HPSI
Human Performance Systems, Inc.

Machine Learning: Classification Course

Day 1

Course Overview & Fundamentals for Machine Learning in Python

HPSI
Human Performance Systems, Inc.

Plan for the Week

- Day 1: Intro & Review
  - Introduction & Course Overview
  - Review of Fundamentals for Machine Learning in Python
- Day 2: Intro to Classification Algorithms
  - Probability Theory
  - Common & Probabilistic Classifiers
- Day 3: Tree-Based Methods for Classification
  - Decision Tree
  - Random Forest
- Day 4: Neural Networks
  - Perceptron (Single & Multilayer) & ADALINE Networks
  - Optimization, Backpropagation, & Learning Rules
- Day 5: Deep Learning (Neural Networks Cont.)
  - Brief Background
  - Convolution Networks
  - Tensorflow & Keras

Learning Objectives

- By the end of the course, students should be able to:
  1. Understand the mathematical and statistical foundations of common classification algorithms
  2. Identify the most appropriate algorithm based on data characteristics and analysis goals
  3. Implement classification algorithms in Python
  4. Evaluate model performance based on classification metrics

HPSI Human Performance Systems, Inc.

Course Structure & Delivery

- Class is from 9:00 AM – 3:30 PM (6.5 hours)
  - This will be broken up into 3 sessions
- Session 1 (2.25 hours)
  - 9:00 AM – 11:15 AM
- Lunch (1 hour)
  - 11:15 AM – 12:15 PM
- Session 2 (2.25 hours)
  - 12:15 PM – 2:30 PM
- Break (10 minutes)
  - 2:30 PM – 2:40 PM
- Session 3 (50 minutes)
  - 2:40 PM – 3:30 PM

HPSI Human Performance Systems, Inc.

Course Structure & Delivery

- Session 1 (2.25 hours)
  - Introduce Topic → ~ 1 hour
  - Guided Example & Walkthrough → ~ 30 minutes
  - In-class Exercise → ~ 45 minutes
- Session 2 (2.25 hours)
  - Same format
- Session 3 (50 minutes)
  - Introduce Light Topic/Next Day
  - Learning Evaluation
  - Questions/Recap of the Day

HPSI Human Performance Systems, Inc.

Plan for the Week

- Day 1: Intro & Review
  - Introduction & Course Overview
  - Review of Fundamentals for Machine Learning in Python
- Day 2: Intro to Classification Algorithms
  - Probability Theory
  - Common & Probabilistic Classifiers
- Day 3: Tree-Based Methods for Classification
  - Decision Tree
  - Random Forest
- Day 4: Neural Networks
  - Perceptron (Single & Multilayer) & ADALINE Networks
  - Optimization, Backpropagation, & Learning Rules
- Day 5: Deep Learning (Neural Networks Cont.)
  - Brief Background
  - Convolution Networks
  - Tensorflow & Keras

Plan for the Week

- Day 1: Intro & Review
  - Introduction & Course Overview
  - Review of Fundamentals for Machine Learning in Python
- Day 2: Intro to Classification Algorithms
  - Probability Theory
  - Common & Probabilistic Classifiers
- Day 3: Tree-Based Methods for Classification
  - Decision Tree
  - Random Forest
- Day 4: Neural Networks
  - Perceptron (Single & Multilayer) & ADALINE Networks
  - Optimization, Backpropagation, & Learning Rules
- Day 5: Deep Learning (Neural Networks Cont.)
  - Brief Background
  - Convolution Networks
  - Tensorflow & Keras

What is Machine Learning?

- "A field of Computer Science that gives computers the ability to learn without being explicitly programmed."
  - Arthur Samuel (Coined the term in 1959 at IBM)
- "The ability [for systems] to acquire their own knowledge, by extracting patterns from raw data."
  - *Deep Learning*, Goodfellow et al
- "A computer program is said to learn from experience E with respect to some set of tasks T and performance measure P if its performance tasks in T, as measured by P, improves with experience E."
  - Tom Mitchell (Computer Scientist & Professor at Carnegie Mellon)

Types of Machine Learning

- There are *three* main types of Machine Learning:
  1. Supervised
  2. Unsupervised
  3. Reinforcement

Types of Machine Learning

- There are *three* main types of Machine Learning:
  1. Supervised
  2. Unsupervised
  3. Reinforcement

Supervised Learning

- For each input, there is a target value (label) associated with it
  - Know the right answer ahead of time
- Classification and Regression are subsets of supervised learning
  - This course will focus specifically on *classification*
- Common Examples
  - Classifying Fraudulent Credit Card Transactions
  - Identifying tumors within X-ray images
  - Predicting the price of a house given its characteristics

Types of Machine Learning

- There are *three* main types of Machine Learning:
  1. Supervised
  2. Unsupervised
  3. Reinforcement

HPSI Human Performance Systems, Inc.

Unsupervised Learning

- For each input, there is *no* target value (label) associated with it
  - Do not know the right answer
  - No error to evaluate in our model
- Examples of unsupervised learning
  - Clustering (Customer Segmentation)
  - Dimensionality Reduction (Principal Components Analysis)
  - Anomaly Detection (Outlier Analysis)

HPSI Human Performance Systems, Inc.

Types of Machine Learning

- There are *three* main types of Machine Learning:
  1. Supervised
  2. Unsupervised
  3. Reinforcement

HPSI Human Performance Systems, Inc.

Reinforcement Learning

- For each input, there is *no* target value (label) associated with it
- Area of ML inspired by behavioral psychology
  - Concerned with how agents should act in their environment so as to maximize some cumulative reward.
- Examples of Reinforcement Learning
  - Self-driving car
  - Chess bot

Programming Languages

- Need framework to implement Machine Learning algorithms
- There are *hundreds* of programming languages
  - https://en.wikipedia.org/wiki/List_of_programming_languages
- This course will use Python

Why Python?

- There are several benefits to using python for Machine Learning
  - **Interpreted** Language
    - No need to compile & easier to implement
  - **Dynamic** Typing
    - No need to declare variable types and then assign them
  - **Strong** Typing
    - Once a value is initialized and saved, it is never coerced. Type casting must be done explicitly
  - **High Level**
    - More human readable (white space matters)
  - Machine Learning & Deep Learning **API**s
    - Scikit-Learn, Tensorflow, Keras, CNTK, Apache MXNET, PyTorch, etc.

Why Do We Need Math?

- There's lots of ML code out there that you can just download and run
  - To get useful insights & meaningful results, you need to understand the mathematical underpinnings
  - Knowing the algorithms helps you know when to choose each one for a given problem
  - Recognize over/underfitting
  - Troubleshooting poor/ambiguous results

Software & Tech Stack

The software and frameworks we will be using are as follows:

- Language
  - Python 3.5 (Anaconda Distribution)
- ML/DL Frameworks
  - scikit-learn & Keras
- IDE
  - Spyder
- Course Content Delivery
  - Github
- Comunication
  - Slack

Setting up Slack

- Invite Link:
  - https://join.slack.com/t/hpsi-classification/shared_invite/enQtMjk5MjQzMTMxNzM1LTA2NDU2NTcyZGVjZDlkZThhNjZmN2MwYThmNWI2ZTZlYTI2OTQyOTkyN2M4MmE4OWZkMzFmNzY5MWJmYjZjNTc

Getting Access to the Course Content on GitHub

- If you do not have a GitHub account, please go to the following link and make one:
  - https://github.com/join?source=header-home
- Send your GitHub username to us through the **#github-usernames** channel on Slack
  - You should receive an email inviting you to collaborate to the *HPSI-Classification* repository
  - Clone the repository
  - Move the repository from your `Downloads` folder to your `Desktop` or `Documents` folder

Tech Stack Setup & Testing of Working Environment

- Follow the instructions in the `README.md` file to:
  - Setup the conda environment
    - To learn more about managing conda environments go to: https://conda.io/docs/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file (Optional)
  - Activate the conda environment
  - Run the `import_test.py` script to ensure your environment has been configured properly

Python Concepts to Know

- Base Python (Assumed)
  - Data Types
    - Booleans, numbers (int/float), strings, lists, tuples, sets, dictionaries
  - Conditionals
    - if/elif/else
  - Loops
    - for/while
  - Functions
- Python Cheat Sheets

Math Review

- Next we are going to review some math we will need for the algorithms and topics discussed throughout the rest of the course
- Machine Learning is primarily based on
  - Linear Algebra
  - Calculus & Optimization
  - Probability Theory

Linear Algebra

- The study of Linear Algebra involves the following types of objects:
  - Scalars
  - Vectors
  - Matrices
  - Tensors

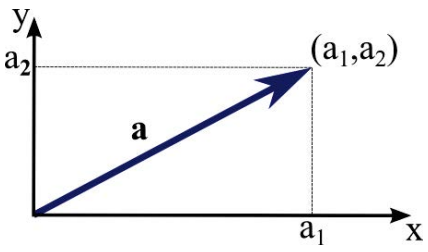
Linear Algebra

- A **scalar** is just a single number
  - Quantity that only has *magnitude*
  - Usually denoted by a lowercase, italicized letter such as *c*

```
# scalar addition
>>> a = 2
>>> b = 4
>>> a + b
6
```

Linear Algebra

- A **vector** is an 1-D array of numbers
  - Has both *magnitude* and *direction*
  - Usually denoted by a lowercase, bold letter such as **x**
  - Written as a *column* enclosed in square brackets
    - $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
    - The dimension of the vector is given by *n*
    - Each element of the vector is identified by an index
      - For example, x_2 represents the second element
  - We can think of vectors as identifying points in space
    - Each element gives the coordinate of a different axis



Linear Algebra

```
# vector is 1-D array
>>> import numpy as np
>>> a = np.arange(4)
>>> a
array([0, 1, 2, 3])

# check number of dimensions
>>> a.ndim
1

# check the number of rows and columns
>>> a.shape
(4,)
```


Linear Algebra

- A **matrix** is a 2-D array of numbers
  - Usually denoted by an uppercase, bold letter such as ***A***
  - Often thought of as an array of column vectors
  - Identified by *two* indices
    - Has *m* rows and *n* columns
    - $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{1,1} & \cdots & \boldsymbol{A}_{1,n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{m,1} & \cdots & \boldsymbol{A}_{m,n} \end{bmatrix}$

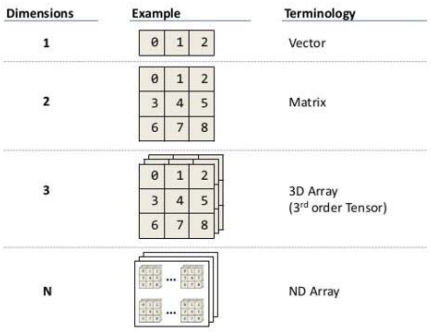
Linear Algebra

```
# matrix is 2-D array
>>> a = np.arange(4).reshape(2,2)
>>> a
array([[0, 1],
       [2, 3]])

# check number of dimensions
>>> a.ndim
2

# check the number of rows and columns
>>> a.shape
(2, 2)
```


Linear Algebra

- A **tensor** is a generalization of vectors and matrices to higher dimensions

```
# tensor is higher dimensional array
>>> a.reshape(1, 2, 2)
array([[[0, 1],
        [2, 3]]])

>>> a.reshape(1, 2, 2).shape
(1, 2, 2)

>>> a.reshape(1, 2, 2).ndim
3
```



Linear Algebra

- Transpose of vectors and matrices
  - The **transpose** of a vector or matrix is when we reflect it across the main diagonal
    - More simply: Swap the rows and columns

Linear Algebra

```
# transposing vectors
>>> a = np.arange(5)
>>> a
array([0, 1, 2, 3, 4])
>>> a.shape
(5,)

# method to transpose
>>> a.T
array([0, 1, 2, 3, 4])
>>> a.T.shape
(5,)

# function to transpose
>>> np.transpose(a)
array([0, 1, 2, 3, 4])
>>> np.transpose(a).shape
(5,)
```

- Why didn't anything happen?

Linear Algebra

```
# transposing vectors continued
>>> a.reshape(-1,1)
array([[0],
       [1],
       [2],
       [3],
       [4]])
>>> a.reshape(-1,1).shape
(5, 1)

>>> a.reshape(-1,1).T
array([[0, 1, 2, 3, 4]])
>>> a.reshape(-1,1).T.shape
(1, 5)
```


Linear Algebra

```
# transposing matrices

>>> b = np.arange(9).reshape(3,3)
>>> b
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])

>>> b.T
array([[0, 3, 6],
       [1, 4, 7],
       [2, 5, 8]])

>>> b.transpose()
array([[0, 3, 6],
       [1, 4, 7],
       [2, 5, 8]])
```

Linear Algebra

- Vector addition
  - Vectors must have the same length
  - Addition is done element-wise
    - $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, \cdots, x_n + y_n)^T$
    - Result is another vector

```
# vector addition

>>> a = np.arange(6)
>>> b = a + 4

>>> a
array([0, 1, 2, 3, 4, 5])
>>> b
array([4, 5, 6, 7, 8, 9])

>>> a + b
array([ 4,  6,  8, 10, 12, 14])
```


Linear Algebra

- Matrix arithmetic
  - If ***A*** and ***B*** are the same size (same number of rows and columns), the resulting matrix has the same size
- $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} \implies \mathrm{C}_{i,j} = \mathrm{A}_{i,j} + B_{i,j}$

```
# matrix addition

>>> a = np.arange(4).reshape(2,2)
>>> b = a + 4

>>> a
array([[0, 1],
       [2, 3]])
>>> b
array([[4, 5],
       [6, 7]])
>>> a + b
array([[ 4,  6],
       [ 8, 10]])
```


Linear Algebra

- The **dot product** of two vectors
  - Results in a *scalar*
  - Multiply corresponding elements, then add the products
  - $\boldsymbol{a} = \boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$

```
>>> a = np.arange(6)
>>> b = a + 4

>>> a
array([0, 1, 2, 3, 4, 5])
>>> b
array([4, 5, 6, 7, 8, 9])

# element-wise multiplication
>>> a*b
array([ 0,  5, 12, 21, 32, 45])

# dot product
>>> a.dot(b)
115
```

Linear Algebra

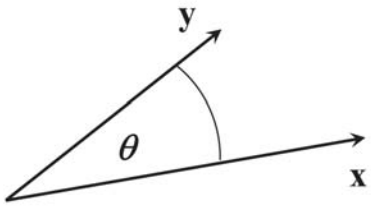
- The **norm** of a vector ***x*** is the length from the origin to ***x***
  - Used for measuring the size of a vector
  - Maps vectors to non-negative values
- L^p Norm
  - $\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$
- L^2 Norm (Euclidean)
  - $\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2} = \sqrt{\sum_{i=1}^{n} |x_i|^2}$
- L^1 Norm (Manhattan/Taxicab)
  - $\|x\|_1 = \left(\sum_{i=1}^{n} |x_i|^1\right)^{1/1} = \sum_{i=1}^{n} |x_i|$
- We will talk more about why you would want to use different distance metrics later on in day 2

Linear Algebra

```
>>> a = np.arange(6)
>>> a
array([0, 1, 2, 3, 4, 5])

# euclidean norm
>>> np.linalg.norm(a)
7.416198487095663

# taxicab norm
>>> np.linalg.norm(a, ord=1)
15.0
```


Linear Algebra

- Alternative representation of the dot product
  - $\boldsymbol{a} = \boldsymbol{x} \cdot \boldsymbol{y} = \|\boldsymbol{x}\|\|\boldsymbol{y}\| \cos\theta$
- Recall
  - $\cos 0° = 1$
  - $\cos 90° = 0$



- The dot product equals 0 when ***x*** ⊥ **y**
- The dot product equals 1 when $\overleftrightarrow{xy}$ (collinear)

Linear Algebra

- Matrix multiplication
  - The inner dimension of the matrices have to match
    - In a chain of matrix multiplications, the *column* dimensions must match the *row* dimensions of the following matrix in the chain
  - The resulting size is the number of *rows* of the first and the number of *columns* of the second

Linear Algebra

```
>>> a = np.arange(4).reshape(2,2)
>>> b = a + 4

>>> a
array([[0, 1],
       [2, 3]])
>>> b
array([[4, 5],
       [6, 7]])

# element-wise multiplcation
>>> a*b
array([[ 0,  5],
       [12, 21]])

# matrix multiplication
>>> a.dot(b)
>>> a@b
>>> np.matmul(a,b)
array([[ 6,  7],
[26, 31]])
```


Linear Algebra

- Matrix multiplication
  - Example matrices
    - **A** is 3x5
    - **B** is 5x1
    - **C** is 5x2
  - Can we compute?
    - **AB**
    - **AC**
    - **BC**
    - **BA**
    - **CA**

Linear Algebra

- Matrix multiplication
  - Example matrices
    - **A** is 3x5
    - **B** is 5x1
    - **C** is 5x2
  - Can we compute?
    - **AB** ✓
    - **AC** ✓
    - **BC** ✗
    - **BA** ✗
    - **CA** ✗

Machine Learning Overview

There are several Python ML frameworks

This course will begin by using scikit-learn
- One of the most popular and user-friendly machine learning libraries
- Plays nice with Numpy, Pandas, & SciPy

Other ML/DL libraries to be aware of:
- Keras
  - TensorFlow
  - Theano
- Torch/Pytorch
- Caffe/Caffe2
- CNTK
- MXNet
- SparkML

Machine Learning Overview

- scikit-learn is a python package that provides efficient implementations of most common ML algorithms via a simple API interface
- scikit-learn has great online documentation
  - http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
- Uniform syntax for "Estimators"
  - Once you know how to code one model, you can code them all

Machine Learning Overview

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

HPSI Human Performance Systems, Inc.

Machine Learning Overview

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

HPSI Human Performance Systems, Inc.

Define the Problem

- What are we classifying?
  - Identify the target variable of interest
  - Decide how that variable ought to be measured
- Choose a *performance metric* based on the goal
  - Accuracy is the simplest classification metric, but it may not be appropriate depending on the application
  - We want to find the metric closest to our business goal that is feasible to evaluate
  - We'll circle back to performance metrics in model evaluation

HPSI Human Performance Systems, Inc.

