Computer Science

CSC380: Principles of Data Science

Linear Models 1

Xinchen Yu

Review: Bernoulli Naïve Bayes MLE

Let $m_c := \sum_{i=1}^{m} \mathbf{I}\{y^{(i)} = c\}$ be number of training examples in class c then,

$$\sum_{i=1}^{m} \log p(\mathcal{D}; \pi, \theta) = \sum_{c=1}^{C} m_c \log \pi_c + \sum_{c=1}^{C} \sum_{i:y^{(i)}=c} \sum_{d=1}^{D} \log p\left(x_d^{(i)}; \theta_{cd}\right)$$

Log-likelihood function is concave in all parameters so…

1. Take derivatives with respect to π and θ separately.
2. Set derivatives to zero and solve

$$\hat{\pi}_c = \frac{m_c}{m}$$

Fraction of training examples from class c

$$\hat{\theta}_{cd} = \frac{m_{cd}}{m_c}$$

Number of “heads” in training set from class c

$$m_{cd} = \sum_{i=1}^{m} I\{y^{(i)} = c, x_d^{(i)} = 1\}$$

Review: making prediction

$$\hat{\pi}_c = \frac{m_c}{m} \qquad \hat{\theta}_{cd} = \frac{m_{cd}}{m_c}$$

Given one data point, it has 4 features (input), compare the probabilities:

$$p(x_1, x_2, x_3, x_4, y = 0) = p(y = 0) \cdot p(x_1, x_2, x_3, x_4 | y = 0)$$
$$= p(y = 0) \cdot p(x_1 | y = 0) \cdot p(x_2 | y = 0) \cdot p(x_3 | y = 0) \cdot p(x_4 | y = 0)$$

$$p(x_1, x_2, x_3, x_4, y = 1) = p(y = 1) \cdot p(x_1, x_2, x_3, x_4 | y = 1)$$
$$= p(y = 1) \cdot p(x_1 | y = 1) \cdot p(x_2 | y = 1) \cdot p(x_3 | y = 1) \cdot p(x_4 | y = 1)$$

Bernoulli Naïve Bayes MLE: issue

no data points of class 2:

$$p(y = 2) = 0$$

no data points in class 1 & 3 is 0 for x1:

$$p(x_1 = 0 | y = 3) = 0$$

$$p(x_1 = 0 | y = 1) = 0$$

y	x_1	x_2
1	1	1
3	1	0
3	1	1
3	1	0
1	1	0
?	0	0

Bernoulli Naïve Bayes MLE: issue

What if there are *no* examples of class c in the training set?

$$\hat{\pi}_c = 0$$

Model will never learn to guess class c

What if all data points i in class c has $x_d^{(i)} = 0$ in the training set?

$$\hat{\theta}_{cd} = 0$$

Model will assign 0 likelihood for test data with $x_d = 1$ for class c (i.e., $p(x|y = c)$).

What does it imply on $p(y = c|x)$? 0!

Training data needs to see *<u>every possible outcome</u> for each feature*

Any ideas how we can fix this problem?

Fixing Bernoulli MLE

We could add a small constant to prevent zero probabilities…

$$\hat{\pi}_c \propto m_c + \alpha \qquad \hat{\theta}_{cj} \propto m_{cj} + \beta \qquad \alpha, \beta > 0$$

Pseudocounts
add-α Smoothing
Laplace smoothing
….

typical choice: set $\alpha = \beta = 1$

Another smoothing method:

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Word count in category c

Vocabulary size in whole corpus

Naïve Bayes in Sentiment Classification

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

$$\boxed{\hat{\pi}_c = \frac{m_c}{m}} \qquad P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

Naïve Bayes in Sentiment Classification

$$\hat{P}(w_i|c) = \frac{count(w_i,c)}{\sum_{w\in V} count(w,c)}$$

smoothing

$$\frac{count(w_i,c)+1}{\left(\sum_{w\in V} count(w,c)\right)+|V|}$$

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Vocabulary size

20 = 14 + 9 - 3

3: the, and, very (duplicate)

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

Naïve Bayes in Sentiment Classification

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable $\boxed{\text{with}}$ no fun

