Linear Regression

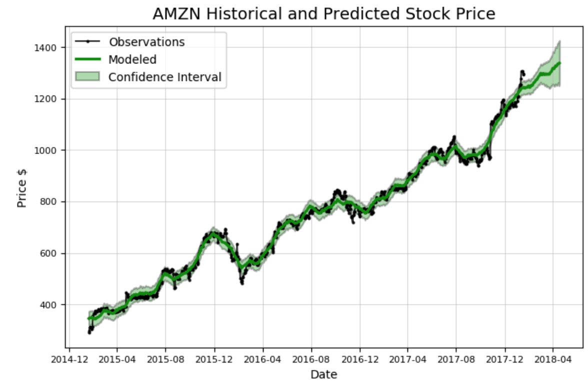
CSC 461: Machine Learning

Fall 2022

Prof. Marco Alvarez
University of Rhode Island

Continuous targets

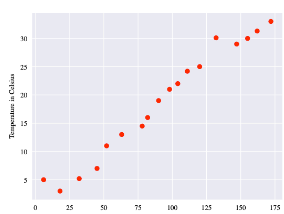
- Certain applications require the prediction of continuous values

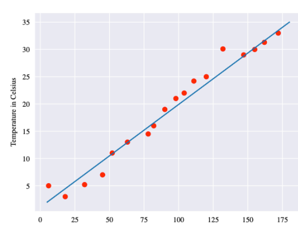


https://towardsdatascience.com/stock-prediction-in-python-b66555171a2

Linear functions

- Assumes the output y is a linear function of the input $\boldsymbol{x}$
  - ✓ can use the function to make predictions, very simple approach, e.g. linear regression

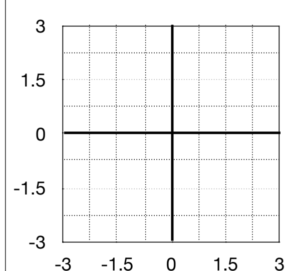


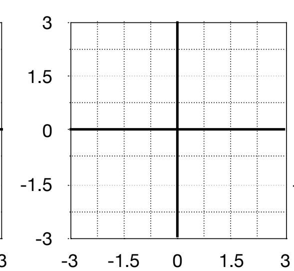


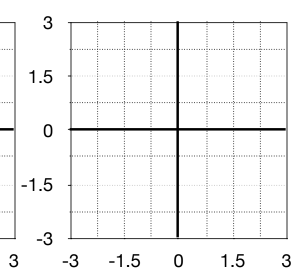
$y = wx + b \quad w, x, b \in \mathbb{R}$

Draw the models

$y = wx + b$ → can also use a vector $\mathbf{w} = [w_0, w_1]^T$, where $w_0 = b$ → $y = w_1 x_1 + w_0$





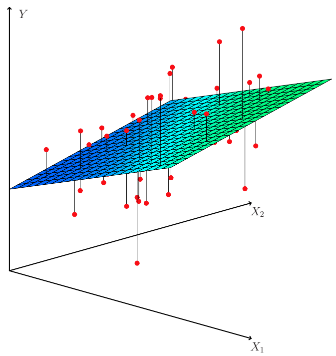


$\mathbf{w} = [0.5, 0]$ $\quad$ $\mathbf{w} = [0, 1.5]$ $\quad$ $\mathbf{w} = [0.5, 0.5]$

Linear functions

- What if we have d features?

why transpose?



$$y = \mathbf{w}^T\mathbf{x} + b \quad \mathbf{w}, \mathbf{x} \in \mathbb{R}^d \quad b \in \mathbb{R}$$

Figure from https://www.statlearning.com/

Linear functions

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

hypothesis weights bias

The weights and bias are the model **parameters** which define the hypothesis and are used to make predictions

Alternative notation

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{d} w_i x_i + b$$

X0	X1	X2	Y
1	0.5	0.1	0.25
1	0.3	0.9	0.5
1	0.3	0.875	1.15
1	0.45	0.15	2.13
...	...	...	...

