

Machine Learning: Introduction to Linear Regression, Logistic Regression, and Neural Networks

Chapter 7 Case Studies

Case Studies

So far in this course:

- We have discussed underlying mathematics and algorithms for Linear Regression, Logistic Regression and Neural Network approaches for Supervised Learning
- We have developed a machine learning framework for Supervised Learning that can apply these approaches
- We have discussed approaches for improving optimization for the training algorithm, measuring performance, regularization, and addressing Underfitting and Overfitting
- In this chapter we apply these ideas to address 3 case studies

Case Studies

Section	Case Study	Type	Description
7.1	House Price Prediction	Regression	This case study uses Linear Regression to predict house prices. It also shows how to use “exploratory data analysis” and “feature preprocessing” to prepare the data.
7.2	Spam Classification	Binary Classification	This case study uses a Neural Network to create a spam filter. It shows an approach for converting text into a feature matrix.
7.3	MNIST Digits Classification	Multiclass Classification	This case study uses a Neural Network to identify digits from images. It shows how to convert an image into a feature matrix.

7.1 Case Study: House Price Prediction

Case Study: House Price Prediction

Goal of this Section:

- Show how to use Linear Regression for House Price Prediction
- Introduce Exploratory Data Analysis and Feature Preprocessing to set up the machine learning problem

House Price Dataset - Citation

- House Price Dataset compiled from data in Sindian Dist., New Taipei City, Taiwan
- Source: (University of California, Irvine, Machine Learning Repository)
<https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set>
- Paper describing work:

Yeh, I. C., & Hsu, T. K. (2018). Building real estate valuation models with comparative approach through case-based reasoning. *Applied Soft Computing*, 65, 260-271.

House Price Dataset

- Data located in folder IntroML/Code/Data_House:
 - 414 data samples
 - 1 Value (price-per-unit-area),
 - 3 features (house-age (years), dist-to-nearest-MRT (meters), num-of-stores)
 - Original dataset has 3 additional features not used here (transaction date & house coordinates)

	A	B	C	D	E
1	price-per-unit-area	house-age	dist-to-nearest-MRT	num-of-stores	
2	37.9	32	84.87882	10	
3	42.2	19.5	306.5947	9	
4	47.3	13.3	561.9845	5	
5	54.8	13.3	561.9845	5	
6	43.1	5	390.5684	5	
7	32.1	7.1	2175.03	3	
8	40.3	34.5	623.4731	7	
9	46.7	20.3	287.6025	6	
10	18.8	31.7	5512.038	1	
11	22.1	17.9	1783.18	3	
12	41.4	34.8	405.2134	1	
13	58.1	6.3	90.45606	9	
14	39.3	13	492.2313	5	
15	23.8	20.4	2469.645	4	
16	34.3	13.2	1164.838	4	

Exploratory Data Analysis & Feature Preprocessing

Exploratory Data Analysis

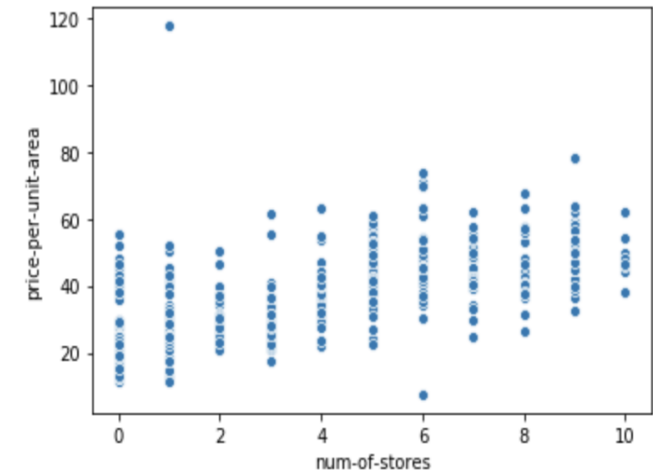
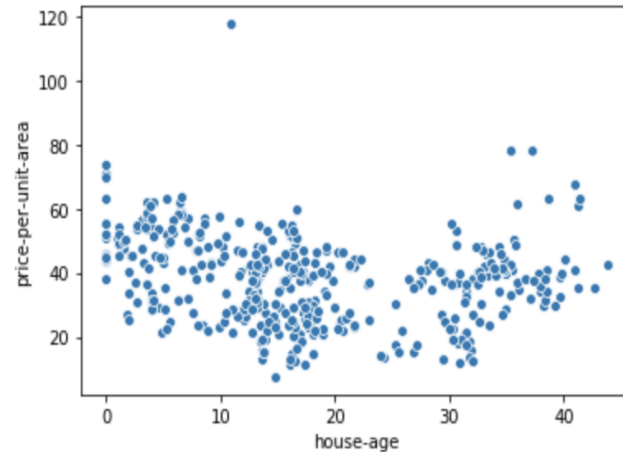
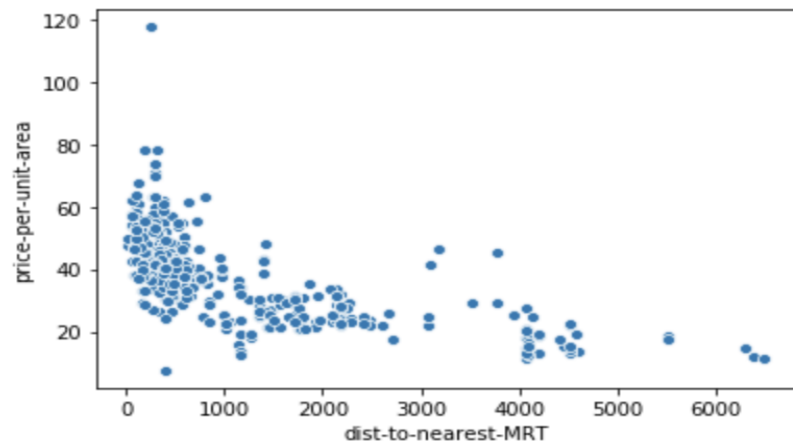
- Approach for analyzing data sets to understand relationships between variables often using visual tools
- For House Price Prediction, do not expect house prices to be linearly dependent on features
- Use Exploratory Data Analysis to determine relationship between prices and features to determine transformations to be made