Machine Learning Overview

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

Machine Learning Overview

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

Explore the Data

- How big is the dataset?
- What types of features do we have?
- Do we have missing data?
  - Do we know why that data is missing?
- How are the data distributed?
  - Class breakdown & Visualization

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

Preprocess the Data

- *Imputing* Missing Values
  - Method: mean, median, mode, interpolation, etc.
  - Sometimes makes sense to drop observation if it's missing several measurements
    - Drop rows based on threshold criterion
- *Encoding* Categorical Features
  - Method determined by whether feature is *nominal* or *ordinal*
- *Structuring* Data for Modeling in scikit-learn
- *Scaling* features
  - Normalization/Standardization
- *Engineering* Features
  - Not explicitly covered in this course
  - This is where domain knowledge comes in hand

Dealing with Missing Data

- What causes missing data?
  - Error in data collection
  - Certain measurements are "Not-Applicable"
  - Fields may have been left blank during data entry
- Null values
  - NaN vs " " vs "_" vs "." vs "N/A" vs "N-A"
- What to do?
  - Impute
    - Mean
    - Median
    - Most Frequent Class
    - Forward-Fill
    - Back-Fill
  - Drop based on a missing-value threshold

Quantifying Missing Data

- Count the number of missing values per column:

```
>>> df.isnull().sum()
```

- Simply remove the corresponding samples from the dataset entirely:

```
# axis = 0 for rows
# axis = 1 for columns
>>> df.dropna(axis=0)

#only drop rows where all columns are NaN
>>> df.dropna(how = 'all')
```


Handling Missing Values

```
# drop rows that have less than 4 real values
>>> df.dropna(thresh = 4)

#only drop rows where NaN appear in specific
columns (here: 'C')
>>> df.dropna(subset = ['C'])
```


Imputing Missing Values

- Usually we have too much valuable data to simply drop columns/rows, in this case we can fill the missing data ("imputation")

```
>>> from sklearn.preprocessing import Imputer
>>> imr = Imputer(missing_values = 'NaN',
                  strategy = 'mean',
                  axis = 0)
>>> imr.fit(df.values)
>>> imputed_data = imr.transform(df.values)
```

Imputing Missing Values

- Here, we replaced each "NaN" value with the corresponding mean, which is separately calculated for each feature column.
- You can impute based on row means by setting `axis = 1`
- Other imputation strategies that are available include setting the `strategy` parameter to:
  - `median`
    - replaces missing values with the median value
  - `most_frequent` (mode)
    - replaces missing values with the most frequent value in the column

Encoding Categorical Features

- Nominal
  - Categorical features that don't imply an order
    - T-shirt colors (Red, Blue, Green)
    - Animals (Dog, Cat, Fish, Tiger)
- Ordinal
  - Categorical features that can be sorted or have an order
    - T-shirt size (S, M, L)
    - Quality (Poor, Fair, Good, Excellent)

Encoding Categorical Features

- Let's make a dummy dataset to walk through encoding categorical features

```
>>> import pandas as pd
>>> df = pd.DataFrame([
...            ['green', 'M', 10.1, 'class1'],
...            ['red', 'L', 13.5, 'class2'],
...            ['blue', 'XL', 15.3, 'class1']])
>>> df.columns = ['color', 'size', 'price', 'classlabel']
>>> df
   color size  price classlabel
0  green    M   10.1     class1
1    red    L   13.5     class2
2   blue   XL   15.3     class1
```

Encoding Categorical Features

- Mapping Ordinal Features
  - To ensure the algorithm interprets the order appropriately, we need to convert the categorical string values into integers.
  - There is no function to automatically derive the correct order of the labels of our size feature, so we have to define the mapping manually
  - Assume the relation XL = L + 1 = M + 2 holds

Encoding Categorical Features

- Mapping Ordinal Features

```
>>> size_mapping = {
...                 'XL': 3,
...                 'L': 2,
...                 'M': 1}
>>> df['size'] = df['size'].map(size_mapping)
>>> df
      color  size  price classlabel
0    green     1   10.1     class1
1      red     2   13.5     class2
2     blue     3   15.3     class1
```


Encoding Categorical Features

- Inverse Mapping Ordinal Features
  - If want to recover the original string representation later on, we can define a reverse-mapping dictionary:

```
inv_size_mapping = {v: k for k, v in size_mapping.items()}
```

  - can use with pandas `map()` method on the transformed feature column, similar to the previous mapping

```
>>> inv_size_mapping = {v: k for k, v in
size_mapping.items()}
>>> df['size'].map(inv_size_mapping)
0     M
1     L
2    XL
Name: size, dtype: object
```

Encoding Class Labels

- Most sklearn classification estimators convert class labels to integers internally
- Still good practice to provide class labels as integer arrays to avoid issues
- Class labels are not ordinal
- It doesn't matter which integer number we assign to a particular string label.
  - Thus, we can simply enumerate the class labels, starting at 0

Encoding Class Labels

- The `LabelEncoder` class in sklearn does this for us:

```
>>> from sklearn.preprocessing import LabelEncoder
>>> class_le = LabelEncoder()
>>> y = class_le.fit_transform(df['classlabel'].values)
>>> y
array([0, 1, 0])
```


Encoding Class Labels

- The `LabelEncoder` class in sklearn does this for us:

```
>>> from sklearn.preprocessing import LabelEncoder
>>> class_le = LabelEncoder()
>>> y = class_le.fit_transform(df['classlabel'].values)
>>> y
array([0, 1, 0])
```

Encoding Class Labels

- Note that the `fit_transform` method is just a shortcut for calling `fit` and `transform` separately
  - `fit` makes the mapping
  - `transform` applies the mapping
  - we can use the `inverse_transform` method to transform the integer class labels back into their original string representation

```
>>> class_le.inverse_transform(y)
array(['class1', 'class2', 'class1'], dtype=object)
```


Performing One-Hot Encoding on Nominal Features

- Sklearn treats class labels as nominal variables, so we were able to use the `LabelEncoder` to encode the strings as integers
- We won't be able to use this method for the nominal features in our data though

```
>>> X = df[['color', 'size', 'price']].values
>>> color_le = LabelEncoder()
>>> X[:, 0] = color_le.fit_transform(X[:, 0])
>>> X
array([[1, 1, 10.1],
       [2, 2, 13.5],
       [0, 3, 15.3]], dtype=object)
```


Performing One-Hot Encoding on Nominal Features

- The new values for the *color* feature are as follows:
  - Blue = 0
  - Green = 1
  - Red = 2
- Although the color values don't come in any particular order, the classifier will assume that green is larger than blue, and red is larger than green
- The solution: **One-hot encoding**
  - Create a new dummy feature for each unique value in the nominal feature column

Performing One-Hot Encoding on Nominal Features

```
>>> from sklearn.preprocessing import OneHotEncoder
>>> ohe = OneHotEncoder(categorical_features=[0])
>>> ohe.fit_transform(X).toarray()
array([[  0. ,   1. ,   0. ,   1. ,  10.1],
       [  0. ,   0. ,   1. ,   2. ,  13.5],
       [  1. ,   0. ,   0. ,   3. ,  15.3]])
```

- Specify columns to one-hot encode when you instantiate
- The `OneHotEncoder` returns a sparse matrix by default
  - We use `toarray` to see it
  - Could also instantiate `OneHotEncoder` with the `sparse=False` parameter

Performing One-Hot Encoding on Nominal Features

- We can also do this with pandas:

```
>>> pd.get_dummies(df[['price', 'color', 'size']])
    price  size  color_blue  color_green  color_red
0    10.1     1           0            1          0
1    13.5     2           0            0          1
2    15.3     3           1            0          0
```


Caution with One-Hot Encoding

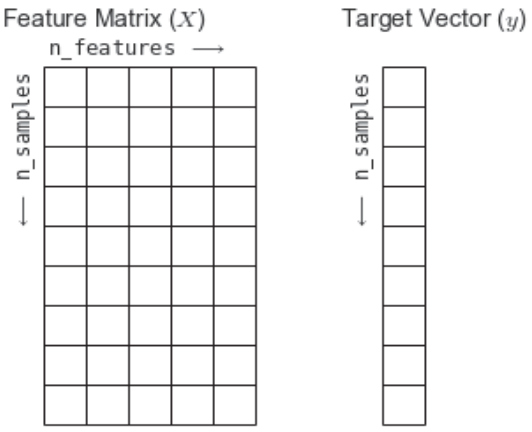
- One-Hot Encoding introduces multicollinearity to our dataset
  - Not great for methods that rely on matrix inversion
- To reduce multicollinearity, we can drop columns that don't give much additional info
  - If we remove the column `color_blue`, the feature information is still preserved since if we observe `color_green=0` and `color_red=0`, it implies that the observation is blue

```
>>> pd.get_dummies(df[['price', 'color', 'size']],
...                drop_first=True)
   price  size  color_green  color_red
0   10.1     1            1          0
1   13.5     2            0          1
2   15.3     3            0          0
```

Structuring Data for Modeling in Sklearn

- **Feature matrix (X)**
  - Scikit-learn assumes a 2-D array of inputs with dimensions [`n_samples`, `n_features`]
  - Scikit-learn expects a feature matrix as a NumPy array or Pandas DataFrame
    - some Scikit-Learn models also accept SciPy sparse matrices.
- **Target vector (y)**
  - Usually a column vector of length `n_samples`
  - scikit-learn expects a NumPy array or Pandas Series.

Structuring Data for Modeling in Sklearn



Structuring Data for Modeling in Sklearn

- We've split our data up into a feature matrix & target vector, but we still aren't ready to model it yet
  - If we fit our model to all the data and test the model on all the data, we would correctly predict almost every sample in the training set
    - Memorizing; not learning
  - Want our models to *generalize*
    - Perform well on unseen (out-of-sample) data
  - Holdout some data to test on
- Split our data into a *training set* and *test set*
  - Learn parameters from the training set and predict/score on the test set

Structuring Data for Modeling in Sklearn

- Sklearn's `train_test_split` function does this for us
  - Its default parameter is to extract 75% for training and 25% for testing

```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test =
train_test_split(iris['data'], iris['target'], random_state=0)
```


Splitting into Training and Test Data

- The `train_test_split` function shuffles the data with a pseudorandom number generator before splitting
  - Without shuffling the iris data, we would only have class 2 in the test set
  - To make sure we get the same splits, we set the seed with the `random_state` parameter
- Dealing with imbalanced classes (credit fraud/cancer examples)
  - When dealing with imbalanced datasets, it is important to ensure the training and test set have the same proportions as the original dataset
  - We can do this by passing the target vector as an argument to the `stratify` parameter

```
train_test_split(iris['data'], iris['target'],
random_state=0, stratify=y)
```


Scaling Features

- Some algorithms are very sensitive to how the data are scaled
  - Neural networks → Vanishing gradient
- There are *four* different ways to scale our data in sklearn:
  - `StandardScaler`
  - `RobustScaler`
  - `MinMaxScaler`
  - `Normalizer`

Scaling Features

- `StandardScaler`
  - Scales each feature so that the mean is zero and the standard deviation is one
    - $x_{std} = \frac{x - \mu_x}{\sigma_x}$
    - All features on same magnitude
    - Does not ensure any particular minimum or maximum values
- `RobustScaler`
  - Similar to the `StandardScaler` in that it guarantees features are on the same scale, but uses the median and quartiles to scale instead of mean and variance
    - $x_{robust} = \frac{x - Q_2}{Q_3 - Q_1}, where\ Q_1, Q_2, Q_3\ are\ the\ 1st, 2nd, and\ 3rd\ quartiles, respectively$
    - Makes the scaler *robust* to outliers

Scaling Features

- `MinMaxScaler`
  - $x_{minmax} = \frac{x - x_{min}}{x_{max} - x_{min}}$
  - Shifts the data such that all features are between zero and one
  - Min-Max scaling is useful when we need values in a bounded interval
  - Standardization can be more practical for many machine learning algorithms especially for optimization algorithms such as gradient descent.
    - Many linear models initialize weights to 0 or small random values close to 0
    - Using standardization, we center the feature columns at mean 0 with standard deviation 1 so that the feature columns takes the form of a normal distribution
    - Standardization maintains useful information about outliers and makes the algorithm less sensitive to them in contrast to min-max scaling, which scales the data to a limited range of values.

Scaling Features

- `Normalizer`
  - $x_{norm} = \frac{x}{\|x\|}$
  - Scales each point such that the feature vector has a Euclidean length of one
  - Projects data point on unit circle (or *n*-dimensional counterpart)
    - Every point is scaled by a different number (inverse of it's length)
  - Used when only the direction/angle of the data matters, not length

Scaling Example

- Open up the `feature-scaling.py` script to see the different scaling methods in action

Feature Engineering & Feature Selection

- *Feature Engineering* is the process of determining how to best represent your data for a particular application
  - Some would consider encoding categorical features to be part of this process
    - Encoding categorical features vs discretizing continuous variables/interaction terms/polynomial features/transformations
  - Case-by-case and model-dependent
    - No need to scale features for tree-based methods
    - Very important to scale features for *k*-NN & neural networks
  - Industry knowledge is very helpful at this stage

Feature Engineering & Feature Selection

- Encoding and engineering features can cause us to increase the dimensionality of the data greatly beyond the original number of features
  - When adding new features in this way, or with high-dimensional datasets in general, it's a good idea to reduce the number of features to only the most useful ones
  - This can lead to simpler models that *generalize* better
- *Feature Selection* is the process of determining which features to include in our model
  - *Three* main methods:
    1. Univariate Statistics (ANOVA)
    2. Model-based Feature Selection (Trees & LASSO)
    3. Iterative Feature Selection (RFE)

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

Scikit-learn Estimator – Steps to Fit a Model & Generate Predictions

1. **Import** appropriate estimator from scikit-learn
2. **Instantiate** estimator with desired hyperparameters
3. **Fit** the model by calling the `fit()` method on the training set
   - ❖What's really happening here?
     - ➢The model is *learning* the relationship between the features/inputs and the response/target
     - ➢Note: All learned parameters are stored as names with trailing underscores
4. **Predict** on unseen data (test set):
   - ❖Predict labels with `predict()` method

Scikit-learn Estimator – Steps to Fit a Model & Generate Predictions

```
# import sklearn methods and estimator
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.neighbors import KNeighborsClassifier
>>> iris = load_iris()
# feature matrix and target vector
>>> X = iris.data
>>> y = iris.target
# split into training and test set
>>> X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
# instantiate estimator
>>> knn = KNeighborsClassifier(n_neighbors=1)
# fit on training set to 'learn' parameters
>>> knn.fit(X_train, y_train)
# predict on the test set
>>> y_pred = knn.predict(X_test)
```

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

Performance Metrics for Binary Classification Models

- In *binary* classification, we have 2 classes—positive and negative
- **Accuracy:** Fraction of *correctly* classified samples
  - number correctly classified / total number of samples
  - (TP+TN)/total
    - A *True Positive* is when the sample is from the *positive* class and we *correctly* predict the *positive* class
    - A *True Negative* is when the sample is from the *negative* class and we *correctly* predict the *negative* class
    - A *False Positive* is when the sample is from the *negative* class and we *incorrectly* predict the *positive* class (Type I Error)
    - A *False Negative* is when the sample is from the *positive* class and we *incorrectly* predict the *negative* class (Type II Error)

Performance Metrics for Binary Classification Models

- Each estimator in scikit-learn has a `score` method that computes the default evaluation criterion for the problem they are designed to solve
  - For almost all classifiers, this will be accuracy
- In our previous example, the following command would calculate the accuracy of the classifier:

```
knn.score(X_test, y_test)
0.97368421052631582
```

- There are also explicit functions in scikit-learn for all the performance metrics

Accuracy in sklearn

```
>>> import numpy as np
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]

# fraction of classified samples
>>> accuracy_score(y_true, y_pred)
0.5

# number of correctly classified samples
>>> accuracy_score(y_true, y_pred, normalize=False)
2
```


When Would You Not Want to Use Accuracy?

- When the number of mistakes does not contain all the information we are interested in
- **Example:** Imagine an automated application to screen for the early detection of cancer. If the test is *negative*, we assume the patient is healthy, and if the test is *positive*, the patient will undergo additional screening.
  - Positive Test → Cancer
  - Negative Test → Healthy
- Our model is never going to be going to be perfect

When Would You Not Want to Use Accuracy?

- What are the consequences of being wrong?
  1. Healthy patient is told they have cancer (False Positive)
     - Costs and inconvenience to patient
  2. Sick patient is told they are healthy (False Negative)
     - Patient will not receive further tests & treatment
     - Undiagnosed cancer could lead to serious issues or ☠
- We obviously want to avoid *False Negatives* as much as possible in this example

When Would You Not Want to Use Accuracy?

- We run into the same issue with imbalanced datasets
- **Example:** Imagine we are looking at credit fraud data where 99% of the transactions were not fraudulent and 1% were fraudulent
  - Say we build a complex ML model that has 99% accuracy in predicting whether a transaction is fraudulent or not
    - Is this actually good?
      - We can just guess every single time that every transaction is not fraudulent and get an accuracy of 99%
      - We actually don't know if the model is good because accuracy doesn't give us enough information as to how predictions were made

Confusion Matrix

- The *confusion matrix* is a more informative way to represent the results of a binary classifier

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Confusion Matrix

- 165 total observations/predictions

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Confusion Matrix

➢Of the 165 total observations, we predicted *yes* 110 times

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

HPSI Human Performance Systems, Inc.

Confusion Matrix

➢Of the 165 total observations, we predicted *no* 55 times

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

HPSI Human Performance Systems, Inc.

Confusion Matrix

➢Of the 165 total observations, 105 are actually *yes*

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

HPSI Human Performance Systems, Inc.

Confusion Matrix

➢Of the 165 total observations, 60 are actually *no*

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

HPSI Human Performance Systems, Inc.

Confusion Matrix

➢Diagonal terms are correct classifications

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

HPSI Human Performance Systems, Inc.

Confusion Matrix

➢Accuracy = (TN + TP) / total = (100 + 50) / 165 = 0.91

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

HPSI Human Performance Systems, Inc.

