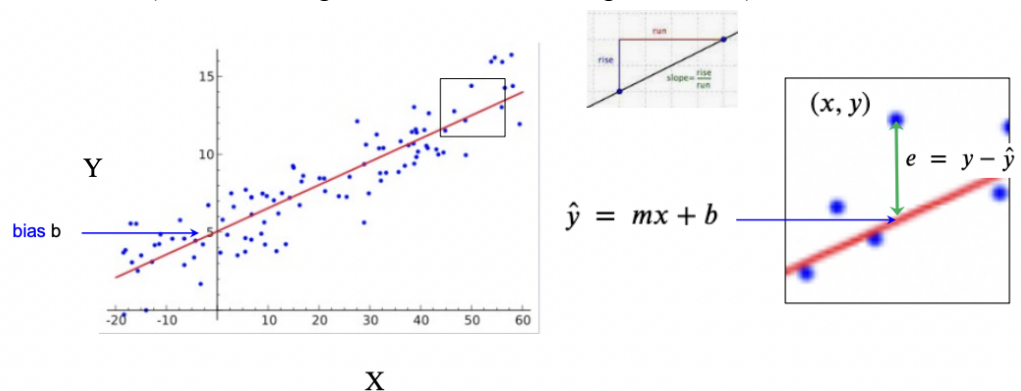


# Prelude to Deep Learning: Linear Regression, Logistic Regression, Classification with Logistic Regression, Finding Solutions with Gradient Descent

## 1. Linear Regression

- a. Linear regression relates some number of independent variables (real numbers)
  - i.  $X_1, X_2, \dots, X_n$
 with a dependent or response variable  $Y$ . The output is the set of parameters (here, slope  $m$  and bias  $b$ ) showing the best approximation of the linear relationship of the variables (model is simplified down into two parameters)



- b.
- c. Model is a simplification of the data
- d. Linear regression can be calculated for any number of dimensions with a magically simple formula from linear algebra:

We thus have  $Y = X \cdot W + E$  or

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \times \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix}.$$

The least-squares estimates for  $W$  are given by the following formula:

$$W = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = (X^T X)^{-1} X^T Y$$

- e.
- f.  $e$  is errors – you collect all of them and use them to check how good the model is ( $e = y - \hat{y}$ )

## 2. Introduction to Deep Learning

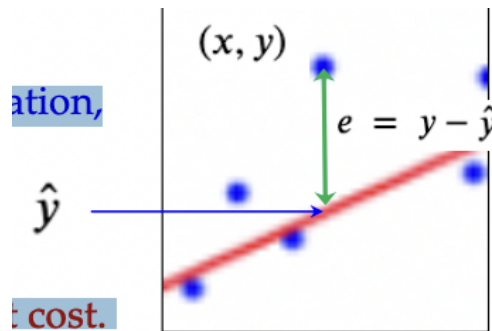
- a. Linear Regression: What is “least” about the least-squares approximation of a line?
- b. In linear regression, we define the error of the prediction as the MSE (mean squared errors) of the predictions

$$\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

```
def MSE(X, Y, m):
    return np.mean( [ (Y[k] - m*X[k])**2 for k in range(len(X)) ] )
```

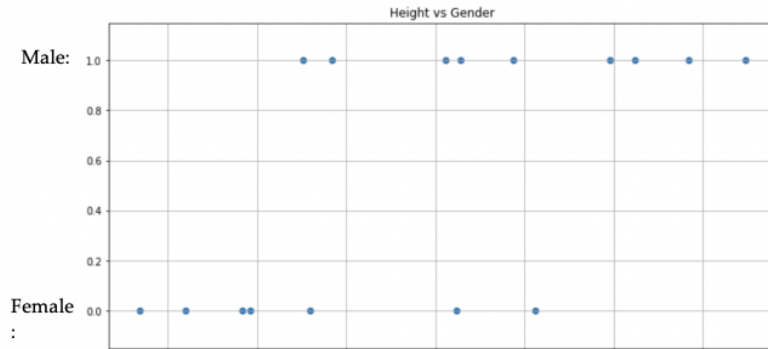
$$MSE = \frac{1}{n} \sum_{i=1}^n e_i^2 = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

- c.
- d. We seek the least amount of error in the approximation, hence the “least squares” line. In machine learning, we generally call this the cost function, so we seek an approximation of least cost.



