Introduction to Data Science Using ScalaTion Release 2 Lesson Plans

John A. Miller
Department of Computer Science
University of Georgia

April 29, 2020

Contents

Chapter 1

Introduction to Data Science

This chapter positions the field of Data Science inclusively between Statistics and Machine Learning. Data Science often focuses on a collecting large dataset(s) to address a problem, selecting appropriate models for making predictions or decisions, training the models on the dataset(s), and then using the trained models to underestand and help address the problem at hand.

1.1 Lesson Plan

Chapter 2

Mathematical Preliminaries

This chapter serves as a quick review of the two principal mathematical foundations: Probability and Linear Algebra (Vectors and Matrices).

2.1 Lesson Plan: Probability

$$\boxed{\textbf{ClassDate : Day 1 – DueDate : Day 2}}$$

Read Sections 5.1-5.5 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class.

Consider a discrete random variable y. By being discrete, the variable can only take on a finite or countably infinite number of values. For example, the number of possible values for the variable for the cases of a coin flip, roll of two dice, and number of coin flips until a head are 2, 11 and ∞, respectively. Knowing the domain of values for the random is useful, but hardly stastifying. Surely, somne the possible values may be more likely or probable than others. We can start with a fixed amount of probabilistic mass and spread it over the domain values in order to guage the likelihood of each possible value. This notion can be captured by the probability mass function (pmf).

$$p_y(y_i) \ = \ P(y = y_i)$$

ITEM 1: Coin flipping experiments may be modeled using a random variable $y \sim \text{Bernoulli}(p)$ where $\sim$ mean distributed as and p indicates the probability of flipping a coin and having it land heads up. The domain for y is 0 (tail) and 1 (head). Since this domain is so simple and regular, it is convenient to represent the possible values using $k \in \{0, 1\}$, as $y_0 = 0$ and $y_1 = 1$. What is the pmf for random variable y?

$$p_y(k) \;=\; p^{?}(1-p)^{?} \qquad \text{for } k \in \{0,1\}$$

ITEM 2: Experiments in which two coins are flipped $y = z_1 + z_2$ where z_1 and $z_2 \sim \text{Bernoulli}(p)$ can be modeled using the Binomial$(p, 2)$ distribution. What is y pmf? In order to determine the form of the pmf, use the fact that the result for each coin is independent, so that the probabiliites multiply.

$$p_y(k) \;=\; \binom{2}{k} p^{?}(1-p)^{?} \qquad \text{for } k \in \{0,1,2\}$$

ITEM 3: The number of flips required to get a head can be modeled using random variable $y \sim \text{Geometric}(p)$. What is y pmf?

2.1.1 Joint Probability

The joint probabilitiy measures the likelihood of multiple random variables taking on certain values. As a simple case, the joint probability for a 2-dimensional random vector $\mathbf{y} = [y_0, y_1]$ is

$$p_{\mathbf{y}}(\mathbf{k}) \;=\; P(y_0 = k_0, y_1 = k_1)$$

If the random variables y_0 and y_1 are independent, then

$$p_{\mathbf{y}}(\mathbf{k}) \;=\; p_{y_0}(k_0)\, p_{y_1}(k_1)$$

For example, after flipping 10 coins with the first 6 being tails and the last 4 being heads, the joint probability for $\mathbf{k} = [0, 0, 0, 0, 0, 0, 1, 1, 1, 1]$ results from taking the product of the ten individual probabilities.

$$p_{\mathbf{y}}(\mathbf{k}) \;=\; (1-p)^6 p^4$$

ITEM 4: For two fair coins $p = .5$, compute the joint probability for all possible values of vector $\mathbf{k} = [k_1, k_2]$. Also, try this for unfair coins with $p = .6$.

$$p_{\mathbf{y}}(\mathbf{k}) \;=\; ?$$

2.1.2 Log-Probability

While probabilities are often combined using multiplication, log-probabilities are often combined using addition. Given the pmf for a random variable $\mathbf{y}$) its log-probability is given by

$$logp_{\mathbf{y}}(\mathbf{k}) \;=\; -\log_2(p_{\mathbf{y}}(\mathbf{k}))$$

The negative log is used since logorithms of values less than 1 are negative. It is common to use base 2 logarithms, although other bases may be used as well.

For the example above of flipping 10 coins, the log-probability is

$$logp_{\mathbf{y}}(\mathbf{k}) \;=\; -6\,\log_2(1-p) - 4\,\log_2 p$$

ITEM 5: Plot the probability and log-probability for the first 16 Binomial(p, n) distributions, i.e., for the number of coins $n = 1, \ldots, 16$. Try with $p = .6$ and $p = .5$.

Chapter 3

Data Management and Preprocessing

This chapter provides a quick overview of the support provided by ScalaTion for data management and data preprocessing.

3.1 Lesson Plan

Chapter 4

Prediction

This chapter focuses on linear models used for making predictions (e.g., Multiple Linear Regression).

4.1 Lesson Plan

Chapter 5

Classification

5.1 Lession Plan: Classification

$\boxed{\textbf{ClassDate : Tuesday, April 7 – DueDate : Wednesday, April 8}}$

Read Sections 5.1-5.5 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class.

This chapter focuses on models used to classify a data/input vector $\mathbf{x}$, amongst k possible values for the response variable y. The response variable may be categorical (e.g., English, French, Spanish) or ordinal (e.g., low, medium, high). For categorical variables, the $<$ (less than) operator is undefined and averages do not make sense (average of English and Spanish is not French), while for ordinal, the $<$ operator is defined and the average may arguably make some sense. For the average to be precise, though, the ordinal number also must be cardinal, meaning that distance between subsequent values is the same. In any case, the number of possible values is finite. Whatever the type of response, its values can be mapped into k integer values. However, when the possible values are countably infinite (like the integers $\mathbb{Z}$) or uncountably infinite (like the reals $\mathbb{R}$), then classification models will not be appropriate.

Classification models may be thought of as applying a function (or algorithm) f, often with parameters $\mathbf{b}$ to an input vector $\mathbf{x}$ to estimate a response $y \in \{0, 1, \ldots, k-1\}$.

$$y \;=\; f(\mathbf{x};\ \mathbf{b}) - \epsilon$$

For making it easier to remember certain Quality of Fit (QoF) measures, negative error will be used. While regression models focus on estimating the conditional expectation of y given $\mathbf{x}$,

$$\hat{y} \;=\; \mathbb{E}\,[y|\mathbf{x}]$$

the classification models often focus on maximizing the conditional probability of y given $\mathbf{x}$, i.e., finding the conditional mode.

$$y^* \;=\; \text{argmax}\; P(y|\mathbf{x}) \;=\; \mathbb{M}\,[y|\mathbf{x}]$$

ITEM 1: When $k = 2$, the classification problem becomes binary, e.g.,

```
0  meaning:   no  or  negative
1  meaning:  yes  or  positive
```

Such problems are very common. Consider the error term ϵ. Before, the error term took on real values (e.g., 2.71828, -3.14159). What values could the error random variable ϵ now take on?

$$\epsilon \;=\; y^* - y \;=\; \;\in \{\ ?\ \}$$

ITEM 2: For regression, the Quality of Fit (QoF) includes sum of squared errors sse, mean squared error mse, sum of absolute errors sae, and mean absolute error mae. Assuming m instances, give a formula involving the error vector $\boldsymbol{\epsilon}$ to calculate mae

$$sae \;=\; ||\ ?\ ||_? \;=\; \sum\ ?$$

ITEM 3: How would sse differ from sae for such binary classification?