the bias can be absorbed into the summation if we augment $\boldsymbol{w}$ and $\boldsymbol{x}$

$$h(\mathbf{x}) = \sum_{i=0}^{d} \tilde{w}_i \tilde{x}_i$$

$$= \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

Augmented vectors

input: $(\overset{+1}{x_0}, x_1, \ldots, x_d)$

model: $(w_0, w_1, \ldots, w_d)$

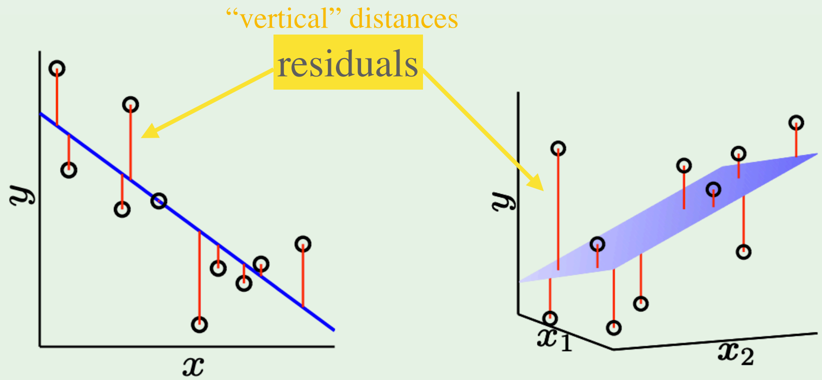
$$h(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^T\mathbf{x}$$

What is the hypothesis space?

all linear functions in $\mathbb{R}^{d+1}$

$\mathscr{H} = \{h_w : h_w(\mathbf{x}) = \mathbf{w}^T\mathbf{x},\ \mathbf{w} \in \mathbb{R}^{d+1}\}$

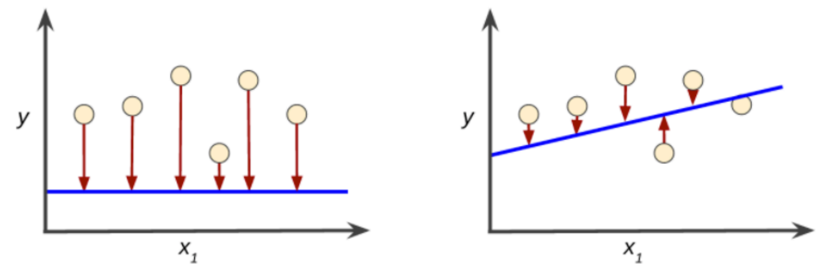
Residuals



Want a linear function with small residuals

http://work.caltech.edu/slides/slides03.pdf

Residuals and Loss



High loss

Low loss

From https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

Goal of learning

- Find a function that best approximates target function (minimize expected loss)

For $h \in \mathscr{H}$ and $\forall (\mathbf{x}^{(i)}, y^{(i)}) \sim P$, we want $h(\mathbf{x}^{(i)}) \approx f(\mathbf{x}^{(\mathbf{i})})$

- What is the expected loss?
  - ✓ cannot calculate, can **approximate** with empirical loss:

$$\mathbb{E}\left[l(h, \mathbf{x}^{(i)}, y^{(i)})\right]_{(\mathbf{x}^{(i)}, y^{(i)}) \sim P} \approx L(h, \mathscr{D})$$

Defining linear regression

- Data $\mathscr{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$

$$\mathbf{x}^{(i)} \in \mathbb{R}^{d+1}, y^{(i)} \in \mathbb{R}$$

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

- Loss Function: Squared Loss (MSE)

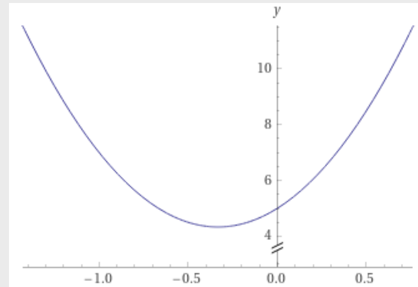
$$l_{sq}(h, \mathbf{x}, y) = \left(h(\mathbf{x}) - y\right)^2$$

$$L_{sq}(h, \mathscr{D}) = \frac{1}{n}\sum_{i=1}^{n} l_{sq}(h, \mathbf{x}^{(i)}, y^{(i)})$$

Closed-form Solution

Normal Equation

min vs. argmin



$f(x) = 6x^2 + 4x + 5$

$\min f(x)$?