Confusion Matrix in sklearn

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [0, 1, 0, 1]
>>> y_pred = [1, 1, 1, 0]
>>> confusion_matrix(y_true, y_pred)
array([[0, 2],
       [1, 1]])

# we can extract the following info in the binary case
>>> tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
>>> print((tn, fp, fn, tp))
(0, 2, 1, 1)
```


Metrics Derived from Confusion Matrix

- **Misclassification Rate:** Proportion *incorrectly* classified
  - (FP+FN)/total
  - Misclassification rate = 1 – Accuracy
- **Precision:** When it predicts *yes*, how often is it correct?
  - TP/(TP + FP)
    - Bounded between [0,1] where 1 is the best
  - *Used when the goal is to limit the number of false positives*
    - Clinical trials are expensive, so you want to know a drug works before investing in them
- **Recall (True Positive Rate):** When it's actually *yes*, how often does it predict *yes*?
  - TP/(TP + FN)
    - Bounded between [0,1] where 1 is the best
  - *Used when we need to identify all positive samples (avoid false negatives)*
    - *Cancer example*

Confusion Matrix

- Misclassification Rate = (FP+FN) / total = (10 + 5) / 165 = 0.09

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Confusion Matrix

➢Precision = TP / (TP + FP) = 100 / (100 + 10) = 0.91

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Confusion Matrix

➢Recall = TP / (TP + FN) = 100 / (100 + 5) = 0.95

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Misclassification Rate, Precision, & Recall in sklearn

```
>>> from sklearn.metrics import accuracy_score, precision_score,
recall_score

>>> miss_rate = 1 - accuracy_score(y_true, y_pred)
>>> precision = precision_score(y_true, y_pred)
>>> recall = recall_score(y_true,y_pred)

>>> print("Misclassification Rate: {:03.2f}".format(miss_rate))
>>> print("Precision: {:03.2f}".format(precision))
>>> print("Recall: {:03.2f}".format(recall))
Misclassification Rate: 0.75
Precision: 0.33
Recall: 0.50
```

Metrics from Confusion Matrix

- ***f1*-score**
  - Harmonic mean of precision and recall
    - Harmonic mean is used when we want to average rates
  - $2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
    - Bounded between [0, 1] where 1 is best
    - Equal contribution between precision and recall
    - Hardest to interpret

f1-score in sklearn

```
>>> from sklearn.metrics import f1_score
>>> print("f1-score: {:03.2f}".format(f1_score(y_true,
y_pred)))

