

Lecture 6: Regression I: Theory and Practice

Modeling Social Data, Spring 2017

Columbia University

Jin Peng

March 2, 2017

1 Introduction

Framework for Models:

- Specify outcome and predictors
- Actually a difficult part (usually handed to you)
- Define loss function
- How close model predicts compared with observed data
- Develop algorithm to find the best model
- Minimize loss function (searching across all possible methods)
- Assess model performance + results

Regressions:

$$Outcomes : \{y_i\}_{i=1}^N \quad (1)$$

$$Predictors : \{x_i\}_{i=1}^N \quad (Input/Features) \quad (2)$$

X can be a vector of multiple dimensions, or features

Figure 1 k-dimensional vector

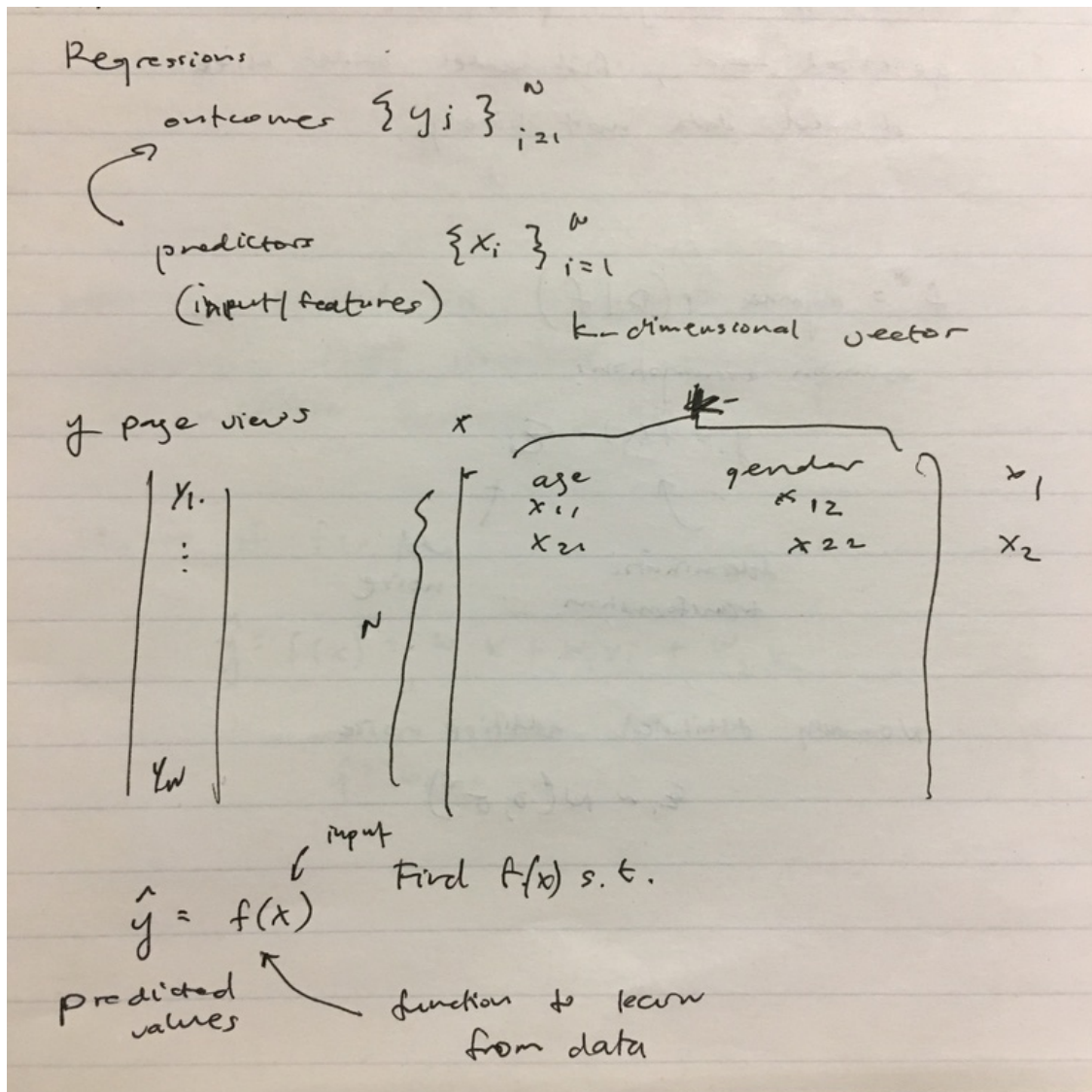


Figure 1: Aim to learn coefficients for vector x to predict y .