Feature Preprocessing

- Apply transformations to feature variables suggested by exploratory data analysis
- “Standardize” or “normalize” variables so they are all of roughly the same size
- Remove outliers

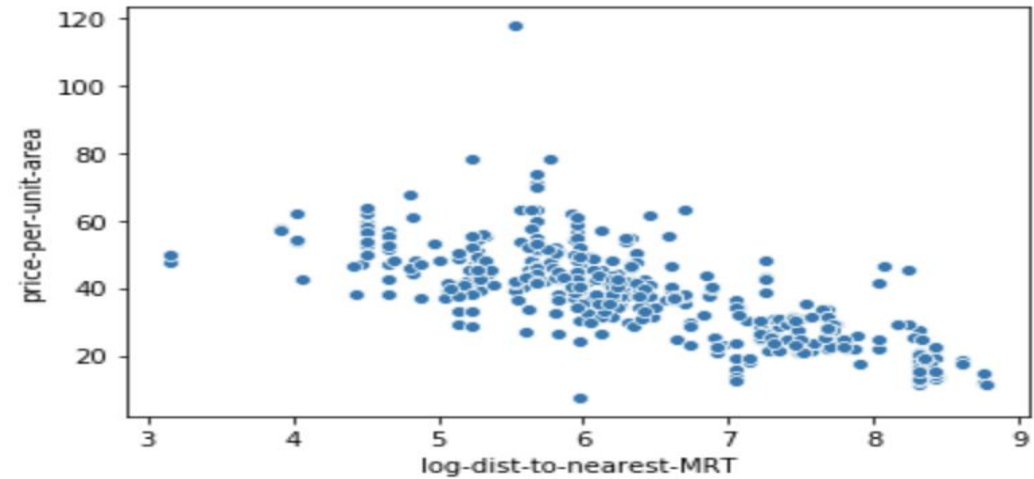
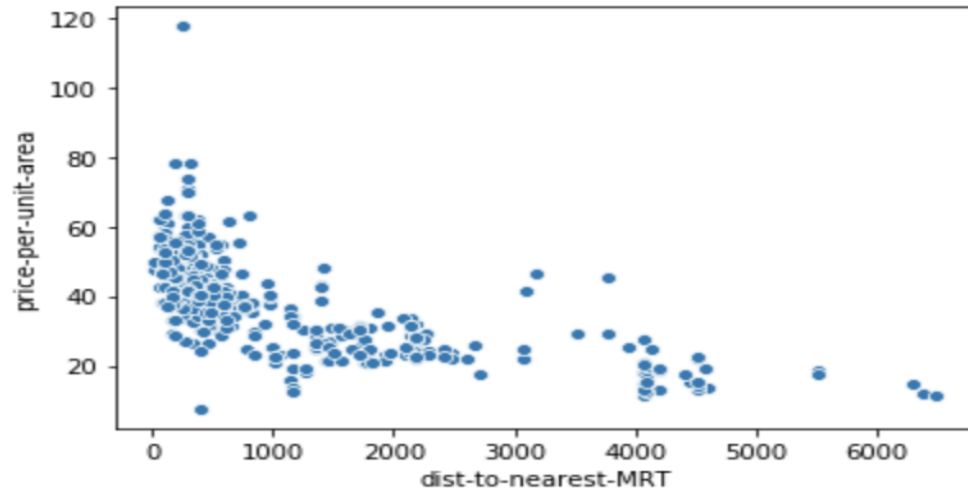
Exploratory Data Analysis

- Scatter plots show relationships between price-per-unit area and each of the features
- Scatter plots suggest non-linear relationships between price-per-unit-area and distance to nearest MRT and house age



