Stats 170A: Project in Data Science

Predictive Modeling: Classification

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine
Reading, Homework, Lectures

• Reference reading:
  Chapters 1 to 4 in Geron’s text, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*
  – See chapter 3 for today’s lecture on classification

• Homework 6
  – Based on Chapter 2 in Geron
  – Due by 2pm Wednesday next week (Monday is a holiday)

• Next Lectures
  – Today: prediction with classification
  – Wednesday next week: text analysis and classification
  – 1 more homework (#7) and then project mode
Jupyter Notebook for Chapter 3 on Classification

Online at https://github.com/ageron/handson-ml/blob/master/03_classification.ipynb
Predictive Modeling

• Two basic types of predictive models
  – Regression: predict a real-valued variable $Y$ given input vector $X$
    • E.g., predict what a customer will spend with a merchant in the next 12 months
  – Classification: predict a categorical variable $Y$ give input vector $X$
    • E.g., predict if a credit card transaction is fraudulent or not

• Both problems can be addressed by statistical models or machine learning algorithms

• Both problems share common mathematical/statistical foundations
Learning a Classification Model

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Zipcode</th>
<th>Age</th>
<th>Test Score</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>18261</td>
<td>92697</td>
<td>55</td>
<td>83</td>
<td>1</td>
</tr>
<tr>
<td>42356</td>
<td>92697</td>
<td>19</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td>00219</td>
<td>90001</td>
<td>35</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>83726</td>
<td>24351</td>
<td>0</td>
<td>35</td>
<td>0</td>
</tr>
</tbody>
</table>

Learning algorithm learns a function that takes values on the left to predict the value (diagnosis) on the right.
Making Predictions with a Classification Model

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Zipcode</th>
<th>Age</th>
<th>Test Score</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>18261</td>
<td>92697</td>
<td>55</td>
<td>83</td>
<td>1</td>
</tr>
<tr>
<td>42356</td>
<td>92697</td>
<td>19</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td>00219</td>
<td>90001</td>
<td>35</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>83726</td>
<td>24351</td>
<td>0</td>
<td>35</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Zipcode</th>
<th>Age</th>
<th>Test Score</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>12837</td>
<td>92697</td>
<td>40</td>
<td>70</td>
<td>??</td>
</tr>
<tr>
<td>72623</td>
<td>92697</td>
<td>32</td>
<td>44</td>
<td>??</td>
</tr>
</tbody>
</table>

We can then use the model to make predictions when target values are unknown.
## Examples of Classification Problems

| Problem                  | Features x                           | Class y               | Used by?                               | Useful to have p(y|x)? |
|--------------------------|--------------------------------------|-----------------------|----------------------------------------|------------------------|
| Spam email               | Presence/absence of words in email   | Spam or not           | Google, Yahoo, Microsoft, etc          | yes                    |
| Speech recognition       | Acoustic/spectral features           | Identity of word     | IBM, Microsoft, Google, etc            | yes                    |
| Loan Approval            | Individuals’ income, job, age, etc   | Will default or not  | Banks, financial companies             | yes                    |
| Cancer screening         | Image features at cell level         | Cancerous or not?    | Medical companies                      | yes                    |
| Personalized genomics    | Gene expression data                 | Cancer or not        | Bioinformatics startups                 | yes                    |
Examples of Classifiers

– Nearest-neighbor: simple baseline

– Logistic regression: simple but useful, widely used in business

– Neural network: non-linear extension of logistic regression

– Support vector machines: complex, not used much in business

– Decision trees: useful general purpose method

– Many more.....
Each dot is a 2-dimensional point representing one person = [AGE, MONTHLY INCOME]
A much more complex boundary – but perhaps overfitting to noise?
Classification with the Logistic Model
Getting Class Probabilities....

Estimates of class probabilities \( P(c \mid x) \) are very useful in practice e.g., for ranking documents to show to a human user.
Getting Class Probabilities....