ITEM 4: How would you characterize the following values for ϵ?

$$\begin{array}{rl} -1 & [\text{ false } | \text{ true }] \text{ but classified } [\text{ negative } | \text{ positive }] \\ 0 & [\text{ false } | \text{ true }] \\ 1 & [\text{ false } | \text{ true }] \text{ but classified } [\text{ negative } | \text{ positive }] \end{array}$$

Note, false means misclassication, while true means correct classification. Also, the "but classified" phrase is usually dropped.

ITEM 5: There are actually two ways of getting true, name them. Now there are four possible outcomes. Imagine a plot of y^* (predicted) versus y (actual) where the horizontal axis has the values for y (0, 1) and the vertical axis has the values for y^* (0, 1). Make four quadrants and place counts for the number of cases given the following two vectors y (`y`) and y^* (`yp`).

```
yp = [ 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0 ]
y  = [ 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0 ]
e  =
```

Note, the plot can be caputured in a 2-by-2 matrix that is referred to as a **confusion matrix**. Correctly label each quadrant with one of the following labels `fn, fp, tn, tp`. Unfortunately, the order of cells in confusion matrices varies based on the source and the data science tool.

ITEM 6: From the confusion matrix, calculate the error rate er, accuracy acc, precision p, recall/sensitivity r, and F$_1$-measure, F_1.

5.2 Null Model

As with other types of data science problems, there is a Null Model for classification problems. The Null Model simply guesses the most likely case based on the response vector $\mathbf{y}$ ignoring the input/data matrix X.

$$y^* \;=\; \text{argmax}\; P(y) \;=\; \mathbb{M}\,[y]$$

Given m instances, the probabilities may be estimated based on frequency counts for each class $c \in \{0, 1, \ldots, k-1\}$.

$$\nu_c(\mathbf{y}) \;=\; \sum_{i=0}^{m-1} I(y_i = c)$$

where the indicator predicate (Boolean function) returns 1 (0) when the condition is true (false). Now the probability is calulated as the ratio of the number instances where y_i is c, divided by the total number of instances m.

$$P(y = c) \;=\; \frac{\nu_c(\mathbf{y})}{m}$$

ITEM 7: Use the Null Model to provide new classifications for `yp`.

```
yp =
```

ITEM 8: Determine the confusion matrix and calculate er, acc, p, r and F_1.

5.3 Lession Plan: Naïve Bayes

$$\boxed{\textbf{ClassDate : Wednesday, April 8} - \textbf{DueDate : Thursday, April 9}}$$

Read Sections 5.6 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class.

The goal is to estimate the conditional probabilities,

$$y^* \;=\; \text{argmax } P(y|\mathbf{x}) \;=\; \mathbb{M}\,[y|\mathbf{x}]$$

which indicate the probability that $y = c$ given the input data vector $\mathbf{x}$, and then pick a class c that is maximal. Clearly the input features $[x_0, x_1, \ldots x_{n-1}]$ should influence the these probabilies.

Conditioning on a vector can be unweildy and performance of models that do so, tends to be less robust. By using Bayes Theorem, one can condition on a single variable and in particular one that only has k values.

$$P(y|\mathbf{x}) \;=\; \frac{P(\mathbf{x}|y)\,P(y)}{P(\mathbf{x})}$$

Finding a class c with maximal conditional probability is now easy since the denonimator is common to all classes and $P(y)$ has already been calculated for the Null Model, i.e., the focus is on calculating the following conditional probabilities:

$$P(\mathbf{x}|y)$$

5.3.1 Factoring the Probability

Such conditional probabilities can be closely approximated using Bayesian Networks, but unfortunately in their unrestricted form are intractable. Making an assumption that each random variable x_j can only conditionally depend (as an approximation) on one other variable x_k leads to either

Tree-Augmented Naïve (TAN) Bayes or Forest-Augmented Naïve (FAN) Bayes models. Ignoring all conditional dependencies between the features x_j leaves each one only dependent on only y. This allows the conditional probability to be factored (independance approximation).

$$P(\mathbf{x}|y) \; = \; \prod_{j=0}^{n-1} P(x_j|y)$$

Thus, one needs for calculate a Conditional Probability Table (CPT) for each feature x_j. For the "Play Tennis?" example given in the textbook in section 5.5, there are four features: x_0 is Outlook, x_1 is Temporature, x_2 is Humdity, and x_3 is Wind.

Item 1: Create a Joint Frequency Table (JFT) for each feature, e.g., for x_0 it is

Table 5.1: JFT for $\mathbf{x}$_0

$x_0 \backslash y$	**0**	**1**
0	2	3
1	0	4
2	3	2

Item 2: From the Joint Frequency Table (JFT) for each feature, calculate the Conditional Probability Table, e.g., for x_0 it is

Table 5.2: CPT for $\mathbf{x}$_0

$x_0 \backslash y$	**0**	**1**
0	2/5	3/9
1	0	4/9
2	3/5	2/9

5.3.2 Laplace Smoothing

When zeros show up in CPTs it can cause problems, because multiplying any value by zero gives 0. The zero may have only showed up due to lack a data, so giving zero as the product could be misleading. It is better to give it small value, rather than zero. One way to do this when calculating the CPT for a given x_j is to add one fake instance that is equally likely of taking on any possible value for x_j. For example if x_0 (Outlook) can take on three values, then add one third to each frequency count.

$$P(x_j = h \,|\, y = c) \;=\; \frac{\nu_{h,c}(\mathbf{x}_{:j}, \mathbf{y}) + 1/vc_j}{\nu_c(\mathbf{y}) + 1}$$

where vc_j is the value count (or number of distant values) for x_j. Note, that this will also ensure that there will be no "divide-by-zero" error.

Item 3: Use Naïve Bayes to provide new classifications for `yp`.

```
yp =
```

ITEM 4: Determine the confusion matrix and calculate er, acc, p, r and F_1.

5.3.3 Entropy

The entropy of a discrete random variable y with probability mass function (pmf) $p_y(y_i)$ captured in a k-dimensional probability vector $\mathbf{p}_y = [p_0, \ldots p_{k-1}]$ is the negative of the expected value of the log of the probability.

$$H(y) \;=\; H(\mathbf{p}_y) \;=\; -\,\mathbb{E}\,[\log_2(\mathbf{p}_y)] \;=\; -\sum_{i=0}^{k-1} \log_2(p_i)\, p_i$$

Under this definition of entropy, $H(y)$ ranges from 0 to $\log_2(k)$. Low entropy (close to 0) means that there is low uncertainty/risk in predicting an outcome of an experiment involving the random variable y, while high entropy (close to $\log_2(k)$) means that there is high uncertainty/risk in predicting an outcome of such an experiment. For binary classification ($k = 2$), the upper bound on entropy is 1.

The entropy may be normalized by setting the base of the logarithm to the dimensionaliry of the probability vector k, in whch case, the entropy will be in the interval $[0, 1]$.

$$H_k(y) \;=\; H_k(\mathbf{p}_y) \;=\; -\,\mathbb{E}\,[\log_k(\mathbf{p}_y)] \;=\; -\sum_{i=0}^{k-1} \log_k(p_i)\, p_i$$

See section 5.11.1 in the textbook, page 154 for more details. Also, see section 2.1 in these lesson plans for background on probability and log-probability.

ITEM 5: A random variable $y \sim \text{Bernoulli}(p)$ may be used to model the flip of a single coin that has a probability of success/head (1) of p. Its pmf is

$$p(y) \;=\; p^y(1-p)^{1-y}$$

Using spreadsheet software, plot its entropy

$$H(y) \;=\; H(\mathbf{p}) \;=\; H([p, 1-p]) \;=\; \log_2(p)\,p + \log_2(1-p)\,(1-p)$$

versus p, as probability p ranges from 0 to 1. Describe the curve.

