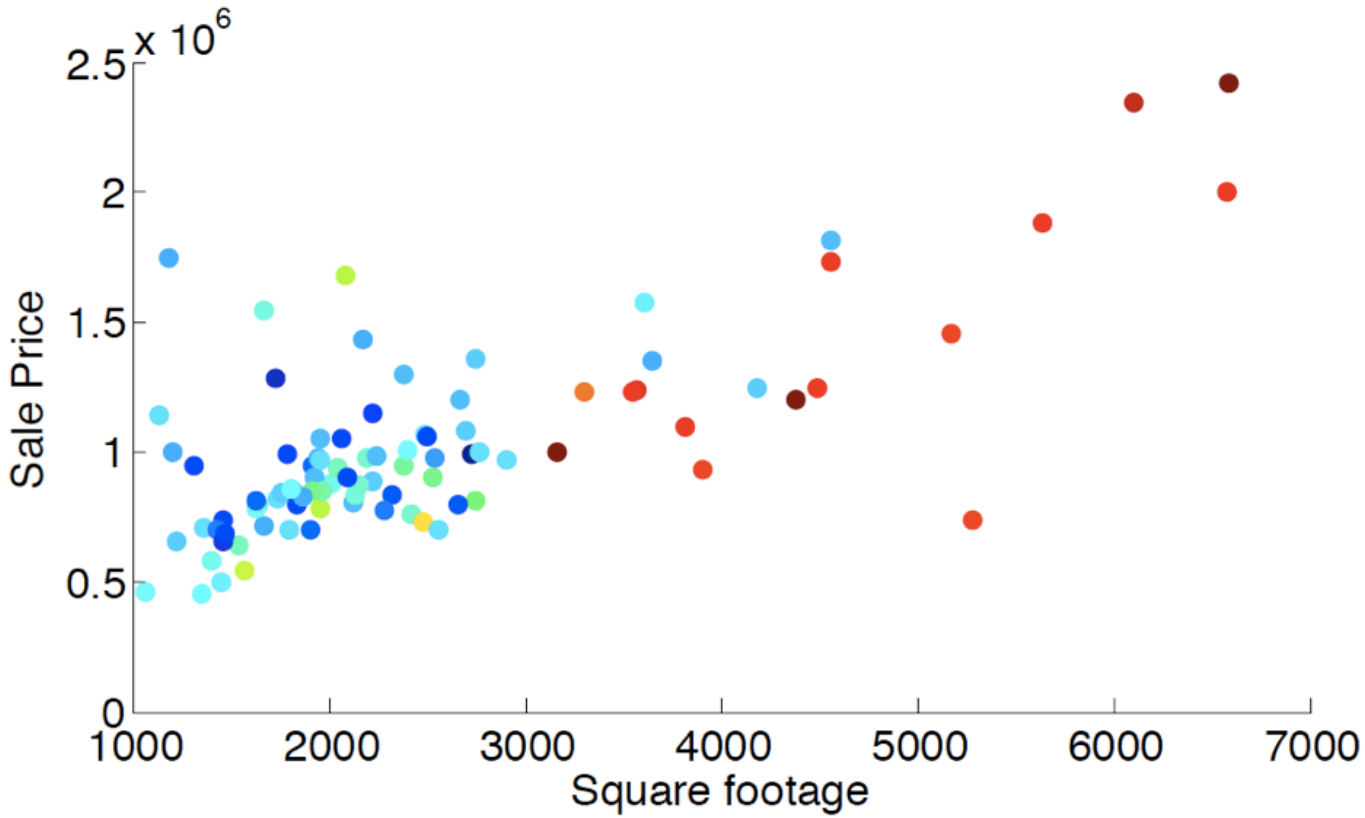
Location Details, Misc. Information & Listing Information

