Perceptron and Linear Regresssion

Professor Ameet Talwalkar

Slide Credit: Professor Fei Sha

Outline

A few announcements

- Homework 2: due now
- Homework 3 and 4 now available online
  - BOTH are due in two weeks

A few announcements

- Homework 2: due now
- Homework 3 and 4 now available online
  - BOTH are due in two weeks
- Read the book(s) to supplement content on slides!

Class Projects (2nd reminder)

- 1-2 students per project
- Grading
  - 30% of total class grade
  - Proposal will be short (roughly 1 page, details next week), but worth 1/4 of grade because planning ahead is important!
- You are responsible for proposing a project!
- You must briefly meet with one the TAs or myself before the proposal submission deadline

How to get started?

- Projects can be theoretical, algorithmic and/or applied in nature
  - Develop new learning algorithm
  - Theoretically analyze an existing or a new algorithm
  - Apply learning techniques on some problem of interest
- Get started by thinking about what you're interested in
  - Research you're already doing?
  - Some domain you've always been excited about (sports, politics, weather, movies, music, etc.)?
  - If you're doing an applied project, finding data is the crucial component. What questions can you ask of your data?

Class Project Timeline

- Before November 5th: Meet with a TA or me
- November 5th: Project Proposal is due
- December 11th: Poster Session; Project Report due

Outline

1. Administration
2. Review of last lecture
   - Generative vs Discriminative
   - Multiclass classification
3. Perceptron
4. Linear regression

Generative vs Discriminative

Discriminative

- Requires only specifying a model for the conditional distribution $p(y|x)$, and thus, maximizes the *conditional* likelihood $\sum_n \log p(y_n|\boldsymbol{x}_n)$.
- Models that try to learn mappings directly from feature space to the labels are also discriminative, e.g., perceptron, SVMs (covered later)

Generative vs Discriminative

Discriminative

- Requires only specifying a model for the conditional distribution $p(y|x)$, and thus, maximizes the *conditional* likelihood $\sum_n \log p(y_n|\boldsymbol{x}_n)$.
- Models that try to learn mappings directly from feature space to the labels are also discriminative, e.g., perceptron, SVMs (covered later)

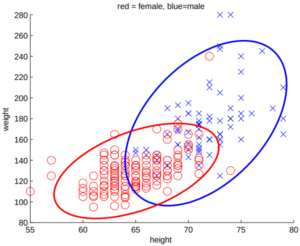
Generative

- Aims to model the joint probability $p(x, y)$ and thus maximize the *joint* likelihood $\sum_n \log p(\boldsymbol{x}_n, y_n)$.
- The generative models we cover (Naive Bayes, QDA, LDA) do so by modeling $p(x|y)$ and $p(y)$

QDA Model of the joint distribution (1D)

$$p(x,y) = p(y)p(x|y)$$
$$= \begin{cases} p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} & \text{if } y=1 \\ p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} & \text{if } y=2 \end{cases}$$

$p_1 + p_2 = 1$ are *prior* probabilities, and $p(x|y)$ is a *class conditional distribution*



QDA Parameter estimation

Log Likelihood in 1D $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ with $y_n \in \{1, 2\}$

$$\begin{aligned}
\log P(\mathcal{D}) &= \sum_n \log p(x_n, y_n) \\
&= \sum_{n:y_n=1} \log \left(p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n-\mu_1)^2}{2\sigma_1^2}} \right) \\
&+ \sum_{n:y_n=2} \log \left(p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n-\mu_2)^2}{2\sigma_2^2}} \right)
\end{aligned}$$

Max log likelihood $(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \arg\max \log P(\mathcal{D})$

QDA Parameter estimation

Log Likelihood in 1D $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ with $y_n \in \{1, 2\}$

$$\begin{aligned}\log P(\mathcal{D}) &= \sum_n \log p(x_n, y_n) \\ &= \sum_{n: y_n = 1} \log \left(p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n - \mu_1)^2}{2\sigma_1^2}} \right) \\ &+ \sum_{n: y_n = 2} \log \left(p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n - \mu_2)^2}{2\sigma_2^2}} \right)\end{aligned}$$