# can do precision, recall, and f1 all together
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred))
                precision recall f1-score support
             0       0.00   0.00    0.00       2
             1       0.33   0.50    0.40       2
avg / total          0.17   0.25    0.20       4
```


Performance Metrics for Binary Classification Models

- The confusion matrix and classification report provided much more information about the predictions than accuracy
  - But, can we get even more information?
    - Yes!
  - We can take uncertainty into account too
    - Most classifiers in sklearn have either a `decision_function` or `predict_proba` method
    - Predictions can be seen as ***thresholding*** the output of one of these functions
      - `decision_function` represents the signed difference of an observation to the plane
        - Assigns all points with a value greater than zero to the positive class
      - `predict_proba` represents the predicted probability of belonging to each class
        - Assigns the point to class based on highest probability

Thresholding

- Depending on the end goal, we can adjust the decision threshold for the probability function
- Recall the cancer screening example where we want a high recall for the positive class
  - Willing to risk more *false positives* in order to reduce *false negatives*
  - We might consider *decreasing* the threshold on the predictions in order to *increase* the recall
    - Recall = TP / (TP + FN)
    - Precision = TP / (TP + FP)
    - Decrease FN → Increase Recall
    - Increase FP → Decrease Precision
- Adjusting the threshold is a trade-off between precision and recall
  - **You should NEVER set your threshold based on the test set**
  - **Always use the training or validation set to determine your optimal threshold**

Precision-Recall Curves

- Thresholding is particularly helpful when we have a set precision or recall score determined by the business project
- Once a precision or recall score is set, a threshold can be set
- Most of the time, we do not have a set point and may wish to look at all possible thresholds and trade-offs at once
  - We can do this with the *precision-recall curve*

Precision-Recall Curve in sklearn

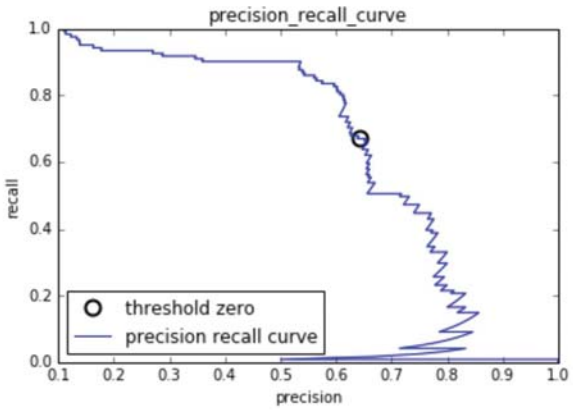
```
>>> from sklearn.metrics import precision_recall_curve
>>> precision,recall,thresholds =
precision_recall_curve(y_test,
                       model.decision_function(X_test))
```

- The `precision_recall_curve` function returns a list of precision and recall values for all possible thresholds (all values that appear in the decision function) in sorted order
- We can use this to plot the curve

Precision-Recall Curve in sklearn

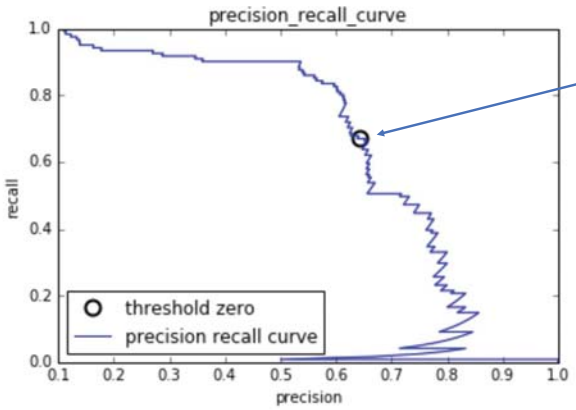
```
# find threshold closest to zero:
>>> close_zero = np.argmin(np.abs(thresholds))
>>> plt.plot(precision[close_zero], recall[close_zero],
'o', markersize=10, label="threshold zero",
fillstyle="none", c='k', mew=2)
>>> plt.plot(precision, recall, label="precision recall
curve")
>>> plt.xlabel("precision")
>>> plt.ylabel("recall")
>>> plt.title("precision_recall_curve")
>>> plt.legend(loc="best")
```


Precision-Recall Curves



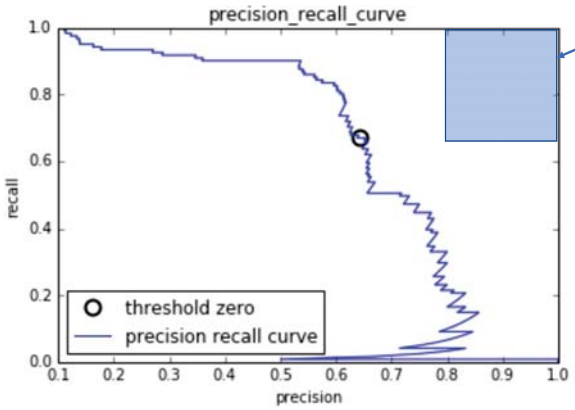
Each point along the curve corresponds to a possible threshold from either `predict_proba` or `decision_function`

Precision-Recall Curves



We can write code to find the threshold closest to zero in the list of returned thresholds and plot it to see where the default decision threshold is

Precision-Recall Curves

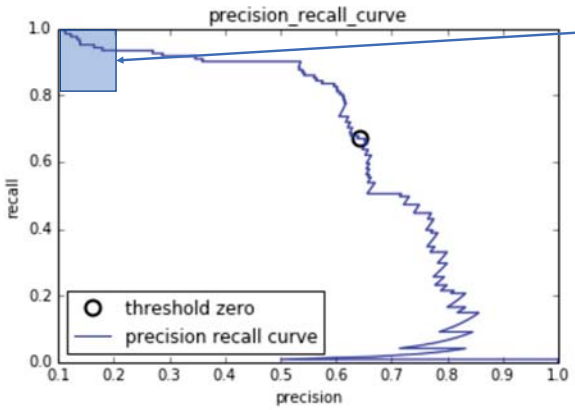


The closer a curve stays to the upper-right corner, the better the classifier

A point in the upper-right means a classifier has high precision & high recall

HPSI Human Performance Systems, Inc.

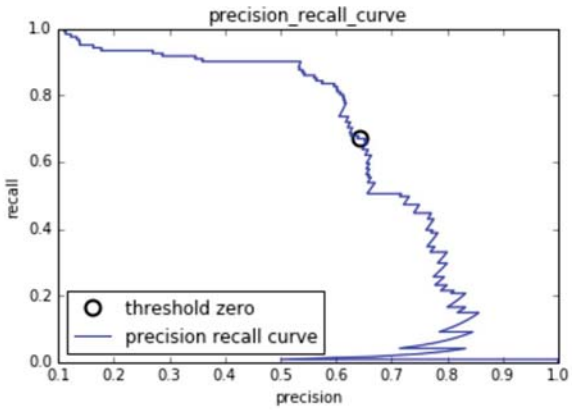
Precision-Recall Curves



The curve starts in the top-left corner, corresponding to a very low threshold, which classifies everything as the *positive* class

HPSI Human Performance Systems, Inc.

Precision-Recall Curves

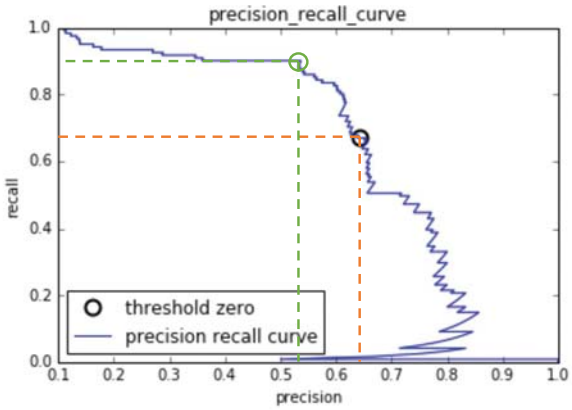


Raising the threshold moves the curve towards *higher precision* and *lower recall*

The more our model keeps *recall* high while increasing *precision*, the better

HPSI Human Performance Systems, Inc.

Precision-Recall Curves

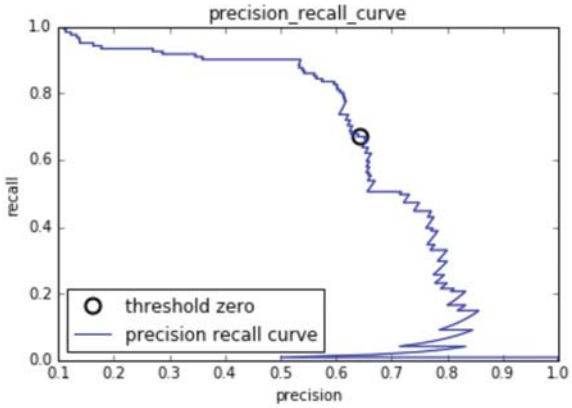


Looking at this curve, it is possible to achieve a much higher recall, while keeping precision above 0.5, by decreasing the threshold

In the cancer example, we would consider moving the threshold to the one in the green circle

HPSI Human Performance Systems, Inc.

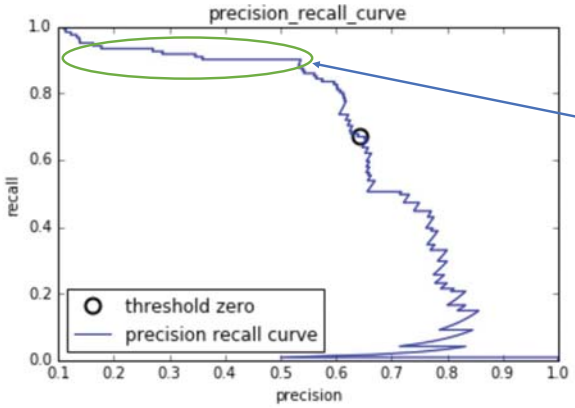
Precision-Recall Curves



Conversely, in this example, getting a high precision is very costly

HPSI Human Performance Systems, Inc.

Precision-Recall Curves

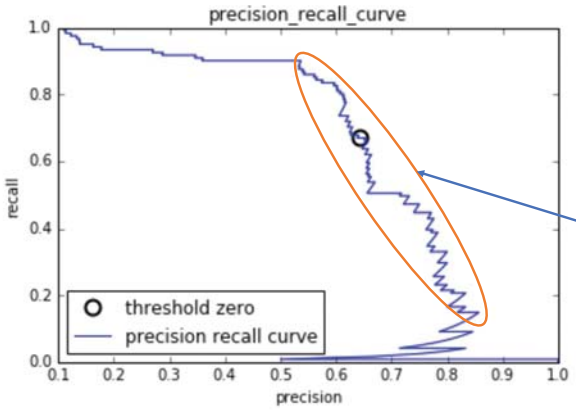


Conversely, in this example, getting a high precision is very costly

This flat region allows us to increase precision at very little cost to recall

HPSI Human Performance Systems, Inc.

Precision-Recall Curves



Conversely, in this example, getting a high precision is very costly

But, for each increase in precision past 0.5, it costs us a lot of recall (evidenced by the steep slope)

ROC Curve

- Another common tool to evaluate a model at different thresholds is the *Receiver Operating Characteristics (ROC) Curve*
- The ROC curve similarly considers all possible thresholds for a given classifier
  - But, instead of reporting precision and recall, it shows the *false positive rate* (FPR) against the *true positive rate* (TPR)
  - Note: the true positive rate is simply another name for *recall*, while the false positive rate is the fraction of false positives out of all negative samples

ROC Curve in sklearn

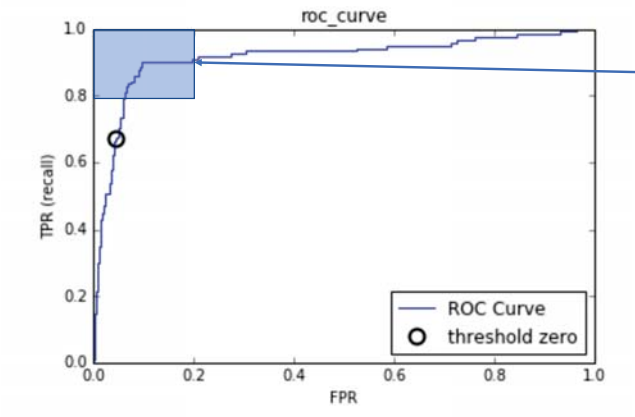
```
>>> from sklearn.metrics import roc_curve
>>> fpr, tpr, thresholds = roc_curve(y_test,
model.decision_function(X_test))
```

- The `roc_curve` function returns a list of false positive rates and true positive rates for all possible thresholds (all values that appear in the decision function) in sorted order
- We can use this to plot the curve

ROC Curve in sklearn

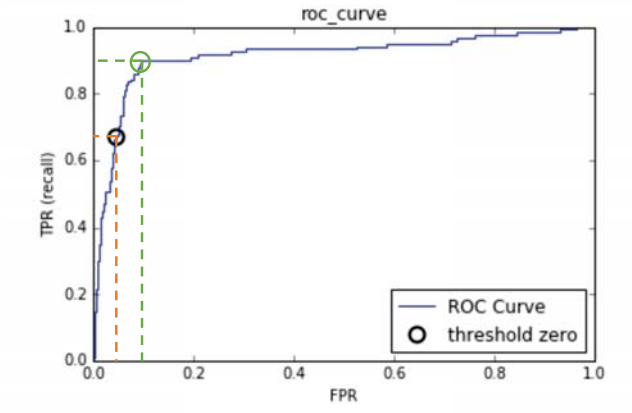
```
# find threshold closest to zero:close_zero =
np.argmin(np.abs(thresholds))
plt.plot(fpr[close_zero], tpr[close_zero], 'o',
markersize=10, label="threshold zero",
fillstyle="none", c='k', mew=2)
plt.legend(loc=4)
plt.plot(fpr, tpr, label="ROC Curve")
plt.xlabel("FPR")
plt.ylabel("TPR (recall)")
plt.title("roc_curve")
```


ROC Curve



For the ROC curve, the ideal curve is close to the top left—you want a classifier that produces a *high recall* while keeping a *low false positive rate*.

ROC Curve



Compared to the default threshold of zero, the curve shows that we could achieve a significantly higher recall (around 0.9) while only increasing the FPR slightly

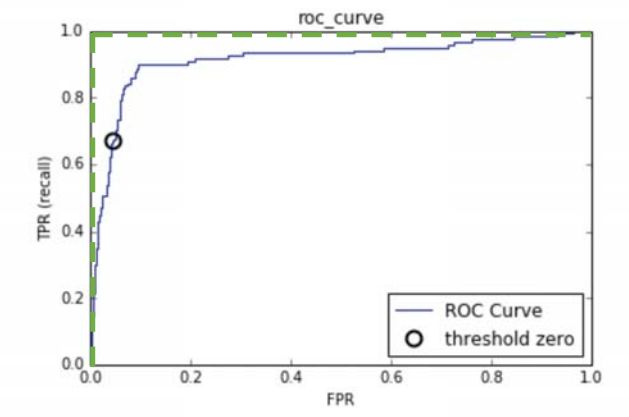
AUC

- If we want to summarize the ROC Curve, we can compute the *area under the curve (AUC)*
  - Bounded between [0, 1]
  - Predicting randomly always produces an AUC of 0.5, not matter how imbalanced the classes in a dataset are
  - This makes it a much better metric for imbalanced classification problems than accuracy
  - The AUC can be interpreted as evaluating the *ranking* of positive samples
  - If you randomly chose one positive and one negative observation, AUC represents the likelihood that your classifier will assign a **higher predicted probability** to the positive observation
  - An AUC of 1 means that all positive points have a higher score than all negative points

AUC in sklearn

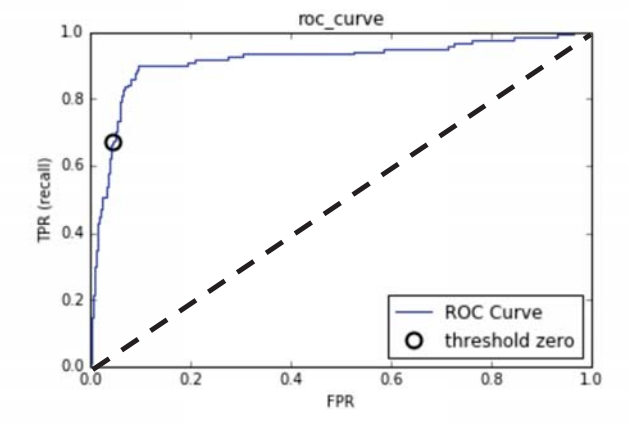
```
>>> from sklearn.metrics import roc_auc_score
# use predicted probability for positive class
>>> auc = roc_auc_score(y_test,
model.predict_proba(X_test)[:, 1])
```


ROC Curve & AUC



A perfect ROC curve is shown in green

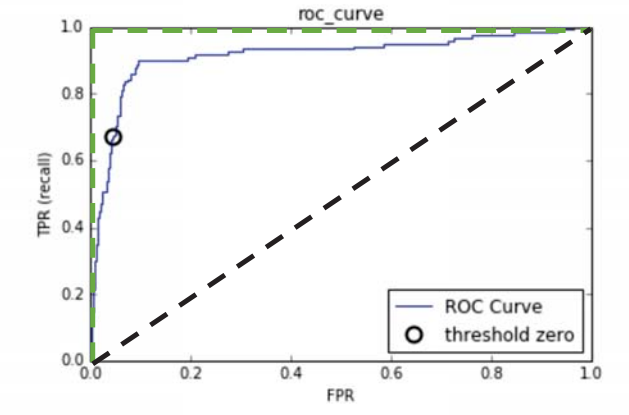
ROC Curve & AUC



A perfect ROC curve is shown in green

If we draw a diagonal line on the ROC plot, this represents a model in which we guess randomly

ROC Curve & AUC

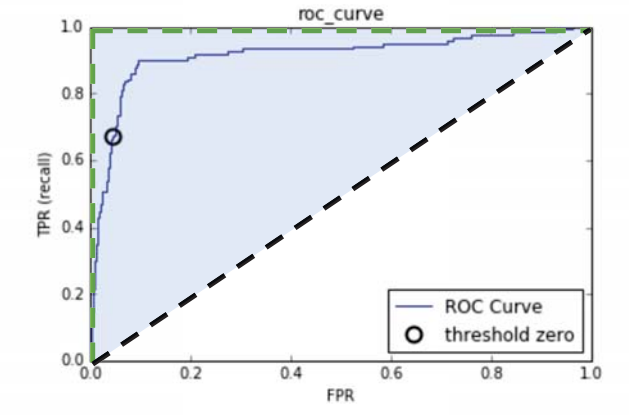


A perfect ROC curve is shown in green

If we draw a diagonal line on the ROC plot, this represents a model in which we guess randomly

Predicting randomly always produces an AUC of 0.5, no matter how imbalanced the classes in a dataset are

ROC Curve & AUC



A perfect ROC curve is shown in green

If we draw a diagonal line on the ROC plot, this represents a model in which we guess randomly

Predicting randomly always produces an AUC of 0.5, no matter how imbalanced the classes in a dataset are

Our model should lie somewhere between the ideal curve and the diagonal line

Final Thoughts on ROC Curve & AUC

- Keep in mind that AUC is just a calculation of the area under the ROC curve
- It does not take the default threshold into account
- We still need to look at the ROC Curve

Machine Learning Overview

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

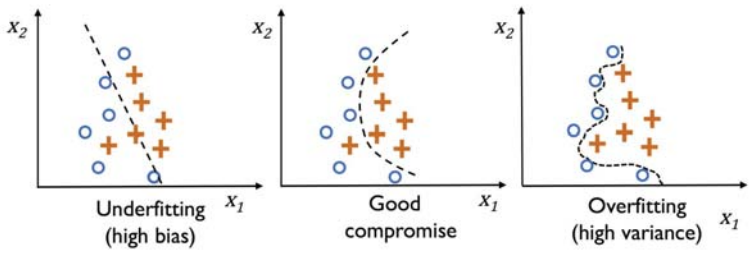
Model Validation

- Recall: one of the main goals in building our classification model is for it perform well on unseen data (*generalize*)
- For this reason, we split our data up into a training set and a test set
- But, when we evaluate the performance of our model, we may find it performs much better on the training set than the test set
- In this section, we will talk more about how make our models generalize better and more robust methods for evaluating generalization performance than a simple split of the original data

Generalization, Overfitting, & Underfitting

- If we notice our model performs much better on the training set than the test set, this is a strong indication of *overfitting*
- ***Overfitting*** occurs when our model is too complex given the underlying training data
  - In this case, we say the model has ***high variance***
  - *Learning the random noise in the data, not the patterns*
- Conversely, ***Underfitting*** occurs when our model is not complex enough to capture the patterns in the training data well and also suffers from low performance on unseen data
  - In this case, we say the model has ***high bias***

Generalization, Overfitting, & Underfitting



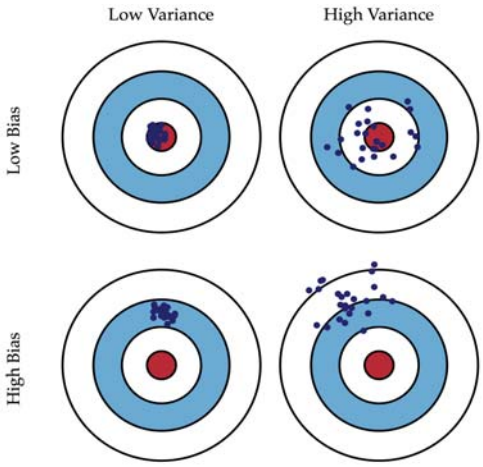
Generalization, Overfitting, & Underfitting

- So we want to avoid *overfitting* because it gives too much predictive power to noise in our training data.
- But in our attempt to reduce overfitting we can also begin to *underfit* or ignore important features in our training data
- So how do we balance the two?
  - This is a problem we call the ***Bias-Variance Tradeoff***

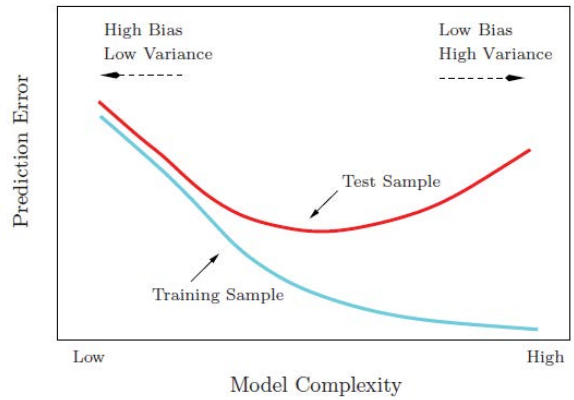
Bias-Variance Tradeoff

- ***Variance*** measures the consistency (or variability) of the model prediction for a particular sample instance if we were to retrain the model multiple times on different subsets of the training dataset
  - The model is sensitive to the randomness in the training data
- ***Bias*** measures how far off the predictions are from the correct values in general if we rebuild the model multiple times on different training datasets
  - Bias is the measure of the systematic error that is not due to randomness

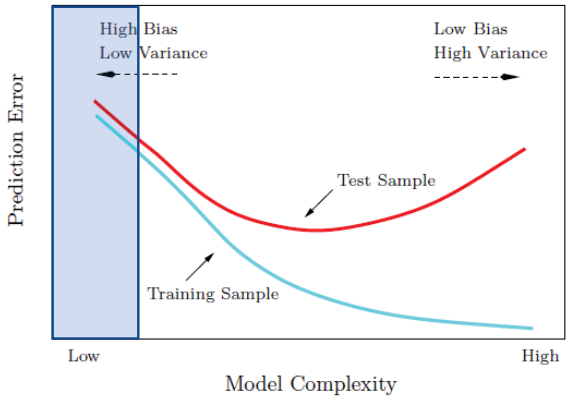
Bias-Variance Tradeoff



Bias-Variance Tradeoff

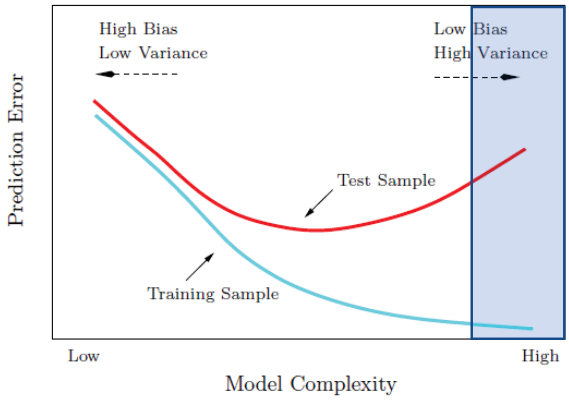


Bias-Variance Tradeoff



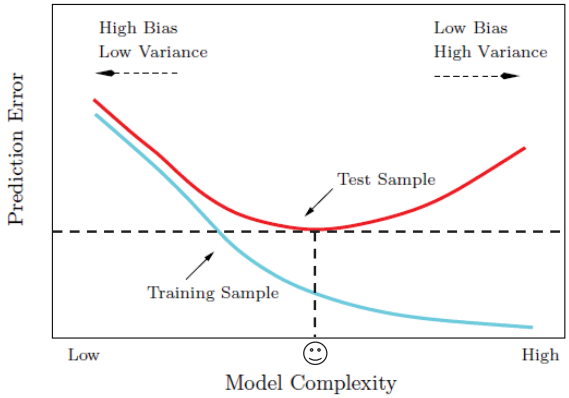
Low complexity models will result in high error (poor accuracy) for both training and test data. This is because the model lacks enough complexity to describe the data.

Bias-Variance Tradeoff



High complexity models will result in a low training error and a high test error. This is because a complex model will be able to model the training data a bit *too* well, and can't generalize to the test data.

Bias-Variance Tradeoff



The ***best complexity*** lies where the *test error* reaches a *minimum.*

Bias-Variance Tradeoff

- As we've seen, overfitting and underfitting have very clear signatures in training and test data
  - *Overfitting* results in *low training error* and *high test error*
  - *Underfitting* results in *high training & test errors*
- To find an acceptable bias-variance tradeoff, we need to evaluate our model carefully
- Often we use more robust methods than a simple training and test set split to find this tradeoff
  - ***Cross-validation***

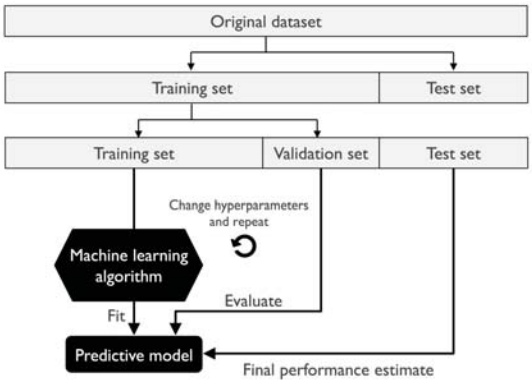
Cross-Validation

- In ***cross-validation***, instead of just splitting the data set in to a training set and a test set, the data is split repeatedly and multiple models are trained
- The *two* most common methods are:
  1. Holdout Cross-Validation
  2. *k*-Fold Cross-Validation

Cross-Validation

- Holdout Cross-Validation
  - Using the holdout method, we split our initial dataset into 3 separate sets:
    1. Training (70 %)
    2. Validation (15 %)
    3. Testing (15 %)
  - The training set is used to fit the models, and the performance on the validation set is then used for the model selection
    - ***Model Selection*** is the process where we try to select the optimal values of tuning parameters (*hyperparameters*) for our models
    - The advantage of having a test set that the model hasn't seen before during the training and model selection steps is that we can obtain a less biased estimate of its ability to generalize to new data

Holdout Cross-Validation



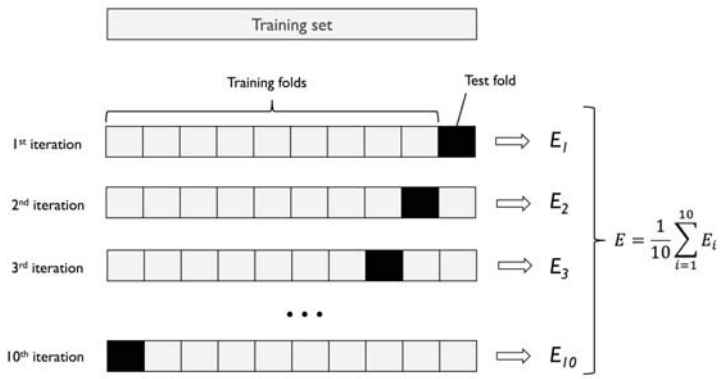
HPSI Human Performance Systems, Inc.

Cross-Validation

- *k*-Fold Cross-Validation
  - In *k*-fold cross-validation, we randomly split the training dataset into *k* folds without replacement, where *k-1* folds are used for the model training, and one fold is used for performance evaluation
  - This procedure is repeated k times so that we obtain k models and performance estimates

HPSI Human Performance Systems, Inc.

k-Fold Cross-Validation



HPSI Human Performance Systems, Inc.

k-Fold Cross-Validation