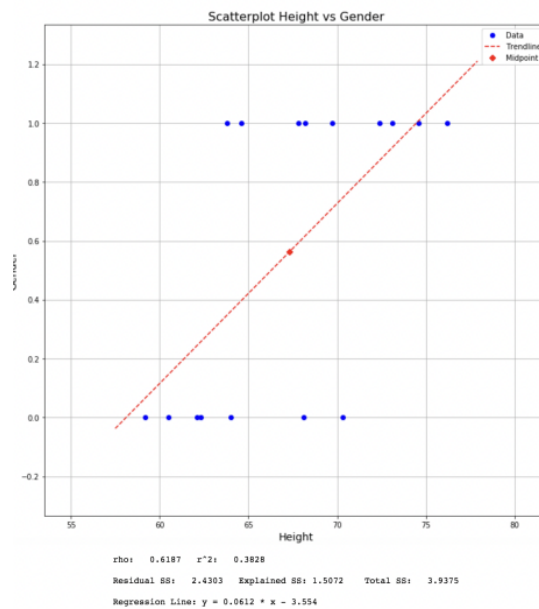
- e.
- f. Linear Regression: Using Gradient Descent to find minimal-cost solution
- g. The magic formula  $W = (X^T X)^{-1} X^T Y$  gives us an analytic solution with the smallest possible cost.
- h. But what if there is no magic formula??
- i. If there is no analytical solution (a formula), then we must use a search algorithm called Gradient Descent to find the parameter values which minimize this cost.
- j. We'll return to Gradient Descent in a bit, but let's look at the most important application of regression to classification problems....
- k. Logistic Regression: A Motivating Example
  - i. But linear regression doesn't work for many problems, especially in NLP! Suppose we attempt to classify 16 people as male or female depending on a single feature: their height. Men in general are taller than women (the average height of an American man is 5' 9" and for women 5' 4"), X = height against Y = gender (1 for male, 0 for female):

Heights: [59.2, 60.5, 62.1, 62.3, 63.8, 64.0, 64.6, 67.8, 68.1, 68.2, 69.7, 70.3, 72.4, 73.1, 74.6, 76.2]  
 Gender: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]



ii.

iii. If we plug this into the linear regression algorithm, we get the following:



iv.

v. There are many issues with this:

1. How can we use this to predict someone's gender from their height?
2. How to give the probability of their gender?
3. There is clearly no linear trend, so what does the line even mean?

l. Logistic Regression: The Logit Transformation

m. In order to solve this, we will transform the scale of Y into a new domain, in this case into the real interval [0..1] used for probabilities. This is called the Logit Transformation and is based on the notion of a sigmoid function  $s: \mathbb{R} \rightarrow [0..1]$  of the form

$$s(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

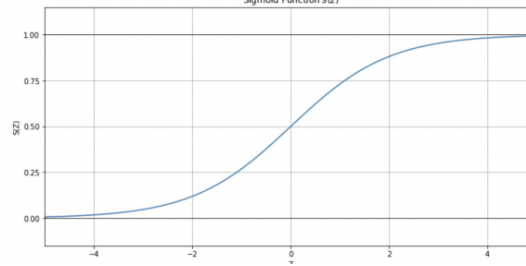
n.

```
def s(z):  
    return 1/(1+np.exp(-z))
```

$$\lim_{z \rightarrow \infty} \frac{1}{1 + e^{-z}} = 1$$

$$\lim_{z \rightarrow -\infty} \frac{1}{1 + e^{-z}} = 0$$

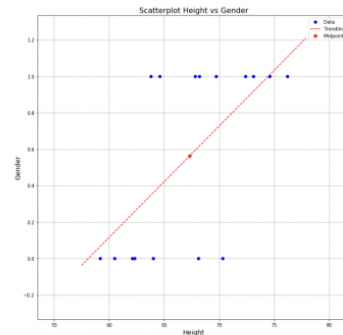
$$s(0) = \frac{1}{1 + e^0} = \frac{1}{1 + 1} = 0.5$$



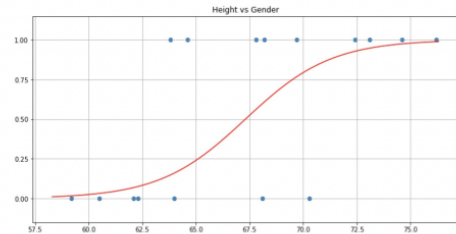
o.