Ignore unknown words

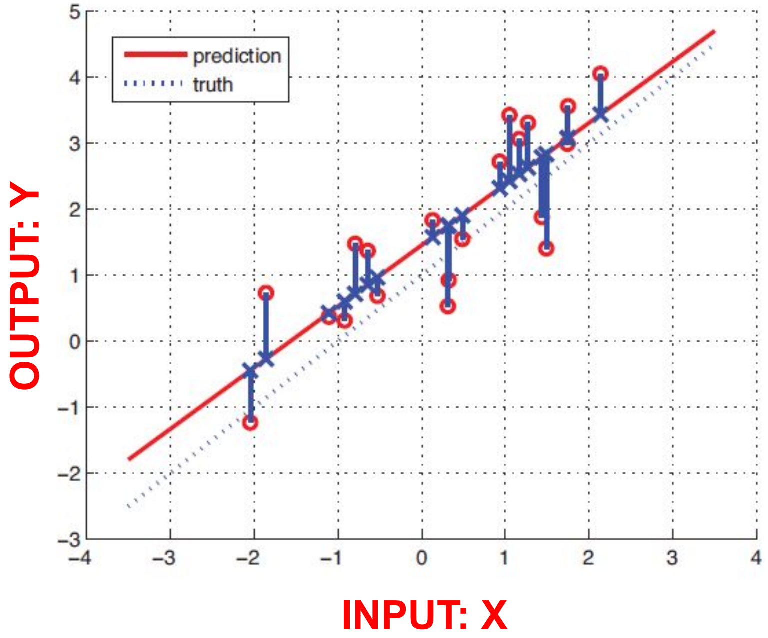
$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

The model thus predicts the class *negative* for the test sentence.

Linear Regression

Linear Regression



Regression Learn a function that predicts outputs from inputs,

$$y = f(x)$$

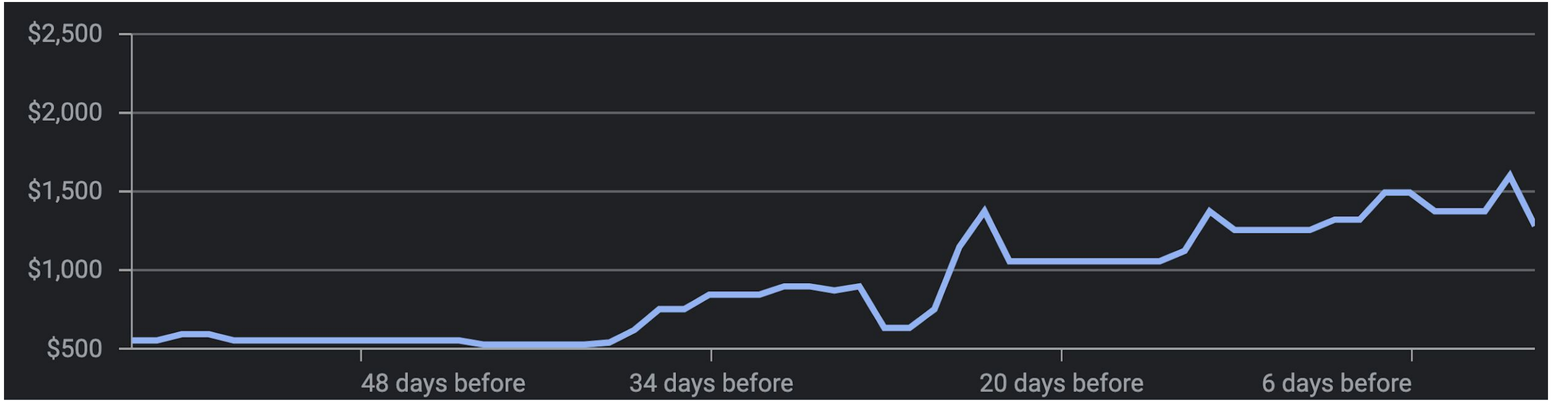
Outputs y are real-valued

Linear Regression As the name suggests, uses a *linear function*:

$$y = w^T x + b$$

$$w^T x := \sum_{d=1}^{D} w_d x_d$$

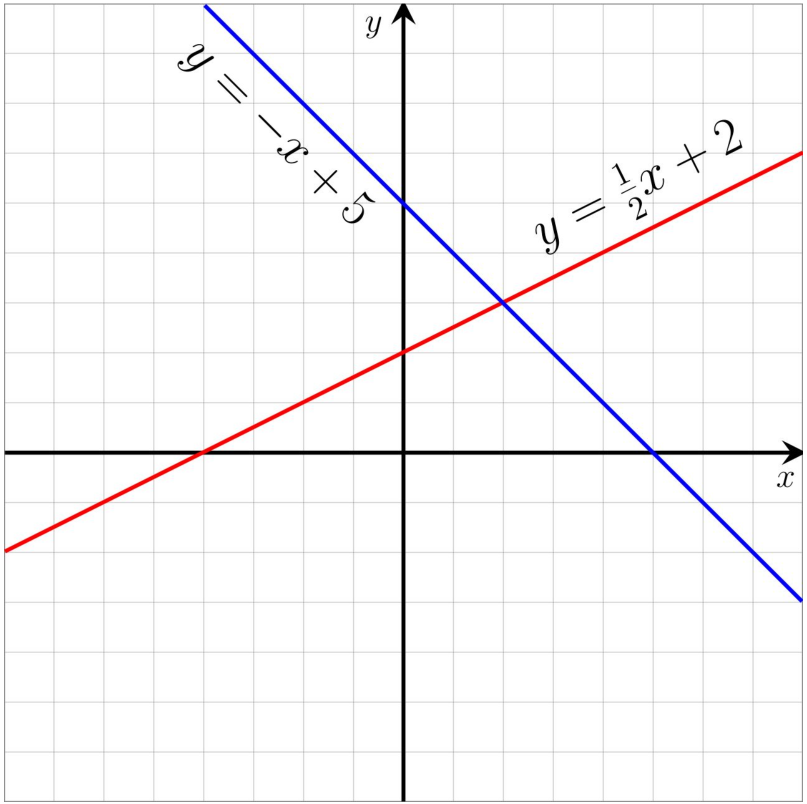
When is linear regression useful?



Price of an airline ticket

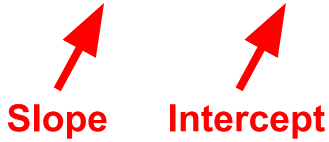
Used anywhere a linear relationship is assumed between inputs / (real-valued) outputs

Line Equation



Recall the equation for a line has a *slope* and an *intercept*,

$$y = w \cdot x + b$$



- Intercept (b) indicates where line crosses y-axis
- Slope controls angle of line
- Positive slope (w) → Line goes up left-to-right
- Negative slope → Line goes down left-to-right

Review: inner product

Two vectors:

$$\vec{x} = \langle 2, -3 \rangle \qquad \mathbf{x} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

$$\vec{y} = \langle 5, 1 \rangle \qquad \mathbf{y} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

Multiply corresponding entries and add:

$$\vec{x} \cdot \vec{y} = \langle 2, -3 \rangle \cdot \langle 5, 1 \rangle = (2)(5) + (-3)(1) = 7$$

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} 2 & -3 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \end{bmatrix} \quad \text{(or just 7)} \quad \text{(so } \vec{x} \cdot \vec{y} \text{ becomes } \mathbf{x}^T\mathbf{y}\text{)}$$

Moving to higher dimensions…

- **1d regression**: regression with 1d input:

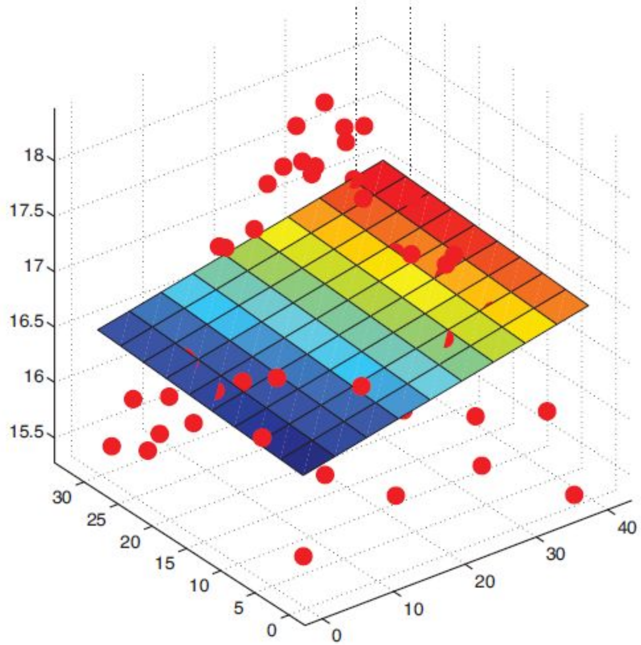
$$y = wx + b$$

- **D-dimensional regression**: input vector is $x \in \mathbb{R}^D$.