Max log likelihood $(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \arg\max \log P(\mathcal{D})$

Max likelihood ($D > 1$) $(p_1^*, p_2^*, \boldsymbol{\mu}_1^*, \boldsymbol{\mu}_2^*, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_2^*) = \arg\max \log P(\mathcal{D})$

QDA vs LDA vs NB

- QDA: Allows distinct, arbitrary covariance matrices for each class
- LDA: Requires the same arbitrary covariance matrix across classes
- GNB in general: Allows for distinct covariance matrices across each class, but these covariance matrices must be diagonal
- GNB in HW2 Problem 1: Requires the same diagonal covariance matrix across classes

Generative versus discriminative: which one to use?

There is no fixed rule

- It depends on how well your modeling assumption fits the data
- LDA and Gaussian Naive Bayes make stronger assumptions than logistic regression
- When data follows this assumption, these generative models will likely yield a model that better fits the data
- But logistic regression is more robust and less sensitive to incorrect modelling assumption

Setup for classifying multiple classes

Suppose we need to predict multiple classes/outcomes: $C_1, C_2, \ldots, C_K$

- Weather prediction: sunny, cloudy, raining, etc
- Optical character recognition: 10 digits + 26 characters (lower and upper cases) + special characters, etc

Two main approaches

- Use binary classifiers as building blocks
- Multinomial logistic regression

The approach of "one versus the rest"

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Train K binary classifiers in total

The approach of "one versus the rest"

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Train K binary classifiers in total

The approach of "one versus one"

- For each *pair* of classes C_k and $C_{k'}$, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel training data with label $C_{k'}$ into NEGATIVE (or '0')
  3. *Disregard* all other data
- Train $K(K-1)/2$ binary classifiers total

Contrast these two approaches

Pros and cons of each approach

- *one versus the rest*: only needs to train K classifiers.

Contrast these two approaches

Pros and cons of each approach

- *one versus the rest*: only needs to train K classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through

Contrast these two approaches

Pros and cons of each approach

- *one versus the rest*: only needs to train K classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through
- *one versus one*: only needs to train a smaller subset of data (only those labeled with those two classes would be involved).

Contrast these two approaches

Pros and cons of each approach

- *one versus the rest*: only needs to train K classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through
- *one versus one*: only needs to train a smaller subset of data (only those labeled with those two classes would be involved).
  - Makes a *big* difference if you have a lot of *data* to go through.

Contrast these two approaches

Pros and cons of each approach

- *one versus the rest*: only needs to train K classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through
- *one versus one*: only needs to train a smaller subset of data (only those labeled with those two classes would be involved).
  - Makes a *big* difference if you have a lot of *data* to go through.

Drawback of both methods: *Combining classifiers' outputs seem to be a bit tricky*.

Definition of multinomial logistic regression

Model

For each class C_k, we have a parameter vector $\boldsymbol{w}_k$ and model the posterior probability as

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^{\mathrm{T}}\boldsymbol{x}}} \qquad \leftarrow \qquad \text{This is called } \textit{softmax} \text{ function}$$

Definition of multinomial logistic regression

Model

For each class C_k, we have a parameter vector $\boldsymbol{w}_k$ and model the posterior probability as

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^{\mathrm{T}}\boldsymbol{x}}} \quad \leftarrow \quad \text{This is called } \textit{softmax} \text{ function}$$

Intuition behind softmax: enforces desired properties of conditional probabilities that we are modelling

- preserves relative ordering of scores
- maps scores to values between 0 and 1 that also sum to 1

Definition of multinomial logistic regression

Model

For each class C_k, we have a parameter vector $\boldsymbol{w}_k$ and model the posterior probability as

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^{\mathrm{T}}\boldsymbol{x}}} \qquad \leftarrow \qquad \text{This is called } \textit{softmax} \text{ function}$$

Intuition behind softmax: enforces desired properties of conditional probabilities that we are modelling