### 3. Logistic Regression: The Logit Transformation

**Linear Regression:**  $-\infty < Y < \infty$



**Logistic Regression:**  $0 \leq Y \leq 1$



$$s(z) = \frac{1}{1 + e^{-z}}$$

$$s : \mathcal{R} \rightarrow [0..1]$$

$$s^{-1} : [0..1] \rightarrow \mathcal{R}$$

$$s^{-1}(p) = \ln\left(\frac{p}{1-p}\right)$$

This is called the “logit” or the “log odds ratio.”

$$y = \frac{e^x}{1 + e^x}$$

$$\Leftrightarrow y + ye^x = e^x$$

$$\Leftrightarrow y = e^x - ye^x$$

$$\Leftrightarrow y = (1 - y)e^x$$

$$\Leftrightarrow \frac{y}{(1 - y)} = e^x$$

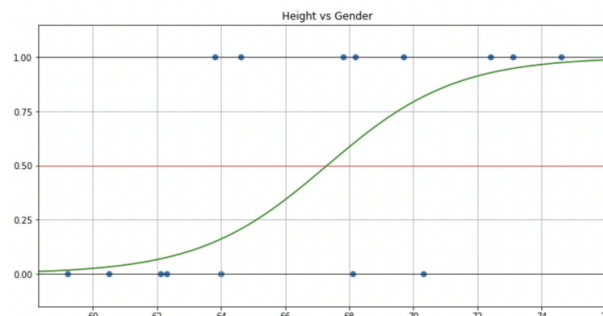
$$\Leftrightarrow \ln\left(\frac{y}{1 - y}\right) = x$$

$$\Leftrightarrow s^{-1}(p) = \ln\left(\frac{p}{1 - p}\right)$$

a.

b. Now, we have probabilities (if we have height, it gives p value for men and women) and preserves the mean


c. The punchline here is that we will transform the regression line into a sigmoid, and use it to give us the probability that a given individual is male, and then define as a decision boundary a threshold (typically 0.5) by which we will decide if the binary output is 1 or 0:



d.

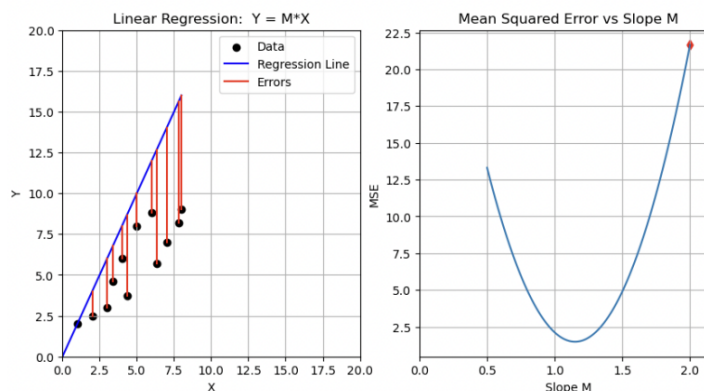
- e. Now, we can go back and forth between the logistic and linear (we can turn the probability into numbers)
    - i. 1 for men, 0 for women (label) → round the probabilities to match
  - f. Caveat: Such decision boundaries are typically not used in neural networks, so the output is between 0 and 1
  - g. However, there is no analytical solution (no magic formula!) once we use the logit transformation, so we will need to use gradient descent to minimize an appropriate cost function.
4. Linear Regression Redux: Gradient descent to find  $\hat{\theta}_0$  and  $\hat{\theta}_1$
- a. In linear regression, we have explicit formulae for finding the parameters for the slope  $m$  and bias  $b$  of the regression line which minimizes the MSE.
  - b. But what if we didn't? We could then use an iterative approximation algorithm called Gradient Descent to find an approximation of the values which minimize the MSE.
  - c. Basic idea: Define a cost or loss function  $J(\dots)$  which gives the cost or penalty measuring how well the model parameters fit the actual data (high cost = bad fit), and then search for the parameters which minimize this cost.

$$J(\hat{m}, \hat{b}) = \frac{1}{N} \sum_{i=1}^N (y_i - (\hat{b} + \hat{m}x_i))^2$$


