Introduction to Linear Regression

Machine Learning Primer Course

Georgios C. Anagnostopolous and Spencer G. Lyon

October 2, 2020

Purpose

- Linear regression is a deep subject
- Our coverage will be quite superficial
- We use linear regression modeling as a stepping stone towards more elaborate, non-linear models

Table of contents

Before we start...

Some important identities regarding partitioned matrices

Assume conformable quantities $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$

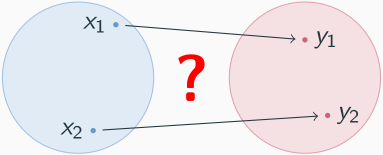
$$\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{C} = \begin{bmatrix} \mathbf{AC} \\ \mathbf{BC} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{D} \end{bmatrix} = \mathbf{AC} + \mathbf{BD}$$

Regression Tasks

Regression

- One of the most common ML tasks
- Based on available data, learn relationships between variables
- Alternatively: learn a mapping from an input (feature) space to an output space
- Falls under the supervised learning paradigm[1]



[1]Although, semi-supervised scenarios also come up

Quick Examples

- Predicting gross income of yet-to-be-released movie
- Real estate pricing prediction
- Any time series forecasting problem
- Tons of examples in (bio)chemistry and reactor dynamics
- Predicting wage based on demographic and educational features

Setting – Data spaces

- Input/Feature space
  - Can be arbitrary, but we will eventually assume $\mathbb{R}^D$
- Output/Target space
  - Typically a "continuous" set, but can be discrete as well
  - Important that elements of the set can be ordered
  - Examples: temperature (continuous), movie ratings (discrete, but ordered)
  - We'll begin by assuming that the output variable is a single number from a continuous range
- Observed Data: (input, output) pairs:

$$\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$$

Setting – Model and Loss

- Model: some family of functions parameterized by a vector of parameters $\mathbf{w}$:

$$\hat{y} = f(\mathbf{x}|\mathbf{w})$$

- Loss function
  - Most typical is squared loss:

$$\ell(y - \hat{y}) = (y - \hat{y})^2$$

  - Sum of Squared Errors:

$$SSE \triangleq \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$$

  - Mean squared error (average loss):

$$MSE(\mathbf{w}) = \frac{1}{N} SSE$$

Setting – Learning

- Loss function:

$$MSE(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 = \frac{1}{N}\left\|\mathbf{y} - \hat{\mathbf{y}}\right\|^2$$

- Learning Algorithm
  - Find $\mathbf{w}^*$ such that $\hat{y}_n \approx y_n$
  - This is done by minimizing the average training loss, or MSE on training set
  - Algorithm and approach varies – depends on loss (ℓ) and model (f) considered
- Prediction

$$\hat{y}^* = f(\mathbf{x}|\mathbf{w}^*)$$

FYI...

- A model f is linear in its parameters when:

$$f(\mathbf{x}|a_1\mathbf{w}_1 + a_2\mathbf{w}_2) = a_1 f(\mathbf{x}|\mathbf{w}_1) + a_2 f(\mathbf{x}|\mathbf{w}_2)$$

- A model f is called linear in its inputs when:

$$f(a_1\mathbf{x}_1 + a_2\mathbf{x}_2|\mathbf{w}) = a_1 f(\mathbf{x}_1|\mathbf{w}) + a_2 f(\mathbf{x}_2|\mathbf{w})$$

Linear Regression

- Linear regression is the most simple[2] regression model
- "Linear" stands for "linear in its parameters"
- The combination linear + squared loss makes linear regression particularly tractable...
  - Optimal parameter values ($\mathbf{w}^*$) can be found in closed form!