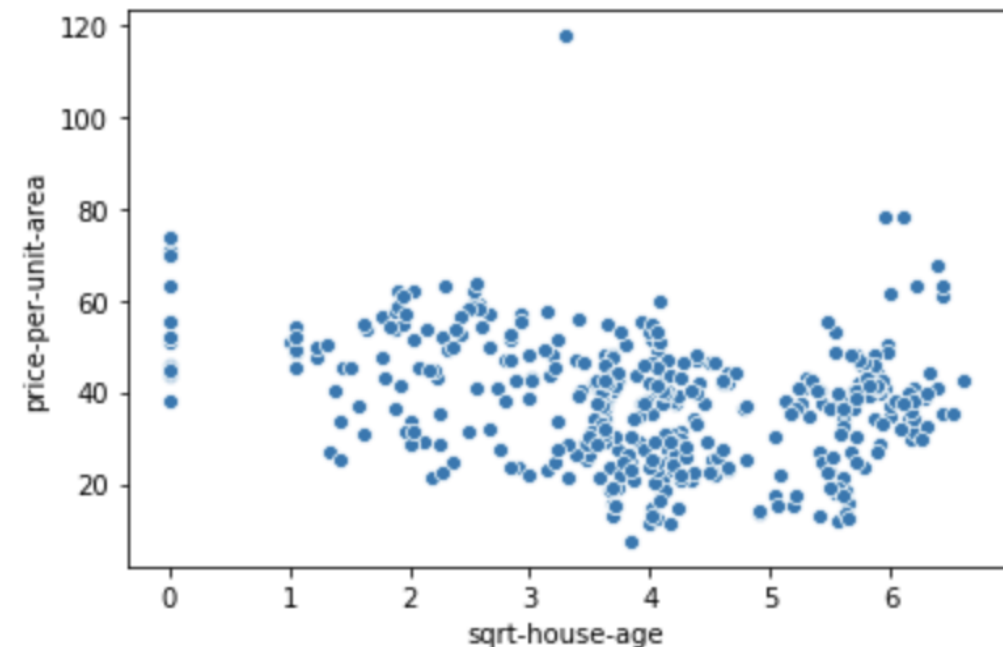
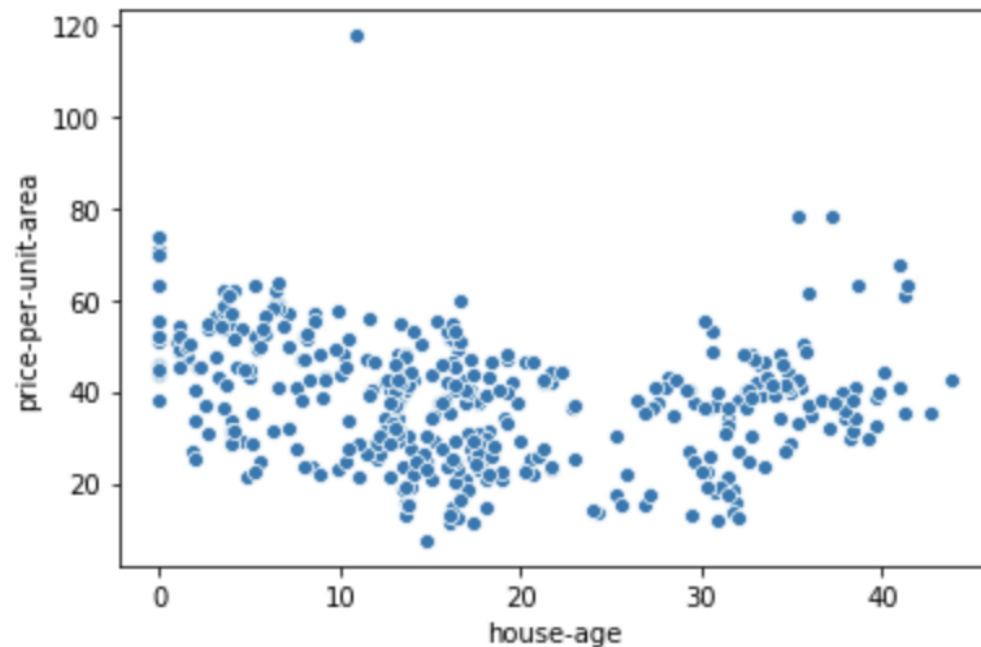
Variable Transformations

- A log transformation is applied to the variable **dist-to-nearest-MRT**
 - Improves linearity between the feature and price



Variable Transformations

- A square root transformation is applied to the variable **house-age**
 - Improves linear relationship between the feature and the price



Feature Standardization

- Feature values are of different magnitudes:
 - Age: roughly 0 – 50
 - Distance to MRT: roughly 0 – 7000
 - Number of stores: 0 - 10
- Standardize Features:
 - Subtract mean and scale by standard deviation of training data
 - Apply same scaling to validation data
 - Results in feature data having mean 0 and standard deviation 1 so that feature values are the same magnitude
- This can improve convergence of optimizer

Value Standardization

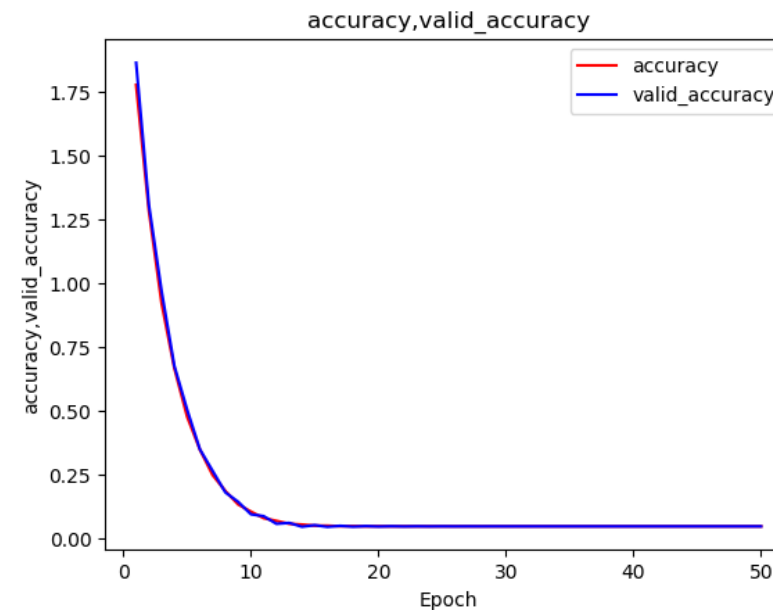
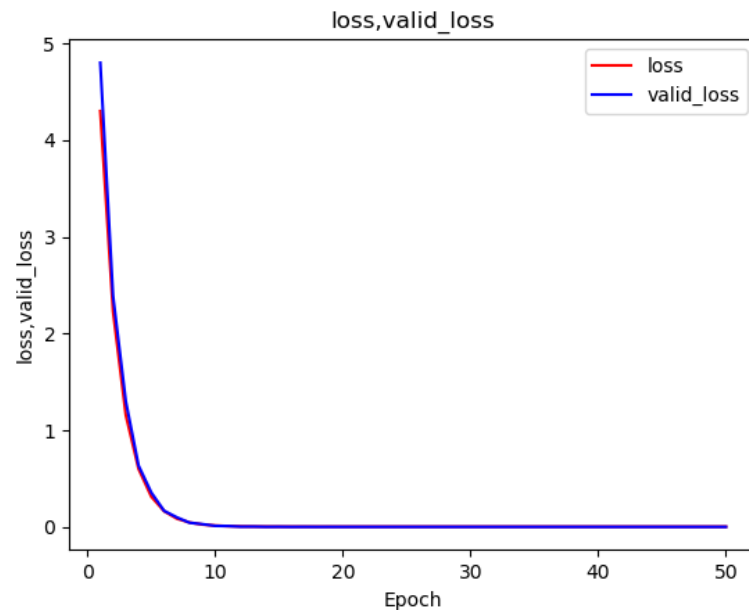
- To improve convergence of optimization, can apply same technique as previous page to standardize price-per-unit-area data
- For simplicity, will simply divide price-per-unit-area data by maximum value in training data set
- Apply exact same scaling factor to validation data set
- Note that predictions of the model will need to be multiplied by this factor to get correct price-per-unit-area