Figure 2 Define a loss function to minimize

The image shows handwritten notes on lined paper. At the top, the text "Loss function" is underlined. Below it, the formula $L_i[f] =$ is written. To the right of this, the text "squared loss" is written above the formula $(y_i - \hat{y}_i)^2$. Under the \hat{y}_i term, there is a bracket pointing to the text $f(x_i)$. Below these, the text "total loss = average loss over points" is written. At the bottom, the formula $L[f] = \frac{1}{N} \sum_{i=1}^N L_i[f]$ is written.

Loss function

$L_i[f] =$

squared loss

$(y_i - \hat{y}_i)^2$

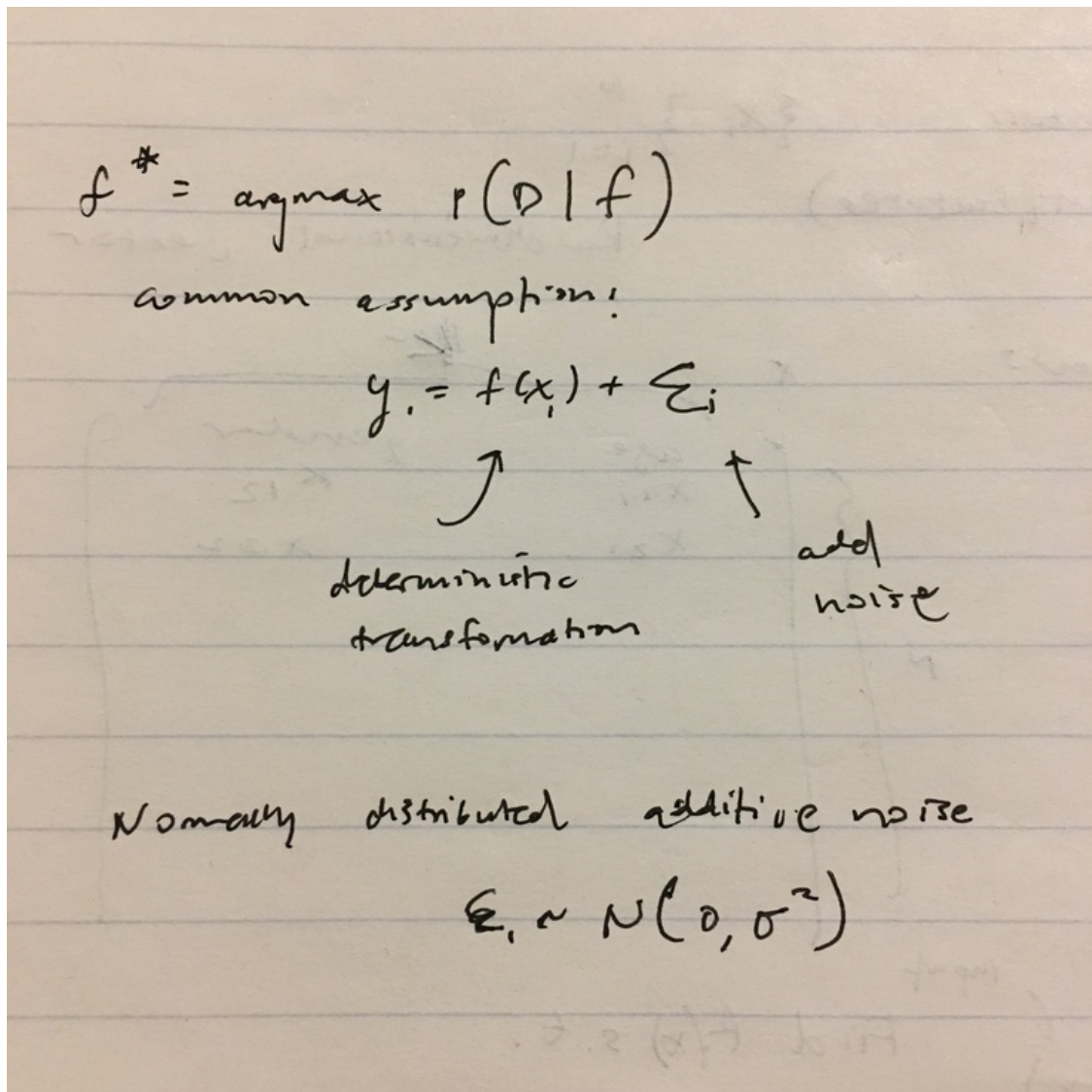
\hat{y}_i
 $f(x_i)$

total loss = average loss over points

$$L[f] = \frac{1}{N} \sum_{i=1}^N L_i[f]$$

Figure 2: Minimizing loss function allows to find coefficients with least error.

Figure 3 Maximum Likelihood


$$f^* = \operatorname{argmax} p(D|f)$$

common assumption:

$$y_i = f(x_i) + \epsilon_i$$

deterministic transformation add noise

Normally distributed additive noise

$$\epsilon_i \sim N(0, \sigma^2)$$

Figure 3: Assume some family of probabilistic models generated the data - we find the model under which observed data most likely

Figure 4 Maximizing given error

maximizing
given error

$$p(\epsilon_i | f) = p(y_i - f(x_i) | f)$$

$$= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y_i - f(x_i))^2}$$

Likelihood

$$p(D|f) = \prod_{i=1}^N p(D_i|f) = \prod_{i=1}^N$$

\Rightarrow simplify

$$p(D|f) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - f(x_i))^2\right\}$$

$$\log p(D|f) = \underbrace{-\frac{N}{2} \log 2\pi\sigma^2}_{\text{indep of } f} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - f(x_i))^2}_{-\mathcal{L}[f]}$$

Figure 4: Minimizing squared loss is equivalent to maximizing (log) likelihood, assuming additive Gaussian noise.