  
 Cost Function  
 = MSE

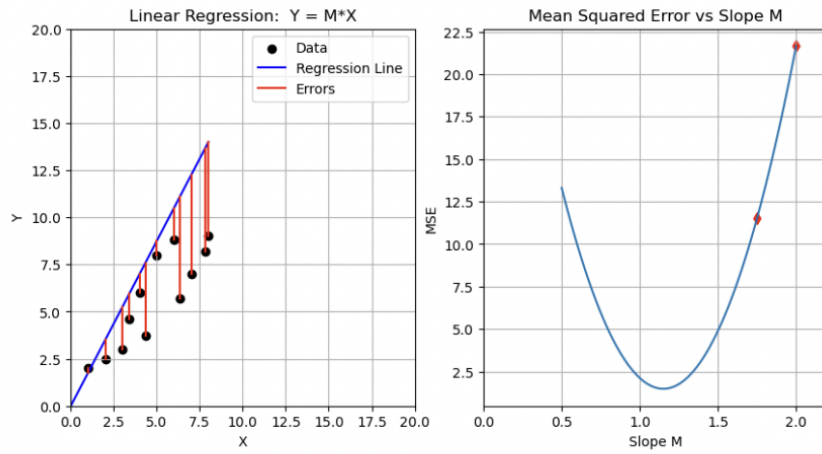
- d.
  - e. The  $J$  in the cost function is used in machine learning and refers to the Jacobian Matrix
5. Introduction to deep learning

- a. Here's a very simple example: Suppose we want to find a regression line satisfying  $Y = m \cdot X$  (i.e., there is no bias term  $b$ ). The cost function is quadratic, so we get a parabola when we graph the slope  $M$  against the MSE:

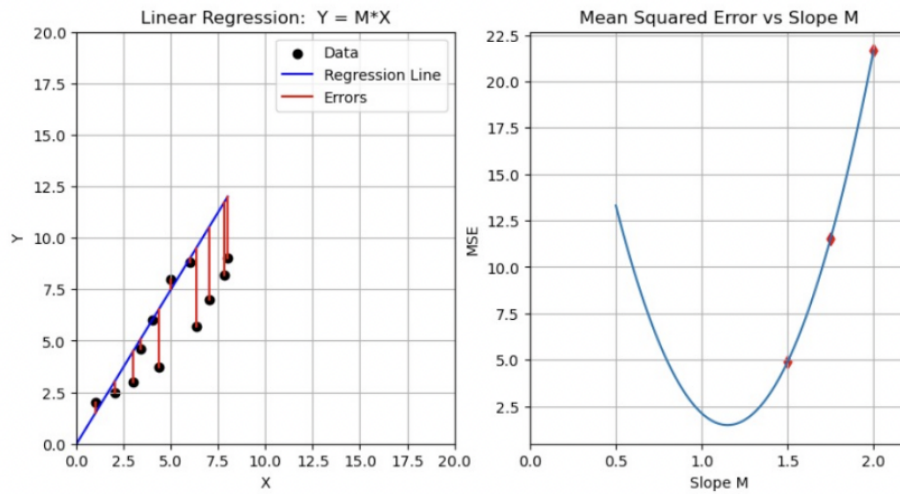


b.

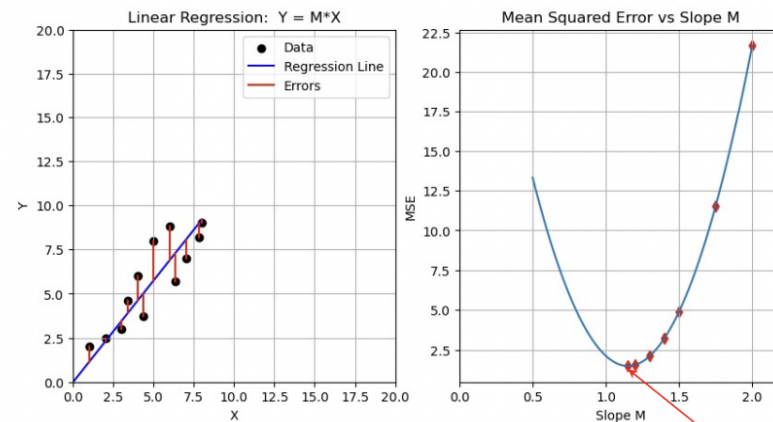
- c. Gradient Descent is an iterative approximation algorithm, which "tweaks" the parameters to move in the direction of lower cost (smaller errors).



d.