- We then calculate the average performance of the models based on the different, independent folds to obtain a performance estimate that is less sensitive to the sub-partitioning of the training data compared to the holdout method
- Generally use *k*-fold cross-validation for model tuning
- Once we have found satisfactory *hyperparameter* values, we can retrain the model on the complete training set and obtain a final performance estimate using the independent test set

Machine Learning Overview

Standard ML workflow for classification:

1. **Define** the problem
2. **Collect** Data (with target labels)
3. **Explore** the Data (EDA)
4. **Preprocess** the Data
5. **Fit** Model on Training Set
6. **Predict** on Test Set
7. **Evaluate** Model Performance
8. **Validate** the Model
9. **Improve** Model Performance/**Select** the Best Model

Improve Model Performance/Select the Best Model

- Gather more training samples
  - Not always possible
- Merge additional datasets
  - Not always possible
- Tune hyperparameters
  - This is model-specific and will be covered in the following days
- Adjust model complexity & flexibility (bias-variance tradeoff)

Machine Learning Overview Wrap-up

- Let's do some more examples and exercises to reinforce the concepts from the machine learning overview!

HPSI Human Performance Systems, Inc.

Day 2

Intro to Classification Algorithms

HPSI Human Performance Systems, Inc.

Day 2 Outline

- Day 1 Recap
- Intro to Classifiers
- K-Nearest Neighbors
  - Algorithm details
  - Choosing "k"
  - Exercise
  - Pros/Cons
- Probability
  - Probabilistic Classifiers
  - Probability Distributions
  - Exercise
  - Sample Spaces
  - Conditional Probability
  - Bayesian Inference
- Naïve Bayes Classifier
  - Algorithm Details
  - Exercise
  - Pros/Cons
- Logistic Regression
  - Algorithm Details
  - Exercise
- Model Comparison

HPSI Human Performance Systems, Inc.

Day 1 Recap

HPSI Human Performance Systems, Inc.

Classification

- Definition
  - Classification can take two distinct meanings in Machine Learning
    - Unsupervised Learning
      - We may be given a set of observations with the aim of establishing the existence of classes or clusters in the data
    - Supervised Learning
      - We may know for certain that there are so many classes, and the aim is to establish a rule that we can use to classify a new observation into one of the existing classes
  - Today we will be discussing methods and algorithms related to Supervised Classification

HPSI Human Performance Systems, Inc.

Classification

- A few classification examples:
  1. A Person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
  2. An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.
  3. On the basis of DN sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are disease-causing and which are not.

HPSI Human Performance Systems, Inc.

Classification

- There are many possible techniques that a classifier might use to predict a qualitative response. Today we will discuss three of the most widely-used classifiers:
  - k-Nearest Neighbors
  - Naïve Bayes
  - Logistic Regression

HPSI Human Performance Systems, Inc.

Classification

A few issues to keep in mind when building a classifier

- **Accuracy.** There is the reliability of the rule, usually represented by the proportion of correct classifications, although it may be that some errors are more serious than others, and it may be important to control the error rate for some key class.
- **Speed.** In some circumstances, the speed of the classifier is a major issue. A classifier that is 90% accurate may be preferred over one that is 95% accurate if it is 100 times faster in testing (and such differences in time-scales are not uncommon in neural networks for example). Such considerations would be important for the automatic reading of postal codes, or automatic fault detection of items on a production line for example.
- **Comprehensibility.** If it is a human operator that must apply the classification procedure, the procedure must be easily understood else mistakes will be made in applying the rule. It is important also, that human operators believe the system.
- **Training Time.** Especially in a rapidly changing environment, it may be necessary to learn a classification rule quickly, or make adjustments to an existing rule in real time. "Quickly" might imply also that we need only a small number of observations to establish our rule.[4]

HPSI Human Performance Systems, Inc.

K-Nearest Neighbors

HPSI Human Performance Systems, Inc.

Simple approach for k-NN

Simple goal:

- Predict the label of a data point by:
  - Looking at the 'k' closest labeled data points (neighbors)
  - Taking a majority vote
- One of the easiest algorithms to interpret, oftentimes used as a baseline for measuring model performance
- Memory-Based Learning
  - Also known as "case-based" or "example-based" learning
- Intuition behind memory-based learning
  - Similar inputs map to similar outputs
    - If true, we just have to define "similar"
    - Not all similarities created equal…

Memory-Based Learning

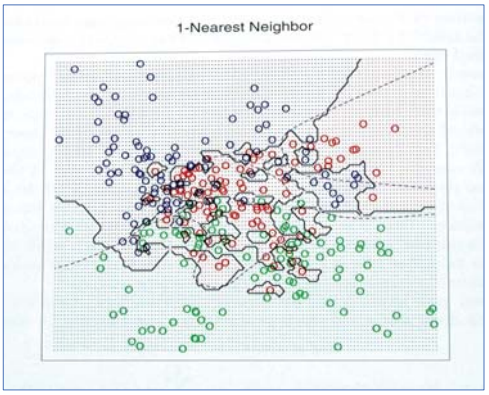
- How do we determine "similar"?
- For instance, if we wanted to:
- Predict Brent's weight
  - Who are the similar people?
  - Similar age, diet, height, waistline, activity level …
- Predict Brent's IQ
  - Similar occupation, writing style, undergraduate degree, SAT score, …
- How would you quantify a comparison for these two?
  - Need some metric…
    - Distance?

k-NN Approach

- Define a distance $d(x_1, x_2)$ between any 2 examples
  - Examples are just feature vectors
  - So we could just use Euclidean distance …
- Training
  - Index the training examples for fast lookup (build a "database")
- Test
  - Given a new x, find the closest neighbor (k=1) from training index
  - Classify x the same as its closest neighbor

kNN Decision Boundaries

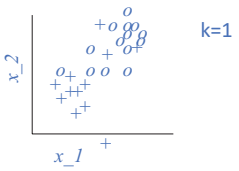
- kNN can learn complex decision boundaries



HPSI Human Performance Systems, Inc.

kNN

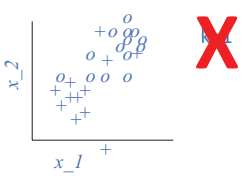
- Instead of picking the (1) nearest neighbor, what if we picked the k-Nearest Neighbors and have them vote?
- Choosing k points is more reliable in the following cases:
  - Noise in training vectors x
  - Noise in training labels y
  - Overlapping classes



HPSI Human Performance Systems, Inc.

kNN

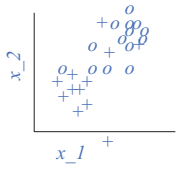
- Instead of picking the (1) nearest neighbor, what if we picked the k-Nearest Neighbors and have them vote?
- Choosing k points is more reliable in the following cases:
  - Noise in training vectors x
  - Noise in training labels y
  - Overlapping classes



HPSI Human Performance Systems, Inc.

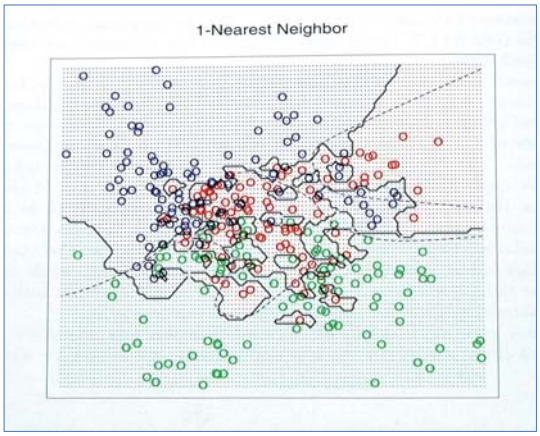
kNN

- Instead of picking the (1) nearest neighbor, what if we picked the k-Nearest Neighbors and have them vote?
- Choosing k points is more reliable in the following cases:
  - Noise in training vectors x
  - Noise in training labels y
  - Overlapping classes
- Why?



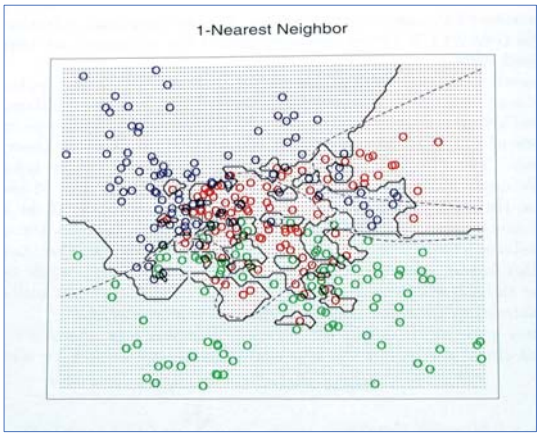
kNN Decision Boundaries

- Consider this example with R,G,B classes with significant overlap



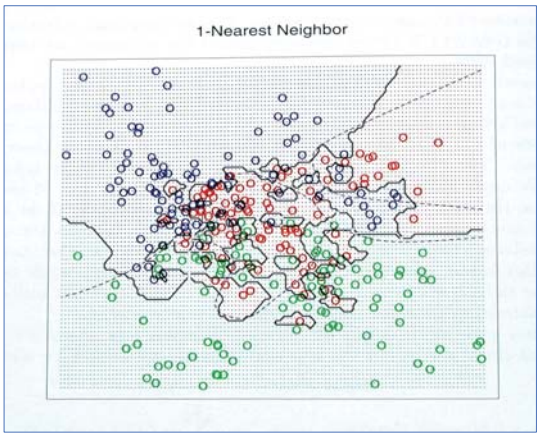
kNN Decision Boundaries

- Consider this example with R,G,B classes with significant overlap
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?



kNN Decision Boundaries

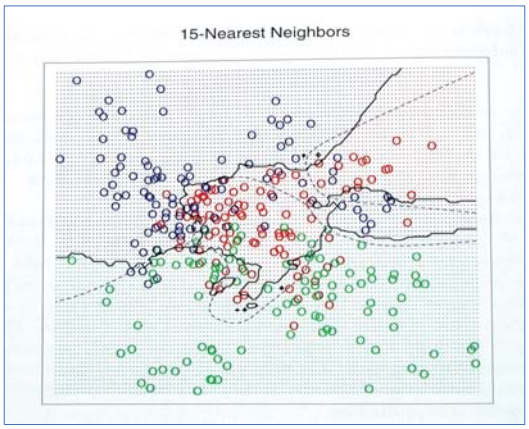
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?
- What if we were to increase k?



HPSI Human Performance Systems, Inc.

kNN Decision Boundaries

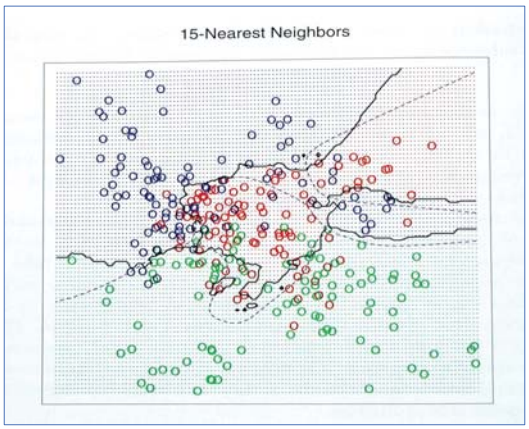
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?
- What if we were to increase k?
  - K=15 Decision boundary



HPSI Human Performance Systems, Inc.

kNN Decision Boundaries

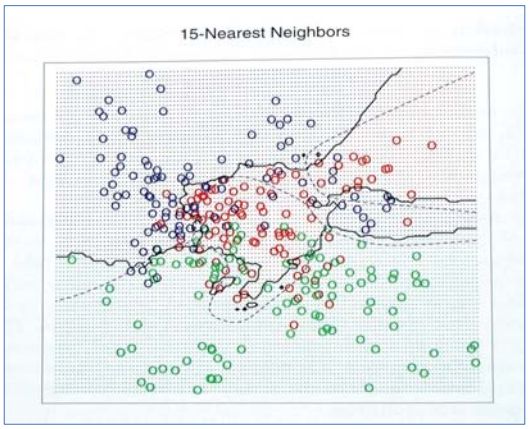
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?
- What if we were to increase k?
  - K=15 Decision boundary
  - Smoother boundaries



HPSI Human Performance Systems, Inc.

kNN Decision Boundaries

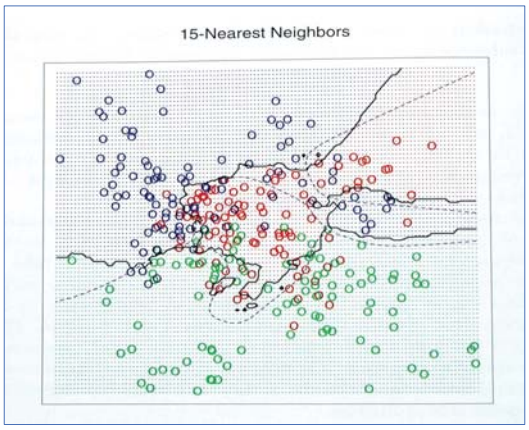
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?
- What if we were to increase k?
  - K=15 Decision boundary
  - Smoother boundaries
  - Generalizes better on unseen data



HPSI Human Performance Systems, Inc.

kNN Decision Boundaries

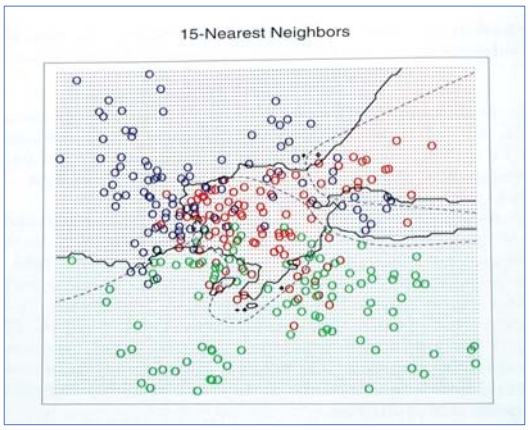
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?
- What if we were to increase k?
  - K=15 Decision boundary
  - Smoother boundaries
  - Generalizes better on unseen data
- What makes the boundaries smoother?



HPSI Human Performance Systems, Inc.

kNN Decision Boundaries

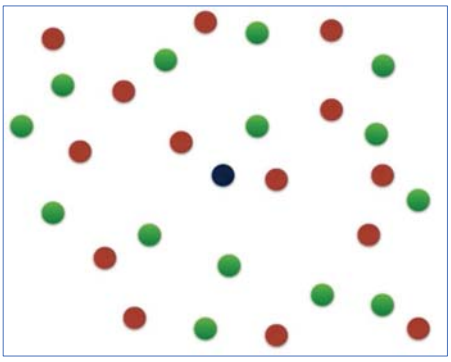
- k=1 Decision Boundary
  - Looks complex
  - Overfitting?
- What if we were to increase k?
  - K=15 Decision boundary
  - Smoother boundaries
  - Generalizes better on unseen data
- What makes the boundaries smoother?
- Let's look at a two-class (binary) example



HPSI Human Performance Systems, Inc.

k-NN Graphical Example

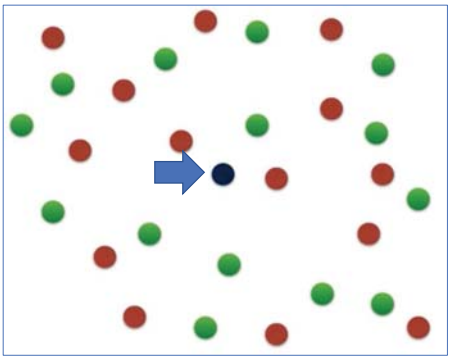
- Consider this two-dimensional dataset with points classified as Red or Green



HPSI Human Performance Systems, Inc.

k-NN Graphical Example

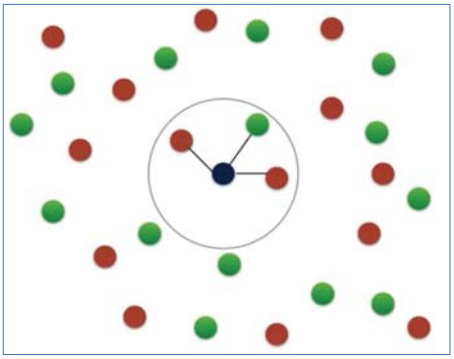
- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point



HPSI Human Performance Systems, Inc.

k-NN Graphical Example

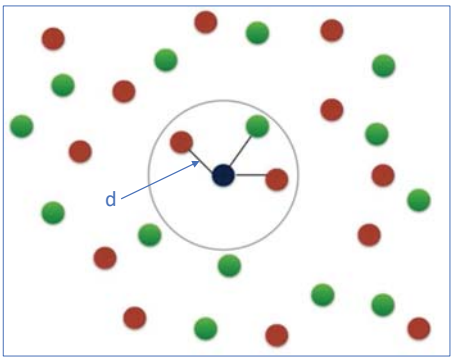
- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors



HPSI Human Performance Systems, Inc.

k-NN Graphical Example

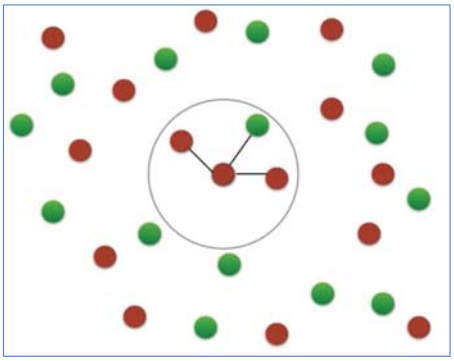
- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors
  - Measured by some distance



HPSI Human Performance Systems, Inc.

k-NN Graphical Example

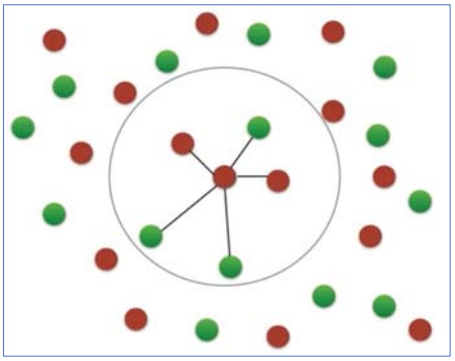
- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors
  - Measured by some distance
  - The point is classified as Red



HPSI Human Performance Systems, Inc.

k-NN Graphical Example

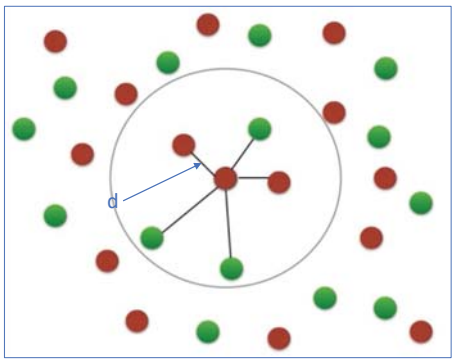
- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors
  - Measured by some distance
  - The point is classified as Red
- If we consider k=5 neighbors



HPSI Human Performance Systems, Inc.

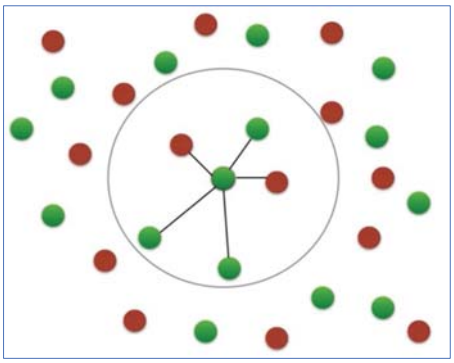
k-NN Graphical Example

- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors
  - Measured by some distance
  - The point is classified as Red
- If we consider k=5 neighbors
  - Measured by some distance



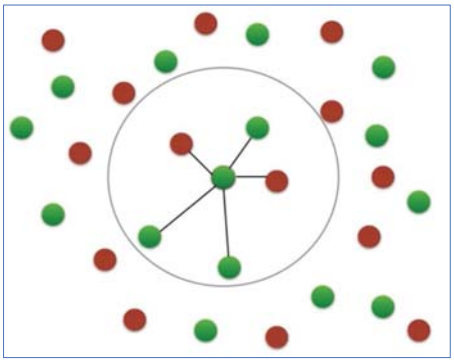
k-NN Graphical Example

- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors
  - Measured by some distance
  - The point is classified as Red
- If we consider k=5 neighbors
  - Measured by some distance
  - The point is classified as Green



k-NN Graphical Example

- Consider this two-dimensional dataset with points classified as Red or Green
- We want to Classify this point
- If we consider k=3 neighbors
  - Measured by some distance
  - The point is classified as Red
- If we consider k=5 neighbors
  - Measured by some distance
  - The point is classified as Green
- So, how do we know what k to choose?



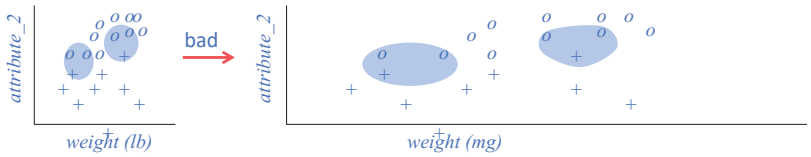
How to choose “k”

- Odd k (often 1, 3, or 5):
  - Avoids problem of breaking ties (in a binary classifier)
- Large k:
  - Less sensitive to noise (particularly class noise)
  - Letter probability estimates for discrete classes
  - Larger training sets allow larger values of k
- Small k:
  - Captures fine structure of problem space better
  - May be necessary with small training sets
- Balance between large and small k

HPSI Human Performance Systems, Inc.

kNN distance problem

- Problem:
  - What if the input represents weight in milligrams?
  - Then small differences in physical weight dimension have a huge effect on distances, overwhelming other features
  - Should really correct for these arbitrary “scaling” issues
    - This leads to Standard Scaling (from yesterday)
    - Rescale weights so that standard deviation = 1



HPSI Human Performance Systems, Inc.

More kNN Details

- Nonparametric - makes no explicit assumptions about the underlying distribution of the input
- Instance/memory-based learning means that this algorithm doesn’t explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as “knowledge” for the prediction phase
- Learns arbitrarily complicated decision boundaries
- Lazy learner - a learning method that generalizes data in the testing phase, rather than during the training phase
  - Don’t do any work until you know what you want to predict
  - A benefit of lazy learning is that it can quickly adapt to changes, since it is not expecting a certain generalized dataset.
  - Very fast training time, but very slow prediction (has to search for the nearest neighbors)

HPSI Human Performance Systems, Inc.

Advantages of k-NN

- Simple and fast to deploy
  - Little to no training time
- Easy to interpret/explain
- Naturally handles multiclass datasets
- Non-parametric
  - Does not assume any probability distributions on the input data

Disadvantages of k-NN

- Storage of model takes a lot of disk space (contains entire training dataset)
- Curse of Dimensionality - often works best with 25 or fewer dimensions
  - There is little difference between the nearest and farthest neighbor in high dimensional data
- Computationally expensive predictions (large search problem to find nearest neighbors)
  - Might be impractical in industry settings
- Need to normalize - suffers from skewed class distributions
  - If one type of category occurs much more than another, classifying an input will be more biased towards that one category (dominates the majority vote since it is more likely to be neighbors with the input)

kNN Exercise

Naïve Bayes

HPSI Human Performance Systems, Inc.

Naïve Bayes

- Naive Bayes is a simple classification technique that relies on conditional probability, and predicts the most probable class given a set of inputs
- It is often used as a baseline for more complex models

HPSI Human Performance Systems, Inc.

A new method: Naïve Bayes

- Naive Bayes is a simple technique for predicting the most probable class/label given a set of features/inputs