Recall the definition of an *inner product*:

$$w^T x = w_1 x_1 + w_2 x_2 + \ldots + w_D x_D \;= \sum_{d=1}^{D} w_d x_d$$

The model is $\quad y = w^T x + b$

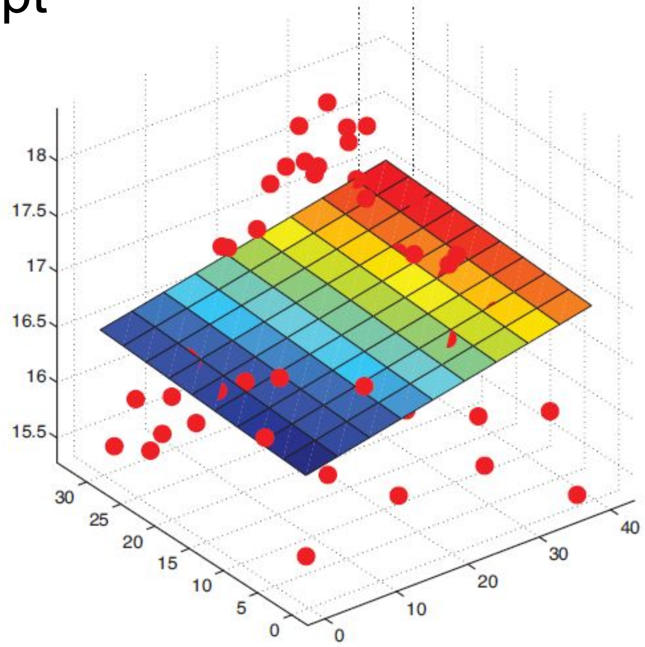


[Image: Murphy, K. (2012)]

Moving to higher dimensions…

Often we simplify this by including the intercept into the weight vector,

$$\widetilde{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_D \\ b \end{pmatrix} \qquad \widetilde{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{pmatrix} \qquad y = \widetilde{w}^T \widetilde{x}$$



Since:

$$\widetilde{w}^T \widetilde{x} = \sum_{d=1}^{D} w_d x_d + b \cdot 1$$
$$= w^T x + b$$

from now on, we assume that $w \in \mathbb{R}^D$ and $x \in \mathbb{R}^D$ already has b and 1 in the last coordinate respectively.

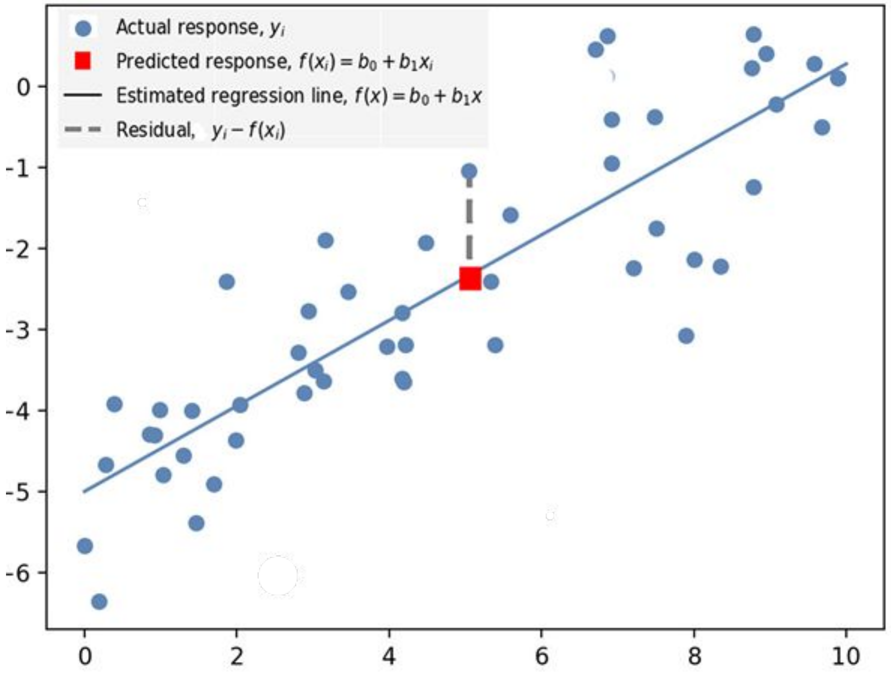
Learning Linear Regression Models

There are several ways to think about fitting regression:

- **Intuitive** Find a plane/line that is close to data
- **Functional** Find a line that minimizes the *least squares* loss
- **Estimation** Find maximum likelihood estimate of parameters

They are all the same thing…

Fitting Linear Regression



Intuition Find a line that is as *close as possible* to every training data point

The distance from each point to the line is the **residual**

$$y - w^T x$$

Training Output **Prediction**

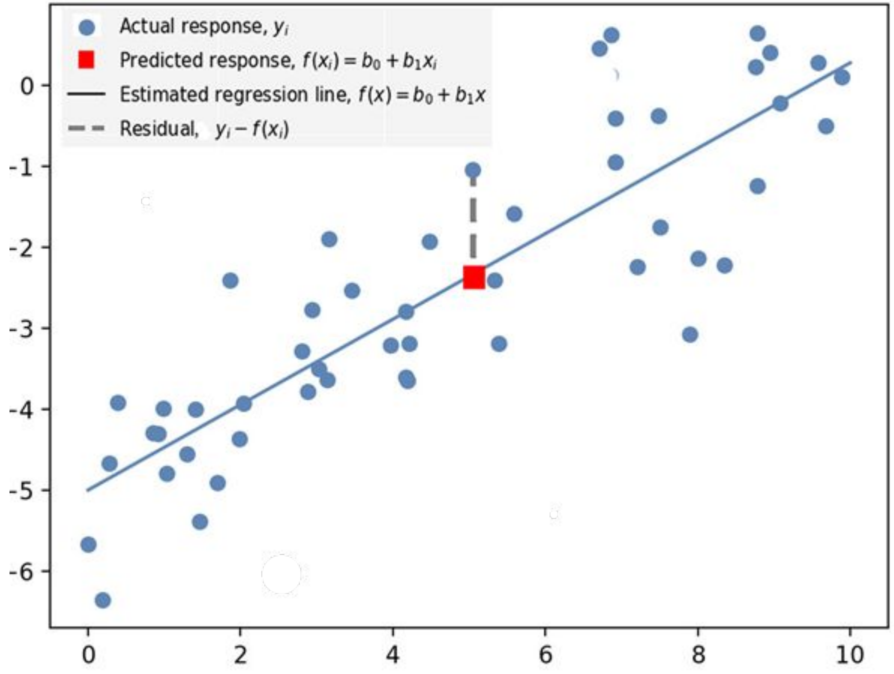
Let's find w that will minimize the residual!

https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

Outline

- Linear Regression
- Least Squares Estimation
- Regularized Least Squares
- Logistic Regression

Least Squares Solution



Functional Find a line that minimizes the sum of squared residuals!

Given: $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$

Compute:

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

Least squares regression

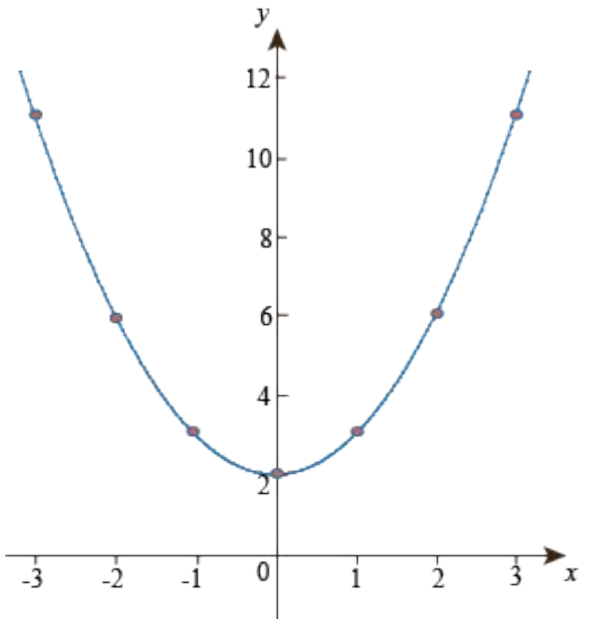
https://www.activestate.com/resources/quick-reads/how-to-run-linear-regressions-in-python-scikit-learn/