- preserves relative ordering of scores
- maps scores to values between 0 and 1 that also sum to 1

Decision boundary: assign $\boldsymbol{x}$ with the label that is the maximum of posterior

$$\arg\max_k P(C_k|\boldsymbol{x}) \rightarrow \arg\max_k \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}$$

Parameter estimation

Discriminative approach: maximize conditional likelihood

$$\log P(\mathcal{D}) = \sum_n \log P(y_n | \boldsymbol{x}_n)$$

Cross-entropy error function

$$\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = -\sum_n \sum_k y_{nk} \log P(C_k | \boldsymbol{x}_n)$$

Properties

- Convex, therefore unique global optimum
- Optimization requires numerical procedures, analogous to those used for binary logistic regression

Outline

Main idea

Consider a linear model for binary classification

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

We use this model to distinguish between two classes $\{-1, +1\}$.

One goal

$$\varepsilon = \sum_n \mathbb{I}[y_n \neq \mathsf{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)]$$

i.e., to minimize errors on the training dataset.

Hard, but easy if we have only one training example

How can we change $\boldsymbol{w}$ such that

$$y_n = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$$

Two cases

- If $y_n = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$, do nothing.
- If $y_n \neq \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$,

$$\boldsymbol{w}^{\mathrm{NEW}} \leftarrow \boldsymbol{w}^{\mathrm{OLD}} + y_n\boldsymbol{x}_n$$

Why would it work?

If $y_n \neq \text{sign}(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)$, then

$$y_n(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n) < 0$$

Why would it work?

If $y_n \neq \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$, then

$$y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) < 0$$

What would happen if we change to new $\boldsymbol{w}^{\mathrm{NEW}} = \boldsymbol{w} + y_n\boldsymbol{x}_n$?

$$y_n[(\boldsymbol{w} + y_n\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{x}_n] = y_n\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + y_n^2\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_n$$

Why would it work?

If $y_n \neq \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$, then

$$y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) < 0$$

What would happen if we change to new $\boldsymbol{w}^{\text{NEW}} = \boldsymbol{w} + y_n\boldsymbol{x}_n$?

$$y_n[(\boldsymbol{w} + y_n\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{x}_n] = y_n\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + y_n^2\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_n$$

We are adding a positive number, so it is possible that

$$y_n(\boldsymbol{w}^{\text{NEW}\,\mathrm{T}}\boldsymbol{x}_n) > 0$$

i.e., we are more likely to classify correctly

Perceptron

Iteratively solving one case at a time

- REPEAT
- Pick a data point $\boldsymbol{x}_n$ (can be a fixed order of the training instances)
- Make a prediction $y = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$ using the *current* $\boldsymbol{w}$
- If $y = y_n$, do nothing. Else,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n$$

- UNTIL converged.

Perceptron

Iterately solving one case at a time

- REPEAT
- Pick a data point $\boldsymbol{x}_n$ (can be a fixed order of the training instances)
- Make a prediction $y = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$ using the *current* $\boldsymbol{w}$
- If $y = y_n$, do nothing. Else,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n \boldsymbol{x}_n$$

- UNTIL converged.

Properties

- This is an online algorithm.
- If the training data is linearly separable, the algorithm stops in a finite number of steps.
- The parameter vector is always a linear combination of training instances.

Outline

Regression

Predicting a continuous outcome variable

- Predicting shoe size from height, weight and gender
- Predicting a company's future stock price using its profit and other financial info
- Predicting annual rainfall based on local flaura / fauna
- Predicting song year from audio features

Regression

Predicting a continuous outcome variable

- Predicting shoe size from height, weight and gender
- Predicting a company's future stock price using its profit and other financial info
- Predicting annual rainfall based on local flaura / fauna
- Predicting song year from audio features

Key difference from classification

Regression

Predicting a continuous outcome variable

- Predicting shoe size from height, weight and gender
- Predicting a company's future stock price using its profit and other financial info
- Predicting annual rainfall based on local flaura / fauna
- Predicting song year from audio features