————– 6 and 7 ADDED AFTER QUIZ

ITEM 6: A random variable $y = z_1 + z_2$ where z_1, z_2 are distributed as Bernoulli(p) may be used to model the sum of flipping two coins. Using spreadsheet software, plot its entropy

$$H(y) \;=\; H(\mathbf{p}) \;=\; H([p^2, 2p(1-p), (1-p)^2])$$

versus p, as probability p ranges from 0 to 1. Describe the curve.

ITEM 7: Plot the entropy H and normalized entropy H_k for the first 16 Binomial(p, n) distributions, i.e., for the number of coins $n = 1, \ldots, 16$. Try with $p = .6$ and $p = .5$.

5.4 Lession Plan: Decision Tree ID3

ClassDate : Wednesday, April 9 – DueDate : Thursday, April 14

Read Section 5.11 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class.

5.4.1 Example Problem

See section 5.11.2 in textbook page 155.

One way to start a Decision Tree for the "Play Tennis?" problem is to try Outlook (x_0) as the root feature/variable. The resulting tree is shown below where the node is designated by the variable (in this case x_0). The edges indicate the values that this variable can take on, while the two numbers $n^- p^+$ indicate the number of negative and positive cases.

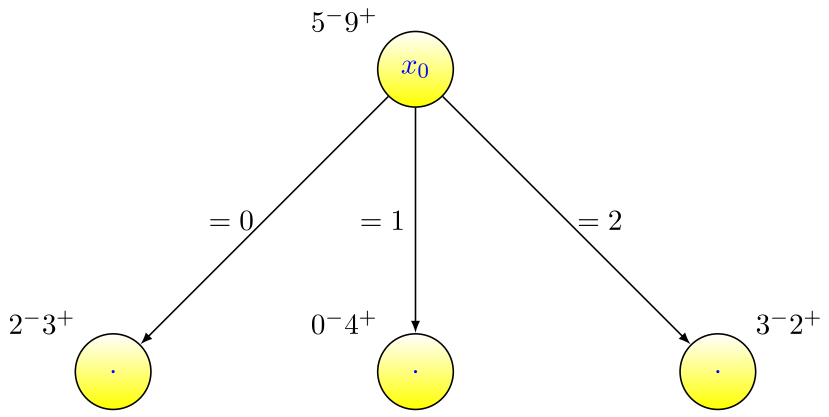

Figure 5.1: Decision Tree for "Play Tennis?" Example

ITEM 1: Redraw figure 5.1. Compute the entropies for each of the four nodes (show steps) and place them next the node. Note, the $n^- p^+$ are all that is needed for computing entropies; convert them to a probability vector $\mathbf{p}$ and calculate $H(\mathbf{p})$.

ITEM 2: Compute the average entropy over all the decision nodes (i.e., the leaves of the tree). The average must be weighted by the fraction of cases applicable to each node. How much did the entropy drop?

ITEM 3: An alternative is to use Humidity (x_2) for the root feature/variable. Draw the Decision Tree with x_2 as the root. Be sure to label nodes with the negative and positive cases.

ITEM 4: For this alternative decision tree, compute the average entropy over all the decision nodes (i.e., the leaves of the tree). How much did the entropy drop in this case. Which is the better choice for the root node?

ITEM 5: Choose a decision/leaf node where the entropy is above a threshold (its entropy is too high). Assume in this case it is 0.1. Pick the first node above the threshold (alternatively the node with the highest entropy) and determine the variable/feature that will result in the greatest entropy drop when the node is replaced with a subtree.

ITEM 6: Under what conditions can Decision Trees when expanded sufficently drive the entropy to zero?

ITEM 7: Might driving the entropy to zero produce overfitting where performance on test data is substantially worse than it was on training data.

In this sense, Decision Trees can be brittle. A more robust modeling technique is to use several simple (or height limited) Decision Trees and have them vote on the decision. This is the idea behind Random Forests.

Chapter 6

Classification: Continuous Variables

This chapter broadens the previous chapter by allowing variables used for making decisions be continuous as well.

6.1 Lesson Plan: Simple Logistic Regression

ClassDate : Tuesday, April 14 – DueDate : Thursday, April 16

Read Section 6.3 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class.

Simple Logistic Regression looks like Simple Regression, except the response variable y is now binary $(0, 1)$. First, let us consider what happens is we ignore this change in y and just create a Simple Regression model.

$$y = \mathbf{b} \cdot \mathbf{x} + \epsilon = b_0 + b_1 x_1 + \epsilon$$

The Motor Trend Cars `mtcars` dataset is available in ScalaTion.

```
val x  = ExampleMtcars.xy.sliceCol (0, 2)
val x1 = ExampleMtcars.xy.col (1)
val y  = ExampleMtcars.xy.col (2).toInt
```

ITEM 1: Train a Simple Regression model using the `mtcars` dataset. The goal is to predict/classify a car's `engine_type` as either V-shaped (0) or Straight (1) based on its `mpg`. Give the QoF measures and plot `y` and `yp` versus `mpg`.

```
val srg = new SimpleRegression (x, y.toDouble}
srg.analyze ()
println (srg.report)
val yq = srg.predict ()
new Plot (x1, y.toDouble, yq, "Simple: y, yq (red) versus mpg")
```

ITEM 2: Simple Regression will make predictions based on the best line that fits the data of the `engine_type` versus `mpg`. The predictions will be continuous values that will be in the vicinity of 0 and 1. Try turning the prediction into a classification by rounding the value of `yp`.

```
val yr = new VectorD (yq.dim)
for (i <- yr.range) yr(i) = round (yq(i)) + 0.04
new Plot (x1, y.toDouble, yr, "Rounded: y, yr (red) versus mpg")
```

ITEM 3: Determine the confusion matrix and calculate er, acc, p, r and F_1.

ITEM 4: Notice that none of the QoF measures in the last item were used in finding optimal values for parameters b_0 and b_1. Prehaps a more direct approach could yield better results. Rather than rounding the response, apply a transformation function that pushes the value to either 0 or 1, such as the *sigmoid* function. Under this transformation, the values will be in the range of $[0, 1]$ and may be interpreted as the conditional probability that $y = 1$ given data instance $\mathbf{x}$, call this p_y. A threshold such as 0.5 may be set for deciding between classifying as 0 or 1.

$$p_y \; = \; \text{sigmoid}(\mathbf{b} \cdot \mathbf{x}) \; = \; \text{sigmoid}(b_0 + b_1 x_1)$$

Note, the *sigmoid* function is a special case of the *logistic* function (hence the name). The inverse of the sigmoid function is the *logit* function, so the above equation may be rewritten as follows:

$$\text{logit}(p_y) \; = \; \mathbf{b} \cdot \mathbf{x} \; = \; b_0 + b_1 x_1$$

Show that the *logit* function is the inverse of the *sigmoid* function.

$$\begin{aligned} \text{sigmoid}(z) &= \frac{1}{1 + e^{-z}} \\ \text{logit}(p) &= \ln \frac{p}{1 - p} \end{aligned}$$

ITEM 5: Use ScalaTion's `SimpleLogisticRegression` class to train a model for the `mtcars` dataset. Plot `y` and `yp` versus `mpg`. Compare with the results from Rounded Simple Regression.

```
val lrg = new SimpleLogisticRegression (x, y}
lrg.train ()
val yp = lrg.classify ()
lrg.confusion (yp)
println (lrg.report)
println (lrg.summary (lrg.parameter))
new Plot (x1, y.toDouble, yp.toDouble, "Logistic: y, yp (red) versus mpg")
```

ITEM 6: Determine the confusion matrix and calculate er, acc, p, r and F_1. Compare the Quality of Fit (QoF) measures for Rounded Simple Regression versus Simple Logistic Regression.

