Data Science

Week 10

Fitting a Model to Data (3)- Regression

Outline

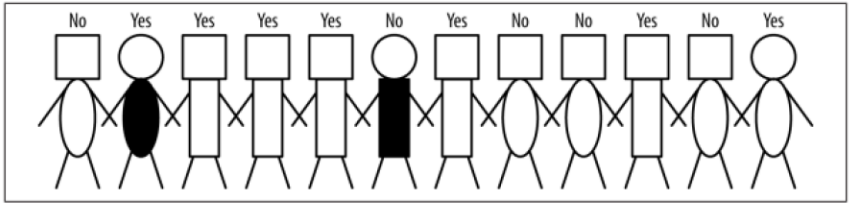
- Recap for predictive model - classification
- Overfitting
- Regression

Predictive Model

- In data science, a predictive model is a formula for estimating the unknown value of interest: the target. The formula could be mathematical, or it could be a logical statement such as a rule.

- Given our division of supervised data mining into classification and regression, we will consider classification models (and class-probability estimation models) and regression models.

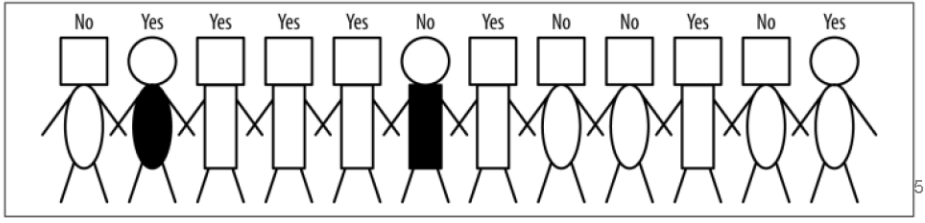
Example for Selecting Informative Attributes -1



- There are two types of heads: square and circular; and two types of bodies: rectangular and oval; and two of the people have gray bodies while the rest are white. These are the attributes we will use to describe the people.
- Above each person is the binary target label, *Yes* or *No*, indicating (for example) whether the person becomes a loan write-off.

How to Create a Tree Model

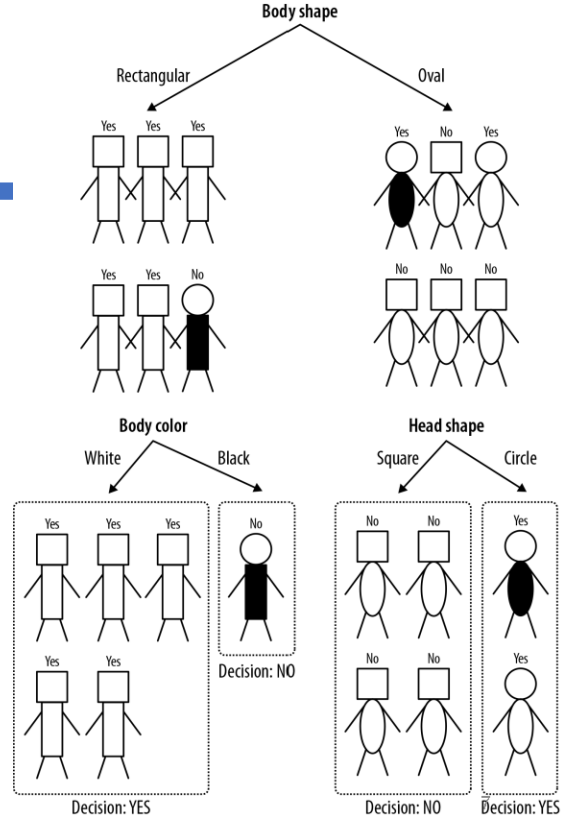
- Combining the ideas introduced above, the goal of the tree is to provide a supervised segmentation—more specifically, to partition the instances, based on their attributes, into subgroups that have similar values for their target variables. We would like for each "leaf " segment to contain instances that tend to belong to the same class.
- Tree induction takes a divide-and-conquer approach, starting with the whole dataset and applying variable selection to try to create the "purest" subgroups possible using the attributes.
- consider the very simple example set shown previously



Summary for creating Tree Model

- In summary, the procedure of classification tree induction is a recursive process of divide and conquer, where the goal at each step is to select an attribute to partition the current group into subgroups that are as pure as possible with respect to the target variable.
- We perform this partitioning recursively, splitting further and further until we are done.
- When are we done? (In other words, when do we stop recursing?) It should be clear that we would stop when the nodes are pure, or when we run out of variables to split on.
- The whole Decision tree is shown in the next page.