House Price Prediction using Linear Regression

- Example: Dataset
 - 331 samples in training dataset (roughly 80% of samples)
 - 83 samples in validation dataset (roughly 20% of samples)
 - Feature matrix for training (3 x 331)
 - Feature matrix for validation (3 x 83)
- Linear Regression
 - W is 1x3 matrix and b is 1x1 scalar
 - $\lambda = 0.0001$ (regularization)
- Optimization:
 - Gradient Descent ($\alpha = 0.5$)
 - 50 epochs batch

House Price Prediction – Summary of Results

- Run with Transformation, Standardization for X and Y = True
 - Training Loss: 0.0059, Training Accuracy: 0.050 (mean absolute error)
 - Validation Loss: 0.0041, Validation Accuracy: 0.050 (mean absolute error)
 - Loss and Accuracy plots indicate no concerning signs of overfitting



Summary of Results

Run	Transformations/Standardization	Results after 50 epochs
1	Transformation: True Standardization X: True Standardization Y: True	Training Accuracy: 0.050 Validation Accuracy: 0.050
2	Transformation: False Standardization X: True Standardization Y: True	Training Accuracy: 0.056 Validation Accuracy: 0.052
3	Transformation: True Standardization X: False Standardization Y: True	Training Accuracy: Blow up – does not converge Validation Accuracy: Blow up - does not converge

New Code for House Price Prediction

Function/component	Input	Description
load_house	train_pct (float) transform (boolean) standardizeX (boolean) standardizeY (boolean)	Loads house price data and perform feature/value preprocessing. Train_pct ranges from 0.0-1.0 and indicates the proportion of the data to used for training. Inputs transform, standardizeX, and standardizeY, determine the preprocessing transformations made to the data Return: Xtrain, Ytrain, Xvalid, Yvalid
driver_casestudy_house		Driver for house price prediction

House Price Prediction Walkthrough

- Code for walkthrough located at:
IntroML/Code/Version4.1
- Demo of pandas for loading and processing data:
IntroML/Examples/Chapter2/PandasDemos.ipynb
- You can implement the code additions and model suggested on the previous slides by adding to an original Version3.3 of the code
 - Have a look at Version4.1 for hints
- We will perform a walkthrough of the code

Further Investigations

- Data:
 - Only 3 of 6 features in original dataset are used in this section - investigate if results are improved if additional features are used
- Transformations
 - Investigate other transformations of the features
- Optimization
 - Investigate use of different learning rate for Gradient Descent or use of other optimizers to overcome loss blow up
- Use of Neural Network:
 - Nothing prevents us from using a neural network for this problem.
 - Investigate use of a neural network. Use relu activation for each layer
 - See `driver_casestudy_house_nn.py` for an example

7.2 Case Study: Spam Classification

Case Study: Spam Classification

Goal of this Section:

- Describe approach for using neural networks for spam classification using the SMS dataset

Text Classification

Text Classification is an application of supervised machine learning

Examples include:

- Spam Classification
 - Binary Classification
 - Training Data: messages and labels (spam or not spam)
 - Goal: predict if new message is spam or not (use to filter email, for example)
- Sentiment Classification of Reviews
 - Multiclass classification
 - Training Data: reviews and labels (1, 2, 3, 4 or 5 star, for example)
 - Goal: predict rating for new reviews

SMS Spam Collection Dataset - Citation

- Source: (University of California, Irvine, Machine Learning Repository)

<http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>

- Paper describing work:

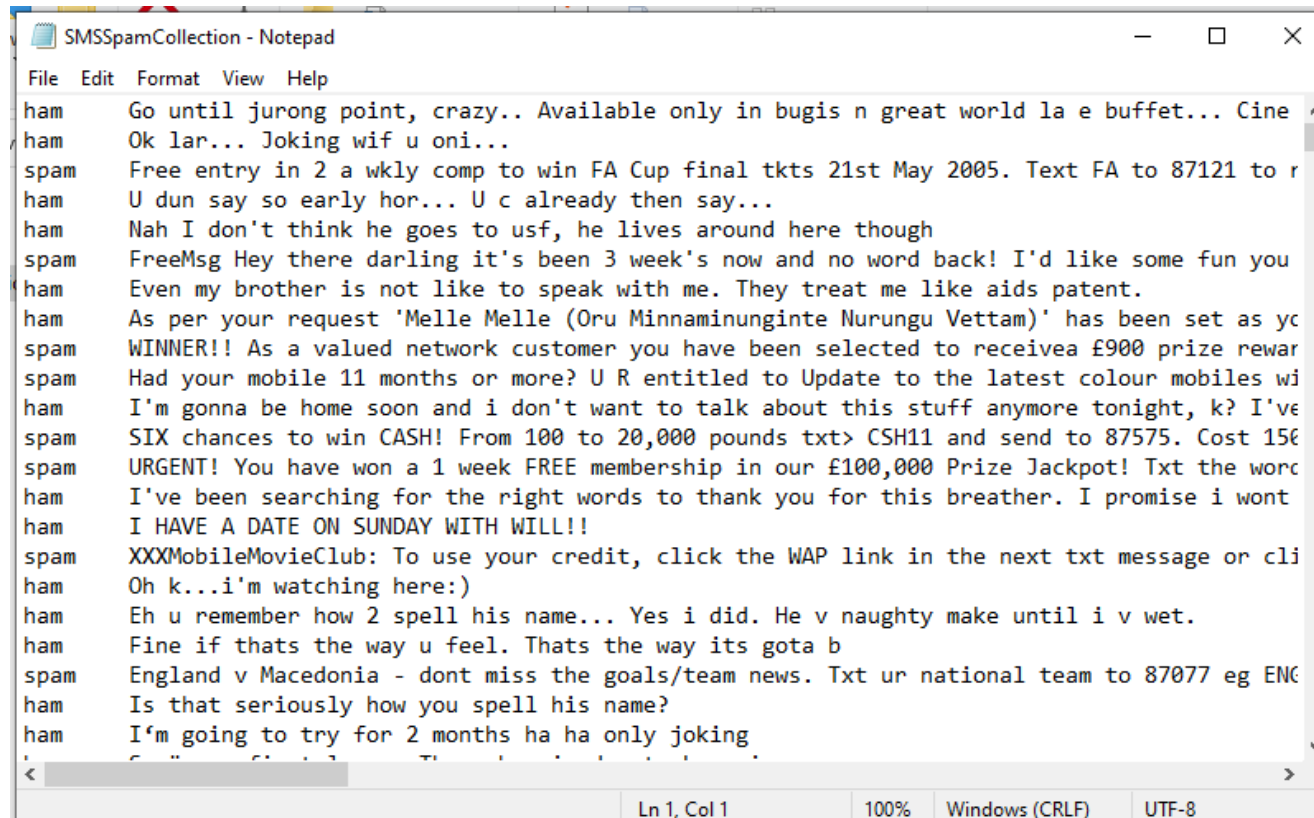
Almeida, T.A., Gómez Hidalgo, J.M., Yamakami, A. Contributions to the study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (ACM DOCENG'11), Mountain View, CA, USA, 2011. (Under review)