Location Information
- Cross Streets: W 36th Pl

Expense Information
- Operating: $37,664

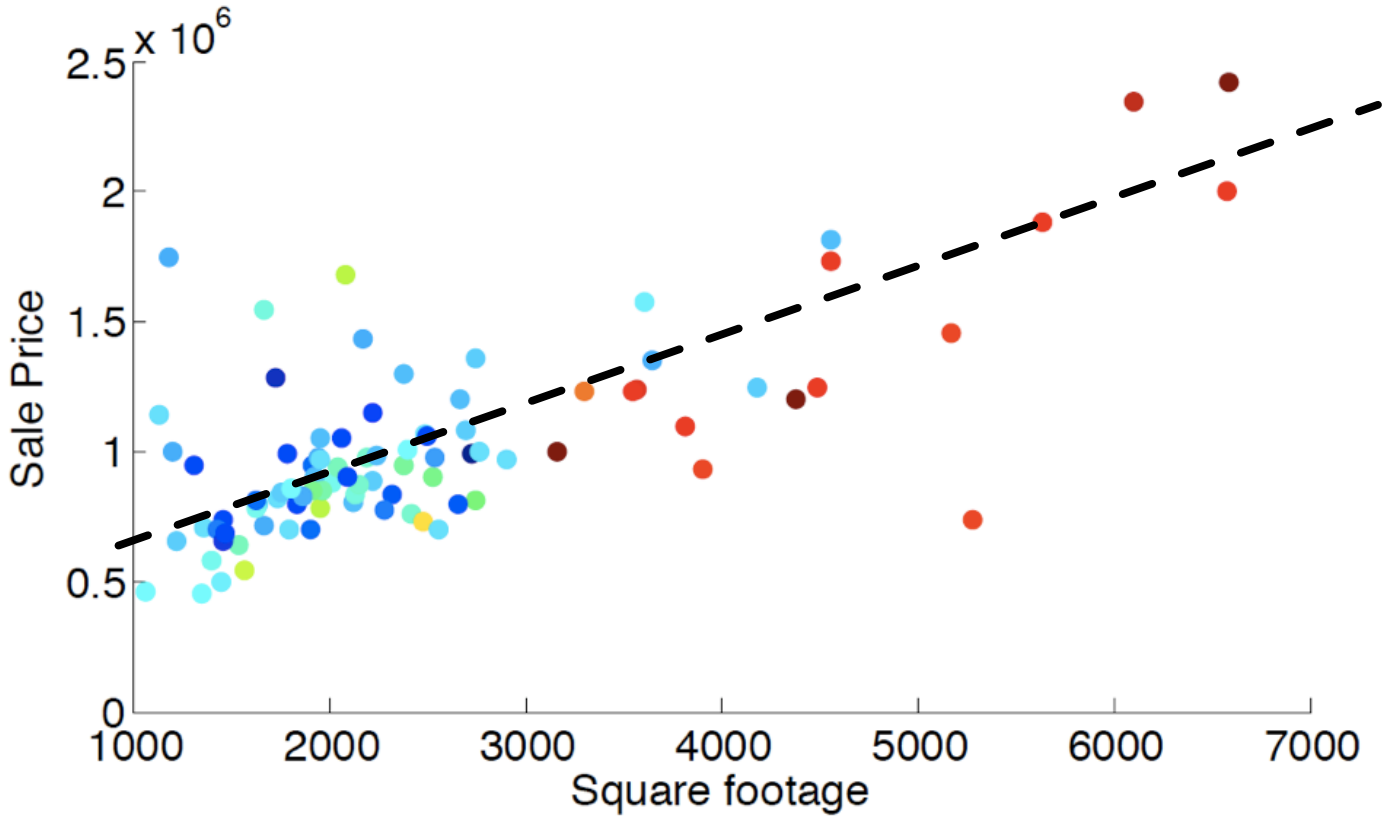
Listing Information
- Listing Terms: Cash, Cash To Existing Loan
- Buyer Financing: Cash