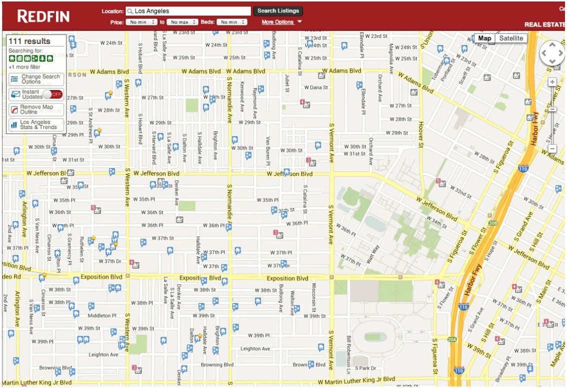
Key difference from classification

- We can measure 'closeness' of prediction and labels, leading to different ways to evaluate prediction errors.
  - Predicting shoe size: better to be off by one size than by 5 sizes
  - Predicting song year: better to be off by one year than by 20 years
- This will lead to 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

3620 South BUDLONG | **$1,510,000** Last Sold Price | **14** Beds | **6** Baths | **4,418** Sq. Ft. $342 / Sq. Ft.
Los Angeles, CA 90007
Status: Closed | Built: 1956 | Lot Size: 9,649 Sq. Ft. | Sold On: Jul 26, 2013

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

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

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
- 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

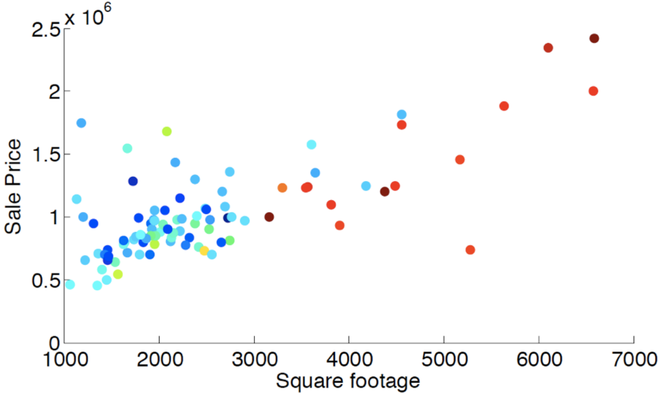
Location Details, Misc. Information & Listing Information

Location Information
- Cross Streets: W 36th Pl

Expense Information
- Operating: $37,664

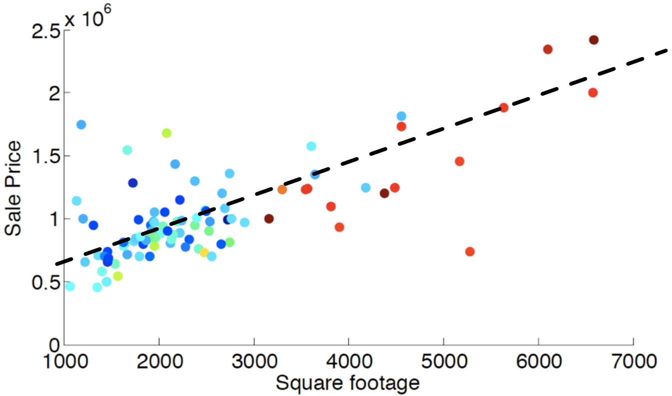
Listing Information
- Listing Terms: Cash, Cash To Existing Loan
- Buyer Financing: Cash

Correlation between square footage and sale price

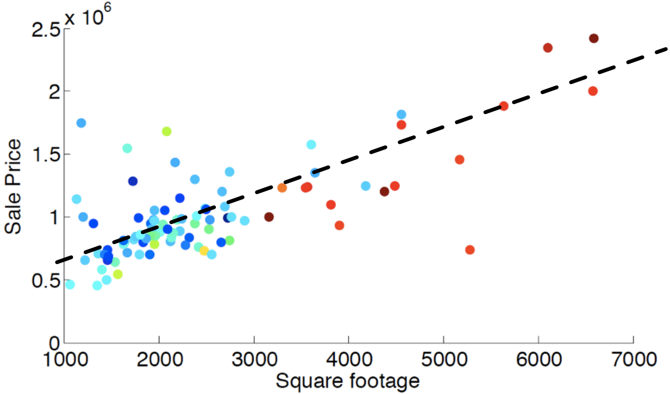


Note: colors here do NOT represent different labels as in classification

Roughly linear relationship



Roughly 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 (*hit* or *miss*) is not appropriate for continuous outcomes.
- How should we evaluate quality of a prediction?