Estimates of class probabilities $P(c \mid x)$ are very useful in practice, e.g., for ranking documents to show to a human user.

Assume for simplicity we have a 2-class binary classification problem.

Say we tried to get a probability of a class with a linear model:

$$P(c \mid x) = f(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_T x_T$$

There is a problem:

$f(x)$ could be negative, could be $>1$, etc.
A Better Approach

\[ P(c \mid x) = f(x) = g(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_T x_T) \]

where \( g(z) = \frac{1}{1 + e^{-z}} \)

As \( z \to \) positive infinity, \( g(z) \to 1, \) \( P(\text{class}) \to 1 \)
As \( z \to \) negative infinity, \( g(z) \to 0, \) \( P(\text{class}) \to 0 \)

This is the logistic regression model

In effect: a linear (weighted sum) model where the sum is transformed to lie between 0 and 1

...and we can interpret \( f(x) \) directly as a probability between 0 and 1
What does the Logistic Function look like?

The Logistic Function can be defined as:

\[ f(x) = g(z) = \frac{1}{1 + e^{-z}} \]

The shape of the Logistic Function is shown in the graph. The function approaches 1 as \( z \rightarrow \) positive infinity, and approaches 0 as \( z \rightarrow \) negative infinity.

The function can be expressed as a weighted sum:

\[ z = \sum_j w_j x_j \]

As \( z \rightarrow \) positive infinity, \( g(z) \rightarrow 1 \), and the probability of the class \( P(\text{class}) \rightarrow 1 \).

As \( z \rightarrow \) negative infinity, \( g(z) \rightarrow 0 \), and the probability of the class \( P(\text{class}) \rightarrow 0 \).
Comparing Linear and Logistic Models

Notation for Classification Learning

Training data:  \( = \{ (x_i, c_i) \} \),  \( i = 1, \ldots, N \)

- The \( i \)th feature vector
- The class label for the \( i \)th example.
- \( c_i \) takes value 0 or 1 for binary classification
Notation for Classification Learning

Training data:  \[ \{ (x_i, c_i) \}, \quad i = 1, \ldots, N \]

The ith feature vector

The class label for the ith example.  
\( c_i \) takes value 0 or 1 for binary classification

Prediction for \( x_i \)  \[ = f(x_i | w), \quad i = 1, \ldots, N \]

The classification model’s prediction for the ith feature vector

Some fixed value of the weight vector \( w \)
Learning: Minimizing a Loss Function

Loss function: \( L(w) = \sum_{i=1}^{N} \text{distance}\left(c_i, f(x_i|w)\right) \)
Learning: Minimizing a Loss Function

Loss function:  \( L(w) = \sum_{i=1}^{N} \text{distance}(c_i, f(x_i|w)) \)

Note that we view the loss function \( L(w) \) as a function of the unknown weights \( w \).

Goal of learning is to minimize this sum over all \( N \) training examples, i.e., learn the weights \( w \) that make the \( f \)'s as close to the \( c \)'s as possible.

We typically use optimization or search algorithms to find the \( w \) vector that minimizes \( L(w) \).
The Squared Error Loss Function

\[ L(w) = \sum_{i=1}^{N} \left( c_i - f(x_i | w) \right)^2 \]

Best suited for problems when \( f \) can lie anywhere on the real line
The Log Loss Function

Used when f represents a probability $P(c \mid x)$ and is between 0 and 1
(Often used to train logistic regression and neural network classifiers)

$$L(w) = \sum_{i:c_i=1} \log \frac{1}{f(x_i \mid w)} + \sum_{i:c_i=0} \log \frac{1}{1 - f(x_i \mid w)}$$
The Log Loss Function

Used when $f$ represents a probability $P(c \mid x)$ and is between 0 and 1
(Often used to train logistic regression and neural network classifiers)

\[
L(w) = \sum_{i:c_i=1} \log \frac{1}{f(x_i \mid w)} + \sum_{i:c_i=0} \log \frac{1}{1 - f(x_i \mid w)}
\]