Decision Tree



Dataset with a Single Linear Split

- The instance-space view is helpful because if we take away the axis-parallel boundaries (see Figure 1) we can see that there clearly are other, possibly better, ways to partition the space.
- For example, we can separate the instances almost perfectly (by class) if we are allowed to introduce a boundary that is still a straight line, but is not perpendicular to the axes (Figure 2).

Figure 1

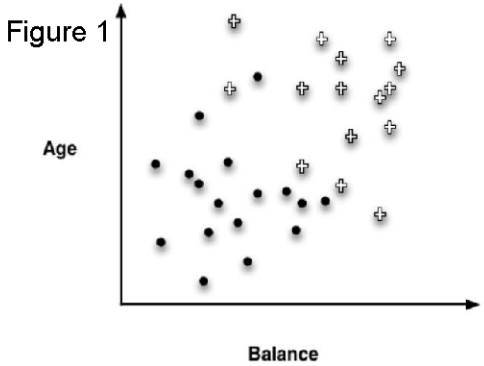
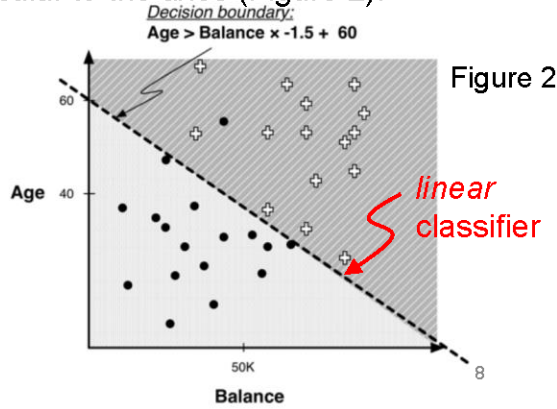


Figure 2



Classification Function

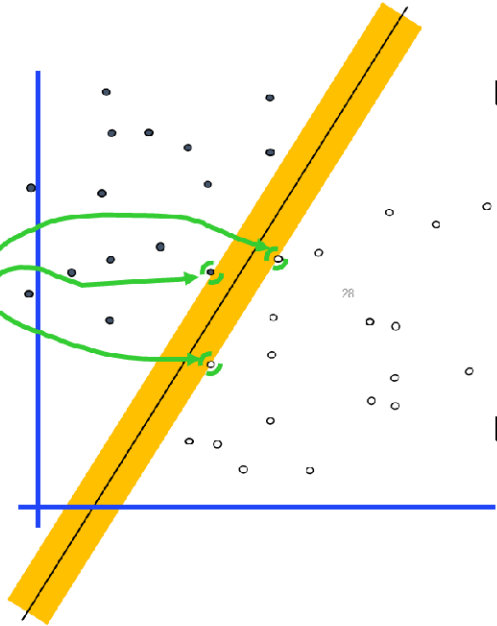
- We would classify an instance x as a + if it is above the line, and as a • if it is below the line. Rearranging this mathematically leads to the function that is the basis of all the techniques discussed in this chapter.
- First, for this example form the classification solution is shown in the following Equation.

Classification function:

$$\text{class}(x) = \begin{cases} + & \text{If} \quad 1.0 \times Age - 1.5 \times Balance + 60 > 0 \\ \bullet & \text{If} \quad 1.0 \times Age - 1.5 \times Balance + 60 \leq 0 \end{cases}$$

Maximum Margin in SVM

- denotes +1
- denotes -1



Support Vectors are those datapoints that the margin pushes up against

- The maximum margin linear classifier is the linear classifier with the maximum margin.
- This is the simplest kind of SVM

Linear SVM Mathematically

- Goal: 1) Correctly classify all training data

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq 1 \quad \text{if } y_i = +1,$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq 1 \quad \text{if } y_i = -1$$
$$y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq 1 \quad \text{for all i}$$

2) Maximize the Margin $M = \frac{2}{\|\mathbf{w}\|}$

same as minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Binary Classification

64 pixel

64 pixel

1 (Cat) vs
0 (Non-Cat)

y

Blue
Green
Red

		255	134	93	22
	255	134	202	22	2
255	231	42	22	4	30
123	94	83	2	192	124
34	44	187	92	34	142
34	76	232	124	94	
67	83	194	202		

Feature

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix}$$

$n_x = 64 \times 64 \times 3 = 12288$

http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression Loss Function

Logistic Regression $\hat{y}^{(i)} = \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right), 0 \leq \hat{y}^{(i)} \leq 1, \quad \sigma(z) = \frac{1}{1+e^{-z}}$

Training set with m samples

$$\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \cdots, \left(\mathbf{x}^{(m)}, y^{(m)}\right)$$

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}, x_0^{(i)} = 1, y^{(i)} \in \{0, 1\}$$

How to choose parameters $\boldsymbol{\theta}$ to make $\hat{y}^{(i)}$ infinitely close to $y^{(i)}$?