Least Squares

$$\min_w \sum_{i=1}^{N} (y^{(i)} - w^T x^{(i)})^2$$

This is just a quadratic function…

- *Convex,* unique minimum
- Minimum given by zero-derivative
- Can find a closed-form solution



Let's see for scalar case with no bias,

$$y = wx$$

Least Squares : Simple Case

$$\frac{d}{dw}\sum_{i=1}^{N}(y^{(i)} - wx^{(i)})^2 =$$

Derivative (+ chain rule)

$$= \sum_{i=1}^{N} 2(y^{(i)} - wx^{(i)})(-x^{(i)}) = 0 \Rightarrow$$

Distributive Property
(and multiply -1 both sides)

$$0 = \sum_{i=1}^{N} y^{(i)} x^{(i)} - w \sum_{j=1}^{N} (x^{(j)})^2$$

Algebra

$$w = \frac{\sum_i y^{(i)} x^{(i)}}{\sum_j (x^{(j)})^2}$$

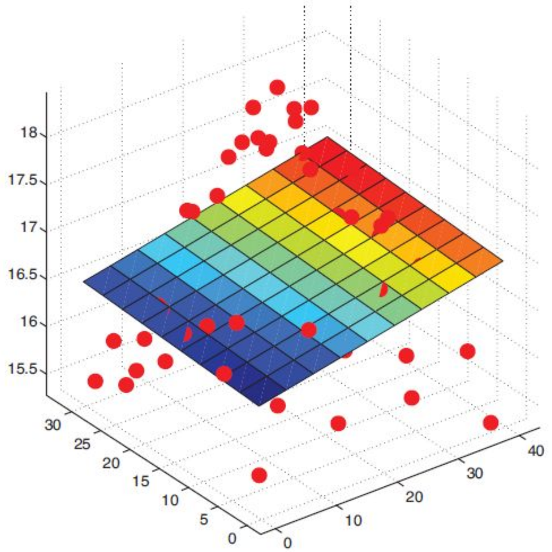
Least Squares: Higher Dimensions

Things are a bit more complicated in higher dimensions and involve more linear algebra,

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & \dots & x_D^{(1)} & 1 \\ x_1^{(2)} & \dots & x_D^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(m)} & \dots & x_D^{(m)} & 1 \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{pmatrix}$$

Design Matrix
(each row is a data point)

Vector of labels



Can write regression over *all training data* more compactly…

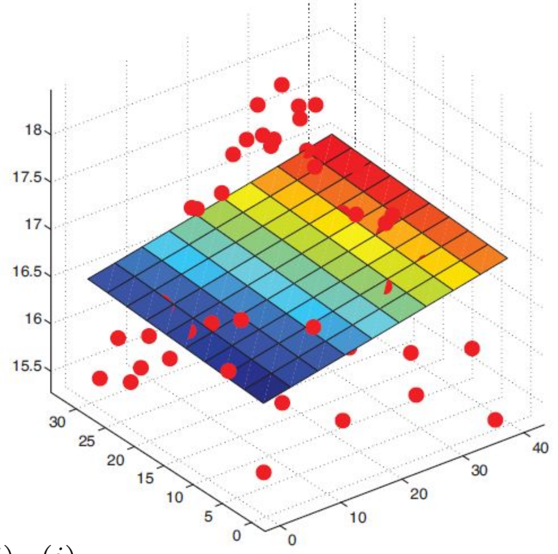
$$\mathbf{y} \approx \mathbf{X}w$$

mx1 Vector

$$= \begin{pmatrix} (x^{(1)})^\top w \\ \dots \\ (x^{(m)})^\top w \end{pmatrix}$$

Least Squares: Higher Dimensions

Least squares can also be written more compactly, $\boxed{\|\boldsymbol{x}\| := \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}.}$



$$\min_w \sum_{i=1}^{N} (y^{(i)} - w^T x^{(i)})^2 = \|\mathbf{y} - \mathbf{X}w\|^2$$

Some slightly more advanced linear algebra gives us a solution,

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

compare with the 1d version: $w = \frac{\sum_i y^{(i)} x^{(i)}}{\sum_j (x^{(j)})^2}$

Ordinary Least Squares *(OLS)* solution

Derivation a bit advanced for this class, but enough to know

- it has a closed-form and why
- we can evaluate it
- generally know where it comes from.