- Drive $f \to 1$ on these examples, and $\log 1/f \to 0$
- Drive $f \to 0$ on these examples, and $1/(1 - f) \to 0$
Weight Regularization

With so many weights it is easy to overfit -> regularization is often useful

\[ L'(w) = \left( \sum_{i:c_i=1} \log \frac{1}{f(x_i|w)} + \sum_{i:c_i=0} \log \frac{1}{1 - f(x_i|w)} \right) + \lambda \sum_{j=0}^{T} w_j^2 \]

Additional term to penalize large weights.... encourages weights to stay at 0 unless data suggests otherwise

\[ \lambda \] typically determined via cross-validation

(Can also use absolute value of w ‘s , called Lasso regularization)
Gradient Descent Algorithm

Simple algorithm that uses the gradient to search for the minimum of $L(w)$

- Start at some random $w$ location
- Compute the gradient $\Delta(w)$
- Move in the direction $-\Delta(w)$
  (typically take small steps)
- Recompute the gradient, iterate....
- Repeat until there is no improvement

Theoretical properties?
If step sizes are small enough,
guaranteed to find a (local) minimum

Simple example of gradient descent for $p = 2$ dimensions
Optimizing $L(w)$ with a single weight $w$

Easy to find the minimum (convex problem with a single global minimum)
Optimizing $L(w)$ with a single weight $w$

Easy to find the minimum (convex problem with a single global minimum)
Example: logistic model

Harder to find the minimum (non-convex problem with local minima)
Example: neural network
Training a Logistic Model with Gradient Descent

• The logistic model combined with the log-loss function is a convex problem
  – So it can be solved relatively easily with gradient descent

• Parameters of the optimization algorithm:
  – Step size or “learning rate”: how far to step in weight space at each iteration
    • Theory recommends that the step size should decrease as iterations increase
    • Note that if the learning rate is too large, the optimization may diverge
  – Convergence criterion
    • Change in log-loss function or magnitude of gradient vector is less than some epsilon
Stochastic Gradient Descent
(Example in 2-dimensional Parameter Space)

Empirically works very well on large data sets: some theoretical support
(See Adam algorithm, by Kingma and Ba, ICLR, 2015)
Example of Stochastic Gradient Optimization

Binary Text Classification Problem
\( d = 47k \) word features
\( N = 781k \) documents
Various forms of linear classifiers

From Leon Bottou, Stochastic Gradient Learning, MLSS Summer School 2011, Purdue University,
http://learning.stat.purdue.edu/mlss/_media/mlss/bottou.pdf
Summary: Training a Logistic Regression Model

• The model:

\[
P(c \mid x) = f(x) = \frac{1}{1 + e^{-(\sum_{j=0}^{T} w_j x_j)}}
\]

• The loss function:

\[
L(w) = \sum_{i:c_i=1} \log \frac{1}{f(x_i \mid w)} + \sum_{i:c_i=0} \log \frac{1}{1 - f(x_i \mid w)}
\]

• Training algorithms:
  – Gradient descent: fast and reliable, need to choose learning rate
  – Stochastic gradient descent:
    • Often used in practice for problems with many documents and many features
Logistic Classifiers: the “Workhorse” of Machine Learning

The logistic model is very widely used in machine learning

- Document classification such as spam email
- Predicting what advertisements to show Web users
- Automatic ranking of search engine results
- Credit scoring
- Fraud detection
- And many more....
- Google, Yahoo!, Microsoft, eBay, Amazon, Yelp, Linkedin, Facebook, Experian, etc

Why so popular?

- Often does very well on high-dimensional problems
- Time and space complexity is linear in d and N (with gradient methods)
- Convex optimization (no local minima problems)
- Relatively interpretable
- Probabilistic outputs useful for many tasks, e.g., ranking
More Complex Classification Models:
Neural Networks and Decision Trees
Example: A Logistic Model as a Simple Neural Network

The machine learning algorithm will learn a weight for each arrow in the diagram.