  - It is often used as a baseline for more complex models

$$\text{Naive Bayes Classifier: } \underset{Y}{\operatorname{argmax}}\ P(Y|\vec{X}) = \underset{Y}{\operatorname{argmax}}\ P(x_1|Y)P(x_2|Y)...P(x_3|Y)P(Y)$$

$$= \underset{Y}{\operatorname{argmax}}\ P(Y)\prod_{i=1}^{\kappa} P(x_i|Y)$$

How can we unpack this?

HPSI Human Performance Systems, Inc.

Understanding Naïve Bayes Classifiers

- To be able to unpack the Naïve Bayes classifier definition, we need a good grasp on the following topics
  - Random variables
  - Distributions
    - Continuous
    - Discrete
  - Statistical Independence
  - Probability
    - Conditional Probability
    - Joint Probability
    - Marginal Probability

Random Variables

- A random variable is a random number determined by chance, or more formally, drawn according to a probability distribution which specifies the probability that its value falls in any given interval.
- Discrete Random Variable
  - Taking any of a specified finite or countable list of values, endowed with a probability mass function characteristic of the random variable's probability distribution
- Continuous
  - Taking any numerical value in an interval or collection of intervals, via a probability density function that is characteristic of the random variable's probability distribution; or a mixture of both types.

Random Variables

- Why do we care about Random Variables?
- Our goal is to predict the target/class
- We are not given the true (presumably deterministic) function
- We are only given observations
- Uncertainty arises through:
  - Noisy measurements
  - Finite size of data sets
  - Ambiguity: The word “bank” can mean (1) a financial institution, (2) the side of a river,or (3) tilting an airplane. Which meaning was intended, based on the words that appear nearby?
- Probability theory provides a consistent framework for the quantification and manipulation of uncertainty
- Allows us to make optimal predictions given all the information available to us, even though that information may be incomplete or ambiguous

Probability

- Probabilities assign numbers to possibilities
- A probability needs to satisfy three properties (Kolmogorov, 1956):
  - A probability must be nonnegative
  - The sum of the probabilities across all events in the entire sample space must be 1
  - For any two mutually exclusive events, the probability that one or the other occurs is the sum of their individual probabilities
    - For example, the probability that a fair six-sided die comes up 3 OR 4 is 1/6 + 1/6 = 2/6.

Probability Distributions

- A probability distribution is simply a list of all possible events and their corresponding probabilities
- There are two kinds of probability distributions
  - Discrete Distribution:
    - Probability of heads or tails
  - Continuous Distribution:
    - Probabilities of people's heights

Discrete Probability Distribution

- When the sample space consists of discrete outcomes (e.g., heads or tails), the probability distribution is a list of probabilities of the outcomes
- The probability of a discrete outcome is called a **probability mass**
- The sum of the probability masses across the sample space must be 1

Discrete Probability Example

- Example
  - Consider the simple experiment of tossing a coin three times. Let X = number of times the coin comes up heads. The 8 possible elementary events and the corresponding values for X are:

Elementary Event	Count of Heads (X)
TTT	0
TTH	1
THT	1
HTT	1
THH	2
HTH	2
HHT	2
HHH	3

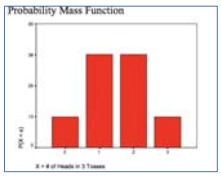
HPSI Human Performance Systems, Inc.

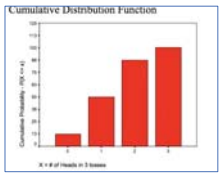
Discrete Probability Example

- Example
  - Therefore, the probability distribution for the number of heads occurring in three coin tosses is

Count of Heads (X)	p(x)	F(x)
0	1/8	1/8
1	3/8	4/8
2	3/8	7/8
3	1/8	1

$$P(x) = \begin{cases} 1/8 & if\ x = 0 \\ 3/8 & if\ x = 1, 2 \\ 1/8 & if\ x = 3 \\ 0 & Otherwise \end{cases}$$





HPSI Human Performance Systems, Inc.

Continuous Probability Distribution

- When the sample space consists of continuous outcomes (ex: people's heights) we cannot use probability mass for a specific outcome.
- Why not?

HPSI Human Performance Systems, Inc.

Continuous Probability Distribution – Probability Density

- When the sample space consists of continuous outcomes (ex: people's heights) we cannot use probability mass for a specific outcome.
- Why not?
  - Because the probability mass for a specific outcome will be zero
  - In other words, the probability of someone's height being exactly 67.2141390842076153…
- Instead, we can:
  - Discretize the space into a finite set of mutually exclusive and exhaustive intervals
  - Calculate the probability mass in each interval
  - Use the ratio of probability mass to interval width
  - This ratio is called the **Probability Density**

HPSI Human Performance Systems, Inc.

Probability Density

- The top panel of this figure shows the discretized intervals and probability mass in each interval
- The second panel shows the probability density
- The third panel shows the narrower intervals and probability mass in each interval
- The bottom panel shows the probability density corresponding to the more narrow intervals
- Generally, the skinnier the intervals are, the more accurate the probability density is

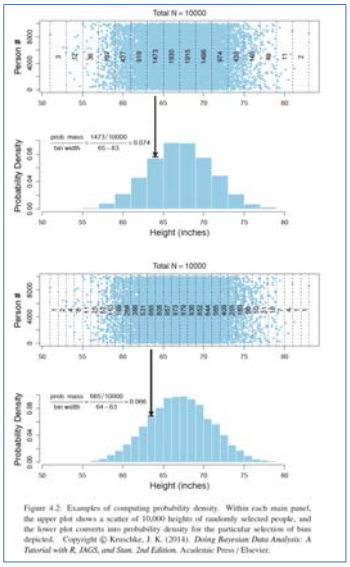


Figure 4.2: Examples of computing probability density. Within each main panel, the upper plot shows a scatter of 10,000 heights of randomly selected people, and the lower plot converts into probability density for the particular selection of bins depicted. Copyright © Kruschke, J. K. (2014). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. 2nd Edition.* Academic Press / Elsevier.

HPSI Human Performance Systems, Inc.

Probability Density

- While probability mass cannot exceed 1, probability densities can
- The upper panel of this figure shows that most of the probability mass is concentrated around 84
- Consequently, the probability density near 84 exceeds 1.0, as shown in the lower panel
- This simply means that there is a high concentration of probability mass relative to the width of the interval

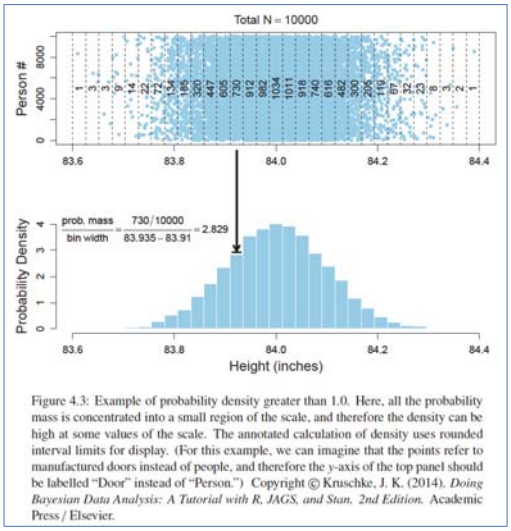


Figure 4.3: Example of probability density greater than 1.0. Here, all the probability mass is concentrated into a small region of the scale, and therefore the density can be high at some values of the scale. The annotated calculation of density uses rounded interval limits for display. (For this example, we can imagine that the points refer to manufactured doors instead of people, and therefore the y-axis of the top panel should be labelled "Door" instead of "Person.") Copyright © Kruschke, J. K. (2014). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. 2nd Edition.* Academic Press / Elsevier.

HPSI Human Performance Systems, Inc.

Properties of Probability Density Functions

- We need to define some notations first
- Let:
  - x be the continuous variable
  - Δx be the width of an interval on x
  - i be an index for the intervals
  - $[x_i, x_i + \Delta x]$ be the interval between x_i and $x_i + \Delta x$
  - $P([x_i, x_i + \Delta x])$be the probability mass of the ith interval
- Then the sum of those probability masses must be 1:

$$\sum_i P([x_i, x_i + \Delta x]) = 1$$

- We can rewrite the equation above in terms of the density of each interval, by dividing and multiplying by x:

$$\sum_i \frac{\Delta x * P([x_i, x_i + \Delta x])}{\Delta x} = 1$$

Properties of Probability Density Functions

- In the limit, as the interval width becomes infinitesimal, we denote:
  - Summation as $\int$ instead of $\sum$
- Then, the previous equation (in terms of density) can be rewritten as:

$$\sum_i \frac{\Delta x * P([x_i, x_i + \Delta x])}{\Delta x} = 1 \Rightarrow \int dx\, p(x) = 1$$

- We use $p(x)$ to represent the probability mass when x is discrete
- Thus, what $p(x)$ represents depends on the context
  - Is x discrete or continuous?

The Normal Probability Density Functions

- Perhaps the most famous probability density function is the normal distribution, also known as the Gaussian distribution
- The probability density function of normal distribution is

$$\mathrm{p}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right)}$$

- Recall, what are σ and μ? what do they control?
- An example of the probability density is shown in the figure where the x axis is divided into a dense comb of small intervals
- The figure also shows that the area under the curve is, in fact, 1

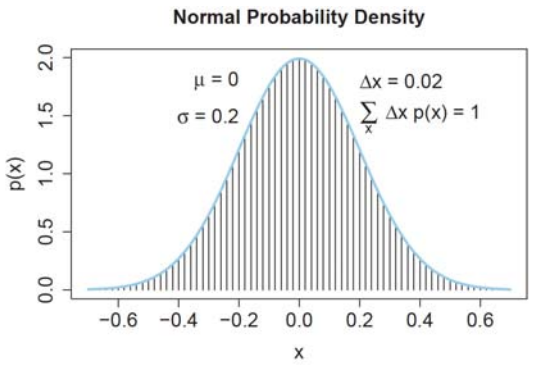
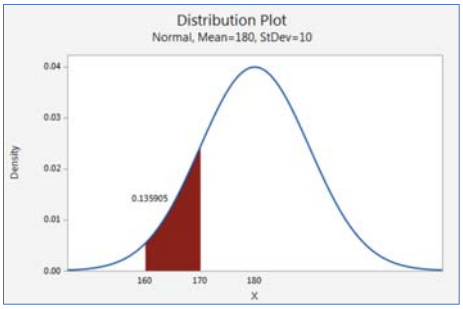


Figure 4.4: A normal probability density function, shown with a comb of narrow intervals. The integral is approximated by summing the width times height of each interval. Copyright © Kruschke, J. K. (2014). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. 2nd Edition.* Academic Press / Elsevier.

Example - Continuous Normal Distribution

- Example of the continuous distribution of weights
  - The continuous normal distribution can describe the distribution of weight of adult males.
  - For example, you can calculate the probability that a man weighs between 160 and 170 pounds.
  - The area of this range is 0.136; therefore, the probability that a randomly selected man weighs between 160 and 170 pounds is 13.6%.
  - The entire area under the curve equals 1.0

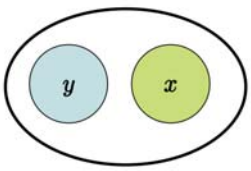
$$\int_{-\infty}^{+\infty} p(x)dx = 1$$



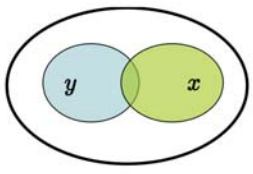
Random Variables/Distributions Exercise

Joint Probability

- Joint Probability
  - Knowing that y occurred reduces the sample space to y
    - The part of y where x also occurred, or the probability of x and y occurring, is:
    - $P(x, y) = P(x \cap y)$
  - Order does not matter:
    - $P(x, y) = P(y, x)$



- Disjoint Sets
- Mutually Exclusive Events
- $x \cap y = \emptyset$



- Intersecting sets

Joint Probability and Marginal Probability

- This table shows the probabilities of various combinations of people's eye/hair color
- Each entry indicates the **joint probability** of particular combinations of eye color (e) and hair color (h), denoted by $p(e, h)$
- The right margin of the table shows the probabilities of the eye colors overall, collapsed across hair colors
- Such probabilities are called **marginal probability**, denoted by $p(e)$:

$$p(e) = \sum_{h} p(e, h)$$

- The marginal probabilities of the hair colors, $p(h)$, are indicated on the lower margin of the table:

$$p(h) = \sum_{e} p(e, h)$$

Table 4.1: Proportions of combinations of hair color and eye color. Some rows or columns may not sum exactly to their displayed marginals because of rounding error from the original data. Data adapted from Snee (1974). Copyright © Kruschke, J. K. (2014). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. 2nd Edition.* Academic Press / Elsevier.

Eye Color	Hair Color: Black	Brunette	Red	Blond	Marginal (Eye Color)
Brown	.11	.20	.04	.01	.37
Blue	.03	.14	.03	.16	.36
Hazel	.03	.09	.02	.02	.16
Green	.01	.05	.02	.03	.11
Marginal (Hair Color)	.18	.48	.12	.21	1.0

HPSI Human Performance Systems, Inc.

Conditional Probability

- Conditional Probability
  - $\mathrm{P}(x|y)$ is the probability of the occurrence of event x, given that y occurred is given as:
    - $\mathrm{P}(x|y) = \frac{P(x \cap y)}{P(y)} = \frac{P(x,y)}{P(y)}$
  - Answers the question:
    - How does the probability of an event change if we have extra information?

Table 4.2: Example of conditional probability. Of the blue-eyed people in Table 4.1, what proportion have hair color h? Each cell shows $p(h|\text{blue}) = p(\text{blue}, h)/p(\text{blue})$ rounded to two decimal points. Copyright © Kruschke, J. K. (2014). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. 2nd Edition.* Academic Press / Elsevier.

Eye Color	Hair Color: Black	Brunette	Red	Blond	Marginal (Eye Color)
Blue	.03/.36 = .08	.14/.36 = .39	.03/.36 = .08	.16/.36 = .45	.36/.36 = 1.0

HPSI Human Performance Systems, Inc.

Conditional Probability – Order Matters

- $\mathrm{P}(x|y) \neq \mathrm{P}(y|x)$
  - Why?
- $\mathrm{P}(cute|puppy) \neq \mathrm{P}(puppy|cute)$

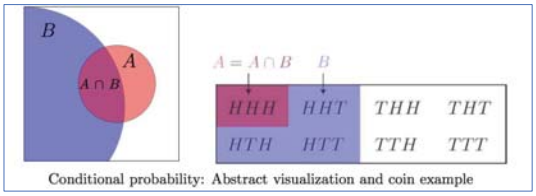
HPSI Human Performance Systems, Inc.

Conditional Probability Example

- Coin Toss Example:
  - Toss a fair coin 3 times
  - What is the probability of 3 heads?
    - Answer:
      - $Sample\ Space = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}$
      - All outcomes are equally likely (if the coin is fair)
      - $P(HHH) = \frac{1}{8}$
  - Suppose we are told that the first toss was heads
  - Given this information, how should we compute the probability of {HHH}?
    - Answer:
      - We have a new (reduced) $Sample\ Space = \{HHH, HHT, HTH, HTT\}$
      - All outcomes are still equally likely (the coin is still fair)
      - $P(HHH) = \frac{1}{4}$

Conditional Probability Example

- We can visualize the conditional probability as follows
  - Think of $P(A)$ as the proportion of the area of the whole sample space taken up by A
  - For $P(A|B)$ we restrict our attention to B
  - $P(A|B)$ is the proportion of B taken up by A
    - $P(A|B) = \frac{P(A \cap B)}{P(B)}$



Conditional probability: Abstract visualization and coin example

Statistical Independence

- Independent Events
  - If x and y are independent then they are unconnected and not related to each other
  - We have:
    $P(x|y) = P(x)$
  - From there it follows that
    $P(x, y) = P(x) * P(y)$
  - In other words, knowing that y occurred does not change the probability that x occurs (and vice versa)
  - Examples of absolute independence include:
    - Eye color and height
    - Hair color and weight

Statistical Independence Example

- Independent Events
  - If we want to calculate the joint probability of two independent events, we can simply multiply each probability together to get the joint probability
  - "Joint Distribution" = "Product Distribution"
  - $\mathrm{P}(x, y) = P(x) * P(y)$
- For Example:
  - Probability of tossing a coin and getting "Heads":

    $P(Heads) = P(x) = \frac{1}{2}$
  - Probability of rolling a dice and getting "3":