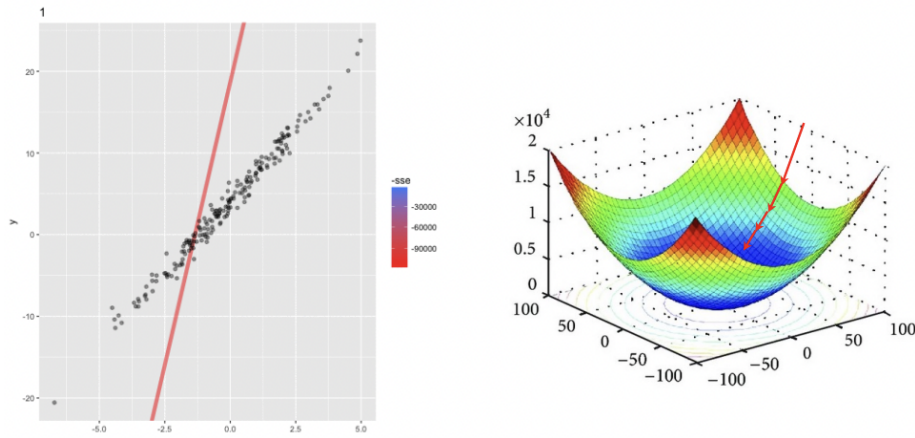
e.



f.

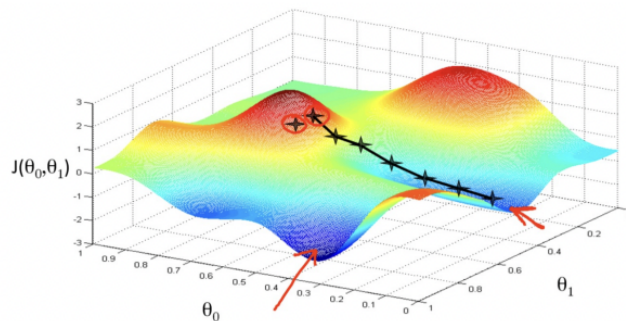


- g. With MSE, we always get a convex cost function, even in higher dimensions:

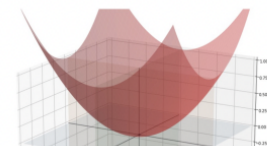


## 6. Linear Regression Redux: Gradient Descent

- The Gradient Descent Algorithm: A gradient is a generalization of a derivative to functions of more than one variable:
- “Like the derivative, the gradient represents the slope of the tangent of the graph of the function. More precisely, the gradient points in the direction of the greatest rate of increase of the function, and its magnitude is the slope of the graph in that direction.” - Wikipedia
- In gradient descent, we pick a place to start, and move down the gradient until we find a minimum point:



When the search space is convex, such as a paraboloid, there will be a single minimum!



d.

## 7. Linear Regression Redux: Gradient Descent to find m and b

- To find the minimum value along one axis we will work with only one of the partial derivatives at a time, say the bias b:

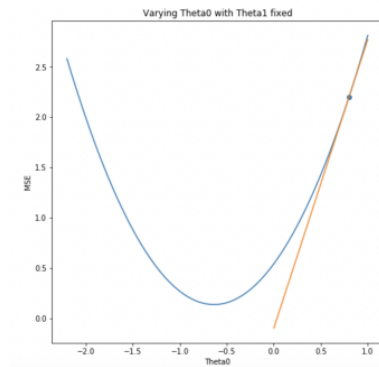
$$J'(b, m) = \begin{bmatrix} \frac{1}{N} \sum_{i=1}^N -2(y_i - (b + mx_i)) \\ \frac{1}{N} \sum_{i=1}^N -2x_i(y_i - (b + mx_i)) \end{bmatrix}$$

Partial derivative of cost function with respect to parameter b.

b.

- Step One: Choose an initial point  $b_0$ .

- ii. Step Two: Choose a step size or learning rate  $\lambda$  and threshold of accuracy  $\epsilon$ .
- iii. Step Three: Move that distance along the axis, in the decreasing direction (the negative of the slope), and repeat until the distance moved is less than  $\epsilon$ .
- iv. Step Four: Output  $b_{n+1}$  as the minimum.