- See website:

<http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/>

SMS Spam Collection Dataset

- Data located in folder IntroML/Code/Data_Spam
 - readme.txt file provides details about dataset
 - SMSSpamCollection contains the data
- Consist of 5574 text messages: 4827 (not spam) 747 (spam)
- Each line has label (ham or spam) and message



```
SMSSpamCollection - Notepad
File Edit Format View Help
ham    Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine
ham    Ok lar... Joking wif u oni...
spam   Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to r
ham    U dun say so early hor... U c already then say...
ham    Nah I don't think he goes to usf, he lives around here though
spam   FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you
ham    Even my brother is not like to speak with me. They treat me like aids patient.
ham    As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as yc
spam   WINNER!! As a valued network customer you have been selected to receive a £900 prize rewar
spam   Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles wi
ham    I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've
spam   SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150
spam   URGENT! You have won a 1 week FREE membership in our £100,000 Prize Jackpot! Txt the worc
ham    I've been searching for the right words to thank you for this breather. I promise i wont
ham    I HAVE A DATE ON SUNDAY WITH WILL!!
spam   XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or cli
ham    Oh k...i'm watching here:)
ham    Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
ham    Fine if thats the way u feel. Thats the way its gota b
spam   England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENG
ham    Is that seriously how you spell his name?
ham    I'm going to try for 2 months ha ha only joking
```

Converting Messages into a Feature Matrix

- Rudimentary Approach: CountVectorizer (from sklearn package)
 1. Build a vocabulary consisting of all words in all messages
 - Default setting is case-insensitive (“my”, “My”, “MY” are the same)
 2. Feature matrix entry X_{ij} is number of times word i appears in message j
 3. Feature Matrix has dimensions (# of words x # of messages)
- Can adjust settings to not include some words (called “stop words”), such as “the”, “to”, “and”,, which probably do not impact classification.
- See <https://scikit-learn.org/stable/index.html> for details

CountVectorizer - Example

- 3 Messages: "Call me soon", "CALL to win", "Pick me up soon"
- 7 unique words
- Feature matrix is (7 words) x (3 messages)

Vocabulary:

call
me
pick
soon
to
up
win

Feature Matrix:

1	1	0
1	0	1
0	0	1
1	0	1
0	1	0
0	0	1
0	1	0

Most Common Words in Vocabulary

- Sum Feature Matrix in col direction to get count of words for all messages

Vocabulary:

call
me
pick
soon
to
up
win

Feature Matrix:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Count:

$$\begin{bmatrix} 2 \\ 2 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Can use numpy argsort function to get indices of most common words:
 - In case of ties, go in order of index
 - Indices of 4 most common words: 0, 1, 3, 2