Figure 5

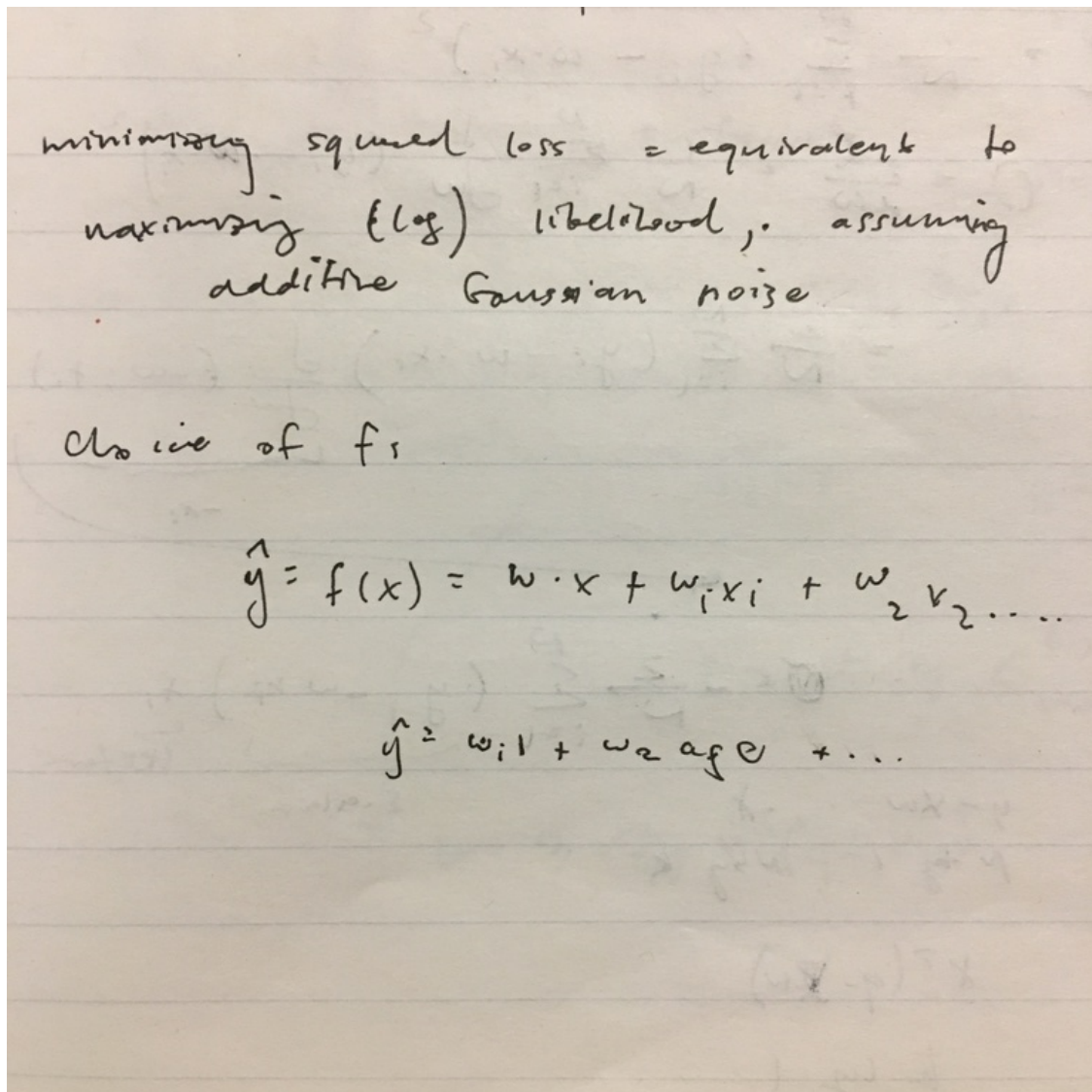


Figure 5: Minimizing squared loss is equivalent to maximizing (log) likelihood, assuming additive Gaussian noise.