Correlation between square footage and sale price



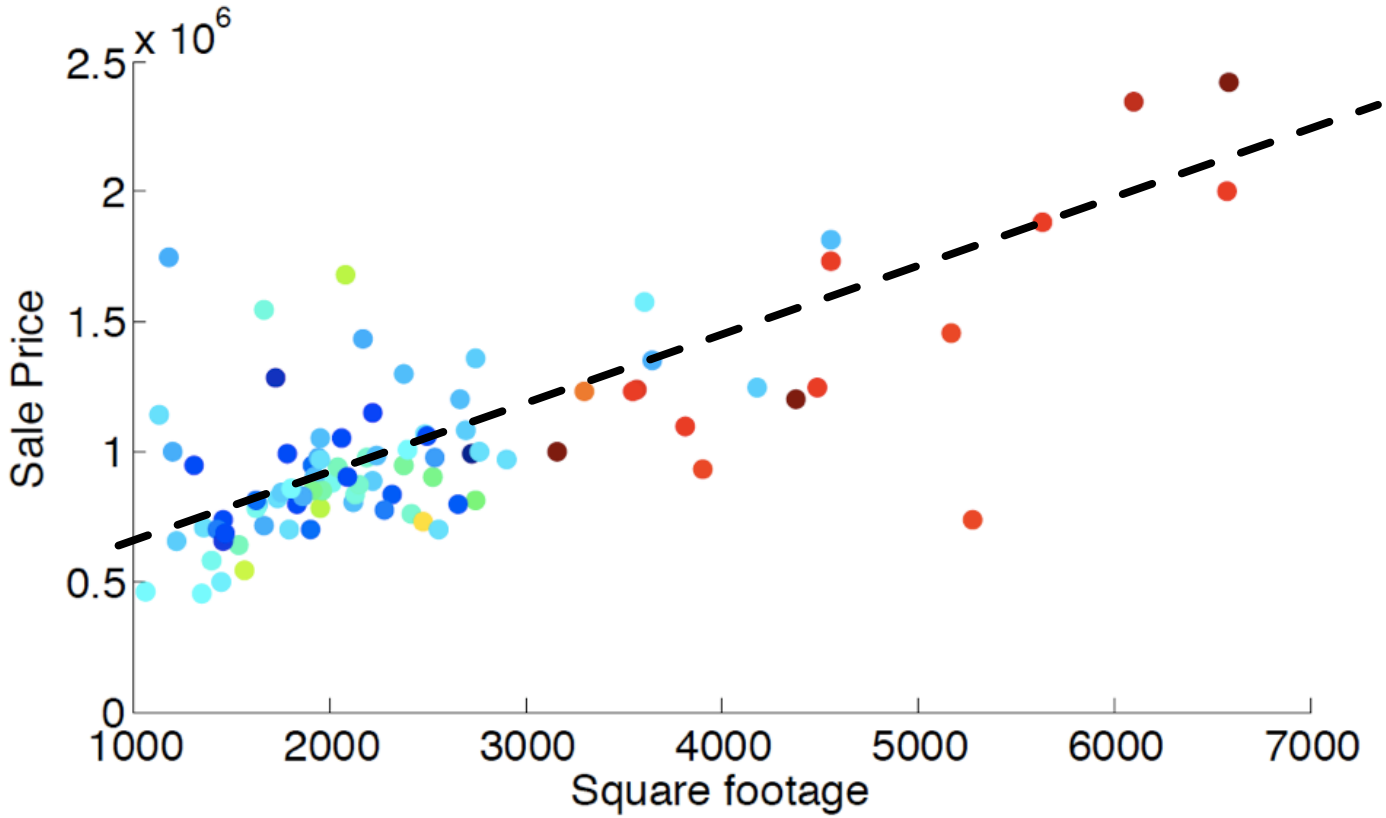
Possibly linear relationship

Sale price $\approx$ **price_per_sqft** $\times$ square_footage $+$ **fixed_expense**



Possibly linear relationship

Sale price $\approx$ **price_per_sqft** $\times$ square_footage $+$ **fixed_expense**
(*slope*) (*intercept*)



How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.

How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.
- We can look at
  - *absolute* error: | prediction - sale price |

How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.
- We can look at
  - *absolute* error: | prediction - sale price |
  - or *squared* error: (prediction - sale price)^2 (**most common**)

How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.
- We can look at
  - *absolute* error: | prediction - sale price |
  - or *squared* error: (prediction - sale price)^2 (**most common**)

Goal: pick the model (unknown parameters) that minimizes the average/total prediction error,

How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.
- We can look at
  - *absolute* error: | prediction - sale price |
  - or *squared* error: (prediction - sale price)^2 (**most common**)

Goal: pick the model (unknown parameters) that minimizes the average/total prediction error, but *on what set*?

How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.
- We can look at
  - *absolute* error: | prediction - sale price |
  - or *squared* error: (prediction - sale price)^2 (**most common**)

Goal: pick the model (unknown parameters) that minimizes the average/total prediction error, but *on what set*?

- test set, ideal but we *cannot use test set while training*

How to learn the unknown parameters?

How to measure error for one prediction?

- The classification error (0-1 loss, i.e. *right* or *wrong*) is *inappropriate* for continuous outcomes.
- We can look at
  - *absolute* error: | prediction - sale price |
  - or *squared* error: (prediction - sale price)^2 (**most common**)

Goal: pick the model (unknown parameters) that minimizes the average/total prediction error, but *on what set*?

- test set, ideal but we *cannot use test set while training*
- training set ✓

Example

Predicted price = **price_per_sqft** × square_footage + **fixed_expense**

one model: price_per_sqft = 0.3K, fixed_expense = 210K

sqft	sale price (K)	prediction (K)	squared error
2000	810	810	0
2100	907	840	67^2
1100	312	540	228^2
5500	2,600	1,860	740^2
...	...	...	...
Total			$0 + 67^2 + 228^2 + 740^2 + \cdots$

Adjust price_per_sqft and fixed_expense such that the total squared error is minimized.