Label Vector

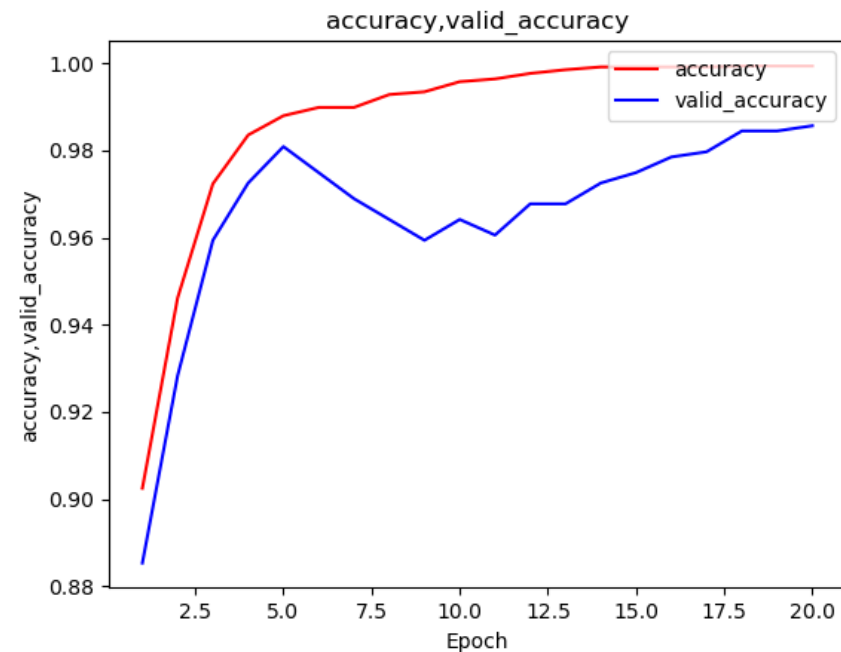
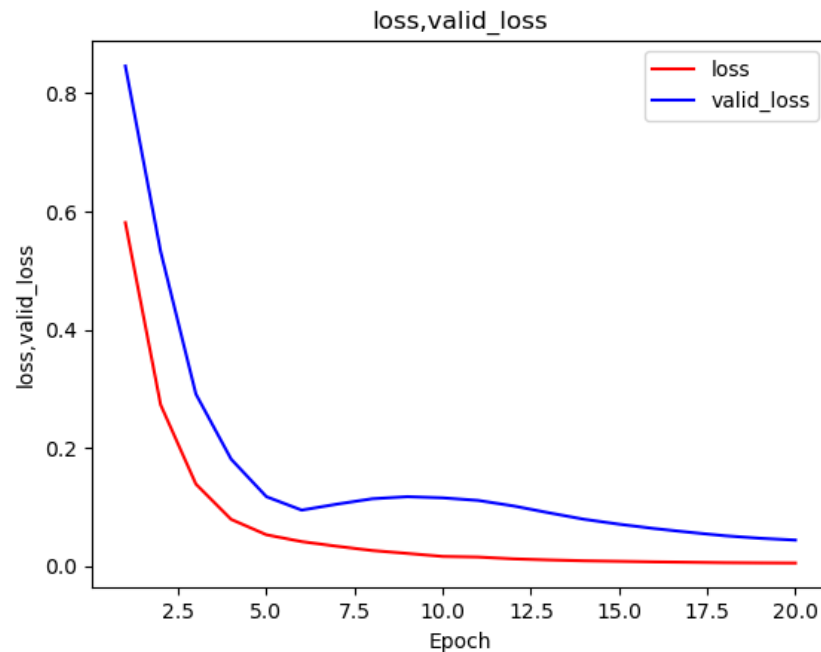
- Label Vector:
 - Convert label ham = 0 and spam = 1

Spam Classification using a Neural Network

- Example: Dataset
 - 4737 (roughly 85%) messages in training dataset
 - 837 (roughly 15%) messages in validation dataset
 - 8572 words in vocabulary
 - Feature matrix for training is (8572 x 4737)
 - Feature matrix for validation is (8572 x 837)
- Neural Network
 - 3 layer neural network
 - Layer 1: 200 units (tanh activation)
 - Layer 2: 50 units (tanh activation)
 - Layer 3: 1 unit (sigmoid activation)
 - Binary Cross Entropy Loss function
 - 1,760,701 total combined entries in $W^{[k]}$ and $b^{[k]}$ in all layers
- Optimization:
 - Adam
 - $\alpha = 0.02, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-7}$
 - 20 epochs (batch_size = 4737 - batch gradient descent)

Spam Classification – Summary of Results

- After 20 epochs:
 - Training Accuracy: 0.9994
 - Validation Accuracy: 0.9857
 - For Validation Set: F1score: 0.9469, Precision: 0.9224, Recall: 0.9727
- Loss and Accuracy plots indicate an overfitting



Spam Classification – Summary of Results

- Confusion matrix for validation dataset:
 - 9 False Positive Messages
 - 3 False Negative Messages

```
Metrics for Validation Dataset
Confusion Matrix
Actual
Predicted 0 1
0 718 3
1 9 107
F1Score: 0.9469026544482732 - Precision: 0.9224137927058561 - Recall: 0.972727272285124
```

```
False Positive messages - Actual = not spam - Predicted = spam
if you text on your way to cup stop that should work. And that should be BUS
How long before you get reply, just defer admission til next semester
I lost 4 pounds since my doc visit last week woot woot! Now I'm gonna celebrate by stuffing my face!
I'm vivek:)i got call from your number.
Why didn't u call on your lunch?
Thanks for being there for me just to talk to on saturday. You are very dear to me. I cherish having you as a brother and role model.
Sorry i missed your call. Can you please call back.
Dhoni have luck to win some big title.so we will win:)
Armand says get your ass over to epsilon
```

```
False Negative messages - Actual = spam - Predicted = not spam
Check Out Choose Your Babe Videos @ sms.shsex.netUN fgkslpoPW fgkslpo
dating:i have had two of these. Only started after i sent a text to talk sport radio last week. Any connection do you think or coincidence?
ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE MINS. INDIA CUST SERVs SED YES. L8ER GOT MEGA BILL. 3 DONT GIV A SHIT. BAILIFF DUE IN DAYS. I O Â£250 3 WANT Â£800
```


New Code for Spam Classification

Method	Input	Description
load_spam	train_pct (float)	Loads spam data base and returns train and validation feature matrices (based on CountVectorizer) and label vectors. Also returns original messages in train and validation datasets. Return: Xtrain, Ytrain, Xvalid, Yvalid, Xtrain_raw, Xvalid_raw
data_analysis	X (numpy array) Y (numpy array) nmostcommon (integer) vectorizer (CountVectorizer instance)	Takes in X feature matrix generated by CountVectorizer and label vector Y and prints nmostcommon words in spam and not spam messages Return: nothing
text_results	Y (numpy array) Y_pred (numpy array) X_raw (numpy array)	Given actual Y and predicted Y_pred label vectors and raw messages, this function prints the false positive and false negative messages Return: nothing
driver_casestudy_spam		Driver for spam classification using a neural network

Spam Classification Walkthrough

- Code for walkthrough located at:
IntroML/Code/Version4.1
- Demo of pandas for loading and processing data in
IntroML/Examples/Chapter2/PandasDemo.ipynb
- Demo of CountVectorizer for text processing in
IntroML/Examples/Chapter2/sklearnDemo.ipynb
- You can implement the code additions suggested on the previous page by adding to a clean Version3.3 of the code
 - Have a look at Version4.1 for hints
- We will perform a walkthrough of the code additions for the spam classification