Logistic Regression Loss Function

Loss (error) function:

$$L(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)}\ln\left(\hat{y}^{(i)}\right) - (1 - y^{(i)})\ln\left(1 - \hat{y}^{(i)}\right)$$

$L(\hat{y}^{(i)}, y^{(i)}) \geq 0$

if $y^{(i)} = 1, \left(1 - y^{(i)}\right) = 0$

when $L(\hat{y}^{(i)}, y^{(i)}) = -\ln\left(\hat{y}^{(i)}\right) \rightarrow 0, \hat{y}^{(i)} \rightarrow 1$

if $y^{(i)} = 0, y^{(i)} = 0$

when $L(\hat{y}^{(i)}, y^{(i)}) = -\ln\left(1 - \hat{y}^{(i)}\right) \rightarrow 0, \hat{y}^{(i)} \rightarrow 0$

Logistic Regression

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$
$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \ln\left(\hat{y}^{(i)}\right) + (1 - y^{(i)}) \ln\left(1 - \hat{y}^{(i)}\right)\right]$$

Learning: find parameter $\boldsymbol{\theta}$ to $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Prediction: given new x output $\hat{y} = \sigma(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{x}}}$

Gradient Descent

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \ln\left(h_{\boldsymbol{\theta}}\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \ln\left(1 - h_{\boldsymbol{\theta}}\left(x^{(i)}\right)\right)\right]$$

Goal: $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Good news: Convex function!
Bad news: No analytical solution

Repeat

{

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$

}

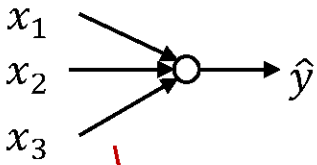
(Simultaneously update all θ_j)

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}\left(x^{(i)}\right) - y^{(i)})\, x_j^{(i)}$$

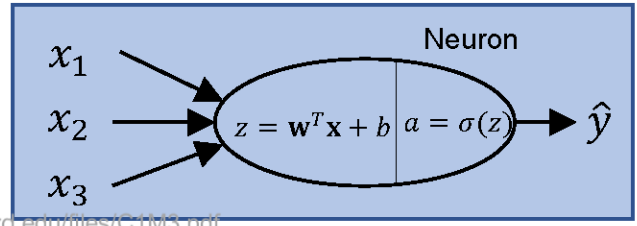
"Neuron" Representation of Logistic Regression

- Logistic Regression

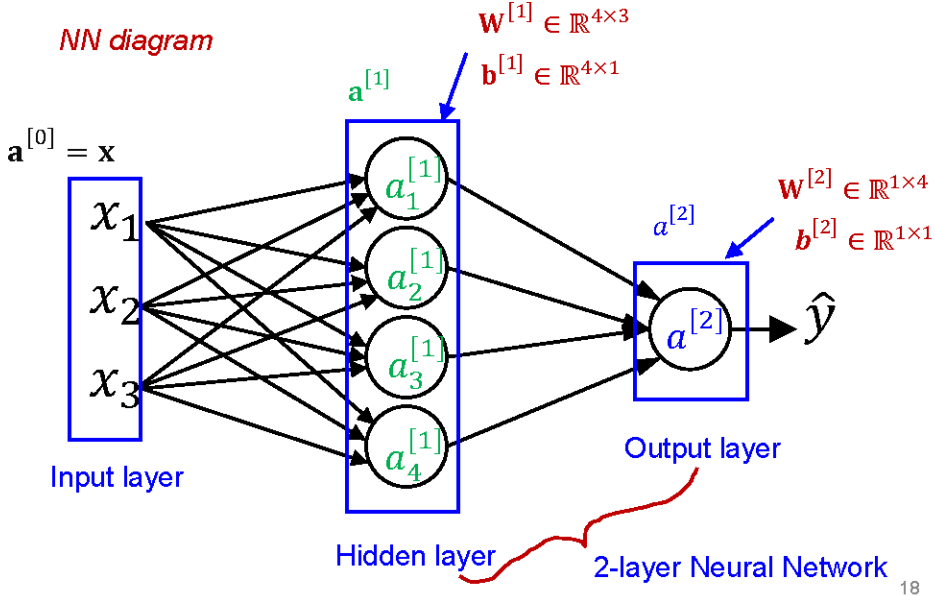
Given $\mathbf{x}, y$, want parameters $\mathbf{w}$ and b to make $\hat{y}$ infinitely close to y