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d$

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}$ (superscript ^{T} stands for transpose),

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}$ (superscript ^{T} stands for transpose), i.e. a *hyper-plane* parametrized by

- $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ (weights, weight vector, parameter vector, etc)
- bias w_0

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}$ (superscript ^{T} stands for transpose), i.e. a *hyper-plane* parametrized by

- $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ (weights, weight vector, parameter vector, etc)
- bias w_0

NOTE: for notation convenience, very often we

- append 1 to each x as the first feature: $\tilde{\boldsymbol{x}} = [1 \; x_1 \; x_2 \; \ldots \; x_{\mathsf{D}}]^{\mathrm{T}}$

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}$ (superscript ^{T} stands for transpose), i.e. a *hyper-plane* parametrized by

- $\boldsymbol{w} = [w_1\ w_2\ \cdots\ w_{\mathsf{D}}]^{\mathrm{T}}$ (weights, weight vector, parameter vector, etc)
- bias w_0

NOTE: for notation convenience, very often we

- append 1 to each x as the first feature: $\tilde{\boldsymbol{x}} = [1\ x_1\ x_2\ \ldots\ x_{\mathsf{D}}]^{\mathrm{T}}$
- let $\tilde{\boldsymbol{w}} = [w_0\ w_1\ w_2\ \cdots\ w_{\mathsf{D}}]^{\mathrm{T}}$, a concise representation of all $D + 1$ parameters

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}$ (superscript ^{T} stands for transpose), i.e. a *hyper-plane* parametrized by

- $\boldsymbol{w} = [w_1\ w_2\ \cdots\ w_{\mathsf{D}}]^{\mathrm{T}}$ (weights, weight vector, parameter vector, etc)
- bias w_0

NOTE: for notation convenience, very often we

- append 1 to each x as the first feature: $\tilde{\boldsymbol{x}} = [1\ x_1\ x_2\ \ldots\ x_{\mathsf{D}}]^{\mathrm{T}}$
- let $\tilde{\boldsymbol{w}} = [w_0\ w_1\ w_2\ \cdots\ w_{\mathsf{D}}]^{\mathrm{T}}$, a concise representation of all $D + 1$ parameters
- the model becomes simply $f(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^{\mathbf{T}} \tilde{\boldsymbol{x}}$

Formal setup for linear regression

Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (features, covariates, context, predictors, etc)

Output: $y \in \mathbb{R}$ (responses, targets, outcomes, etc)

Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Linear model: $f : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$, with $f(\boldsymbol{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \boldsymbol{w}^{\mathbf{T}} \boldsymbol{x}$ (superscript ^{T} stands for transpose), i.e. a *hyper-plane* parametrized by

- $\boldsymbol{w} = [w_1\ w_2\ \cdots\ w_{\mathsf{D}}]^{\mathrm{T}}$ (weights, weight vector, parameter vector, etc)
- bias w_0

NOTE: for notation convenience, very often we

- append 1 to each x as the first feature: $\tilde{\boldsymbol{x}} = [1\ x_1\ x_2\ \ldots\ x_{\mathsf{D}}]^{\mathrm{T}}$
- let $\tilde{\boldsymbol{w}} = [w_0\ w_1\ w_2\ \cdots\ w_{\mathsf{D}}]^{\mathrm{T}}$, a concise representation of all $D + 1$ parameters
- the model becomes simply $f(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^{\mathbf{T}} \tilde{\boldsymbol{x}}$
- sometimes just use $\boldsymbol{w}, \boldsymbol{x}, \mathsf{D}$ for $\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}}, \mathsf{D} + 1$!

Goal

Minimize total squared error

$$\sum_n (f(\boldsymbol{x}_n) - y_n)^2 = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

Goal

Minimize total squared error

- **Residual Sum of Squares** (RSS), a function of $\tilde{\boldsymbol{w}}$

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (f(\boldsymbol{x}_n) - y_n)^2 = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

Goal

Minimize total squared error

- **Residual Sum of Squares** (RSS), a function of $\tilde{\boldsymbol{w}}$

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (f(\boldsymbol{x}_n) - y_n)^2 = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

- find $\tilde{\boldsymbol{w}}^* = \underset{\tilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{D}+1}}{\operatorname{argmin}} \mathrm{RSS}(\tilde{\boldsymbol{w}})$, i.e. **least (mean) squares solution**
  (more generally called **empirical risk minimizer**)

Goal

Minimize total squared error

- **Residual Sum of Squares** (RSS), a function of $\tilde{\boldsymbol{w}}$

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (f(\boldsymbol{x}_n) - y_n)^2 = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

- find $\tilde{\boldsymbol{w}}^* = \underset{\tilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{D}+1}}{\operatorname{argmin}} \mathrm{RSS}(\tilde{\boldsymbol{w}})$, i.e. **least (mean) squares solution**
  (more generally called **empirical risk minimizer**)