[2]In terms of model complexity

Simple Linear Regression

Setting

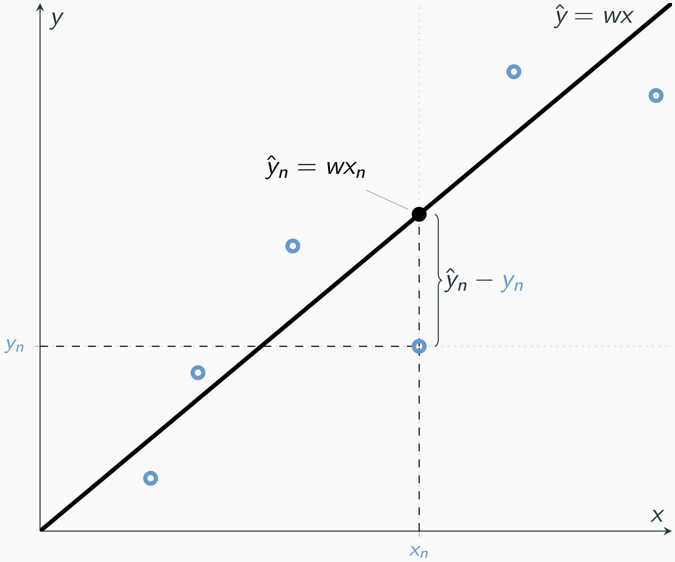
- Model: Learn a linear map of one parameter w:

$$\hat{y} = wx$$

- Data: (x, y) pairs

Setting – Graphically

Model: All straight lines that pass through origin



Average Training Loss

Average training loss on $\{(x_y, y_n)\}_{n=1}^N$ is

$$\begin{aligned}
MSE(w) &= \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}(y_n - wx_n)^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(y_n^2 - 2wx_ny_n + w^2x_n^2\right) \\
&= \frac{1}{N}\left(\sum_{n=1}^{N}y_n^2 - 2w\sum_{n=1}^{N}x_ny_n + w^2\sum_{n=1}^{N}x_n^2\right) \\
&= \frac{1}{N}\left(||\mathbf{y}||_2^2 - 2w\mathbf{x}^T\mathbf{y} + w^2||\mathbf{x}||_2^2\right)
\end{aligned}$$

Average Training Loss (continued)

- MSE can be re-written by "completing the square"[3]

$$
\begin{aligned}
MSE(w) &= \frac{1}{N}\left(||\mathbf{y}||_2^2 - 2w\mathbf{x}^T\mathbf{y} + w^2||\mathbf{x}||_2^2\right) \\
&= \frac{1}{N}||\mathbf{x}||_2^2\left(w - \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||_2^2}\right)^2 + \frac{1}{N}\left(||\mathbf{y}||_2^2 - \frac{(\mathbf{x}^T\mathbf{y})^2}{||\mathbf{x}||_2^2}\right)
\end{aligned}
$$

- Optimal w (called w^*) minimizes MSE

[3] First line repeated from previous slide

Average Training Loss (continued)

- MSE can be re-written by "completing the square"[3]

$$MSE(w) = \frac{1}{N}\left(||\mathbf{y}||_2^2 - 2w\mathbf{x}^T\mathbf{y} + w^2||\mathbf{x}||_2^2\right)$$
$$= \frac{1}{N}||\mathbf{x}||_2^2\left(w - \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||_2^2}\right)^2 + \frac{1}{N}\left(||\mathbf{y}||_2^2 - \frac{(\mathbf{x}^T\mathbf{y})^2}{||\mathbf{x}||_2^2}\right)$$

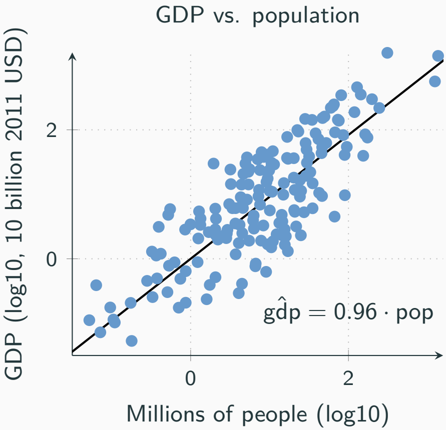
- Optimal w (called w^*) minimizes MSE
- ... so we have that

$$w^* \triangleq \underset{w}{\operatorname{argmin}}\, MSE(W) = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||_2^2}$$
$$MSE(w^*) = \frac{1}{N}\left(||\mathbf{y}||_2^2 - \frac{(\mathbf{x}^T\mathbf{y})^2}{||\mathbf{x}||_2^2}\right)$$