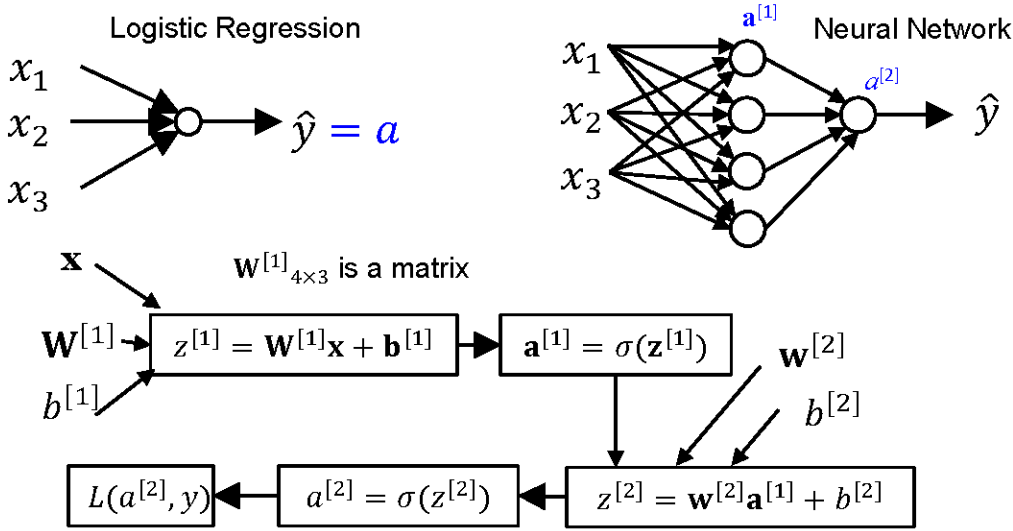
- Where, $\mathbf{x} \in \mathbb{R}^{n_x}, 0 \leq \hat{y} \leq 1$, $\hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b)$ is the probability of the given $\mathbf{x}$ belongs to the class 1 (binary classification case
- Parameters: $\mathbf{w} \in \mathbb{R}^{n_x}, b \in \mathbb{R}$



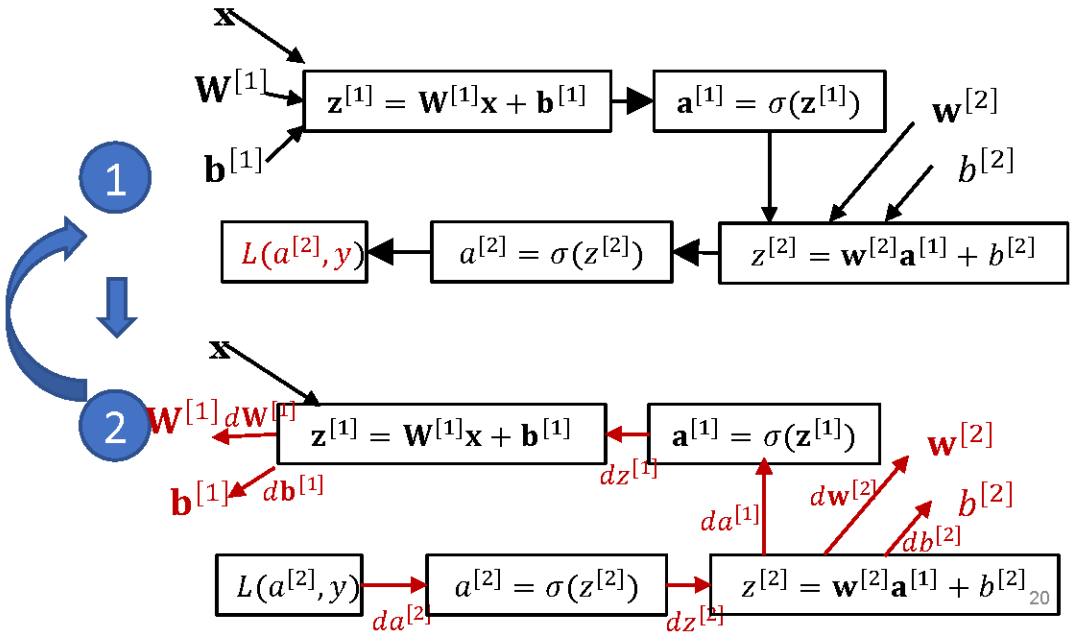
Neural Network (NN) Representation



Neural Network Representation and Output



Training in Neural Network



Gradient Descent for Training Neural Networks

Parameters: $\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{w}^{[2]}, b^{[2]}$

Cost function: $J\left(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{w}^{[2]}, b^{[2]}\right) = \frac{1}{m}\sum_{i=1}^{m} L(a^{[2]}, y)$

Gradient descent:

Repeat

{

Compute predicts $(\hat{y}^{(i)})$ ← Forward Propagation

$d\mathbf{W}^{[1]} = \frac{\partial J}{\partial \mathbf{W}^{[1]}}, d\mathbf{b}^{[1]} = \frac{\partial J}{\partial \mathbf{b}^{[1]}}$

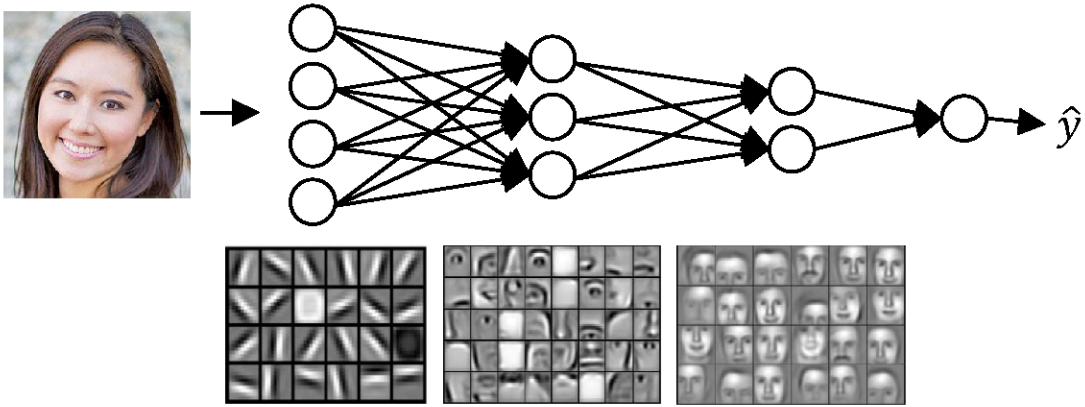
$d\mathbf{W}^{[2]} = \frac{\partial J}{\partial \mathbf{W}^{[2]}}, d\mathbf{b}^{[2]} = \frac{\partial J}{\partial \mathbf{b}^{[2]}}$ ← Backpropagation

$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \alpha \cdot d\mathbf{W}^{[1]}, \mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \alpha \cdot d\mathbf{b}^{[1]}$

$\mathbf{W}^{[2]} = \mathbf{W}^{[2]} - \alpha \cdot d\mathbf{W}^{[2]}, \mathbf{b}^{[2]} = \mathbf{b}^{[2]} - \alpha \cdot d\mathbf{b}^{[2]}$ ← Updating

}

Intuition about deep representation



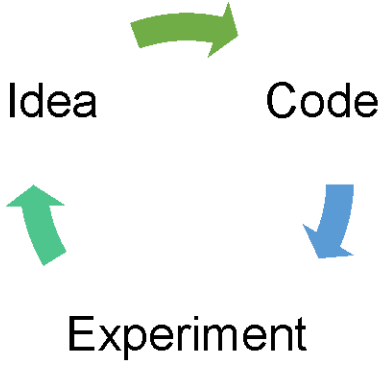
http://cs230.stanford.edu/files/C1M4.pdf

Outline

- Recap for predictive model - classification
- Overfitting
- Regression

Applied deep learning is a highly iterative process

- Data
- Computation
- Algorithms and hyperparameters
  - #layers
  - # hidden units
  - learning rate
  - activation function…



Dataset

- How to effectively make a progress in development?
  - Analyze based on the result on dataset

Training Dataset: The sample of data used to fit the model.

Validation Dataset: The sample of data used for model selection.

Test Dataset: The sample of data used for assessment of the final chosen model.

Example for using different datasets

- Example

  For different setup of hyperparameters

  1. Train model with training dataset
  2. Evaluate current model using validation dataset
  3. Save model and evaluate result/training result

  End

  4. Choose model based on results
  5. Evaluate the chosen model based on test dataset
  6. Output the final evaluation result as the performance of the model

- More Example: cross validation

Evaluation metric

Precision, Recall, F1 score

Predicted/Classified

Actual		Negative	Positive
	Negative	998	0
	Positive	1	1

Actual

Predicted

	Negative	Positive
Negative	True Negative	False Positive
Positive	False Negative	True Positive

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{True\ Positive}{Total\ Predicted\ Positive}$$

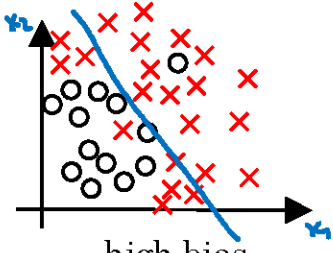
$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