- *reduce machine learning to optimization*

Goal

Minimize total squared error

- **Residual Sum of Squares** (RSS), a function of $\tilde{\boldsymbol{w}}$

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (f(\boldsymbol{x}_n) - y_n)^2 = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

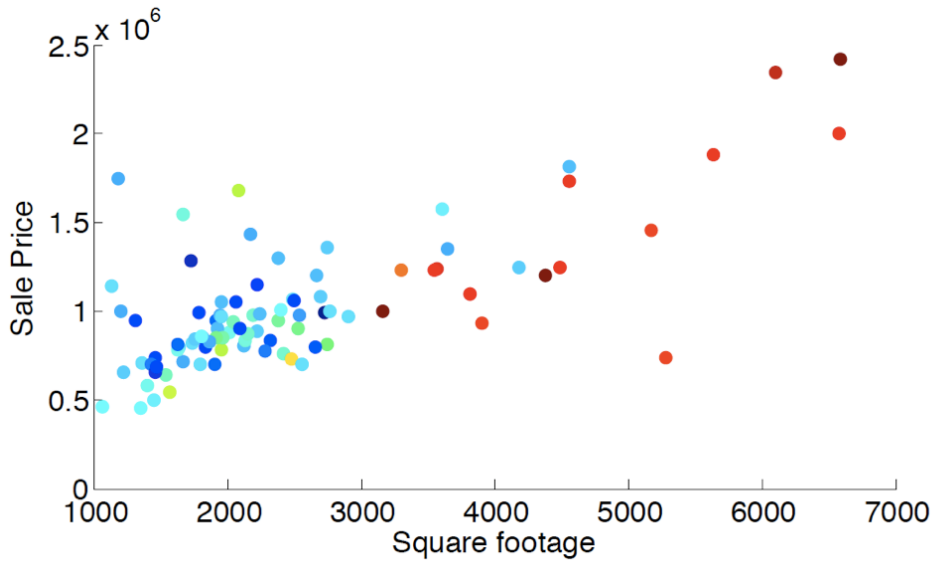
- find $\tilde{\boldsymbol{w}}^* = \operatorname*{argmin}_{\tilde{\boldsymbol{w}} \in \mathbb{R}^{\mathrm{D}+1}} \mathrm{RSS}(\tilde{\boldsymbol{w}})$, i.e. **least (mean) squares solution**
  (more generally called **empirical risk minimizer**)

- *reduce machine learning to optimization*

- in principle can apply any optimization algorithm, but linear regression admits a *closed-form solution*

Warm-up: D = 0

Only one parameter w_0: constant prediction $f(x) = w_0$



f is a horizontal line, where should it be?

Warm-up: D = 0

Optimization objective becomes

$$\mathrm{RSS}(w_0) = \sum_n (w_0 - y_n)^2$$

(it's a *quadratic* $aw_0^2 + bw_0 + c$)

Warm-up: D = 0

Optimization objective becomes

$$\mathrm{RSS}(w_0) = \sum_n (w_0 - y_n)^2 \qquad \text{(it's a } \textit{quadratic}\ aw_0^2 + bw_0 + c\text{)}$$

$$= Nw_0^2 - 2\left(\sum_n y_n\right) w_0 + \text{cnt.}$$

Warm-up: D = 0

Optimization objective becomes

$$\mathrm{RSS}(w_0) = \sum_n (w_0 - y_n)^2 \qquad \text{(it's a } \textit{quadratic}\ aw_0^2 + bw_0 + c\text{)}$$

$$= Nw_0^2 - 2\left(\sum_n y_n\right) w_0 + \text{cnt.}$$

$$= N\left(w_0 - \frac{1}{N}\sum_n y_n\right)^2 + \text{cnt.}$$

Warm-up: D = 0

Optimization objective becomes

$$\text{RSS}(w_0) = \sum_n (w_0 - y_n)^2 \qquad \text{(it's a } \textit{quadratic}\ aw_0^2 + bw_0 + c\text{)}$$

$$= Nw_0^2 - 2\left(\sum_n y_n\right) w_0 + \text{cnt.}$$

$$= N\left(w_0 - \frac{1}{N}\sum_n y_n\right)^2 + \text{cnt.}$$

It is clear that $w_0^* = \frac{1}{N}\sum_n y_n$, i.e. the **average**

Warm-up: D = 0

Optimization objective becomes

$$\mathrm{RSS}(w_0) = \sum_n (w_0 - y_n)^2 \qquad \text{(it's a } \textit{quadratic}\ aw_0^2 + bw_0 + c\text{)}$$

$$= N w_0^2 - 2 \left(\sum_n y_n \right) w_0 + \text{cnt.}$$

$$= N \left(w_0 - \frac{1}{N} \sum_n y_n \right)^2 + \text{cnt.}$$

It is clear that $w_0^* = \frac{1}{N} \sum_n y_n$, i.e. the **average**

Exercise: what if we use absolute error instead of squared error?