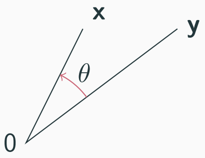
[3] First line repeated from previous slide

Example

- Input: log10 country population
- Output: log10 10 billion dollars country GDP
- 167 samples
- Source: Penn World Tables



Geometry of w^*



- Notice that:

$$\mathbf{x}^T\mathbf{y} = ||\mathbf{x}||_2||\mathbf{y}||_2\cos(\theta)$$

- We can re-write MSE at w^* as:

$$MSE(w^*) = \frac{1}{N}||\mathbf{y}||_2^2\left(1 - \cos^2(\theta)\right)$$

- Interpretation:
  - The more the proportional the y's are to the x's
  - $\implies \theta$ is closer to 0
  - $\implies$ Cosine of θ will be closer to 1
  - $\implies$ Optimal MSE approaches 0.

Comments

- We saw that training amounts to computing the optimal w from the available, observed data
- In the scenario we are considering, training is trivial in the sense that the optimal parameter value was found in closed form
- The optimal parameter can even be found visually by graphing MSE vs. w

Multiple Linear Regression

Setting

- Model: All hyperplanes that pass through the origin:

$$\hat{y} = \mathbf{x}^T \mathbf{w}$$

- Data:

$$\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}, \quad \mathbf{x}_n \in \mathbb{R}^D$$

- Model predictions on training samples:

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \cdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \mathbf{w} \\ \cdots \\ \mathbf{x}_N^T \mathbf{w} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{x}_1^T \\ \cdots \\ \mathbf{x}_N^T \end{bmatrix}}_{\triangleq \mathbf{X}} \mathbf{w} = \mathbf{X}\mathbf{w}$$

MSE

In this setting we write the MSE as

$$
\begin{aligned}
MSE(\mathbf{w}) &= \frac{1}{N}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 \\
&= \frac{1}{N}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{w} - \mathbf{X}\mathbf{w}) \\
&= \frac{1}{N}\left(\mathbf{y}^T - \mathbf{w}^T\mathbf{X}^T\right)(\mathbf{y} - \mathbf{X}\mathbf{w}) \\
&= \frac{1}{N}\left(\mathbf{w}^T\mathbf{R}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + ||\mathbf{y}||_2^2\right)
\end{aligned}
$$

where

$$\mathbf{R} \triangleq \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{D\times D}$$

Optimal Weights

- The MSE is a convex quadratic in $\mathbf{w}$, bounded from below...
  - So there is a unique minimum MSE
- Finding the weights:

$$\left.\frac{dMSE(\mathbf{w})}{d\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^*} = \left.\frac{d}{d\mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^*} \frac{1}{N}\left(\mathbf{w}^T\mathbf{R}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + ||\mathbf{y}||_2^2\right)$$

$$= \frac{1}{N}\left(2\mathbf{R}\mathbf{w}^* - 2\mathbf{X}^T\mathbf{y}\right)$$

$$\implies \quad \text{(set = 0 to optimize)}$$

$$\mathbf{w}^* = \underbrace{\mathbf{R}^{-1}\mathbf{X}^T}_{\triangleq \mathbf{X}^\dagger}\mathbf{y} = \mathbf{X}^\dagger\mathbf{y}$$

- Optimal MSE can be shown to be

$$MSE(\mathbf{w}^*) = \frac{1}{N}\mathbf{y}^T\left(\mathbf{I}_N - \mathbf{X}\mathbf{X}^\dagger\right)y$$

Comments About R

We have

$$\mathbf{R} \triangleq \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{D \times D}$$