This a simple model: one weight per input
A Simple Neural Network

Here the model learns 3 different functions and then combines the outputs of the 3 to make a prediction.

This is more complex and has more parameters than the simple model.
A Simple Neural Network

Here the model learns 3 different functions and then combines the outputs of the 3 to make a prediction.

This is more complex and has more parameters than the simple model.
A Simple Neural Network

Here the model learns 3 different functions and then combines the outputs of the 3 to make a prediction.

This is more complex and has more parameters than the simple model.
Deep Learning: Models with More Hidden Layers

We can build on this idea to create “deep models” with many hidden layers.

Very flexible and complex functions
Figure from Lee et al., ICML 2009
Example of a Network for Image Recognition

Mathematically this is just a function (a complicated one)

Figure from http://parse.ele.tue.nl/
Training Neural Networks

• Learning algorithm: stochastic gradient descent on the loss function

• Many “hyperparameters” to select
  – Number of layers
  – Size of each hidden layer
  – Type of non-linearity
  – ...and more

• Strengths
  – More powerful than simpler models like logistic regression
  – Works very well on “low-level” features like pixels

• Weaknesses
  – May need very large labeled data sets (millions)
  – Search over hyperparameters may be expensive (human and computer time)
  – Models are not easy to interpret
Example: Decision Trees and CellPhone “Churn” Prediction

Historical data set of 20,000 customers.
The target variable is whether the customer stayed with the company or left (“churned”)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLEGE</td>
<td>Is the customer college educated?</td>
</tr>
<tr>
<td>INCOME</td>
<td>Annual income</td>
</tr>
<tr>
<td>OVERAGE</td>
<td>Average overcharges per month</td>
</tr>
<tr>
<td>LEFTOVER</td>
<td>Average number of leftover minutes per month</td>
</tr>
<tr>
<td>HOUSE</td>
<td>Estimated value of dwelling (from census tract)</td>
</tr>
<tr>
<td>HANDSET_PRICE</td>
<td>Cost of phone</td>
</tr>
<tr>
<td>LONG_CALLS_PER_MONTH</td>
<td>Average number of long calls (15 mins or over)</td>
</tr>
<tr>
<td>AVERAGE_CALL_DURATION</td>
<td>Average duration of a call</td>
</tr>
<tr>
<td>REPORTED_SATISFACTION</td>
<td>Reported level of satisfaction</td>
</tr>
<tr>
<td>REPORTED_USAGE_LEVEL</td>
<td>Self-reported usage level</td>
</tr>
<tr>
<td>LEAVE (Target variable)</td>
<td>Did the customer stay or leave (churn)?</td>
</tr>
</tbody>
</table>

Example and figures from Provost and Fawcett, *Data Science for Business*, 2013
Example: Cellular Phone Churn-Prediction Problem

Classification tree learned from the training data set of 20k examples

73% accuracy on the training set

Example and figures from Provost and Fawcett, *Data Science for Business*, 2013
Example: Cellular Phone Churn-Prediction Problem

Classification tree learned from the training data set of 20k examples

73% accuracy on the training set

Questions:

1. How likely are we to get 73% accuracy on another (new) set of customers?

2. Is 73% as good as we can do?

Example and figures from Provost and Fawcett, *Data Science for Business*, 2013
Data Probability Contours/Decision Boundaries of Different Classifiers

Nearest Neighbors

Linear SVM

Decision Tree

Random Forest

Neural Net

kNN

Linear

Tree

Random Forest

Neural Network

Figure from scikit-learn tutorial material
scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
- Identifying to which set of categories a new observation belong to.
- **Applications**: Spam detection, image recognition.
- **Algorithms**: SVM, nearest neighbors, random forest. ...