Warm-up: D = 1

Optimization objective becomes

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (w_0 + w_1 x_n - y_n)^2$$

Warm-up: D = 1

Optimization objective becomes

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (w_0 + w_1 x_n - y_n)^2$$

General approach: find *stationary points*, i.e., points with *zero gradient*

$$\begin{cases} \frac{\partial \mathrm{RSS}(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \\ \frac{\partial \mathrm{RSS}(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \end{cases} \Rightarrow \begin{array}{ll} \sum_n (w_0 + w_1 x_n - y_n) & = 0 \\ \sum_n (w_0 + w_1 x_n - y_n) x_n & = 0 \end{array}$$

Warm-up: D = 1

Optimization objective becomes

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (w_0 + w_1 x_n - y_n)^2$$

General approach: find *stationary points*, i.e., points with *zero gradient*

$$\begin{cases} \frac{\partial \mathrm{RSS}(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \\ \frac{\partial \mathrm{RSS}(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \end{cases} \Rightarrow \begin{array}{ll} \sum_n (w_0 + w_1 x_n - y_n) & = 0 \\ \sum_n (w_0 + w_1 x_n - y_n) x_n & = 0 \end{array}$$

$$\Rightarrow \begin{array}{ll} N w_0 + w_1 \sum_n x_n & = \sum_n y_n \\ w_0 \sum_n x_n + w_1 \sum_n x_n^2 & = \sum_n y_n x_n \end{array} \quad \textbf{(a linear system)}$$

Warm-up: D = 1

Optimization objective becomes

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (w_0 + w_1 x_n - y_n)^2$$

General approach: find *stationary points*, i.e., points with *zero gradient*

$$\begin{cases} \frac{\partial \mathrm{RSS}(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \\ \frac{\partial \mathrm{RSS}(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \end{cases} \Rightarrow \begin{array}{ll} \sum_n (w_0 + w_1 x_n - y_n) & = 0 \\ \sum_n (w_0 + w_1 x_n - y_n) x_n & = 0 \end{array}$$

$$\Rightarrow \begin{array}{ll} N w_0 + w_1 \sum_n x_n & = \sum_n y_n \\ w_0 \sum_n x_n + w_1 \sum_n x_n^2 & = \sum_n y_n x_n \end{array} \quad \textbf{(a linear system)}$$

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

Least square solution for D = 1

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

(assuming the matrix is invertible)

Least square solution for D = 1

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

(assuming the matrix is invertible)

Are stationary points minimizers?

Least square solution for D = 1

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

(assuming the matrix is invertible)

Are stationary points minimizers?

- yes for **convex** objectives (RSS is convex in $\tilde{\boldsymbol{w}}$)

Least square solution for D = 1

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

(assuming the matrix is invertible)

Are stationary points minimizers?

- yes for **convex** objectives (RSS is convex in $\tilde{\boldsymbol{w}}$)
- not true in general

General least square solution

Objective

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

General least square solution

Objective

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

Again, find stationary points (**multivariate calculus**)

$$\nabla \mathrm{RSS}(\tilde{\boldsymbol{w}}) = 2 \sum_n \tilde{\boldsymbol{x}}_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)$$

General least square solution

Objective

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

Again, find stationary points (**multivariate calculus**)

$$\nabla \mathrm{RSS}(\tilde{\boldsymbol{w}}) = 2 \sum_n \tilde{\boldsymbol{x}}_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n) \propto \left(\sum_n \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} - \sum_n \tilde{\boldsymbol{x}}_n y_n$$

General least square solution

Objective

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

Again, find stationary points (**multivariate calculus**)

$$\begin{aligned} \nabla \mathrm{RSS}(\tilde{\boldsymbol{w}}) &= 2 \sum_n \tilde{\boldsymbol{x}}_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n) \propto \left(\sum_n \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} - \sum_n \tilde{\boldsymbol{x}}_n y_n \\ &= (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}) \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} \end{aligned}$$

where

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

General least square solution

Objective

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2$$

Again, find stationary points (**multivariate calculus**)

$$\begin{aligned} \nabla \mathrm{RSS}(\tilde{\boldsymbol{w}}) &= 2 \sum_n \tilde{\boldsymbol{x}}_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n) \propto \left(\sum_n \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} - \sum_n \tilde{\boldsymbol{x}}_n y_n \\ &= (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}) \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} = \boldsymbol{0} \end{aligned}$$

where

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

General least square solution

$$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = \mathbf{0} \quad \Rightarrow \quad \tilde{\boldsymbol{w}}^{*} = (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

assuming $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is invertible for now.