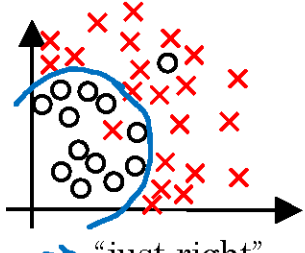
$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{True\ Positive}{Total\ Actual\ Positive}$$

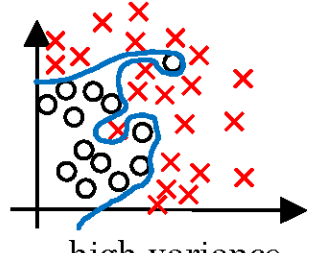
$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Bias and Variance







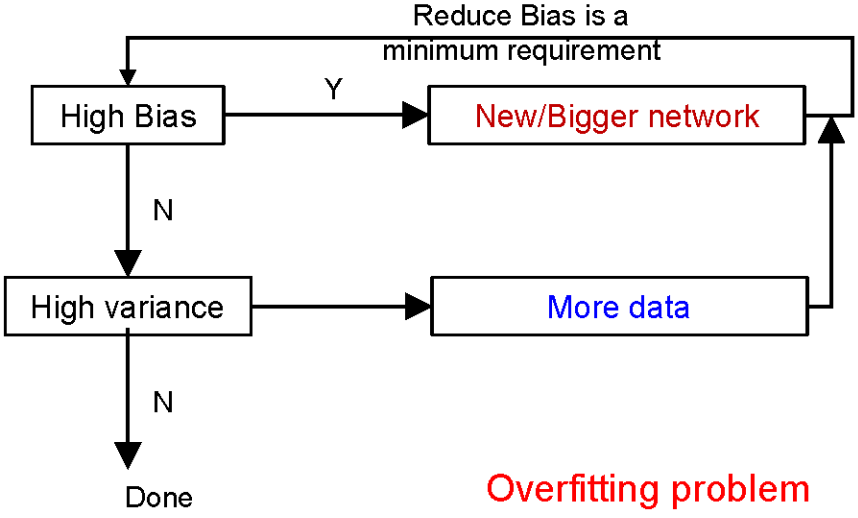
Bias and Variance

Cat classification

Train set error:	1%	15%	15%	0.5%
Validation set error:	11%	16%	30%	1%
	High Variance	High Bias	High Bias High Variance	Good

Overfitting

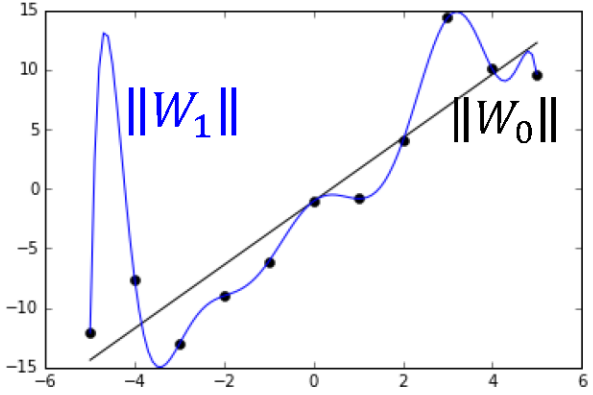
Basic recipe for Neural Network



Overfitting

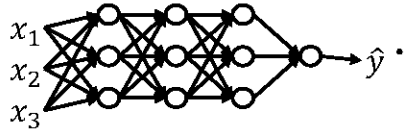
- Overfitting occurs when you achieve a good fit of your model on the training data, while it does not generalize well on new, unseen data.
- We can identify overfitting by looking at validation metrics, like loss or accuracy.
- Usually, the validation metric stops improving after a certain number of epochs and begins to decrease afterward. The training metric continues to improve because the model seeks to find the best fit for the training data.

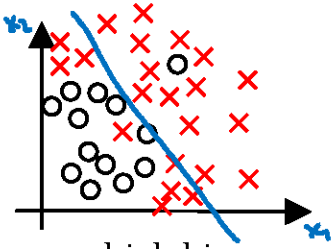
What happen in overfitting

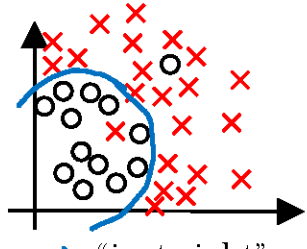


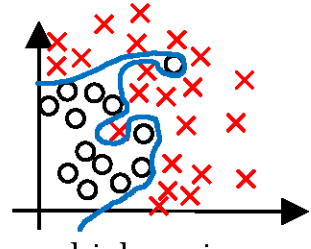
$\|W_1\|$ vs $\|W_0\|$

How to reduce overfitting









Regularization

Loss function: $J(W; X, y) + \frac{1}{2}\lambda \cdot ||W||^2$

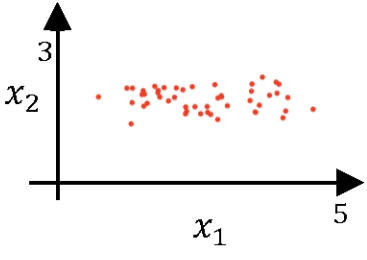
$$J_{regularized} = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right)\right)}_{\text{cross-entropy cost}} + \underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l}\sum_{k}\sum_{j}W_{k,j}^{[l]2}}_{\text{L2 regularization cost}}$$