- If $N < D$ (more features than samples) then **R** will not be invertible
- In this case $\mathbf{w}^*$ is not uniquely defined
  - Example: N $=$ 1, D $=$ 2
  - There are infinitely many planes that pass through the origin and the sample
  - In all cases the MSE is 0
- Therefore, we need at least one sample per feature

Comments About $\mathbf{X}^\dagger$

We have

$$\mathbf{X}^\dagger \triangleq \mathbf{R}^{-1}\mathbf{X}^T = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \in \mathbb{R}^{D\times N}$$

- Called the Moore-Penrose Pseudo-Inverse of $\mathbf{X}$
- Consider the system of equations:

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

- If $\mathbf{X}$ is square (and invertible) we can solve $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$
- If $\mathbf{X}$ has more rows than columns (more samples than features), then $\mathbf{X}^{-1}$ is not defined
- In this setting, we use $\mathbf{X}^\dagger$ as if it were $\mathbf{X}^{-1}$ and compute $\mathbf{w} = \mathbf{X}^\dagger\mathbf{y}$

Computing $\mathbf{w}^*$ in Practice

- We know we can solve for $\mathbf{w}^*$ by...
  1. Computing $\mathbf{R} = \mathbf{X}^T\mathbf{X}$
  2. Inverting $\mathbf{R}$ to get $\mathbf{R}^{-1}$
  3. Computing $\mathbf{X}^\dagger = \mathbf{R}^{-1}\mathbf{X}^T$
  4. Computing $\mathbf{w}^* = \mathbf{X}^\dagger\mathbf{y}$

Computing $\mathbf{w}^*$ in Practice

- We know we can solve for $\mathbf{w}^*$ by...
  1. Computing $\mathbf{R} = \mathbf{X}^T\mathbf{X}$
  2. Inverting $\mathbf{R}$ to get $\mathbf{R}^{-1}$
  3. Computing $\mathbf{X}^\dagger = \mathbf{R}^{-1}\mathbf{X}^T$
  4. Computing $\mathbf{w}^* = \mathbf{X}^\dagger\mathbf{y}$
- However, in practice we compute $\mathbf{w}^*$ by iteratively minimizing the MSE
- This is more numerically stable than the direct approach[4]
- It also might be the only feasible approach if the data is "big"

[4] especially inverting $\mathbf{R}$

Adding an Intercept

- So far we've restricted models to pass through origin
- We can add an intercept term by adding the $\mathbf{1}$ vector as a column to $\mathbf{X}$:

$$
\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} + b \\
&= \underbrace{\begin{bmatrix} \mathbf{X} & \mathbf{1} \end{bmatrix}}_{\triangleq \tilde{\mathbf{X}}} \underbrace{\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}}_{\tilde{\mathbf{w}}} \\
&= \tilde{\mathbf{X}}\tilde{\mathbf{w}}
\end{aligned}
$$

- Solving for $\tilde{\mathbf{w}}^*$ follows same procedure as solving for $\mathbf{w}$:

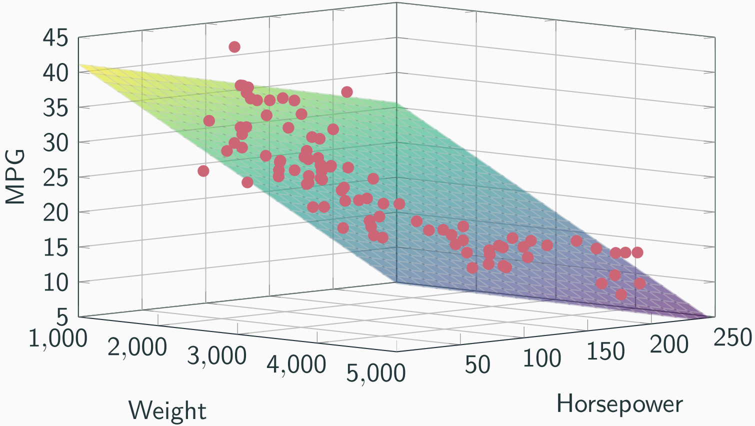
$$\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^\dagger \mathbf{y}$$