ITEM 7: Try changing the classification/decision threshold hyper-parameter `cThresh` from its default of 0.5 to see how it affects the false positive rate (fpr) and the false negative rate (fnr). As `cThresh` increares, fpr does what? As `cThresh` increares, fnr does what?

6.2 Lesson Plan: Maximum Likelihood Estimation

$$\boxed{\textbf{ClassDate : Wednesday, April 15 – DueDate : Thursday, April 16}}$$

Read Section 6.3 in the **Introduction to Data Science in ScalaTion** textbook. No Quiz.

In this section, rather than estimating parameters using *Least Sqaures Estimation* (LSE), *Maximum Likelihood Estimation* (MLE) will be used. Given a dataset with m instances, the model will produce an error for each instance. When the error is large, the model is in disagreement with the data. When errors are normally distributed, the probability density will low for a large error, meaning this is an unlikely case. If this is true for many of the instances, the problem is not the data, its the values given for the parameters. The parameter vector $\mathbf{b}$ should be set to maximize the likelihood of seeing instances in the dataset. This notion is captured in the likelihood function $L(\mathbf{b})$. Note, for the Simpler Regression model there is only a single parameter, the slope b.

Given an m instance dataset $(\mathbf{x}, \mathbf{y})$ where both are m-dimensional vectors and a Simpler Regression model

$$y = bx + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$. Since $\epsilon_i = y_i - bx_i$, we may write the likelihood function $L(b)$ as the product of m Normal density functions (making the assumption that the instances are independent).

$$L(b) \;=\; \prod_{i=0}^{m-1} \frac{1}{\sqrt{2\pi}\sigma} e^{-(y_i - bx_i)^2/2\sigma^2}$$

Taking the natural logarithm gives the log-likelihood function $l(b)$

$$l(b) \;=\; \sum_{i=0}^{m-1} -\ln(\sqrt{2\pi}\sigma) - (y_i - bx_i)^2/2\sigma^2$$

The derivative of $l(b)$ w.r.t. b is

$$\frac{dl}{db} \;=\; \sum_{i=0}^{m-1} -2x_i(y_i - bx_i)/2\sigma^2$$

For optimization, the derivative may be aet to zero

$$\sum_{i=0}^{m-1} x_i(y_i - bx_i) \;=\; 0$$

Solving for b gives

$$b = \frac{\sum x_i y_i}{\sum x_i^2} = \frac{\mathbf{x} \cdot \mathbf{y}}{\mathbf{x} \cdot \mathbf{x}}$$

6.3 Final Exam: Two Day Exam

Start: Thursday, April 30 at 12:00 noon - same as start of UGA scheduled final exam

End: Friday, May 1 at 12:00 midnight - email pdf file with answers by this time

Each Question requires a two page answer (1 page or 3 page answers not accepted)

1. Topic: Prediction

   Question: Consider the following 3 approaches for conducting Feature/Variable Selection for Multiple Linear Regression: (1) Pick the x_j that has the highest absolute correlation with y, and repeat. (2) Pick the x_j that has the smallest p value, and repeat. (4) Pick the x_j that improves a QoF measure (e.g., $\bar{R}^2$) the most, and repeat. Explain why each approach could potentially work. Will these approaches result in the same features/variables being selected? Explain. Compare and rank the three approaches in terms of how well they work for feature selection. Explain.

2. Topic: Classification

   Question: Consider a Naïve Bayes Classifier. Take Equation 5.6 on page 138 in the ScalaTion textbook and simplify it for the case of Binary Classification. Explain why each part of your new simplified equation is needed. Explain the theorem and assumption that underly this technique.

3. Topic: Neural Networks

   Question: Define epoch. Define mini-batch. During a single epoch, for a given mini-batch explain how backpropogation is used to improve the model's parameters (weights and biases). What problems may occur and explain how different optimization algorithms and loss functions can be used to address these problems.

4. Topic: Convolutional Neural Networks

   Question: Given 10,000 grayscale images with 28 by 28 pixels, describe the construction of a Convolutional Neural Network consisting of one input layer, one convolutional-pooling layer, followed by a 10 node fully-connected output layer. For the combined convolutional-pooling layer, assume there are two 5 by 5 filters, with each using 2 by 2 max-pooling (the max value in each non-overlapping 2 by 2 region is selected). Give the sizes for all the components in the network and explain how they are connected. Assume the (convolutional) stride is 1 and padding is not used. How many parameters (weights and biases) are there to be trained? After training, explain how the network classifies an image as one of 10 digits.

Sources for CNN:

1. "Chapter 9: Convolutional Networks"
   http://www.deeplearningbook.org/contents/convnets.html
2. "An Intuitive Explanation of Convolutional Neural Networks"
   https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets
3. "Convolutional Neural Networks (CNNs / ConvNets)"
   https://cs231n.github.io/convolutional-networks
4. "1D Convolutional Neural Networks and Applications – A Survey"
   https://arxiv.org/pdf/1905.03554.pdf
5. "Understanding Convolutional Neural Networks"
   https://davidstutz.de/wordpress/wp-content/uploads/2014/07/seminar.pdf

General Sources:

1. Textbook (ScalaTion): Introduction to Data Science Using ScalaTion, 2018. John A. Miller
2. Introduction to Data Science Using ScalaTion: Lesson Plans, 2020. John A. Miller
3. Textbook (ISL): An Introduction to Statistical Learning, 2013. Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
4. TextBook (ESL): The Elements of Statistical Learning, 2nd Ed., 2009. Trevor Hastiei, Robert Tibshirani and Jerome Friedman
5. Neural Networks and Deep Learning, 2018. Michael Nielsen

Additional Rules:

1. May only use the ten sources listed above.

2. Do not copy-paste or copy-paraphrase from any of the sources
3. Answers must be developed individually (no group work).
4. Answers must be emailed as one pdf file (may have embedded images)
5. Do not send multiple emails (only the first one will be graded).

6.4 Term Project

1. Serve as Zoom cohost to present your slides at the time scheduled by the TA.

2. May have suggestions for improvement given to boost your score if you introduce the change/addition before submitting the project.

3. Submit the term project following the TA instruction by midnight on Friday, April 24.

4. Scores for each of the nine items on pages 38-40 will be assigned. Each items except 8 (Reporting of Results) will be worth 10 points, while 8 will be worth 20 points.

5. Please make it easy for the TA to reproduce your results by having a good ReadMe file and scripts that make it easy to run the software.

Chapter 7

Generalized Linear Models

This chapter focuses on familty of models that become linear when a link function is used to transform the data.

7.1 Lesson Plan

Chapter 8

Generalized Additive Models

This chapter focuses on

8.1 Lesson Plan

Chapter 9

Non-Linear Models and Neural Networks

This chapter focuses on models that are nonlinear in their parameters. The majority of the coverage is on Neural Networks.

9.1 Lesson Plan: Perceptron

9.2 Lession Plan: Multi-Output Prediction

$$\boxed{\textbf{ClassDate : Tuesday, March 31 – DueDate : Wednesday, April 1}}$$

Read Section 9.3 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class.

9.2.1 Model Equation

For multi-output prediction, the response becomes a vector.

$$\mathbf{y} \;=\; \mathbf{f}(B \cdot \mathbf{x}) + \boldsymbol{\epsilon} \;=\; \mathbf{f}(B^t\mathbf{x}) + \boldsymbol{\epsilon}$$

where $\mathbf{y}$ is an n_y-dimensional output/response random vector,

$\mathbf{x}$ is an n_x-dimensional input/data vector,

B is an n_x-by-n_y parameter matrix,

$\mathbf{f} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ is a function mapping vectors to vectors, and

$\boldsymbol{\epsilon}$ is an n_y-dimensional residual/error random vector.