Regression
- Predicting a continuous value for a new example.
- **Applications**: Drug response, Stock prices.
- **Algorithms**: SVR, ridge regression, Lasso, ...

Clustering
- Automatic grouping of similar objects into sets.
- **Applications**: Customer segmentation, Grouping experiment outcomes.
- **Algorithms**: k-Means, spectral clustering, mean-shift, ...

Dimensionality reduction
- Reducing the number of random variables to consider.
- **Applications**: Visualization, Increased efficiency.
- **Algorithms**: PCA, Isomap, non-negative matrix factorization.

Model selection
- Comparing, validating and choosing parameters and models.
- **Goal**: Improved accuracy via parameter tuning
- **Modules**: grid search, cross validation, metrics.

Preprocessing
- Feature extraction and normalization.
- **Application**: Transforming input data such as text for use with machine learning algorithms.
- **Modules**: preprocessing, feature extraction.

News
- On-going development: What's new (changelog)

Community
- Questions? See stackoverflow # scikit-learn
- Mailing list: scikit-learn-

Who uses scikit-learn?
Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through total modularity, minimalism, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Supports arbitrary connectivity schemes (including multi-input and multi-output training).
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: Python 2.7-3.5.

Guiding principles

- Modularity. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can
Picking the Best Model from a Set of Models
Complexity versus Goodness of Fit

Training data

\[ y \]

\[ x \]
Complexity versus Goodness of Fit

Here the red circles represent training data and the blue curves are models fitted to the data.

Too simple?
Complexity versus Goodness of Fit

Here the red circles represent training data and the blue curves are models fitted to the data.
Complexity versus Goodness of Fit

Here the red circles represent training data and the blue curves are models fitted to the data.
Different Polynomial Models

Example from Geron Text, Chapter 4
Complexity and Generalization

Loss Function
- e.g., mean squared error

Optimal model complexity

Model Complexity (e.g., number of parameters)

$E_{train}(\theta)$

$E_{test}(\theta)$
Complexity and Generalization

Loss Function
e.g., mean squared error

Model Complexity

Amount of overfitting

$E_{\text{train}}(\theta)$

$E_{\text{test}}(\theta)$

Underfitting

Overfitting

UCIrvine

Complexity and Generalization

Loss Function
e.g., mean squared error

$E_{\text{train}}(\theta)$

$E_{\text{test}}(\theta)$

High bias
Low variance

Optimal model complexity

Low bias
High variance
Training and Test Data

• **Classification accuracy on our training data will tend to be optimistic**
  – Classifier can “memorize” the training data

• **Test set performance**
  – A more accurate estimate of accuracy can be gotten by reserving some of our data as an independent/unseen/holdout test data set
  – Train the model on the training data
  – Evaluate the model’s true accuracy on the data it did not see (the test set)

• **Cross-validation**
  – We can repeat the process of splitting our data into train-test sets multiple times to get an even more robust estimate of accuracy
  – **V-fold cross-validation**
    • V train-test splits of the data, train V models and evaluate on V test sets
    • Our final accuracy estimate is the average over the V test “folds”
Example of 5-fold Cross-Validation

Repeatedly partition data randomly into disjoint training and validation sets

More robust than using a single partition of the data

Evaluate the performance of each model by taking the average performance across the validation sets
Example: Cellular Phone Churn-Prediction Problem

Classification tree learned from the training data set of 20k examples
73% accuracy on the training set

Question:
How likely are we to get 73% accuracy on another (new) set of customers?