General least square solution

$$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{0} \quad \Rightarrow \quad \tilde{\boldsymbol{w}}^{*} = (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

assuming $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is invertible for now.

Again by convexity $\tilde{\boldsymbol{w}}^{*}$ is the minimizer of RSS.

General least square solution

$$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{0} \quad \Rightarrow \quad \tilde{\boldsymbol{w}}^{*} = (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

assuming $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is invertible for now.

Again by convexity $\tilde{\boldsymbol{w}}^{*}$ is the minimizer of RSS.

Verify the solution when $\mathrm{D} = 1$**:**

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

General least square solution

$$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{0} \quad \Rightarrow \quad \tilde{\boldsymbol{w}}^{*} = (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

assuming $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is invertible for now.

Again by convexity $\tilde{\boldsymbol{w}}^{*}$ is the minimizer of RSS.

Verify the solution when $\mathrm{D} = 1$**:**

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

when $\mathrm{D} = 0$**:** $(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1} = \frac{1}{N}$, $\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y} = \sum_n y_n$

Another approach

RSS is a quadratic:

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2 = \|\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2$$

Another approach

RSS is a quadratic:

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2 = \|\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2$$

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

Another approach

RSS is a quadratic:

$$\begin{aligned}\mathrm{RSS}(\tilde{\boldsymbol{w}}) &= \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2 = \|\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2 \\ &= \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right) \\ &= \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} + \text{cnt.}\end{aligned}$$

Another approach

RSS is a quadratic:

$$\begin{aligned}
\mathrm{RSS}(\tilde{\boldsymbol{w}}) &= \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2 = \|\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2 \\
&= \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right) \\
&= \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} + \text{cnt.} \\
&= \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}\right) \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}\right) + \text{cnt.}
\end{aligned}$$

Another approach

RSS is a quadratic:

$$\begin{aligned}
\mathrm{RSS}(\tilde{\boldsymbol{w}}) &= \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2 = \|\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2 \\
&= \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right) \\
&= \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} + \text{cnt.} \\
&= \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}\right) \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}\right) + \text{cnt.}
\end{aligned}$$

Note: $\boldsymbol{u}^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}\right) \boldsymbol{u} = \left(\tilde{\boldsymbol{X}} \boldsymbol{u}\right)^{\mathrm{T}} \tilde{\boldsymbol{X}} \boldsymbol{u} = \|\tilde{\boldsymbol{X}} \boldsymbol{u}\|_2^2 \geq 0$ and is 0 if $\boldsymbol{u} = 0$.

Another approach

RSS is a quadratic:

$$\begin{aligned}
\mathrm{RSS}(\tilde{\boldsymbol{w}}) &= \sum_n (\tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - y_n)^2 = \|\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2 \\
&= \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}\right) \\
&= \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} + \text{cnt.} \\
&= \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}\right) \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}\right) + \text{cnt.}
\end{aligned}$$

Note: $\boldsymbol{u}^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}}\right) \boldsymbol{u} = \left(\tilde{\boldsymbol{X}} \boldsymbol{u}\right)^{\mathrm{T}} \tilde{\boldsymbol{X}} \boldsymbol{u} = \|\tilde{\boldsymbol{X}} \boldsymbol{u}\|_2^2 \geq 0$ and is 0 if $\boldsymbol{u} = 0$.

So $\tilde{\boldsymbol{w}}^* = (\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}$ is the minimizer.

Computational complexity

Bottleneck of computing

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

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

- aka *pseudo-inverse*[1] denoted by $(\cdot)^\dagger$, i.e. $\tilde{\boldsymbol{X}}^\dagger = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}$
- naively need $O(\mathrm{D}^3)$ time

see `https://en.wikipedia.org/wiki/Moore-Penrose_inverse`

Computational complexity

Bottleneck of computing

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

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

- aka *pseudo-inverse*[1] denoted by $(\cdot)^\dagger$, i.e. $\tilde{\boldsymbol{X}}^\dagger = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}$
- naively need $O(\mathrm{D}^3)$ time
- there are many faster approaches

see `https://en.wikipedia.org/wiki/Moore-Penrose_inverse`

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

What does that imply?

See https://sites.math.washington.edu/~burke/crs/308/LeastSquares.pdf

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

What does that imply?

Recall $\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)\boldsymbol{w}^{*} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$.

See `https://sites.math.washington.edu/~burke/crs/308/LeastSquares.pdf`

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

What does that imply?