Figure 6 Two Dimensional Representation of Loss Function

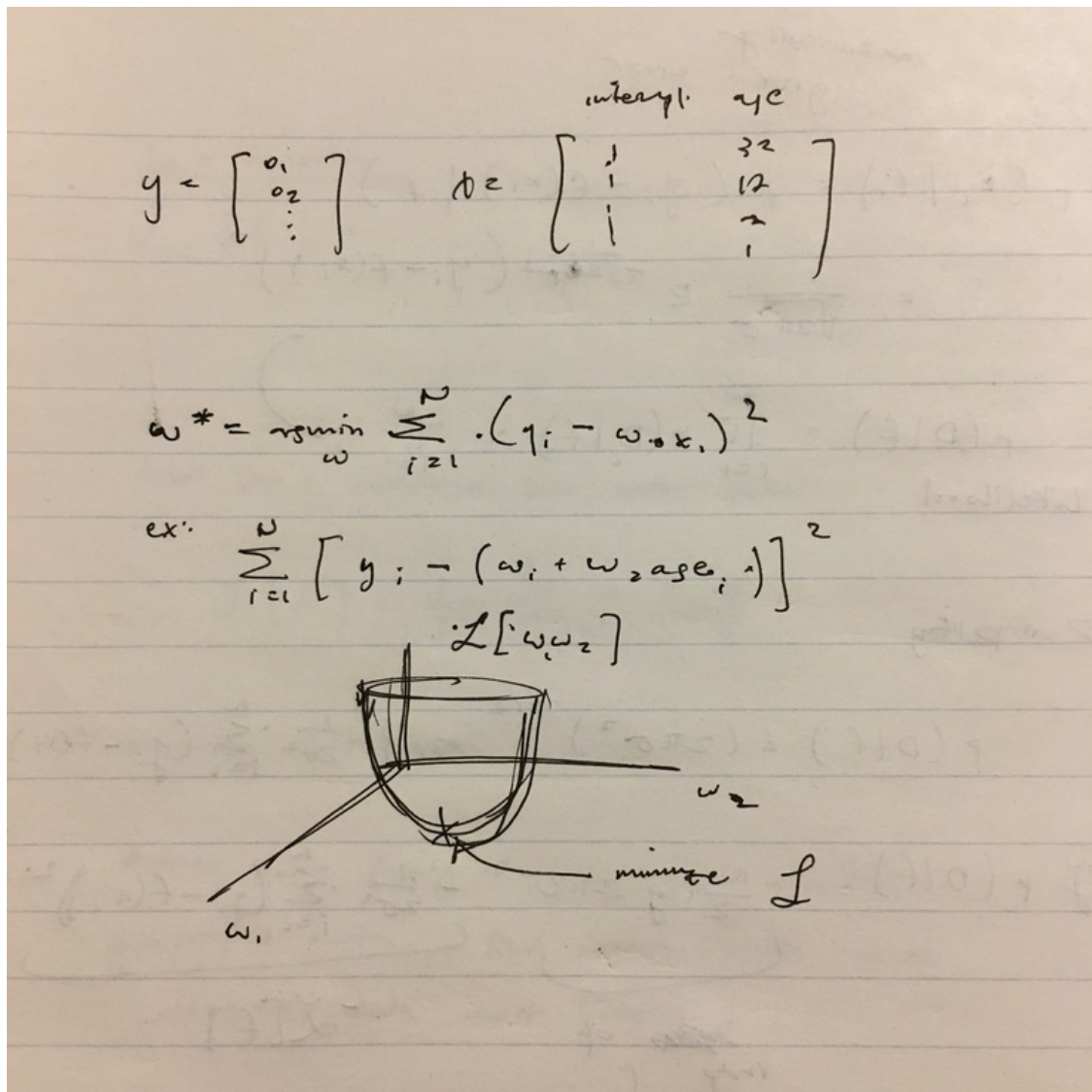


Figure 6: If we are working with two dimensions(labels), easier to visualize minimizing L to find coefficients with least loss

Figure 7 Mathematical derivation

$$\begin{aligned}
 L &= \frac{1}{N} \sum_{i=1}^N (y_i - w \cdot x_i)^2 \\
 0 &= \frac{dL}{dw} = \frac{1}{N} \sum_{i=1}^N \frac{d}{dw} (y_i - w \cdot x_i)^2 \\
 &= \frac{2}{N} \sum_{i=1}^N (y_i - w \cdot x_i) \underbrace{\frac{d}{dw} (-w \cdot x_i)}_{-x_i} \\
 0 &= -\frac{2}{N} \sum_{i=1}^N \underbrace{(y_i - w \cdot x_i)}_{\text{Scalar}} \underbrace{x_i}_{\text{Vector}}
 \end{aligned}$$

$\frac{dL}{dw}$
 $k \text{ by } 1$

$y - Xw$
 $N \text{ by } 1$

$\cdot x$
 $N \text{ by } k$

$x^T (y - Xw)$
 $k \text{ by } 1$

Figure 7: Take derivative of loss function (averaged over all values) and set to 0 to solve for our vector of coefficients.

Figure 8 Converting to matrix operations

$$0 = X^T (y - Xw)$$
$$X^T y = X^T X w$$
$$\hat{w} = (X^T X)^{-1} X^T y$$
$$O(Nk^2 + k^3)$$

↑
cubic
to invert
 $X^T X$

problematic
when you have
wide data (k)
not number of observations,
width of data

Gradient Descent

Figure 8: Since we are working with vectors, we can use linear algebra and matrix multiplication to solve for our coefficient vector w . Note, it becomes problematic when we have wide data (high dimensions k) as matrix operations are extremely expensive in this manner - number of observations is less impactful on runtime than width of data