The `PredictorMat2` serves as an abstract base class for several types of models, including Muli-Variate Regression and many types of Neural Networks.

```
abstract class PredictorMat2 (x: MatriD, y: MatriD,
                              protected var fname: Strings,
                              hparam: HyperParameter)
         extends Predictor with Error
```

Much of the basic functionality for such models is implemented in this abstract class. For example, the important capabilities for forward selection are provided.

ITEM 1: The Concrete Dataset has 7 predictor variables (not including the intercept) and 3 response variables. Define the dimensionality for $\mathbf{x}$, $\mathbf{y}$ and B.

ITEM 2: If $\mathbf{f}$ is the identity function (maps a values to itself), then an `MV_Regression` model may be used. Removing $\boldsymbol{\epsilon}$ from the model equation gives the prediction equation for $\hat{\mathbf{y}}$. Write the equation for computing the $\hat{\mathbf{y}}$ vector.

ITEM 3: Drill down to the element level and write the equations for

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 & = & ? \cdot ? \\ \hat{y}_1 & = & ? \cdot ? \\ \hat{y}_2 & = & ? \cdot ? \end{bmatrix}$$

Hint: Draw a matrix for B^t [?] and a column vector for $\mathbf{x}$ [?] and carry out a matrix-vector product operation. Each formula is the dot product of two vectors.

9.2.2 Forward Selection

Forward selection starts with a given set of columns (predictor variables) and adds the variable x_j that improves the model the most. Improvement is measured in terms of the `index_q` Quality of Fit (QoF) measure, which defaults to `index_rSqBar`. A new model is tried by calling the `buildModel` abstract method that's implemented in the subclasses. For example, its implementation in `NeuralNet_3L` class is shown below.

```
def buildModel (x_cols: MatriD): NeuralNet_3L =
{
    new NeuralNet_3L (x_cols, y, -1, null, hparam, f0, f1, itran)
} // buildModel
```

ITEM 4: Translate ScalaTion's `forwardSel` method Scala to Python. The `forwardSelAll` or something equivalent will need to be implemented for Project 2 (but is not part of the quiz).

```
def forwardSel (cols: Set [Int], index_q: Int = index_rSqBar): (Int, PredictorMat2) =
{
    var j_mx   = -1                                            // best column, so far
    var rg_mx  = null.asInstanceOf [PredictorMat2]             // best model, so far
    var fit_mx = noDouble                                      // best fit, so far
    for (j <- x.range2 if ! (cols contains j)) {
        val cols_j = cols + j                                  // try adding variable x_j
        val x_cols = x.selectCols (cols_j.toArray)             // x projected onto cols_j
        val rg_j   = buildModel (x_cols)                       // regress with x_j added
        rg_j.train ().eval ()                                  // train model, evaluate QoF
        val fit_j = rg_j.fitA(0).fit(index_q)                  // new fit for first response
        if (fit_j > fit_mx) { j_mx = j; rg_mx = rg_j; fit_mx = fit_j }
    } // for
    if (j_mx == -1) {
        flaw ("forwardSel", "could not find a variable x_j to add: j = -1")
    } // if
    (j_mx, rg_mx)                                               // return best column, model
} // forwardSel
```

Here is a partial translation to Python.

```
# forwardSel.py
# Sample Python code for ScalaTion's forwardSel method
# Replace Model class with a one from a Python library

import sys

# class definition for Model

class Model:
    def __init__ (self, xx):
       self.x = xx
       self.q = 1
    def train (self):
       print ("train the model")
    def eval (self):
       print ("evaluate the model")
    def fit (self, index_q):
       print ("return the QoF")
       self.q = self.q + 1
       return self.q

# definition of buildModel function

def buildModel (x_cols):
    print ("build a model using ", x_cols)
    return Model (x_cols)
```

```
# definition of forwardSel function

def forwardSel (cols, index_q, x, xcolumns):
    j_mx   = -1                                          # best column, so far
    fit_mx = - sys.float_info.max                        # best fit, so far
    print ("start for loop")
    for j in xcolumns:
        print ("process column ", j)
        if not j in cols:
            cols_j = cols.union ({j})                    # try adding variable x_j
            x_cols = [ x[index] for index in cols_j ]    # x projected onto cols_j
            rg_j   = buildModel (x_cols)                 # regress with x_j added
            rg_j.train ()                                # train model
            rg_j.eval ()                                 # evaluate QoF
            fit_j = rg_j.fit(index_q)                    # new fit for first response
            if fit_j > fit_mx:
               j_mx   = j
               fit_mx = fit_j
    if j_mx == -1:
        print ("forwardSel: could not find a variable x_j to add: j = -1")
    return j_mx                                          # return best column

# main

x = [[ 1.0, 3.0],                                        # marrix stored column-wise
    [2.0, 4.0]]
xcolumns = [0, 1]
print (x)
cols = {0}
index_q = 0
next_j =forwardSel (cols, index_q, x, xcolumns)
print ("the next variable/column to add is ", next_j)

# end
```

The **forwardSel** method works by trying each variable currently not in the model and picking the one giving the most improvement in the first response variable $y(0)$. If a data science software package does not support Forward Selection, this logic can be readily added using the language for that package. It is straighforward to write a method that calls **forwardSel** until thesre is no further improvement.

```
def forwardSelAll (index_q: Int = index_rSqBar): (Set [Int], MatriD) =
{
    val rSq  = new MatrixD (x.dim2 - 1, 3)                          // R^2, R^2 Bar, R^2 cv
    val cols = Set (0)                                              // start with x_0 in model
    breakable { for (l <- 0 until x.dim2 - 1) {
        val (j, rg_j) = forwardSel (cols)                           // find most predictive x_j
        if (j == -1) break
        cols     += j                                               // add variable x_j
        val fit_j = rg_j.fitA(0).fit
        rSq(l)    = Fit.qofVector (fit_j, rg_j.crossValidate ())  // use new model, rg_j
        if (DEBUG) {
            val k = cols.size + 1
            println (s"forwardSel: add (#$k) variable $j, qof = ${fit_j(index_q)}")
        } // if
    }} // breakable for
    (cols, rSq.slice (0, cols.size-1))
} // forwardSelAll
```

ITEM 5: Two-Page Term Project Proposal.

A **two-page project proposal** giving a detailed description of the application you propose to develop must be submitted as part of this quiz. The project includes data collection, data analytics, interpretation and recommendations for a real-world project. May use ScalaTion, R, Keras, Spark, Scikit-Learn or other approved data science/machine learning toolkit. The term project including a 20-minute presentation and demo will be presented during the last week of class. It must address the nine points/questions listed below. Submit the Presentation (e.g., PowerPoint) slides by midnight on the last day of class. Worth twice the points of regular projects.

For additional information about these nine points, read section 1.3 in the textbook.

1. **Problem Statement**. Describe a problem that can be partially addressed by collecting data to train models. The trained models will then be used to make predictions, forecasts or decisions/classifications. All projects must be of the supervised learning type. Pick only one of the three.

$$\begin{array}{rclr} \hat{y} & = & f(\mathbf{x}) \ \in \ \mathbb{R} & \text{Prediction} \\ \hat{y}_t & = & f(\mathbf{y}_{[t-1,t-p]}, \mathbf{x}) \ \in \ \mathbb{R} & \text{Forecasting} \\ \hat{y} & = & f(\mathbf{x}) \ \in \ \{0,1\} & \text{Binary Classification} \end{array}$$

The study should be focused and the purpose of the study should be clearly stated. An example of a forecasting problem would be to predict the future curve (number infected vs. time) for the Coronvirus (Covid-19). An example of a classification problem would be to classify an individual based upon demographics and symptoms for the Coronvirus (Covid-19). Only two groups may work on the same problem and the same type of supervised learning.