Example and figures from Provost and Fawcett, *Data Science for Business*, 2013
Example: Cellular Phone Churn-Prediction Problem

10-fold cross-validation accuracies for logistic regression (top) and decision trees (bottom)

Solid lines indicate the mean accuracy for each model

Note that the mean accuracy for trees is lower than the 73% on the training data

Figure from Provost and Fawcett, *Data Science for Business*, 2013
Example: Cellular Phone Churn-Prediction Problem

Learning curves: accuracy as a function of number of training examples

The tree is a more flexible model than logistic regression, so for this data set at least it can learn a more accurate model than logistic regression given more data

Example and figures from Provost and Fawcett, *Data Science for Business*, 2013
“All models are wrong, but some are useful”

Professor G. E. P. Box, 1987
Practical Advice

- Evaluate a model using test or validation data that was unseen to the model when it was trained
  - This is the only true way to assess model performance
  - Otherwise complex models always “win” on the training data

- More complex models require more training data to be effective
  - Logistic models: 100’s or 1000’s of training examples
  - Neural networks: tens of 1000’s or even millions of training examples
  - Trees are often useful when dealing with real and categorical variables
  - So different models are useful in different contexts

- It’s always a good idea to start with simpler models (e.g., logistic) and then compare with more complex “challenger” models
Success Indicators for Predictive Modeling

• Do you have enough labeled data to train a model?
  – How expensive will it be to obtain labeled data?

• What are the costs of making errors?
  – Predictive models are usually trained to maximize accuracy
  – But real-world environments may have significant asymmetric costs

• How fast does your environment change?
  – Predictive modeling presupposes that future data is like past data
  – How will you detect if the environment for the model has changed?
  – How will the models get updated/retrained?

• What are the hidden costs in operationalizing a predictive model?
Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley, gholt, dgg, edavydov, toddphillips}@google.com
Google, Inc.

Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Sculley et al, NIPS 2015 Conference
Classification Examples from the Real World
EXAMPLE: RECOMMENDING FRIENDS ON FACEBOOK

L. Backstrom, invited keynote talk at ESWC 2011 Conference
Online video and slides at http://videolectures.net/eswc2011_backstrom_facebook/

L. Backstrom and J. Leskovec
Supervised Random Walks: Predicting and Recommending Links in Social Networks
*ACM Conference on Web Search and Data Mining (WSDM)*, 2011
Learning to Suggest Friends on Facebook

Problem: automatically suggest friends

Restrict to “friends of friends”
  – Still leaves 40,000 possibilities on average
Learning to Suggest Friends

Solution: learn a prediction model

Target: user clicks or not on the recommendation

Input Features: mutual friends, age, geography, etc

Models: combination of logistic regression and decision trees

Significant engineering:
feature computation in real-time, plus real-time feedback
Learning to Suggest Friends

Significant improvement in click-through rate (y-axis) when system went live
Each question mark represents an “ad slot”.

In a fraction of a second, algorithms predict which ads you are most likely to click on (from 1000’s of ads).
The ads that are most likely to lead to a click are selected using classification models such as logistic regression and then displayed to you.
Applications in Financial Forecasting

Deep Learning for Mortgage Risk, Sirignano, Sadwhani, Giesecke, ArXiv 2016

Problem:
Predicting whether a mortgage will remain current, be paid off, or default

Data:
120 million mortgages between 1995 and 2014 (70% of all US mortgages)
3.5 billion borrower-month observations

Prediction Models:
Deep neural networks (up to 5 hidden layers)
Logistic regression
Applications in Astronomy


**Problem:**
Predicting whether an astronomical image is a star or galaxy

**Data:**
Approximately 50,000 images for training, 15,000 for test
Two different telescope data sets
Manually labeled

**Prediction Models:**
Convolutional neural networks on raw pixel data, 11 layers of weights
Existing random forest classifier for this problem using predefined features

**Data Set 1: CFHTLenS**

<table>
<thead>
<tr>
<th>classifier</th>
<th>AUC</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet</td>
<td>0.9948</td>
<td>0.0112</td>
</tr>
<tr>
<td>TPC\textsubscript{morph}</td>
<td>0.9924</td>
<td>0.0109</td>
</tr>
<tr>
<td>TPC\textsubscript{phot}</td>
<td>0.9876</td>
<td>0.0189</td>
</tr>
</tbody>
</table>