Figure 9 Gradient Descent

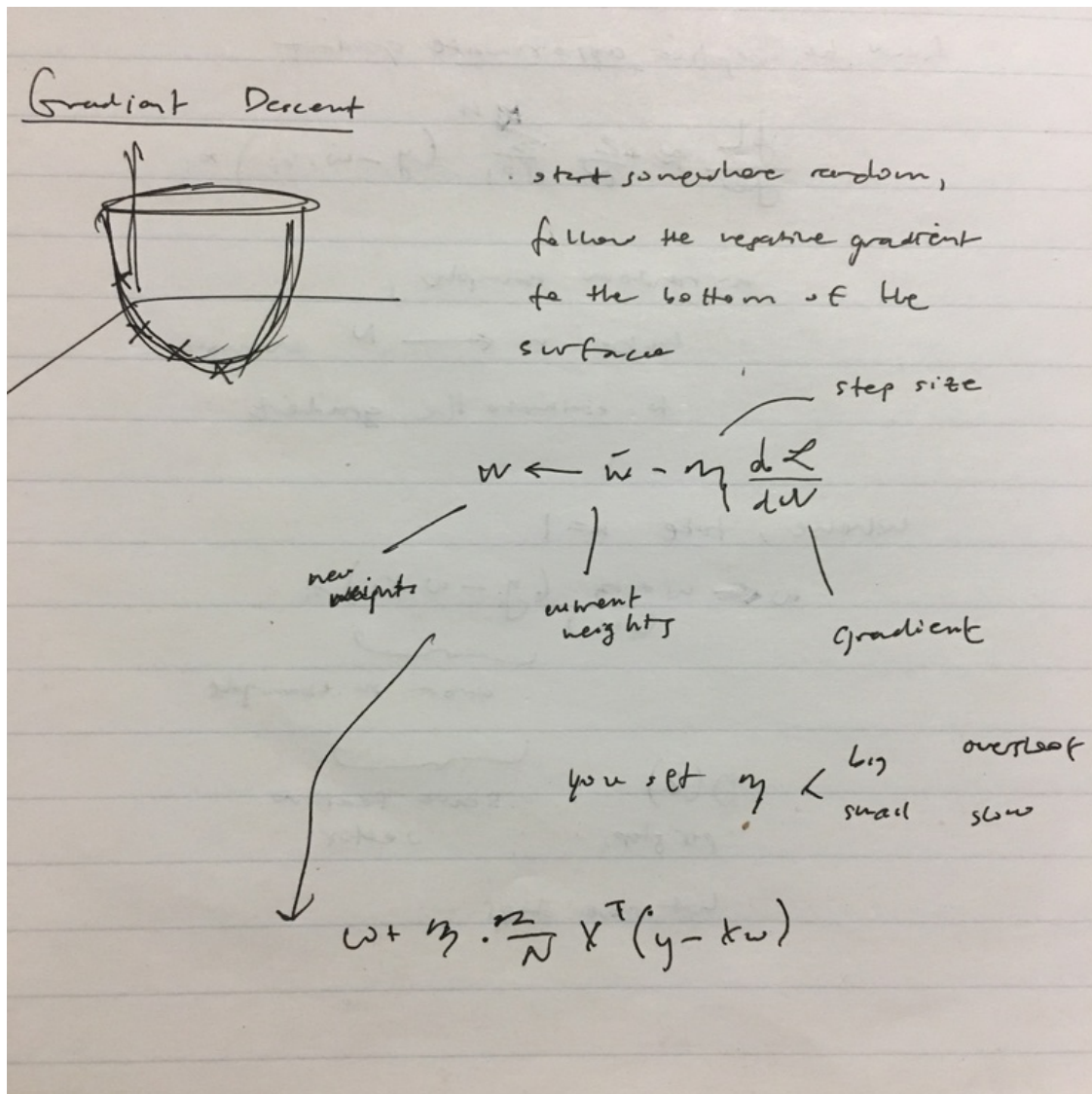


Figure 9: Start somewhere random, follow the negative gradient to the bottom of the surface. You set the learning rate (can be variable) - too big and it will overshoot the minimum, too small and gradient descent will take a long time