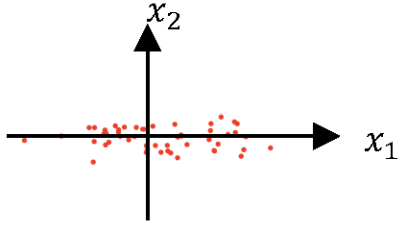
- We can view the learning procedure as a compromise between minimizing E (the error) and minimizing the sum of the squares of the weights.
- An additional parameter, λ, is added to allow control of the strength of the regularization.

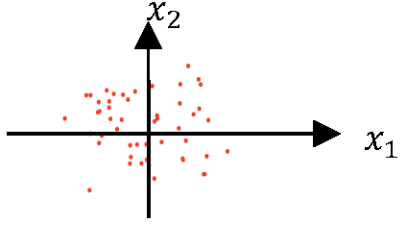
https://towardsdatascience.com/understanding-the-scaling-of-l%C2%B2-regularization-in-the-context-of-neural-networks-e3d25f8b50db

Regularization: Data augmentation

Normalizing training sets





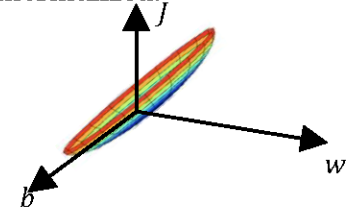


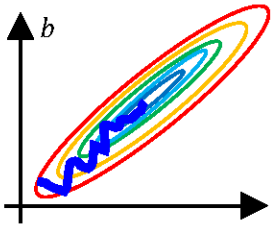
Why normalize inputs?

Same scale (learning rate) for adjusting w and b

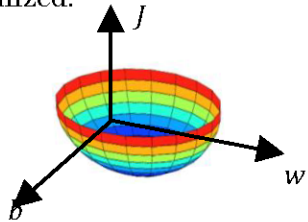
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

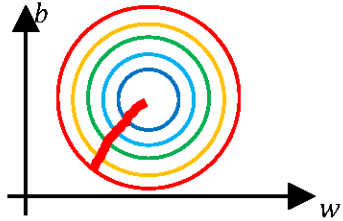
Unnormalized:





Normalized:





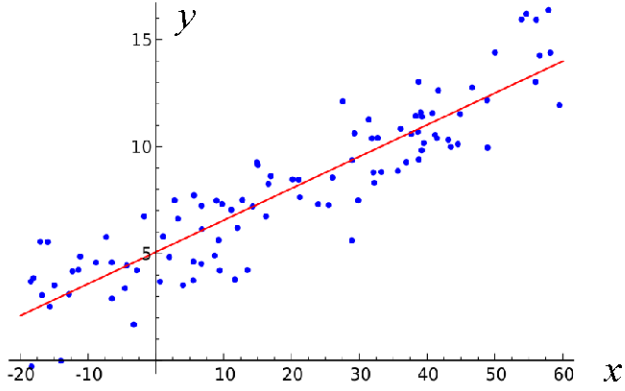
Outline

- Recap for predictive model - classification
- Overfitting
- Regression

Logistic regression is a misnomer

- Above we mentioned that the name logistic regression is a misnomer under the modern use of data science terminology. Recall that the distinction between classification and regression is whether the value for the target variable is categorical or numeric. For logistic regression, the model produces a numeric estimate (the estimation of the log-odds). However, the values of the target variable in the data are categorical.
- Debating this point is rather academic. What is important to understand is what logistic regression is doing. It is estimating the log-odds or, more loosely, the probability of class membership (a numeric quantity) over a categorical class. So we consider it to be a class probability estimation model and not a regression model, despite its name.

(Simple) Linear Regression



Linear Model: $y = wx + b$

Linear Regression: modelling the relationship between x and y using linear function

Problem Statement of Linear Regression

Assumption

- There is a linear relationship between x and y with intercept and slope.

  x and y are given, inter and slope are unknown.