2. **Collection and Description of Datasets**. The study must include at least two large datasets. Their relevance to the study and relationship to each other must be clearly explained. Unrelated datasets are not permitted.

3. **Data Preprocessing Techniques Applied**. It is convenient to load data as is into the toolkit's data frame or associated database. First remove key/id columns and columns with zero variance and then convert strings into integers. Next, missing values should be handled by (i) removing columns with too many missing values, or (ii) imputing values for them. Finally, extract a matrix from the data frame or database. Read chapter 3 for details.

4. **Visual Examination**. Pick a response column/variable from the matrix and see how the other columns/variables relate to it. Before modeling techniques are chosen, perform **Exploratory Data Analysis (EDA)**, by plotting the response variable versus each of the other (predictor) variables. The correlations between each of the columns may be examined by looking at the correlation matrix.

5. **Modeling Techniques Chosen**. All projects must include models from five Complexity Classes: (i) NullModel, (ii) Simple, easy to explain models (e.g., Simple Linear Regression

for highest correlated feature), (iii) Standard, easy to explain models (e.g., Multiple Linear Regression), (iv) Intermediate, performant models (e.g., Quadratic Cross Regression, Extreme Learning Machines) (v) Complex, time-consuming models (e.g., Neural Networks). At least one modeling technique should be chosen from each class.

Table 9.1: Example Modeling Techniques per Problem Type and Complxity Class

Complexity Class	**Prediction**	**Forecasting**	**Classification**
NullModel	NullModel	NullModel	NullModel
Simple	SimpleRegression	RandomWalk	NaïveBayes
Standard	Regression	AR(1)	Logistic Regression
Intermediate	QuadXRegression	ARIMA	Random Forest
Complex	NeuralNet_XL	LSTM	NeuralNet_Classif_XL

Avoid these two problems with datasets:

- Avoid datasets that are too easy as Simple and Standard modeling techniques perform so well that there is no room for improvement left for advanced modeling techniques.
- Avoid datasets that are too hard as no commonly used modeling technique seems to not work at all. Could be that there is no relationship between the response variable and the predictor variables, or complex domain-specific, theory-based models are needed.

Replace the dataset with another one in either case.

6. **Explanation of Why Techniques Were Chosen**. For the Intermediate and Complex models, ideally try more than one modeling technique and have a reason for the ones that were picked.

7. **Feature Selection**. Discuss when in the modeling process and how features were eliminated.

8. **Reporting of Results**. Explain the experimental setup in sufficient detail that the TA can reproduce your results. Produce tables and plots to summarize your results. Be sure to provide the Quality of Fit (QoF) measures. Finally, plot the actual response and the predicted response vs. the instance index. Ideally, sort on the the actual response to made this plot easier to interpret.

9. **Interpretation of Results**. What insights or understanding of the system or process under study can be gained from the results? What issues are brought forward by the study. What changes do the results suggest? As a concise summary, list a few overall **Recommendations of the Study**.

9.3 Lesson Plan: Three-Layer Neural Network

ClassDate : Wednesday, April 1 – DueDate : Thursday, April 2

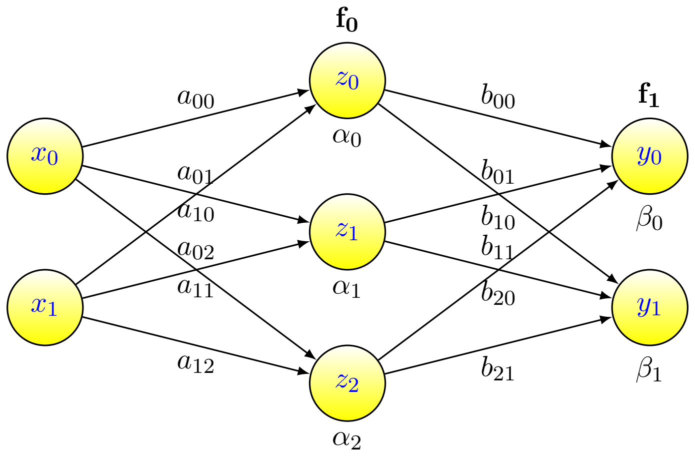

Figure 9.1: Three-Layer (input, hidden, output) Neural Network

Table 9.2: Componenets of Three-Layer Neural Network

Component	**Elements**	**Description**
$\mathbf{x}$	$[x_0, x_1]$	input vector
$\mathbf{z}$	$[z_0, z_1, z_2]$	hidden vector
$\mathbf{y}$	$[y_0, y_1]$	output vector
A	$[a_{jh}] \in \mathbb{R}^{n_x \times n_z}$	first weight matrix
$\boldsymbol{\alpha}$	$[\alpha_0, \alpha_1, \alpha_2]$	first bias vector
B	$[b_{hk}] \in \mathbb{R}^{n_z \times n_y}$	second weight matrix
$\boldsymbol{\beta}$	$[\beta_0, \beta_1]$	second bias vector
$\mathbf{f_0}$	$\mathbf{f_0} : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_z}$	first activation function
$\mathbf{f_1}$	$\mathbf{f_1} : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_y}$	second activation function

Read Section 9.5 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class date.

9.3.1 Model Equation

Recall the Model Equation for Three-Layer (with one hidden) Neural Networks.

$$\mathbf{y} \;=\; \mathbf{f_1}(B \cdot \mathbf{f_0}(A \cdot \mathbf{x})) + \boldsymbol{\epsilon} \;=\; \mathbf{f_1}(B^t \mathbf{f_0}(A^t \mathbf{x})) + \boldsymbol{\epsilon}$$

It is not uncommon for the output/response $\mathbf{y}$ to be a scalar rather than vector, i.e., y. In this case the output layer consists of a single node.

ITEM 1: Draw a Three-Layer Neural Network with $n_x = 2$ input nodes, $n_z = 3$ hidden nodes and $n_y = 1$ output node. Write out the 2-by-3 parameter/weight matrix $A = [a_{jh}]$ and label the first set of 6 edges in the neural network with the approprate a_{jh} weight.

$$A \;=\; [\;?\;]$$

Do the same for the 3-by-1 parameter/weight matrix $B = [b_{hk}]$ where $k = 0$ and label the second set of 3 edges in the neural network with the approprate b_{h0} weight.

$$B \;=\; [\;?\;]$$

Note: for simplicity, the 3-dimensional bias vector $\boldsymbol{\alpha}$ to be added before hidden layer and the 1-dimensional bias vector $\boldsymbol{\beta}$ to be added before the ouput layer are ignored. Try adding these into the equations (not part of the quiz).

ITEM 2: When the error term $\boldsymbol{\epsilon}$ is removed, the model equation can be used for making predictions. In this form it can be divided into two parts. The first part computes intermediate values (or hidden/complex features) $\mathbf{z}$.

$$\mathbf{z} \;=\; \mathbf{f_0}(A \cdot \mathbf{x}) \;=\; \mathbf{f_0}(A^t \mathbf{x})$$

The second part computes a predicted output/response value $\hat{y}$.

$$\hat{y} \;=\; \mathbf{f_1}(B \cdot \mathbf{z}) \;=\; \mathbf{f_1}(B^t \mathbf{z})$$

Write the three equations for $\mathbf{z}$, one for z_0, one for z_1 and one for z_2. Write the one equation for $\hat{y}$.