Figure 10 Vector of residuals/errors

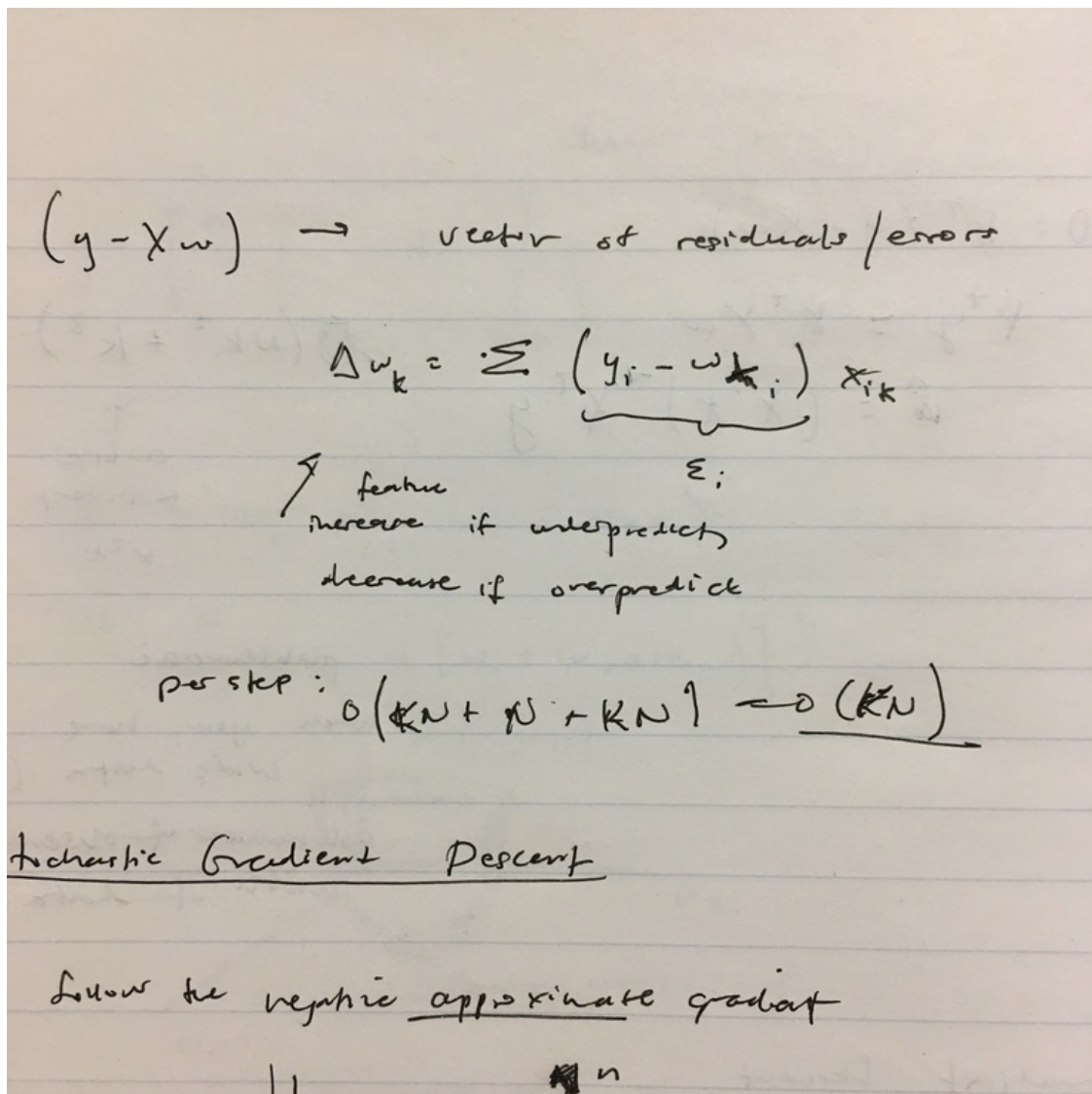


Figure 10: Increase features w if underpredict, decrease features w if overpredict