- The intercept b is the last element of $\tilde{\mathbf{w}}^*$

An Example: MPG prediction

- Features: weight, Horsepower
- Target: MPG
- Source: Matlab's "carsmall" dataset

$$\hat{MPG} = 47.7694 - 0.0066 * \text{weight} - 0.042 * \text{horsepower}$$



How to Guide

How to fit a linear regression model:

1. Obtain N training samples, each with D features
2. Store the input features as rows of $\mathbf{X} \in \mathbb{R}^{N \times D}$
3. Store the corresponding outputs in a vector $\mathbf{y} \in \mathbb{R}^{N}$
4. If the model needs an intercept, augment $\mathbf{X}$:
   $\mathbf{X} = \begin{bmatrix} \mathbf{X} & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{N \times (D+1)}$
5. Train/fit the model (find $\mathbf{w}^*$):
   - Less Robust: $\mathbf{w}^* = \mathbf{R}^{-1}\mathbf{X}^T\mathbf{y}$, where $\mathbf{R} = \mathbf{X}^T\mathbf{X}$

How to Guide

How to fit a linear regression model:

1. Obtain N training samples, each with D features
2. Store the input features as rows of $\mathbf{X} \in \mathbb{R}^{N \times D}$
3. Store the corresponding outputs in a vector $\mathbf{y} \in \mathbb{R}^{N}$
4. If the model needs an intercept, augment $\mathbf{X}$:
   $\mathbf{X} = \begin{bmatrix} \mathbf{X} & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{N \times (D+1)}$
5. Train/fit the model (find $\mathbf{w}^*$):
   - ~~Less Robust: $\mathbf{w}^* = \mathbf{R}^{-1}\mathbf{X}^T\mathbf{y}$, where $\mathbf{R} = \mathbf{X}^T\mathbf{X}$~~
   - More Robust[5]: $\mathbf{w}^* = \mathbf{X}^\dagger y$, where $\mathbf{X}^\dagger = \mathbf{R}^{-1}\mathbf{X}$

[5] Compute $\mathbf{X}^\dagger$ using `np.linalg.pinv(X)`

How to Guide

How to fit a linear regression model:

1. Obtain N training samples, each with D features
2. Store the input features as rows of $\mathbf{X} \in \mathbb{R}^{N \times D}$
3. Store the corresponding outputs in a vector $\mathbf{y} \in \mathbb{R}^N$
4. If the model needs an intercept, augment $\mathbf{X}$:
   $\mathbf{X} = \begin{bmatrix} \mathbf{X} & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{N \times (D+1)}$
5. Train/fit the model (find $\mathbf{w}^*$):
   - ~~Less Robust: $\mathbf{w}^* = \mathbf{R}^{-1}\mathbf{X}^T\mathbf{y}$, where $\mathbf{R} = \mathbf{X}^T\mathbf{X}$~~
   - ~~More Robust[5]: $\mathbf{w}^* = \mathbf{X}^\dagger y$, where $\mathbf{X}^\dagger = \mathbf{R}^{-1}\mathbf{X}$~~
   - Most robust: use a routine that iteratively solves:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, MSE(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}}\, ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2$$

[5] Compute $\mathbf{X}^\dagger$ using `np.linalg.pinv(X)`

Using $\mathbf{w}^*$

- If intercept was used, its optimal value is the last element of $\mathbf{w}^*$
- The minimum MSE can be computed as

$$MSE(\mathbf{w}^*) = \frac{1}{N}||\mathbf{y} - \mathbf{X}\mathbf{w}^*||_2^2$$

- Predicted outputs on training set are

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}^*$$

- For arbitrary test samples arranged in matrix $\mathbf{X}_{\text{test}}$, predictions are given by[6]

$$\hat{\mathbf{y}}_{\text{test}} = \mathbf{X}_{\text{test}}\mathbf{w}^*$$

[6] Note that $\mathbf{X}_{\text{test}}$ must have columns in same order as $\mathbf{X}$ used for training, including the column of ones if an intercept was used