**Data Set 2: SDSS**

<table>
<thead>
<tr>
<th>classifier</th>
<th>AUC</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet</td>
<td>0.9952</td>
<td>0.0182</td>
</tr>
<tr>
<td>TPC\textsubscript{morph}</td>
<td>0.9967</td>
<td>0.0099</td>
</tr>
<tr>
<td>TPC\textsubscript{phot}</td>
<td>0.9886</td>
<td>0.0283</td>
</tr>
</tbody>
</table>
A Deep Neural Network for Image Recognition

From Nguyen, Yosinski, Clune, CVPR 2015
A Deep Neural Network for Image Recognition

From Nguyen, Yosinski, Clune, CVPR 2015

Images used for Training

New Images
A Deep Neural Network for Image Recognition

From Nguyen, Yosinski, Clune, CVPR 2015
A Deep Neural Network for Image Recognition

From Nguyen, Yosinski, Clune, CVPR 2015
Reading, Homework, Lectures

• Reference reading:
  Chapters 1 to 4 in Geron’s text, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*
  – See chapter 3 for today’s lecture on classification

• Homework 6
  – Based on Chapter 2 in Geron
  – Due by 2pm Wednesday next week (Monday is a holiday)

• Next Lectures
  – Today: prediction with classification
  – Wednesday next week: text analysis and classification
  – 1 more homework (#7) and then project mode
BACKUP SLIDES:
ADDITIONAL EVALUATION METHODS
Evaluation Methods in General

- When we build a predictive model how can we evaluate the model?

- As mentioned before, the standard approach is to keep a holdout or test data set (that we haven’t used in any aspect of model training or model selection) and evaluate our error or objective function on that, e.g.,
  - Squared error or classification error

- But this is often only part of the story....its important to look at other aspects of performance, particularly if we want to further improve our model

- In the next few slides we will discuss different aspects of evaluating or “diagnosing” model performance
  - These techniques can be very useful in practical applications
## Classification: Confusion Matrices

Count the pairs of (predicted, actual) class labels in test data set. Patterns in the off-diagonal cells can be informative.

<table>
<thead>
<tr>
<th>Model’s Predictions</th>
<th>True Class Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
</tr>
<tr>
<td>Class 1</td>
<td>400</td>
</tr>
<tr>
<td>Class 2</td>
<td>2</td>
</tr>
<tr>
<td>Class 3</td>
<td>5</td>
</tr>
</tbody>
</table>
Binary Classification: Ranking Metrics

- In many applications we have a set of test items and want to rank them.

- Often rank by $P(C = 1 | x)$, where $C = 1$ indicates the class of interest. And where $P(C = 1 | x)$ is produced by our prediction model, e.g., by logistic regression.

- **Examples:**
  - Ranking loan applicants by likelihood of repaying the loan.
  - Ranking Web users by likelihood of clicking on an ad.
  - Ranking patients by likelihood that they need surgery.
  - And so on.

- In practice we could select top K-ranked items, e.g., to award a loan to. There are algorithms that “learn to rank” directly.
Binary Classification: Ranking Metrics

• To evaluate using a ranking metric we do the following
  – take a test set with N data vectors x
  – compute a score for each item, say $P(C = 1 \mid x)$, using our prediction model
  – sort the N items from largest to smallest score

• This gives us 2 lists, each of length N
  – A list of predictions, with decreasing score values
  – A corresponding list of “ground truth” values, 0’s and 1’s for binary class labels

• A variety of evaluation metrics can be computed based on these 2 lists
  – Precision/recall
  – Receiver-operating characteristics
  – Lift curves
  – And so on.
### Ranking Terminology

<table>
<thead>
<tr>
<th>True Labels</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>TP</td>
<td>True positive</td>
</tr>
<tr>
<td>Negative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>FP</td>
<td>False positive</td>
</tr>
<tr>
<td>Negative</td>
<td>FN</td>
<td>False negative</td>
</tr>
<tr>
<td>Negative</td>
<td>TN</td>
<td>True negative</td>
</tr>
</tbody>
</table>