Figure 11 Stochastic Gradient Descent

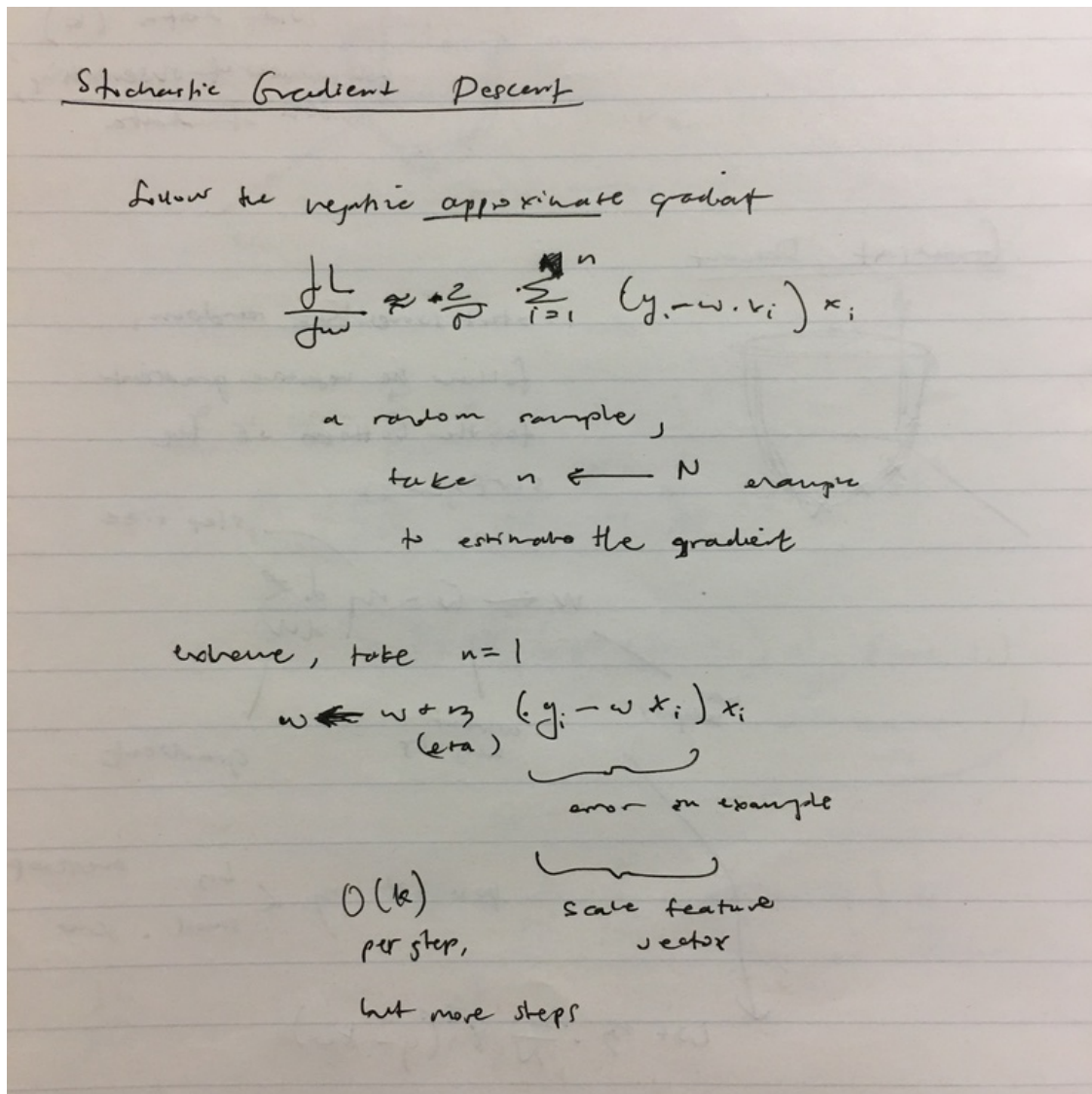


Figure 11: Follow the negative approximate gradient - take an n random sample to estimate the gradient