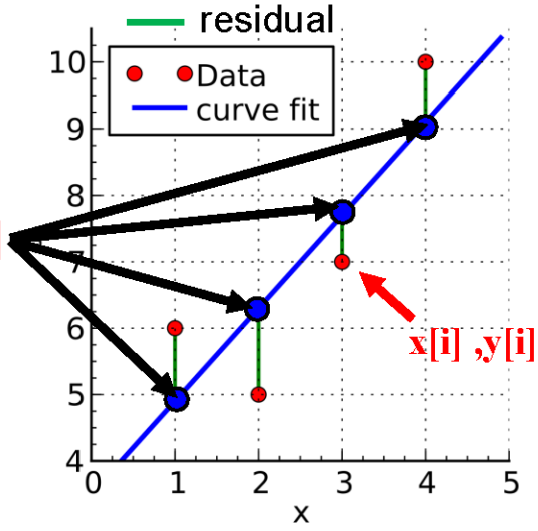
Objective

- We expect to find a best-fitting straight line (a line is represented by inter and slope)

→ make y be inter + slope * x.

Notations in Linear Regression

$\hat{y}[i] = inter + slope * x[i]$



Loss Function for Linear Regression

- If we get the parameters inter and slope wrong, the residuals get bigger, so it makes sense that the parameters we want are the ones that minimize the residuals:

- Objective is to minimize the sum of squared residuals

$$\sum_{i} (y[i] - (inter + slope \cdot x[i]))^2$$

Techniques for Linear Regression

1st Method: Least squares

2nd Method: Gradient descent

Least Squares for Simple Linear Regression

Loss Function: $L(w, b) = \sum_{i=1}^{m} (y[i] - (w\ x[i] + b))^2$

m is the number of sample data

Slope→w, Inter→b for easy notation

Partial Derivative of Loss Function for **each** unknown parameters:

$$\begin{cases} \frac{\partial \sum_{i=1}^{m} (y[i] - (w\ x[i] + b))^2}{\partial w} = 0 \\ \frac{\partial \sum_{i=1}^{m} (y[i] - (w\ x[i] + b))^2}{\partial b} = 0 \end{cases}$$

Result:

$$\begin{cases} w = \frac{\sum_{i=1}^{m} (x[i] - \bar{x})(y[i] - \bar{y})}{\sum_{i=1}^{m} (x[i] - \bar{x})^2} \\ b = \bar{y} - w\bar{x} \end{cases} \quad \text{if } \sum_{i=1}^{m} (x[i] - \bar{x})^2 \neq 0$$

Check yourself

https://mathworld.wolfram.com/LeastSquaresFitting.html

Visualization of Gradient Descent

Illustration of how the gradient descent algorithm works

https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

Gradient Descent

1. Initialization: w = 0, b = 0, Learning rate: α= 0.0001.
2. Partial derivative of the loss function

$\hat{y}$ denotes the predicted value of y

m is the number of sample data

$$\frac{\partial L(\mathrm{w},\mathrm{b})}{\partial \mathrm{w}} = \frac{\partial \sum_{i=1}^{m}(\mathrm{y[i]} - (\mathrm{wx[i]} + \mathrm{b}))^2}{\partial \mathrm{w}} = \frac{-2}{m}\sum_{i=1}^{m}\mathrm{x[i]}(\mathrm{y[i]} - \hat{\mathrm{y}}) = \frac{-2}{m}\sum_{i=1}^{m}\mathrm{x[i]}(\mathrm{y[i]} - (\mathrm{wx[i]} + \mathrm{b}))$$

$$\frac{\partial L(\mathrm{w},\mathrm{b})}{\partial \mathrm{b}} = \frac{\partial \sum_{i=1}^{m}(\mathrm{y[i]} - (\mathrm{w\ x[i]} + \mathrm{b}))^2}{\partial \mathrm{b}} = \frac{-2}{m}\sum_{i=1}^{m}(\mathrm{y[i]} - \hat{\mathrm{y}}) = \frac{-2}{m}\sum_{i=1}^{m}(\mathrm{y[i]} - (\mathrm{w\,x[i]} + \mathrm{b}))$$

3. Updating the current value of **w** and **b**

$$\mathrm{w} = \mathrm{w} - \alpha\frac{\partial L(\mathrm{w},\mathrm{b})}{\partial \mathrm{w}} \qquad \mathrm{b} = b - \alpha\frac{\partial L(\mathrm{w},\mathrm{b})}{\partial \mathrm{b}}$$

4. Repeat step 2 and 3 for *k* iterations.

Comparison between Least squares and Gradient descent

- Least squares
  - Globally optimal solution, simple
  - The error(residuals) follow a normal distribution.
  - If the solution can be found analytically
- Gradient descent
  - Locally optimal solution, approximate, iterative
  - Can be used for linear model and other model

Data Science

Week 11

Test