$\arg\min_{x} f(x)$?

Solving linear regression (least squares)

find these parameters

$$\mathbf{w}^* = \arg\min_{h \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} \left(h(\mathbf{x}^{(i)}) - y^{(i)}\right)^2$$

given this objective function

unconstrained optimization

Solving linear regression (least squares)

$$\mathbf{w}^* = \arg\min_{h \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} \left(h(\mathbf{x}^{(i)}) - y^{(i)}\right)^2$$

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

Understanding matrix form

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

input vector $\mathrm{x}^{(1)}$

$$\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \approx \begin{bmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(n)}\right)^T \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$$

Norms

- A **norm** is a function that assigns a strictly positive length to each vector in a vector space
  - ✓ except for the zero vector

ℓ_1-norm: $\quad \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$

manhattan distance from origin

ℓ_2-norm: $\quad \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$

euclidean norm, euclidean distance from origin

Using matrix notation

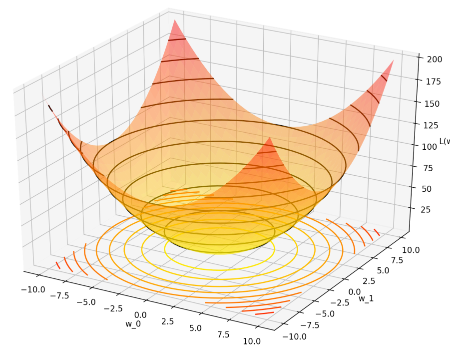
$$\arg\min_{h \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

$$\downarrow$$

$$\arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

How to minimize it?

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

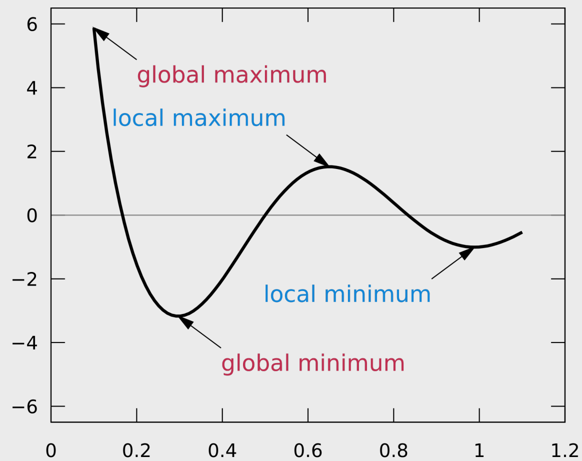


continuous, differentiable, **convex**

optimal solution

https://cnl.salk.edu/~schraudo/teach/NNcourse/linear1.html

(some) Critical points



https://en.wikipedia.org/wiki/Maxima_and_minima

Closed form solution

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\mathrm{X}^{\intercal}(\mathrm{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$$

$$\mathrm{X}^{\intercal}\mathrm{X}\mathbf{w} = \mathrm{X}^{\intercal}\mathbf{y}$$

$$\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y} \quad \text{where} \quad \mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$$

$\mathrm{X}^{\dagger}$ is the 'pseudo-inverse' of X

There are other methods for finding the optimal solution
e.g. gradient descent, MLE

http://work.caltech.edu/slides/slides03.pdf

The algorithm

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

$$\mathrm{X} = \underbrace{\begin{bmatrix} -\mathbf{x}_1^{\intercal}- \\ -\mathbf{x}_2^{\intercal}- \\ \vdots \\ -\mathbf{x}_N^{\intercal}- \end{bmatrix}}_{\text{input data matrix}}, \qquad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2: Compute the pseudo-inverse $\mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$.

3: Return $\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y}$.

http://work.caltech.edu/slides/slides03.pdf

Show me the code

```
w = np.linalg.pinv(Xtr).dot(Ytr)

pred = Xte.dot(w)

loss = np.mean((pred-Yte) ** 2)
```

vectorized computation

Computational complexity?

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$nd^2 + d^3 + nd + d^2$$

$$O(nd^2 + d^3)$$

Training might be computationally expensive

Colab notebook

https://colab.research.google.com/drive/1GIovpb0ij4bSK1-jPKjNTlmXHSwTWwou