ITEM 3: Since the B consists of a single column, it may be replaced by a vector $\mathbf{b}$. Show the resulting simplification to the model equation or one of its parts.

ITEM 4: Next, it may be useful to allow the neural network signals to scale out to the original scale for the response/output variable y. This can be done making the last activation function $\mathbf{f_1}$ be the identity function (maps a value to itself). Show the resulting simplification of this step to the model equation or one of its parts.

ITEM 5: Finally, what would happen is the first activation function $\mathbf{f_0}$ was also the identity function. Show the resulting simplification of this step to the model equation.

ITEM 6: What other modeling technique is the simplified Model Equation now equivalent to. Write both prediction equations.

$$\begin{aligned} \hat{y} &= ? && \text{Simplified Neural Network} \\ \hat{y} &= ? && \text{Other Modeling Technique} \end{aligned}$$

$\boxed{\textbf{ClassDate : Thursday, April 2 – DueDate : Tuesday, April 7}}$

Turn in all the ITEMS listed until the next class date.

9.3.2 Training/Optimization

Start with the Matrix Version of the training/optimization equations. Recall that m is the number of instances currently in use (e.g., entire dataset, training dataset, or current mini-batch). The numbers of nodes per layer are n_x, n_z and n_y, for the input, hidden and output layers, respectively.

$$\begin{aligned} \text{Dataset :} \quad & X \;=\; [x_{ij}] \;\in\; \mathbb{R}^{m \times n_x} \;, \quad & Y \;=\; [y_{ik}] \;\in\; \mathbb{R}^{m \times n_y} \\ \text{Parameters :} \quad & A \;=\; [a_{jh}] \;\in\; \mathbb{R}^{n_x \times n_z} \;, \quad & B \;=\; [b_{hk}] \;\in\; \mathbb{R}^{n_z \times n_y} \end{aligned}$$

Recall from the textbook that the biases are carried along in the `NetParam` class, so this quiz will ignore them. The simplification of the `NetParam` abstraction allows a weight matrix and its bias vector to be treated as a matrix-like construct.

ITEM 1: The m-by-n_z **hidden layer matrix** Z has a row per instance and a column per node in the hidden layer.

$$Z \;=\; \mathbf{f_0}(XA)$$

Letting vector $\mathbf{z_{:h}}$ be the h^{th} column in the Z matrix, derive its formula from the matrix equation.

$$\mathbf{z_{:h}} \;=\; \mathbf{f_0}(\dots)$$

Hint: Consider how information flows from all the nodes in the input layer to the h^{th} node in the hidden layer.

ITEM 2: The m-by-n_y **prediction matrix** $\hat{Y}$ has a row per instance and a column per per node in the output layer.

$$\hat{Y} \;=\; \mathbf{f_1}(ZB)$$

Letting vector $\hat{\mathbf{y}}_{\mathbf{:k}}$ be the k^{th} column in the $\hat{Y}$ matrix, derive its formula from the matrix equation.

$$\mathbf{\hat{y}_{:k}} \;=\; \mathbf{f_1}(\dots)$$

ITEM 3: The m-by-n_y **negative error matrix** E is the predicted minus the actual/target values.

$$E \;=\; \hat{Y} \;-\; Y$$

Letting vector $\mathbf{e_{:k}}$ be the k^{th} column in the E matrix, derive its formula from the matrix equation.

ITEM 4: The m-by-n_y **delta one matrix** Δ^1, used for adjusting parameters B, is the elementwise matrix (Hadamard) product of $\mathbf{f}_1'(ZB)$ and E.

$$\Delta^1 \;=\; \mathbf{f}_1'(ZB) \circ E$$

Letting vector $\boldsymbol{\delta}^1_{:k}$ be the k^{th} column in the Δ^1 matrix, derive its formula from the matrix equation.

ITEM 5: The m-by-n_z **delta zero matrix** Δ^0 matrix, used for adjusting parameters A, is the elementwise matrix (Hadamard) product of $\mathbf{f}_0'(XA)$ and $\Delta^1 B^t$.

$$\Delta^0 \;=\; \mathbf{f}_0'(XA) \circ (\Delta^1 B^t)$$

Letting vector $\boldsymbol{\delta}^0_{:h}$ be the h^{th} column in the Δ^0 matrix, derive its formula from the matrix equation.

ITEM 6: The n_z-by-n_y **one parameter matrix** B, connecting the hidden layer to the output layer, is updated based on the Δ^1 matrix.

$$B \;=\; B \;-\; Z^t \Delta^1 \,\eta$$

Letting vector $\mathbf{b_{:k}}$ be the k^{th} column in the B matrix, derive its formula from the matrix equation.

ITEM 7: The n_x-by-n_z **zero parameter matrix** A, connecting the input layer to the hidden layer, is updated based on the Δ^0 matrix.

$$A \;=\; A \;-\; X^t \Delta^0 \,\eta$$

Letting vector $\mathbf{a_{:h}}$ be the h^{th} column in the A matrix, derive its formula from the matrix equation.

$$\boxed{\textbf{ClassDate : Tuesday, April 7 – DueDate : Wednesday, April 8}}$$

Turn in all the ITEMS listed until the next class date.

9.3.3 Example Error Calculation Problem

This example Neural Network calculation is partially worked out in section 9.5.5 in the textbook. It follows a single ($m = 1$) instance vector

$$\mathbf{x} \;=\; [x_0, x_1]$$

through the Neural Network showing how it could be used to update the parameters (weights and biases) in the network. In practice, a mini-batch of instances would be used rather than a single instance.

ITEM 1: Draw the Neural Network corresponding to the example problem. The number of nodes per layer are $n_x = 2$, $n_z = 2$ and $n_y = 1$ for the input, hidden and output layer, respectively.

ITEM 2: Compute the negative **error** matrix E for the first iteration. Do this by overlaying all the calculations leading to E on the drawn neural network.

$$E \;=\; [\;\epsilon_0\;] \;=\; [\;?\;]$$

ITEM 3: Compute the **delta** (slope adjusted error) matrices Δ^1 and Δ^0 for the first iteration (see Exerice 1 in section 9.5.6). Do this by overlaying all the calculations for the delta matrices on the drawn neural network.

$$\Delta^1 \;=$$
$$\Delta^0 \;=$$

ITEM 4: Compute the updates to the B and A **parameters** for the first iteration (see Exerice 2 in section 9.5.6). Give the amount of change as well as the new updated values.

B	$=$	weight matrix
$\boldsymbol{\beta}$	$=$	bias vector
A	$=$	weight matrix
$\boldsymbol{\alpha}$	$=$	bias vector

The rest of the exercises in 9.5.6 are recommended, but need not be turned in.

9.4 Lesson Plan: Deep Neural Network

$$\boxed{\textbf{ClassDate} : \textbf{Wednesday}, \textbf{April 8} - \textbf{DueDate} : \textbf{Thursday}, \textbf{April 9}}$$

Read Section 9.6 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class date.

9.4.1 Model Equation

Deep Neural Networks utilize more than the one hidden layer used by traditional Neural Networks.

ITEM 1: Rewrite the model equation for Three-Layer Neural Networks by denoting parameter matrix A as B_0 and parameter matrix B as B_1.

$$\mathbf{y} = \mathbf{f_1}(\dots) + \boldsymbol{\epsilon}$$

ITEM 2: Add a second hidden layer, splitting $\mathbf{z}$ into $\mathbf{z_1}$ and $\mathbf{z_2}$, and introduction a third parameter matrix B_2 and a third activation function $\mathbf{f_2}$.

$$\mathbf{y} = \mathbf{f_2}(\dots) + \boldsymbol{\epsilon}$$

ITEM 3: By designating $\mathbf{x}$ as $\mathbf{z_0}$ and $\mathbf{y} + \boldsymbol{\epsilon}$ as $\mathbf{z_3}$, the equation may be rewritten recursively where the values for layer $l+1$ are determined by layer l,

$$\mathbf{z_{l+1}} = \mathbf{f_l}(\dots)$$

9.4.2 Training/Optimization

Recall that the training equation is basically the model equation, except that multiple instances are used.