**Model’s Predictions**

<table>
<thead>
<tr>
<th>Positive</th>
<th></th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td></td>
<td>FN</td>
</tr>
<tr>
<td>Negative</td>
<td></td>
<td>TN</td>
</tr>
</tbody>
</table>

**Precision**

\[
\text{Precision} = \frac{TP}{(TP + FP)} = \text{ratio of correct positives predicted to total positive predicted}
\]

**Recall**

\[
\text{Recall} = \frac{TP}{(TP + FN)} = \text{ratio of correct positives predicted to actual number of positives}
\]

Typically will get high precision for low recall, and low precision at high recall
Simple Example of Precision and Recall

- Test set with 10 items, binary labels, 5 from each class \((TP + FN = 5)\)

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>
Simple Example of Precision and Recall

• Test set with 10 items, binary labels, 5 from each class (TP + FN = 5)

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

Threshold

TP = 2, FP = 0
Precision = TP/(TP+FP) = 100%
Recall = TP/(TP + FN) = 2/5 = 40%
Simple Example of Precision and Recall

- Test set with 10 items, binary labels, 5 from each class (TP + FN = 5)

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

Threshold

TP = 3, FP = 2
Precision = TP/(TP+FP) = 60%
Recall = TP/(TP + FN) = 3/5 = 60%
Simple Example of Precision and Recall

- Test set with 10 items, binary labels, 5 from each class (TP + FN = 5)

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

TP = 5, FP = 8  
Precision = TP/(TP + FP) = 62%  
Recall = TP/(TP + FN) = 5/5 = 100%
How are Precision and Recall used?

• Precision @ K is often used as a metric, where K is the top K items or the top K% of the sorted prediction list
  – E.g., useful for evaluating search engine results

• Precision-recall curves can be plotted for varying thresholds
  – Can be useful in classifying text documents
## ROC Plots

<table>
<thead>
<tr>
<th>Model’s Predictions</th>
<th>True Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: True positive</td>
</tr>
<tr>
<td>Negative</td>
<td>FN: False negative</td>
</tr>
</tbody>
</table>

### True Positive Rate (TPR)

\[
TPR = \frac{TP}{TP + FN} = \frac{\text{True positives predicted}}{\text{Actual number of positives}}
\]

(same as recall, sensitivity, hit rate)

### False Positive Rate (FPR)

\[
FPR = \frac{FP}{FP + TN} = \frac{\text{Incorrect negatives predicted}}{\text{Actual number of negatives}}
\]

(same as false alarm rate)

### Receiver Operating Characteristic

Plots TPR versus FPR as threshold varies.

As we decrease our threshold, both the TPR and FPR will increase, both ending at [1, 1]
ROC for Binary Classification

- Test set with 10 items, binary labels, 5 from each class (TP + FN = 5)

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

Threshold

TPR = TP/(TP+FN) = 2/5 = 0.4
FPR = FP /(FP+TN) = 0/5 = 0.0
ROC for Binary Classification

- Test set with 10 items, binary labels, 5 from each class (TP + FN = 5)

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

Threshold: TPR = TP/(TP+FN) = 2/5 = 0.4
FPR = FP/(FP+TN) = 0/5 = 0.0

Threshold: TPR = TP/(TP+FN) = 5/5 = 1.0
FPR = FP/(FP+TN) = 3/5 = 0.6
Each point corresponds to a particular operating threshold (N+1 points).
ROC Plot for Data from Previous Slide

Diagonal line is the theoretical performance of a random classifier (random ordering)

Each point corresponds to a particular operating threshold (N+1 points)

True Positive Rate ("Hit Rate")

False Positive Rate ("False Alarm Rate")
The Area under the Curve (AUC) provides a measure of performance as a single number. Here AUC = 0.76.