Reduce prediction error

How to measure errors?

- The classification error (*hit* or *miss*) is not appropriate for continuous outcomes.
- How should we evaluate quality of a prediction?
  - *absolute* difference: | prediction - sale price|
  - *squared* difference: (prediction - sale price)^2 [differentiable]

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
$\cdots$	$\cdots$			

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				$8100 + 107^2 + 38^2 + 0 + \cdots$

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)

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)
- Model: $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

  $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$: *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!

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)
- Model: $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

  $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$: *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!
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

How do we learn parameters?

Minimize prediction error on training data

- Use squared difference to measure error
- Residual sum of squares

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

A simple case: x is just one-dimensional ($D=1$)

Residual sum of squares

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

A simple case: x is just one-dimensional ($D=1$)

Residual sum of squares

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

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

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \Rightarrow -2 \sum_n [y_n - (w_0 + w_1 x_n)] = 0$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \Rightarrow -2 \sum_n [y_n - (w_0 + w_1 x_n)] x_n = 0$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \Rightarrow -2 \sum_n [y_n - (w_0 + w_1 x_n)] = 0$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \Rightarrow -2 \sum_n [y_n - (w_0 + w_1 x_n)] x_n = 0$$

Simplify these expressions to get "Normal Equations"

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \Rightarrow -2\sum_n [y_n - (w_0 + w_1 x_n)] = 0$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \Rightarrow -2\sum_n [y_n - (w_0 + w_1 x_n)] x_n = 0$$

Simplify these expressions to get "Normal Equations"

$$\sum y_n = N w_0 + w_1 \sum x_n$$

$$\sum x_n y_n = w_0 \sum x_n + w_1 \sum x_n^2$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_0} = 0 \Rightarrow -2\sum_n [y_n - (w_0 + w_1 x_n)] = 0$$

$$\frac{\partial RSS(\tilde{\boldsymbol{w}})}{\partial w_1} = 0 \Rightarrow -2\sum_n [y_n - (w_0 + w_1 x_n)] x_n = 0$$

Simplify these expressions to get "Normal Equations"

$$\sum y_n = N w_0 + w_1 \sum x_n$$

$$\sum x_n y_n = w_0 \sum x_n + w_1 \sum x_n^2$$

We have two equations and two unknowns! Do some algebra to get:

$$w_1 = \frac{\sum (x_n - \bar{x})(y_n - \bar{y})}{\sum (x_i - \bar{x})^2} \quad \text{and} \quad w_0 = \bar{y} - w_1 \bar{x}$$

where $\bar{x} = \frac{1}{n}\sum_n x_n$ and $\bar{y} = \frac{1}{n}\sum_n y_n$.

Why is minimizing RSS sensible?

Probabilistic interpretation

- Noisy observation model

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

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

Why is minimizing RSS sensible?

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)

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

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)

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

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

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

Probabilistic interpretation (cont'd)

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

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

What is the relationship between minimizing RSS and maximizing the log-likelihood?

Maximum likelihood estimation

Estimating σ, w_0 and w_1 can be done in two steps

- 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 can be done in two steps

- 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 can be done in two steps

- 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)

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

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

How does this probabilistic interpretation help us?

- It gives a solid footing to our intuition: minimizing $\text{RSS}(\tilde{\boldsymbol{w}})$ is a sensible thing based on reasonable modeling assumptions
- Estimating σ^* tells us how much noise there could be in our predictions. For example, it allows us to place confidence intervals around our predictions.