ITEM 4: Take the recursive equation and convert vector $\mathbf{z_l}$ to matrix Z_l.

$$Z_{l+1} = \mathbf{f_l}(\dots)$$

The rest of matrix equations are roughly the same as for Three-Layer Neural Networks. In `train0` these matrix equations are iteratively computed inside the following loop.

```
    for (epoch <- 1 to maxEpochs) {
```

With Deep Neural Networks, **overfitting** may easily happen.

ITEM 5: What happens to the relative positions of the R^2 and R^2_{cv} curves that likely indicates overfitting? Note, the curves correspond to those produced for Project 2 where the horizontal axis is the number of paremeters in the model.

ITEM 6: How does reducing the number of nodes in the hidden layers affect these curves? Try it and see (do not just speculate).

ITEM 7: In Keras, try utilizing a validation set to reduce overfitting and plot the loss function vs. epoch on training data and on the validation data. How do these curves differ?

ITEM 8: List other techniques that may be used to reduce overfitting. Give a brief explanation of why each on is thought to work.

The exercises in section 9.6.6 in the textbook are recommended, but are not part of this quiz. In particular, exercise 6 on **tuning hyper-parameters** should be helpful for Project 2.

9.5 Lesson Plan: Extreme Learning Machines

$$\boxed{\textbf{ClassDate : Thursday, April 9 – DueDate : Tuesday, April 14}}$$

Read Section 9.8 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class date.

9.5.1 Model Equation

The model equation for an Extreme Learning Machine (ELM) is similar to a Three-Layer Neurral Network. The differnece is that the first set of parameters A are initialized, but are not updated. This avoids the use of the slow back-propogation algorithm. Only the second set of parameters are optimized (B). Although `TranRegression` may be used when there is a second activation function $\mathbf{f_1}$, we consider the case where the second activation function is the identity function.

For simplicity, we only condsider the `ELM_3L1` class that supports only a single output/response variable.

$$y \;=\; \mathbf{b} \cdot \mathbf{f_0}(A^t \mathbf{x}) \,+\, \boldsymbol{\epsilon}$$

Note, for multiple output/response variables, the `ELM_3L` class may be used.

ITEM 1: Consider the example calculation for the Three-Layer Neural Network. Now suppose the dataset $(X, \mathbf{y})$ consits of $m = 3$ instances.

$$X \;=\; \begin{bmatrix} 2.0 & 1.0 \\ 1.5 & 0.9 \\ 1.0 & 1.1 \end{bmatrix}, \qquad \mathbf{y} \;=\; \begin{bmatrix} 0.8 \\ 0.7 \\ 0.6 \end{bmatrix}$$

Using the same initializations for the bias vector $\mathbf{a^b}$ and the weight matrix A as in the textbook's example in section 9.5.5, Calculate the m-by-n_z Z matrix.

$$Z \;=\; \mathbf{f_0}(XA)$$

ITEM 2: Notice that the Z matrix will never change, so that multiplying it by the parameter vector $\mathbf{b}$ will yield a predicted output value.

$$\hat{\mathbf{y}} = Z\mathbf{b}$$

Using the B weight matrix given in section 9.5.5, calculate the predicted response vector $\hat{\mathbf{y}}$. Note the single column matrix B may be treated as vector $\mathbf{b}$. Ignore the biases for the time being.

ITEM 3: Calculate the error vector $\boldsymbol{\epsilon}$ and the sum of squared errors (sse).

ITEM 4: Notice that the parameter vector $\mathbf{b}$ is unoptimized. Propose an efficient algorithm for optimizing these parameters, not using an algorithm in the Gradient Descent family.

ITEM 5: Write the vector equation to solve for parameter vector $\mathbf{b}$.

$$\mathbf{b} = ?$$

ITEM 6: Carry out the calculation to compute optimized values for parameter vector $\mathbf{b}$. Hint: Try using Spreadsheet software.

ITEM 7: Recalculate the error vector $\boldsymbol{\epsilon}$ and the sum of squared errors (sse). Has there been improvement?

Chapter 10

Time-Series/Temporal Models

This chapter focuses on data that have a sequential order (e.g., time-series data). The typical assumption that the instances making up a dataset are independent does not hold for such data (dependencies between adjacent instance may be very high).

10.1 Lesson Plan: Time-Series Models

ClassDate : Teusday, April 14 – DueDate : Wednesday, April 15

Read Sections 10.1-2 in the **Introduction to Data Science in ScalaTion** textbook. Turn in all the ITEMS listed until the next class date.

Forecasting is similar to predictions, but differs in the following ways:

1. The instances are orderred, for example in time.
2. They are no longer Identically Distributed (ID), but may be highly dependent.
3. Forecasting depends heavily on previous values in the time-series.

Consequently, the model equation involves previous values of the response variable.

10.1.1 Model Equation

The value to be forecasted is y_t is dependent on its last p values as well as possibly other exogeneous variables $\mathbf{x}$.

$$y_t \;=\; f(\mathbf{y}_{[t-1,t-p]}, \mathbf{x}) \;+\; \epsilon_t$$

The vector $\mathbf{y}_{[t-1,t-p]}$ hold the most recent past values.

10.1.2 Auto-Correlation Function

The Auto-Correlation Function (AFC) measures correlation over time. The following two vectors $[y_t, y_{t-1}, y_{t-2}, \dots, y_1]$ and $[y_{t-1}, y_{t-2}, y_{t-3} \dots, y_0]$ are likely be highly correlated. As the time gap between the two vectors increases, the correlation is likely to decrease. At some point, distantly past values will have less relevance and need not be included in the model.

The k^{th} lag auto-covariance (auto-correlation), γ_k (ρ_k) is the covariance (correlation) of time-series y_t and times-series y_{t-k}.

$$\begin{aligned} \gamma_k \;&=\; \mathbb{C}\,[y_t, y_{t-k}] && \text{Auto} - \text{Covariance} \\ \rho_k \;&=\; corr(y_t, y_{t-k}) && \text{Auto} - \text{Correlation} \end{aligned}$$

10.1.3 Auto-Regressive (AR) Models

Chapter 11

Clustering

This chapter focuses on techniques for clustering or grouping data instances/vectors according to the similarity or distance from each other.

11.1 Lesson Plan

Chapter 12

Dimensionality Reduction

This chapter focuses on techniques for reducing the number variables in a model by tranforming the space in which the vectors reside. Due to multi-collinearity of vectors in an apparently high-dimensional space, transformation may allow a much lower dimensional model to be effectively used.

12.1 Lesson Plan

Chapter 13

Functional Data Analysis

This chapter focuses on the data that are assumed to be produced by an underlying continuous system or process. Basis functions are used to create smooth continuous functions that match the given data points.

13.1 Lesson Plan

Chapter 14

Simulation Models

This chapter focuses on building models that capture more details about the system or process under study. The models should capture internal structure sufficient to allow the model to mimic the behavior of the actual system or process under study. Due to the more intricate modeling, simulation models are more appropriate for anwering ”what-if” question and generally support greater extrapolation than other types of model.

14.1 Lesson Plan

Chapter 15

Optimization Used in Data Science

This chapter provides a brief introduction to optimization techniques used for fitting model parameters to a dataset.

15.1 Lesson Plan