Further Investigations

- Feature Matrix:
 - Investigate other approaches to create Feature Matrix
 - For example, TfidfVectorizer (see scikit-learn documentation for more details)
- Overfitting
 - Go through procedures described in previous chapter to address overfitting in this problem
 - More data – this is not feasible given fixed data set
 - Simpler neural network
 - Regularization
 - Perform hyperparameter tuning to find best performance
 - Sample program using a 1 layer neural network (Logistic Regression) is included as `driver_casestudy_spam_logistic.py`

7.3 Case Study: MNIST Digits Classification

MNIST Digits Classification

Goal of this Section:

- Describe approach for using neural networks for image classification using the MNIST Digits dataset

Machine Learning - Image Classification

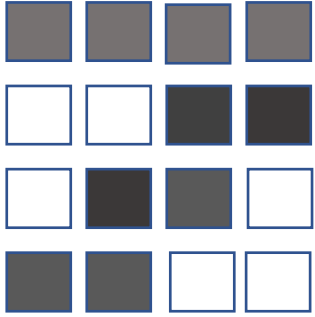
- Image classification (cats and dogs, animals, x-rays, scans, etc) is a principal application of supervised machine learning
- Binary or multi-class
- Training data:
 - Input information = image data
 - Output information = labels
- Goal is to be able to predict label for new images
- Question: how does one convert images into feature matrix to employ neural network approach?

Representation of Images

- Images typically are composed of rectangular arrays of pixels
- For black and white images, intensity of greyscale for each pixel is represented by a number (white = 0 to 255 = black)
- Feature vector for image is vector of intensities for all pixels
- For colour images, each pixel represented by 3 values – intensities of red, blue, and green components for that pixel – feature vector will have 3 x number of pixels as in black and white case

Converting Image to Feature Matrix

Original Image:
Greyscale 4x4 =16 pixels



Intensity Matrix
4x4 (white=0 to 255=black)

190	190	190	190
0	0	220	220
0	220	200	0
200	200	0	0



Feature Vector 16x1

190
190
190
190
0
0
220
220
0
220
200
0
200
200
0
0

Choice of Labels

- For Binary Classification, arbitrarily assign 0, 1 to the classes
 - Example: for classification of cats and dogs (assign 0 for cat and 1 for dog)
 - Example: X-rays assign (0 normal and 1 for broken)
 - Choice is arbitrary (can use 1 for cat and 0 for dog) – doesn't matter
- For Multiclass Classification (c classes) assign $0, 1, \dots, c-1$ to classes
 - For digits classification, 10 classes – obviously assign 0 to 0, 1 to 1, ..., 9 to 9
 - For pictures of cats, dogs, rabbits, ferrets, ducks (5 classes), assign 0 to cats, 1 to dogs, 2 to rabbits, 3 to ferrets, and 4 to ducks.

MNIST Digits Database

- NIST is acronym for National Institute of Standards and Technology, which is a physical sciences laboratory and a non-regulatory agency of the United States Department of Commerce
- MNIST (Modified National Institute of Standards and Technology) digits database is a large collection of black and white handwritten digit images used for training and testing of machine learning algorithms
- Digit images are uniform (28x28 resolution = 784 pixels)
- 60,000 individual digit images (0 – 9 with labels) for training
- 10,000 individual digit images (0 – 9 with labels) for testing
- Data Source: <http://yann.lecun.com/exdb/mnist/>

Sample of Digit Images



- Collage of 160 individual digit images
- Citation for above image

By Josef Steppan - Own work, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=64810040>

MNIST Digits – Format of Data Files

- Each row represents label and intensities for one image
 - First column is the digit label (0,1,...,9)
 - Columns 2 – 785 are the intensities
 - Take transpose to convert feature matrix and value vector to correct format
 - Standard practice is to divide pixel values by 255 so between 0 and 1

Normal

Page Break Preview

Page Layout

Custom Views

Workbook Views

</

Training and Validation Data

- Data located in folder IntroML/Code/Data_MNIST:
 - 60000 training data samples split into 2 files (because of Github limitations)
 - MNIST_train_set1_30K.csv
 - MNIST_train_set2_30K.csv
 - 1 data sample for each row consisting of digit label plus $784=28 \times 28$ pixel values
 - 10000 validation data samples in file:
 - MNIST_valid_10K.csv
 - 1 data sample for each row consisting of digit label plus $784=28 \times 28$ pixel values