    $P(Roll\ \text{"3"}) = P(y) = \frac{1}{6}$

    $P(Heads) * P(Roll\ \text{"3"}) = \mathrm{P}(x, y) = \frac{1}{2} * \frac{1}{6} = \frac{1}{12}$

Bayesian Inference

- The Naïve Bayes Classifier relies on three things:

1. Independence assumption
2. The notion of conditional probability
3. **Bayesian Inference -** a method of statistical inference in which Bayes Theorem is used to update the probability for a hypothesis as more evidence becomes available.

Bayesian Inference Example

- We observe that the sidewalk is wet
- What are the possible causes?

Bayesian Inference Example

- We observe that the sidewalk is wet
- What are the possible causes?
  - It rained recently
  - Sprinkler
  - Broken water main (pipe)
  - Person spilled a drink
  - Dog marked his territory

Bayesian Inference Example

- Based on information that we have, we have some notion that certain probabilities are greater than others
- For example:
  - P(recent rain) > P(sprinkler)
  - P(recent rain) > P(spilled drink)
- Bayesian inference incorporates previous knowledge (prior probabilities)

Bayesian Inference Example

- Observation A:
  - Suppose we *observe* that the sidewalk is wet, in addition to the grass, the trees, the street, and the parked cars
  - How do the probabilities change given this new information?

Bayesian Inference Example

- Observation A:
  - Suppose we *observe* that the sidewalk is wet, in addition to the grass, the trees, the street, and the parked cars
  - How do the probabilities change given this new information?
  - P(recent rain | Observation A)

HPSI Human Performance Systems, Inc.

Bayesian Inference Example

- Observation A:
  - Suppose we *observe* that the sidewalk is wet, in addition to the grass, the trees, the street, and the parked cars
  - How do the probabilities change given this new information?
  - P(recent rain | Observation A) ↑

HPSI Human Performance Systems, Inc.

Bayesian Inference Example

- Observation A:
  - Suppose we *observe* that the sidewalk is wet, in addition to the grass, the trees, the street, and the parked cars
  - How do the probabilities change given this new information?
  - P(recent rain | Observation A) ↑
  - P(spilled drink | Observation A)

HPSI Human Performance Systems, Inc.

Bayesian Inference Example

- Observation A:
  - Suppose we *observe* that the sidewalk is wet, in addition to the grass, the trees, the street, and the parked cars
  - How do the probabilities change given this new information?
  - P(recent rain | Observation A) ↑
  - P(spilled drink | Observation A) ↓

Bayesian Inference Example

- Observation B:
  - Now suppose that instead we *observe* that the sidewalk is wet, but it is localized to a small area next to an empty water bottle
  - How do the probabilities change given this new information?

Bayesian Inference Example

- Observation B:
  - Now suppose that instead we *observe* that the sidewalk is wet, but it is localized to a small area next to an empty water bottle
  - How do the probabilities change given this new information?
  - P(recent rain | Observation B)

Bayesian Inference Example

- Observation B:
  - Now suppose that instead we *observe* that the sidewalk is wet, but it is localized to a small area next to an empty water bottle
  - How do the probabilities change given this new information?
  - P(recent rain | Observation B) ↓

HPSI Human Performance Systems, Inc.

Bayesian Inference Example

- Observation B:
  - Now suppose that instead we *observe* that the sidewalk is wet, but it is localized to a small area next to an empty water bottle
  - How do the probabilities change given this new information?
  - P(recent rain | Observation B) ↓
  - P(spilled drink | Observation B)

HPSI Human Performance Systems, Inc.

Bayesian Inference Example

- Observation B:
  - Now suppose that instead we *observe* that the sidewalk is wet, but it is localized to a small area next to an empty water bottle
  - How do the probabilities change given this new information?
  - P(recent rain | Observation B) ↓
  - P(spilled drink | Observation B) ↑

HPSI Human Performance Systems, Inc.

Applying Bayesian Inference to parameters/data

- Naïve Bayes is a machine learning method that can be used to predict the likelihood that an event will occur given evidence that is present in your data
- The Naïve Bayes Classifier relies on Bayesian inference at its core
- Makes two "Naïve" assumptions over attributes

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Assumptions

- Fundamental assumption:
  - Each feature makes an **independent** and **equal** contribution to the outcome
    - We assume that no pair of features are dependent on one another in any way (complete independence)
      - Temperature has nothing to do with humidity, and has no effect on whether or not it is windy
    - Each feature is given the same weight/importance
      - We assume that none of the attributes is irrelevant, and that they are all contributing equally to the outcome.

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier

- To illustrate the inner workings of the Naïve Bayes Classifier, we will consider an example:
  - We have recorded weather features about the last 14 times that we played golf.
  - We also recorded the result of whether the conditions were favorable or not.
  - We will demonstrate how a Naïve Bayes Classifier can be used to determine whether or not the a specific set of weather conditions supports playing golf.

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- Weather Independent Variables:
  - Outlook
    - (Rainy, Overcast, Sunny)
  - Temperature
    - (Hot, Mild, Cool)
  - Humidity
    - (High, Normal)
  - Windy
    - (False, True)
- Weather Dependent Variable:
  - Play Golf
    - (YES, NO)

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier

How does the Naïve Bayes Classifier work?

- Step 1:
  - Convert the data set into frequency tables
  - Use the frequency tables to calculate likelihood tables
- Step 2:
  - Use the product rule to obtain a joint conditional probability for the attributes
- Step 3:
  - Use Bayes Rule to calculate the posterior probability for each class variable
  - Once this has been done for all classes, output the class with the highest probability

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- Step 1:
  - Create a frequency table for the Class:
    - P(C) or P(CLASS)
    - P(YES)

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- Step 1:
  - Create a frequency table for the Class:
    - P(C) or P(CLASS)
    - P(YES) = 9/14

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- Step 1:
  - Create a frequency table for the Class:
    - P(C) or P(CLASS)
    - P(YES) = 9/14
    - P(NO) = 5/14

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- Step 1:
  - Create a frequency table for the Class:
    - P(C) or P(CLASS)
    - P(YES) = 9/14
    - P(NO) = 5/14
- Class Frequency Table:

Play	
YES	9
NO	5
Total	**14**

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier

How does the Naïve Bayes Classifier work?

- Step 1:
  - Convert the data set into frequency tables
  - Use the frequency tables to calculate likelihood tables
- Step 2:
  - Use the product rule to obtain a joint conditional probability for the attributes
- Step 3:
  - Use Bayes Rule to calculate the posterior probability for each class variable
  - Once this has been done for all classes, output the class with the highest probability

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- Step 1:
  - P(C) or P(CLASS)
  - P(YES) = 9/14 = 0.643
  - P(NO) = 5/14 = 0.357
- The Class Frequency Table can be used to create the Class Likelihood Table by dividing each class frequency by the total (relative probability)
- Likelihood Table:

Play		P(YES)
YES	9	9/14 = 0.643
NO	5	5/14 = 0.357
Total	**14**	**14/14 = 100%**

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

- The next step is to repeat this process for each weather feature and compute their corresponding Feature Frequency Tables
- These Frequency tables can then be used to create Likelihood tables as previously demonstrated for each feature (weather condition) in our dataset

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

➢**Outlook** Frequency and Likelihood

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

Outlook				
	YES	NO	P(YES)	P(NO)
Sunny	2	3	2/9 = 0.222	3/5 = 0.6
Overcast	4	0	4/9 = 0.444	0/5 = 0
Rainy	3	2	3/9 = 0.33	2/5 = 0.4
Total	9	5	9/9 = 1	5/5 = 1

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

➢**Temperature** Frequency and Likelihood

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

Temperature				
	YES	NO	P(YES)	P(NO)
Hot	2	2	2/9 = 0.22	2/5 = 0.4
Mild	4	2	4/9 = 0.44	2/5 = 0.4
Cool	3	1	3/9 = 0.33	1/5 = 0.2
Total	9	5	9/9 = 1	5/5 = 1

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

➢**Humidity** Frequency and Likelihood

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

Humidity				
	YES	NO	P(YES)	P(NO)
High	3	4	3/9 = 0.33	4/5 = 0.8
Normal	6	1	6/9 = 0.66	1/5 = 0.2
Total	9	5	9/9 = 1	5/5 = 1

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Example

➢**Windy** Frequency and Likelihood

	outlook	temperature	humidity	windy	play_golf
0	Rainy	Hot	High	False	NO
1	Rainy	Hot	High	True	NO
2	Overcast	Hot	High	False	YES
3	Sunny	Mild	High	False	YES
4	Sunny	Cool	Normal	False	YES
5	Sunny	Cool	Normal	True	NO
6	Overcast	Cool	Normal	True	YES
7	Rainy	Mild	High	False	NO
8	Rainy	Cool	Normal	False	YES
9	Sunny	Mild	Normal	False	YES
10	Rainy	Mild	Normal	True	YES
11	Overcast	Mild	High	True	YES
12	Overcast	Hot	Normal	False	YES
13	Sunny	Mild	High	True	NO

Windy				
	YES	NO	P(YES)	P(NO)
FALSE	6	2	6/9 = 0.66	2/5 = 0.4
TRUE	3	3	3/9 = 0.33	3/5 = 0.6
Total	9	5	9/9 = 1	5/5 = 1

Bayes Rule applied to data

➢Bayes rule is merely the mathematical relation between the prior allocations of credibility and the posterior reallocation of credibility (conditional on data)

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Bayes Rule Explained

Posterior Probability of class (C) given predictor (X)

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Bayes Rule Explained

Posterior Probability of class (C) given predictor (X)

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

This represents the probability of C being true, provided X is true

Bayes Rule Explained

Likelihood - the conditional probability of the predictor-given the class

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Bayes Rule Explained

Likelihood - the conditional probability of the predictor-given the class

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

This represents the probability of X being true provided C is true

Bayes Rule Explained

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Prior Probability of the Class

Bayes Rule Explained

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Prior Probability of the Class

This represents the observed probability of the class out of all the observations

Bayes Rule Explained

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Prior Probability of the Predictor

Bayes Rule Explained

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Prior Probability of the Predictor

This represents the observed probability of the predictor out of all the observations

Naïve Bayes Classifier

How does the Naïve Bayes Classifier work?

- Step 1:
  - Convert the data set into frequency tables
  - Use the frequency tables to calculate likelihood tables
- Step 2:
  - Use the product rule to obtain a joint conditional probability for the attributes
- Step 3:
  - Use Bayes Rule to calculate the posterior probability for each class variable
  - Once this has been done for all classes, output the class with the highest probability

Bayes Rule Explained

Prior Probability of the Predictor can be estimated directly by multiplying the individual relative frequencies of each predictor due to the naive independence assumption

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Bayes Rule Explained

Prior Probability of the Predictor can be estimated directly by multiplying the individual relative frequencies of each predictor (due to the naive independence assumption)

Key Idea: the "naive assumption" allows us to multiply the probabilities

$$P(C|X) = \frac{[P(X_1|C) * P(X_2|C) * P(X_3|C) \dots * P(X_n|C)] * P(C)}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we should golf or not

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

$$P(C|X) = \frac{[P(X_1|C) * P(X_2|C) * P(X_3|C) \dots * P(X_n|C)] * P(C)}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we should golf or not

Outlook	Temperature	Humidity	Windy?	Play?
X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE	YES

$$P(YES|X) = \frac{[P(Sunny|YES) * P(Cool|YES) * P(High|YES) * P(TRUE|YES)] * P(YES)}{P(X)}$$

Naïve Bayes Classifier Example

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

- Outlook = Sunny
  - P(Sunny|YES)=0.22
  - P(Sunny|NO)=0.6

Outlook				
	YES	NO	P(YES)	P(NO)
Sunny	2	3	**2/9 = 0.222**	**3/5 = 0.6**
Overcast	4	0	4/9 = 0.444	0/5 = 0
Rainy	3	2	3/9 = 0.33	2/5 = 0.4
Total	9	5	9/9 = 1	5/5 = 1

Naïve Bayes Classifier Example

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

- Temperature = Cool
  - P(Cool|YES) = 0.33
  - P(Cool|NO) = 0.2

Temperature				
	YES	NO	P(YES)	P(NO)
Hot	2	2	2/9 = 0.22	2/5 = 0.4
Mild	4	2	4/9 = 0.44	2/5 = 0.4
Cool	3	1	**3/9 = 0.33**	**1/5 = 0.2**
Total	9	5	9/9 = 1	5/5 = 1

Naïve Bayes Classifier Example

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

- Humidity = High
  - P(High|YES) = 0.33
  - P(High|NO) = 0.8

Humidity				
	YES	NO	P(YES)	P(NO)
High	3	4	**3/9 = 0.33**	**4/5 = 0.8**
Normal	6	1	6/9 = 0.66	1/5 = 0.2
Total	9	5	9/9 = 1	5/5 = 1

Naïve Bayes Classifier Example

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

- Windy = True
  - P(TRUE|YES) = 0.33
  - P(TRUE|NO) = 0.6

Windy				
	YES	NO	P(YES)	P(NO)
FALSE	6	2	6/9 = 0.66	2/5 = 0.4
TRUE	3	3	**3/9 = 0.33**	**3/5 = 0.6**
Total	9	5	9/9 = 1	5/5 = 1

Naïve Bayes Classifier Example

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

- Play = YES
  - P(C) = P(YES) = 0.64
  - P(C) = P(NO) = 0.36