V.

1. Choose  $b_0$ ;
2. Choose  $\lambda$ ;
3. Repeat  $b_{n+1} = b_n - J'(b_n) \cdot \lambda$   
Until  $|b_{n+1} - b_n| < \epsilon$
4. Output  $b_{n+1}$ .

## 8. Linear Regression Redux: Gradient descent to find $\hat{\theta}_0$ and $\hat{\theta}_1$

- a. Gradient Descent for Linear Regression:
- b. To find a point in multiple dimensions, we simply do all dimensions in the same way at the same time. Here is the algorithm:

```
def update_weights(m, b, X, Y, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        # Calculate partial derivatives
        # -2x(y - (mx + b))
        m_deriv += -2*X[i] * (Y[i] - (m*X[i] + b))

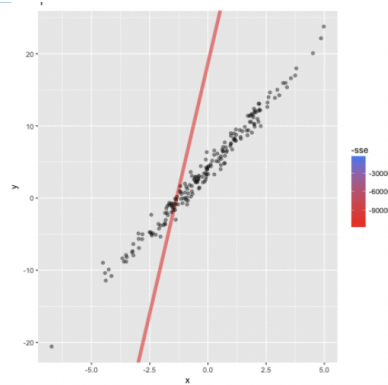
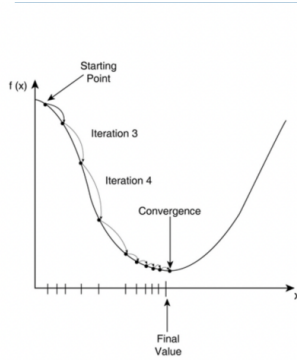
        # -2(y - (mx + b))
        b_deriv += -2*(Y[i] - (m*X[i] + b))

    # We subtract because the derivatives point in direction of steepest ascent
    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate

    return m, b
```

- c.
- d. As the parameters are “tuned” to minimize the cost (= measuring how well the parameters fit the model) you get a better and better fit between the model and the data. You can run the gradient descent model as long as you wish to get a better fit. Obviously, defining the cost function and picking the learning rate and threshold are critical decisions, and much research has been devoted to different cost models and different approaches to gradient descent





e.

## 9. Text Classification: Definition

a. Input:

- i. a document  $d$
- ii. a fixed set of labels/classes  $C = \{c_1, c_2, \dots, c_J\}$

b. Output: a predicted class  $c$  in  $C$

c. Caveats: In general, an algorithm will return probabilities for all document classes: this can be used to find the single best class, or—by setting a threshold or a bound on the number of classes—a set of classes.

## 10. Classification Methods: Hand-coded rules

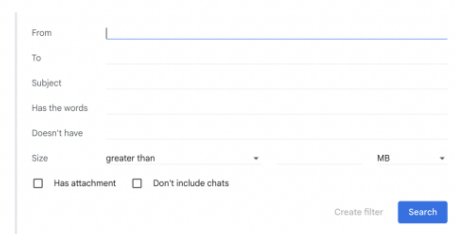
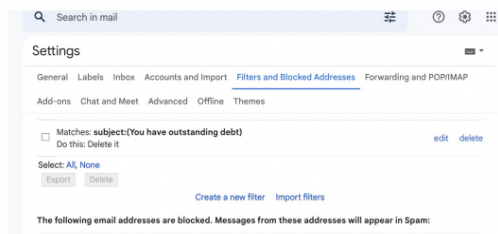
a. Rules based on combinations of words or other features

- i. spam: black-list-address OR (“dollars” AND “you have been selected”)

b. Accuracy can be high

- i. If rules carefully refined by expert

c. But building and maintaining these rules is expensive



d.

## 11. Classification Methods: Supervised ML

a. Input:

- i. a fixed set of classes  $C = \{c_1, c_2, \dots, c_J\}$
- ii. a randomly-permuted set of labeled documents  $(d_1, c_1), \dots, (d_n, c_n)$  split into
  1. a training set  $(d_1, c_1), \dots, (d_m, c_m)$
  2. a testing set  $d_{m+1}, \dots, d_n$  (labels withheld) §

b. Output:

- i. A classifier  $\gamma : d \rightarrow c$  trained the training set
- ii. The testing set with labels calculated by  $\gamma$
- iii. Test results (confusion matrix, metrics, etc.)