Recall $\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)\boldsymbol{w}^{*} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$. If $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ not invertible, this equation aka *Normal Equations* has

- infinitely many solutions

See `https://sites.math.washington.edu/~burke/crs/308/LeastSquares.pdf`

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

What does that imply?

Recall $\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)\boldsymbol{w}^{*} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$. If $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ not invertible, this equation aka *Normal Equations* has

- infinitely many solutions

See `https://sites.math.washington.edu/~burke/crs/308/LeastSquares.pdf`

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

What does that imply?

Recall $\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)\boldsymbol{w}^{*} = \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$. If $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ not invertible, this equation aka *Normal Equations* has

- infinitely many solutions ($\Rightarrow$ infinitely many minimizers)
- This is because *Normal Equations* are always *consistent*[2] meaning a solution *always* exists! It may not be unique though.

See `https://sites.math.washington.edu/~burke/crs/308/LeastSquares.pdf`

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

Why would that happen?

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

Why would that happen?

One situation: $\mathsf{N} < \mathsf{D} + 1$, i.e. not enough data to estimate all parameters.

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

Why would that happen?

One situation: $\mathrm{N} < \mathrm{D} + 1$, i.e. not enough data to estimate all parameters.

Example: $\mathrm{D} = \mathrm{N} = 1$

sqft	sale price
1000	500K

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

Why would that happen?

One situation: $\mathrm{N} < \mathrm{D} + 1$, i.e. not enough data to estimate all parameters.

Example: $\mathrm{D} = \mathrm{N} = 1$

sqft	sale price
1000	500K

Any line passing through this single point is a minimizer of RSS.

How to resolve this issue?

Intuition: what does inverting $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ do?

eigendecomposition: $$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \lambda_{\mathsf{D}} & 0 \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}+1} \end{bmatrix} \boldsymbol{U}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}+1} \geq 0$ are **eigenvalues**.

How to resolve this issue?

Intuition: what does inverting $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ do?

eigendecomposition: $$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \lambda_{\mathsf{D}} & 0 \\ 0 & \cdots & 0 & \lambda_{\mathsf{D}+1} \end{bmatrix} \boldsymbol{U}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{\mathsf{D}+1} \geq 0$ are **eigenvalues**.

inverse: $$(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \frac{1}{\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{1}{\lambda_{\mathsf{D}}} & 0 \\ 0 & \cdots & 0 & \frac{1}{\lambda_{\mathsf{D}+1}} \end{bmatrix} \boldsymbol{U}$$

i.e. just inverse the eigenvalues

How to solve this problem?

Non-invertible $\Rightarrow$ some eigenvalues are 0.

How to solve this problem?

Non-invertible $\Rightarrow$ some eigenvalues are 0.

One natural fix: add something positive

$$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda \boldsymbol{I} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 + \lambda & 0 & \cdots & 0 \\ 0 & \lambda_2 + \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \lambda_{\mathrm{D}} + \lambda & 0 \\ 0 & \cdots & 0 & \lambda_{\mathrm{D}+1} + \lambda \end{bmatrix} \boldsymbol{U}$$

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

How to solve this problem?

Non-invertible $\Rightarrow$ some eigenvalues are 0.

One natural fix: add something positive

$$\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda \boldsymbol{I} = \boldsymbol{U}^{\mathrm{T}} \begin{bmatrix} \lambda_1 + \lambda & 0 & \cdots & 0 \\ 0 & \lambda_2 + \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \lambda_{\mathrm{D}} + \lambda & 0 \\ 0 & \cdots & 0 & \lambda_{\mathrm{D}+1} + \lambda \end{bmatrix} \boldsymbol{U}$$

where $\lambda > 0$ and $\boldsymbol{I}$ is the identity matrix. Now it is invertible:

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

Fix the problem

The solution becomes

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

- not a minimizer of the original RSS

Fix the problem

The solution becomes

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

- not a minimizer of the original RSS

This in fact comes from minimizing **regularized** RSS (covered in next lecture)!

$$\min_{\boldsymbol{w}} \|\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{y}\|_2^2$$

Fix the problem

The solution becomes

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

- not a minimizer of the original RSS

This in fact comes from minimizing **regularized** RSS (covered in next lecture)!

$$\min_{\boldsymbol{w}} \|\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{y}\|_2^2$$

λ is a *hyper-parameter*, can be tuned by cross-validation.

Comparison to NNC

Parametric versus non-parametric

- **Parametric methods**: the size of the model does *not grow* with the size of the training set N.
  - e.g. linear regression, $D + 1$ parameters, independent of N.
- **Non-parametric methods**: the size of the model *grows* with the size of the training set.
  - e.g. NNC, the training set itself needs to be kept in order to predict. Thus, the size of the model is the size of the training set.