MNIST Distribution of Digits

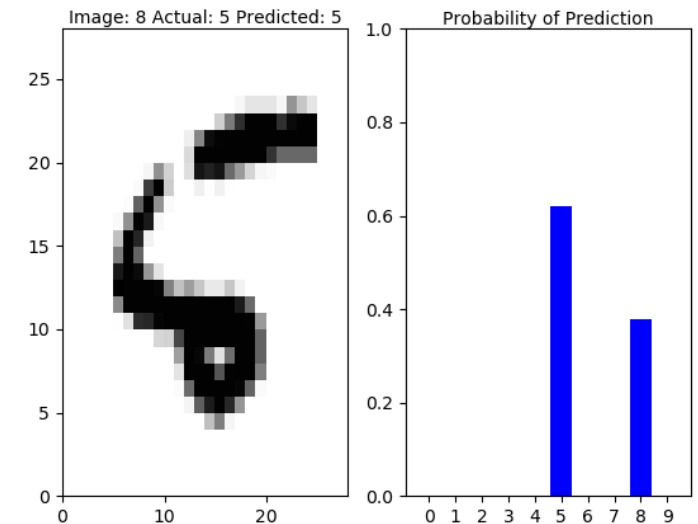
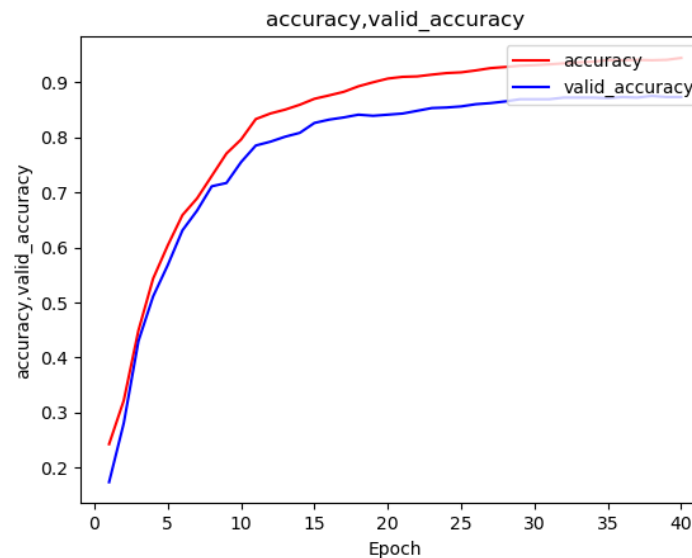
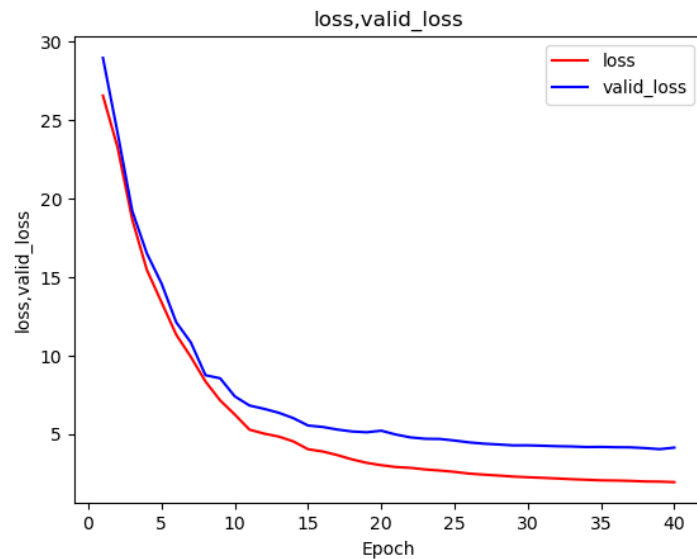
Digit	# Training 60K	# Valid 10K	# Training 6K	# Validation 1K
0	5923	980	592	85
1	6742	1135	671	126
2	5958	1032	581	116
3	6131	1010	608	107
4	5842	982	623	110
5	5421	892	514	87
6	5918	958	608	87
7	6265	1028	651	99
8	5851	974	551	89
9	5949	1009	601	94

MNIST Digit Classification using a Neural Network

- Example: Dataset
 - 6000 images (28x28 resolution) in training dataset
 - 1000 images in validation dataset
 - Feature matrix for training is (784 x 6000)
 - Feature matrix for validation is (784 x 1000)
- Neural Network
 - 2 layer neural network
 - Layer 1: 128 units (relu activation)
 - Layer 2: 10 unit (softmax activation)
 - Cross Entropy Loss function
 - 101,770 combined total entries in $W^{[k]}$ and $b^{[k]}$ for all layers
- Optimization:
 - Adam
 - $\alpha=0.02, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-7}$
 - 40 epochs (batch_size = 6000 batch gradient descent)

Digit Classification – Summary of Results

- After 40 epochs:
 - Training Accuracy: 0.944
 - Validation Accuracy: 0.873
- Loss and Accuracy plots indicate an underfitting (also an overfitting)
 - Expect training accuracy to be higher – should be close to 100%
- Plot of Image and Probability:
 - Probability bar chart obtained from final activation for validation image (index 8)
 - Actual image is 5 and predicted is 5
 - Bar chart shows that probability of prediction of 5 is ~63% and prediction of 8 is ~37%



Digit Classification – Summary of Results

- Confusion Matrix:
 - Actual 2 is predicted as 8 on 11 occasions (roughly 10% of time)
 - Actual 3 is predicted as 5 on 10 occasions (roughly 10% of time)

		Confusion Matrix									
		Actual									
Predicted	0	0	1	2	3	4	5	6	7	8	9
	0	79	0	0	0	0	1	5	0	1	0
	1	0	122	0	0	0	0	0	1	0	0
	2	2	1	95	3	1	2	1	3	2	0
	3	0	2	1	82	0	2	0	0	3	1
	4	0	0	0	1	100	0	1	0	1	2
	5	1	0	3	10	1	75	2	0	4	1
	6	2	1	1	3	3	1	77	0	0	0
	7	0	0	4	3	0	5	0	87	2	3
	8	1	0	11	3	1	1	1	0	76	7
	9	0	0	1	2	4	0	0	8	0	80

New Code for Digits Classification

Method	Input	Description
load_mnist	ntrain (integer) nvalid (integer)	Loads MNIST database Return: Xtrain, Ytrain, Xvalid, Yvalid
driver_casestudy_mnist		Driver for performing mnist training
plot_results_mnist_animation	X (numpy array) Y (numpy array) Y_pred (numpy array) Afinal (numpy array) nframe (integer)	Shows animation of digit images (X) and prints actual label (Y) and predicted label (Y_pred), as well as probabilities for each digit (Afinal) for nframe images Return: nothing

MNIST Digits Classification Walkthrough

- Code for walkthrough located at:
IntroML/Code/Version4.1
- Demo of pandas for loading and processing data in
IntroML/Examples/Chapter2/PandasDemo.ipynb
- You can implement the code additions suggested on the previous page by adding to a clean Version3.3 of the code
 - Have a look at Version4.1 for hints
- We will perform a walkthrough of the code additions for MNIST Digit Classification

Further Investigations

- Results indicate underfitting has occurred
 - Investigate if adding additional layers and/or units to neural network addresses underfitting
 - Experiment with optimization parameters or optimizer and number of epochs