Play		P(YES)
YES	9	**9/14 = 0.64**
NO	5	**5/14 = 0.36**
Total	14	14/14 = 100%

Naïve Bayes Classifier

How does the Naïve Bayes Classifier work?

- Step 1:
  - Convert the data set into frequency tables
  - Use the frequency tables to calculate likelihood tables
- Step 2:
  - Use the product rule to obtain a joint conditional probability for the attributes
- Step 3:
  - Use Bayes Rule to calculate the posterior probability for each class variable
  - Once this has been done for all classes, output the class with the highest probability

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

$$P(YES|X) = \frac{[P(Sunny|YES) * P(Cool|YES) * P(High|YES) * P(TRUE|YES)] * P(YES)}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

$$P(YES|X) = \frac{[0.22 * 0.33 * 0.33 * 0.33] * 0.64}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

$$P(YES|X) = \frac{0.0053}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

$$P(NO|X) = \frac{[P(Sunny|NO) * P(Cool|NO) * P(High|NO) * P(TRUE|NO)] * P(NO)}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

$$P(NO|X) = \frac{[0.6 * 0.2 * 0.8 * 0.6] * 0.36}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

$$P(NO|X) = \frac{0.0206}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

Likelihood of playing golf under these conditions:

$$P(YES|X) = \frac{0.0053}{P(X)}$$

Likelihood of *NOT* playing golf under these conditions:

$$P(NO|X) = \frac{0.0206}{P(X)}$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

Likelihood of playing golf under these conditions:

$$P(YES|X) = \frac{0.0053}{P(X)}$$

Likelihood of *NOT* playing golf under these conditions:

$$P(NO|X) = \frac{0.0206}{P(X)}$$

Probability of Play = YES:

$$P(YES|X) = \frac{0.0053}{(0.0053 + 0.0206)} = 20.5\%$$

Probability of Play = NO:

$$P(NO|X) = \frac{0.0206}{(0.0053 + 0.0206)} = 79.5\%$$

Naïve Bayes Classifier Example

Lets consider the following case, and try to predict whether we can golf or not

	Outlook	Temperature	Humidity	Windy?
X	X_1= Sunny	X_2 = Cool	X_3 = High	X_4 = TRUE
P(X|YES)	0.22	0.33	0.33	0.33
P(X|NO)	0.6	0.2	0.8	0.6

	Play?
C	YES
P(YES)	0.64
P(NO)	0.36

Likelihood of playing golf under these conditions:

$$P(YES|X) = \frac{0.0053}{P(X)}$$

Likelihood of *NOT* playing golf under these conditions:

$$P(NO|X) = \frac{0.0206}{P(X)}$$

Probability of Play = YES:

$$P(YES|X) = \frac{0.0053}{(0.0053 + 0.0206)} = 20.5\%$$

Probability of Play = NO:

$$P(NO|X) = \frac{0.0206}{(0.0053 + 0.0206)} = 79.5\%$$

Naïve Bayes Classifier Results

- Probability of Play = YES
  - 0.0053/(0.0053 + 0.0206) = 20.5%

- Probability of Play = NO
  - 0.0206 / (0.0053 + 0.0206) = 79.5%

HPSI Human Performance Systems, Inc.

Naïve Bayes Classifier Results

The Naïve Bayes Classifier predicts that we should **not** play golf under the conditions in question

Outlook	Temperature	Humidity	Windy?	Play?
Sunny	Cool	High	TRUE	YES

HPSI Human Performance Systems, Inc.

Naïve Bayes Exercise

HPSI Human Performance Systems, Inc.

Properties of Bayes Classifiers

- Incrementality:
  - With each training example, the prior and the likelihood can be updated dynamically (Flexible)
  - Combines prior knowledge and observed data:
  - Prior probability of a hypothesis multiplied with the probability of the hypothesis given the training data
- Probabilistic hypotheses:
  - Outputs not only a classification, but a probability distribution over all classes
- Performs well with multi-class prediction
- Meta-classification:
  - The outputs of several classifiers can easily be combined (ensembled)
    - E.g. by multiplying the probabilities that all classifiers predict for a given class

Improving Naïve Bayes

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques "Laplace Correction" to predict the class of test data set
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance
- Not about tuning parameters
  - Naive Bayes classifiers have few parameters to tune
  - Preprocessing is more effective

Logistic Regression

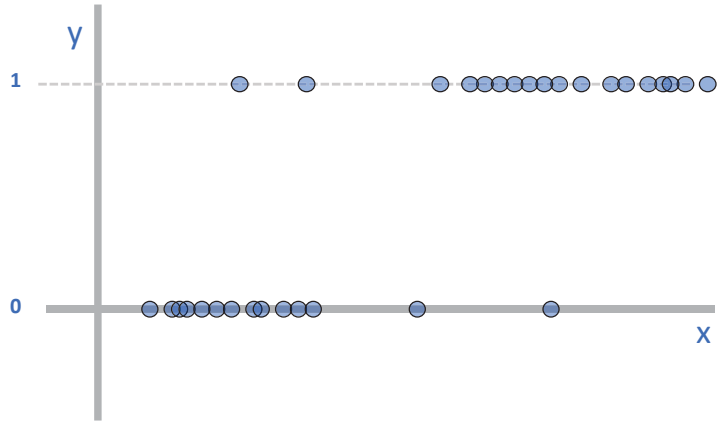
Logistic Regression

- When we think of linear regression, we usually think of predicting a numeric quantity
- Logistic Regression is somewhat misleading in that it serves primarily as a binary classification model
- Categorical response (y) with 2 levels (binary: 0 and 1)
  - Passing or failing a test
  - Surviving a plane crash or not
  - Hospitalisation required or not
  - Diagnosis of diabetes (yes / no)
  - Labelling (over/under some threshold)
- Predictor variables (x_i) can take on any form: binary, categorical, and/or continuous

HPSI Human Performance Systems, Inc.

Logistic Regression Classification

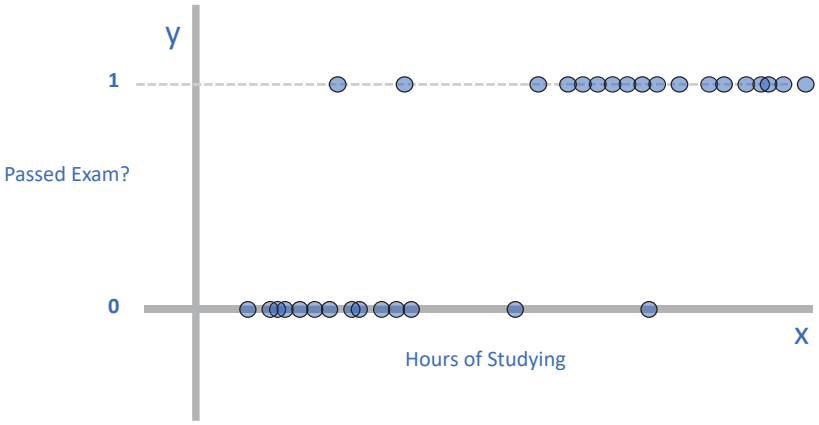
Consider this set of binary data



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

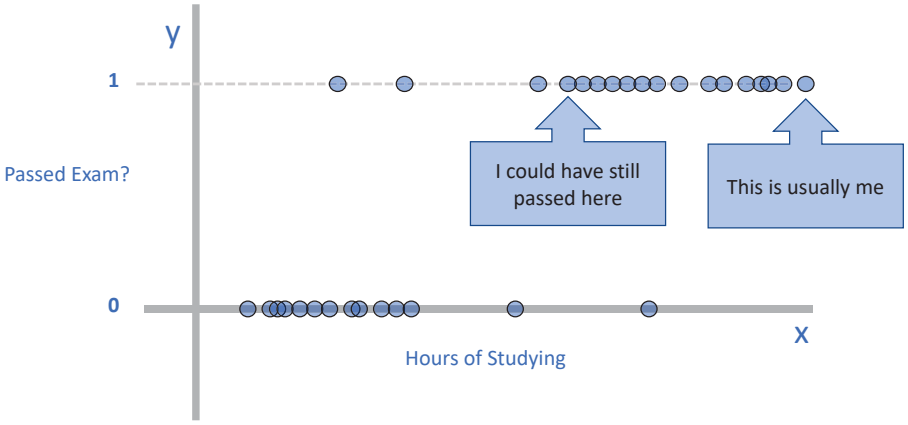
Consider this set of binary data



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

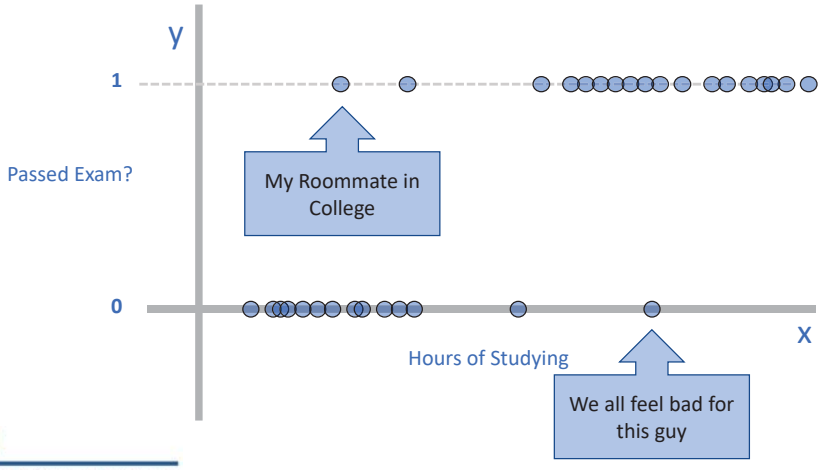
Consider this set of binary data



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

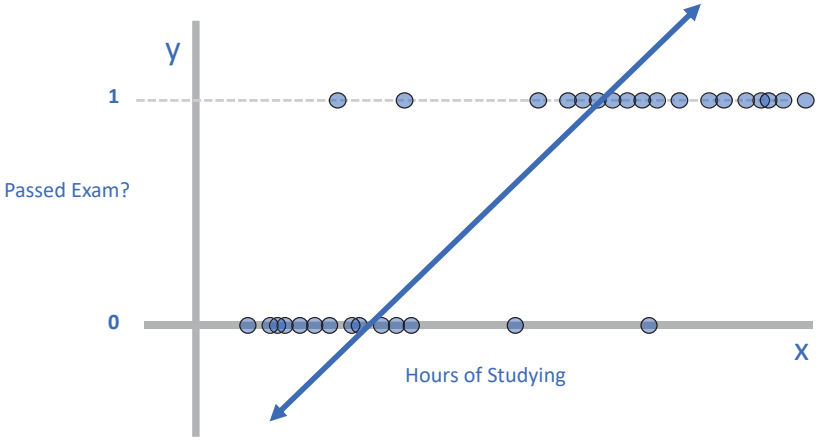
Consider this set of binary data



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

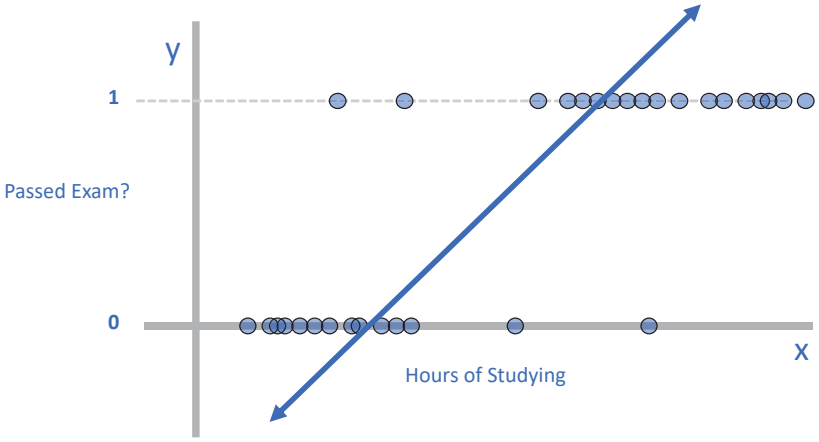
➢Linear Model? – Aside from being binary, there's nothing special about (y)



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

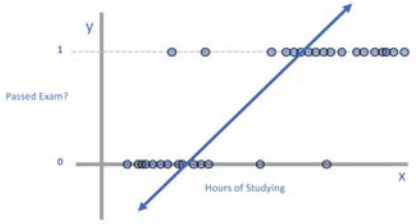
- The value of "Passed Exam" is higher if a student studies more



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

- Linear Model
  - $Pass = \beta_0 + \beta_1 hours\ of\ studying + \varepsilon$
- Problem
  - We want to see what makes the dependent variable change from a 0 to a 1
  - This can also be interpreted as what increases the likelihood of passing, or $P(pass = 1)$ which we can simply denote as p.
  - We should then be able to re-write the linear model as
  - $P(pass = 1) = p = \beta_0 + \beta_1 hours\ of\ studying + \varepsilon$
  - Every additional hour of studying increases the probability of passing by $x\%$

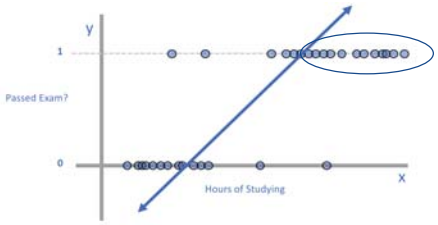


HPSI Human Performance Systems, Inc.

Logistic Regression Classification

$$P(pass = 1) = \beta_0 + \beta_1 hours\ of\ studying + \varepsilon$$

- Recall
  - Probabilities are bounded by $0 \leq p \leq 1$
  - If we were to look at students who studied this many hours, our model would predict a probability greater than 1

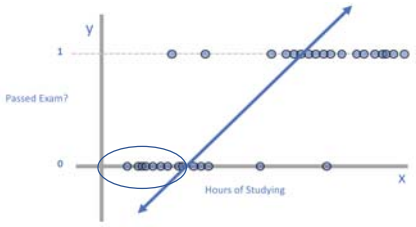


HPSI Human Performance Systems, Inc.

Logistic Regression Classification

$$P(pass = 1) = \beta_0 + \beta_1 hours\ of\ studying + \varepsilon$$

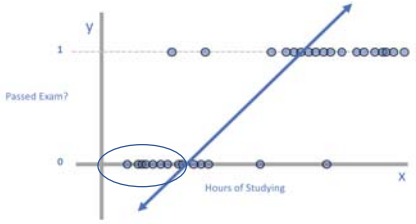
- Recall
  - Probabilities are bounded by $0 \leq p \leq 1$
  - If we were to look at students who studied this many hours, our model would predict a probability greater than 1
  - Likewise, here our model would predict a probability less than 0



Logistic Regression Classification

$$P(pass = 1) = \beta_0 + \beta_1 hours\ of\ studying + \varepsilon$$

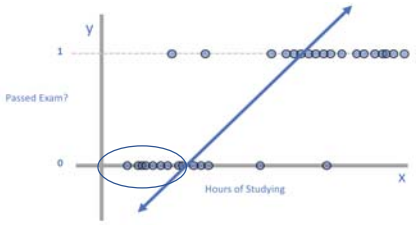
- Recall
  - Probabilities are bounded by $0 \leq p \leq 1$
  - If we were to look at students who studied this many hours, our model would predict a probability greater than 1
  - Likewise, here our model would predict a probability less than 0
- In addition to violating the laws of probability, our model would not maintain normally distributed residuals (which is another requirement of a linear regression model)



Logistic Regression Classification

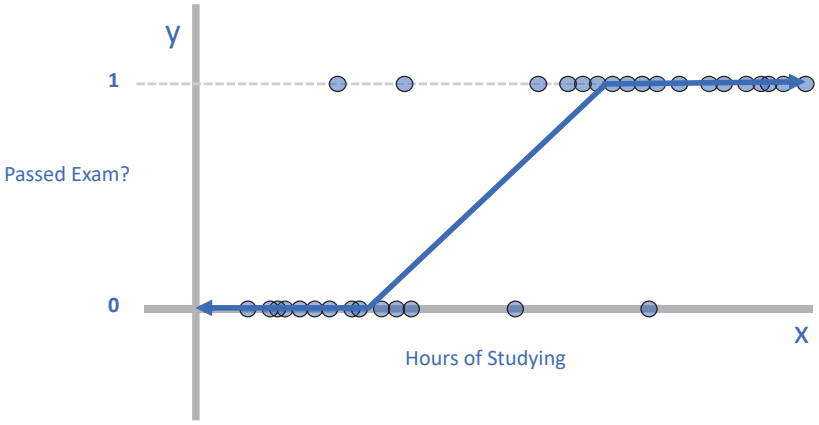
$$P(pass = 1) = \beta_0 + \beta_1 hours\ of\ studying + \varepsilon$$

- Recall
  - Probabilities are bounded by $0 \leq p \leq 1$
  - If we were to look at students who studied this many hours, our model would predict a probability greater than 1
  - Likewise, here our model would predict a probability less than 0
- In addition to violating the laws of probability, our model would not maintain normally distributed residuals (which is another requirement of a linear regression model)
- What can we do to fix this?



Logistic Regression Classification

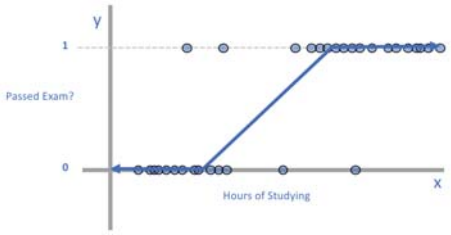
- We could cap the probabilities at 0 and 1



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

- Fixing the prior approach
  - We need to somehow constrain p such that $0 \leq p \leq 1$
  - We know $P(pass) = f(hours\ of\ studying)$ but the linear function didn't work
  - Let's try to develop a new function $f(hours\ of\ studying)$ that satisfies this criteria



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

- Two Steps
  1. It must always be positive (since $0 \leq p(pass)$)
     - $|x|$?
     - x^2 ?
     - What about $p(pass) = e^{\beta_0+\beta_1*hours\ of\ studying}$?
     - This works, but there are times when it would be greater than 1
  2. It must always be less than 1 ($p(pass) \leq 1$)
     - If you think about proportions, any number that is divided by a number slightly greater than it will give us a number smaller than 1
     - What if we just add 1 to the denominator?
     - $p(pass) = \frac{(e^{\beta_0+\beta_1*hours\ of\ studying})}{(e^{\beta_0+\beta_1*hours\ of\ studying})+1}$
     - Note that we could have added any small number (ε) and this condition would have been met, but we use 1 for reasons that will become clear shortly

HPSI Human Performance Systems, Inc.

Logistic Regression Classification

➢The previous expression:

$$p(pass) = p = \frac{(e^{\beta_0+\beta_1*hours\ of\ studying})}{(e^{\beta_0+\beta_1*hours\ of\ studying}) + 1}$$

➢After applying some algebra, can be re-written as:

$$\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 * hours\ of\ studying$$

Logistic Regression Classification

$$\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 * hours\ of\ studying$$

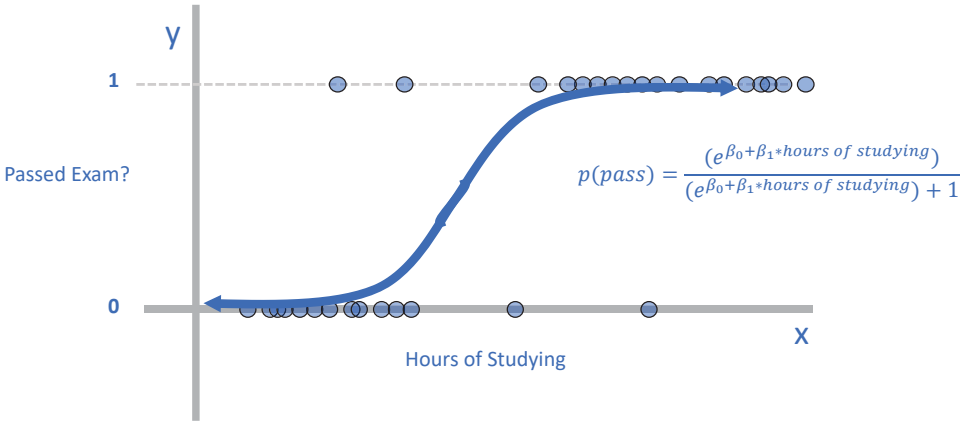
➢Does this look familiar?
- ❖Yes!
- ❖It's in the form of a standard linear model

➢So, even though the probability of a student passing is not a linear function of study-hours, the simple transformation is a linear function of study-hours

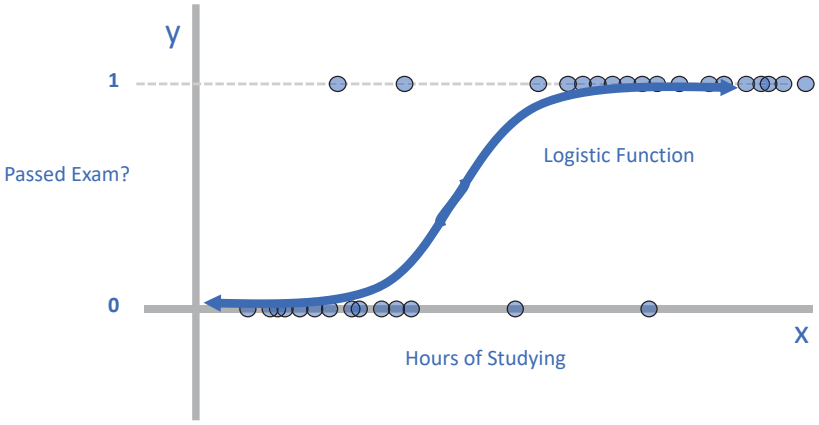
➢This is the equation used in logistic regression

Logistic Regression Classification

Logistic Regression

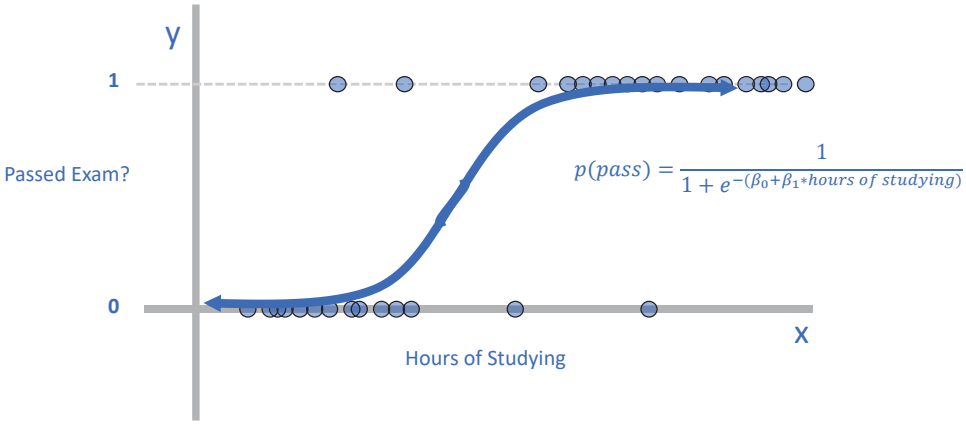


Logistic Regression



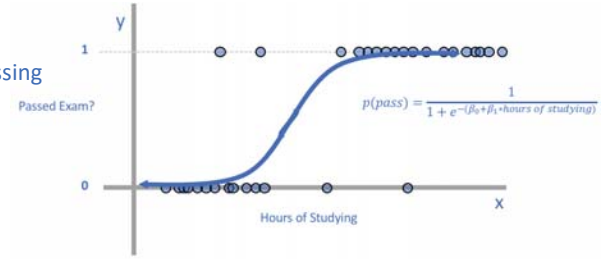
HPSI Human Performance Systems, Inc.

Logistic Regression



HPSI Human Performance Systems, Inc.

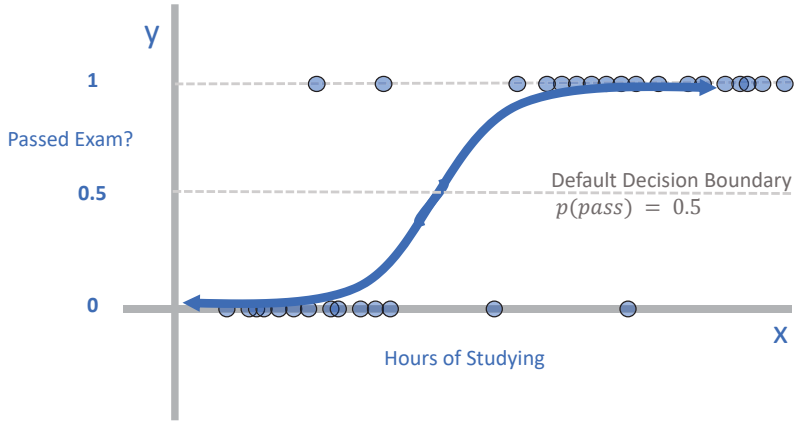
- Note that the probability of passing is now between 0 and 1
  - $0 \leq p \leq 1$
- We now have a continuous function
- As study-hours approach 0, the probability of passing goes (asymptotically) to zero
- As study-hours approach infinity, probability of passing goes (asymptotically) to 1



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

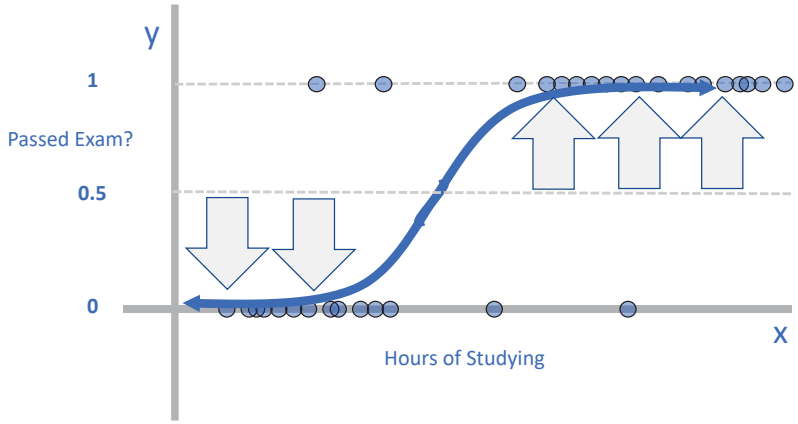
Decision Boundary



HPSI Human Performance Systems, Inc.

Logistic Regression Classification

Decision Boundary



HPSI Human Performance Systems, Inc.

Interpreting Binary Logistic Regression Model Output

- Interpreting the coefficients from a Logistic Regression model is different from a Linear Regression model
- Consider these results from a fitted logistic regression model
  - $\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 * study\_hrs$
  - $\beta_0 = -2.524$
  - $\beta_1 = 0.614$

HPSI Human Performance Systems, Inc.

Interpreting Binary Logistic Regression Model Output

- Interpreting the coefficients from a Logistic Regression model is different from a Linear Regression model
- Consider these results from a fitted logistic regression model
  - $\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 * study\_hrs$
  - $\beta_0 = -2.524$
  - $\beta_1 = 0.614$
  - $\ln\left(\frac{p}{1-p}\right) = -2.524 + 0.614 * study\_hrs$

Interpreting Binary Logistic Regression Model Output

- Interpreting the coefficients from a Logistic Regression model is different from a Linear Regression model
- Consider these results from a fitted logistic regression model
  - $\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 * study\_hrs$
  - $\beta_0 = -2.524$
  - $\beta_1 = 0.614$
  - $\ln\left(\frac{p}{1-p}\right) = -2.524 + 0.614 * study\_hrs$
  - For every additional unit increase in $study\_hrs$, $\ln\left(\frac{p}{1-p}\right)$ increases by 0.614 units?

Interpreting Binary Logistic Regression Model Output

- Interpreting the coefficients from a Logistic Regression model is different from a Linear Regression model
- Consider these results from a fitted logistic regression model
  - $\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 * study\_hrs$
  - $\beta_0 = -2.524$
  - $\beta_1 = 0.614$
  - $\ln\left(\frac{p}{1-p}\right) = -2.524 + 0.614 * study\_hrs$
  - For every additional unit increase in $study\_hrs$, $\ln\left(\frac{p}{1-p}\right)$ increases by 0.614 units?
  - But what does that mean?

Interpreting Binary Logistic Regression Model Output

- If we have

$$y^* = logit = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 * study\_hrs$$

- y^* is called the "logit" function
  - A logit is defined as the log base e (log) of the odds
- Exponentiating both sides of equation results in:

$$\frac{p}{1-p} = e^{(\beta_0 + \beta_1 * study\_hrs)}$$

- We can exponentiate each of the coefficients to generate their corresponding odds ratios

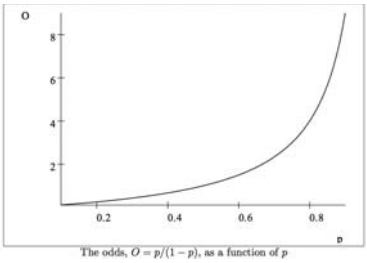
HPSI Human Performance Systems, Inc.

Interpreting Binary Logistic Regression Model Output

- Odds

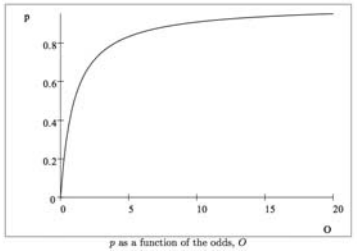
$$Odds(O) = \frac{P}{1-P}$$

  - The odds ratio tells you how a unit increase or decrease in the corresponding input variable affects the odds of passing the test
    - When the odds ratio is greater than 1, it describes a positive relationship
    - An odds ratio less than 1 implies a negative relationship



The odds, $O = p/(1-p)$, as a function of p

Note that $0 \le O$, and O is undefined for $p = 1$. Solving $O = \frac{p}{1-p}$ for p, $p = \frac{O}{(O+1)}$

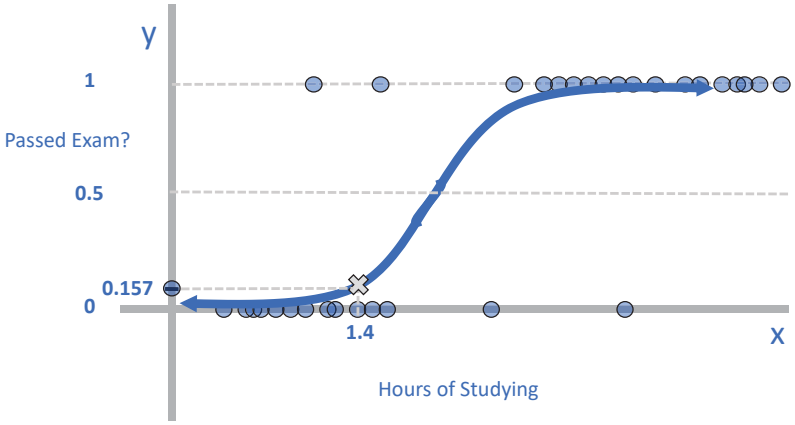


p as a function of the odds, O

HPSI Human Performance Systems, Inc.

Logistic Regression Classification

Visualizing the Logistic Regression Model



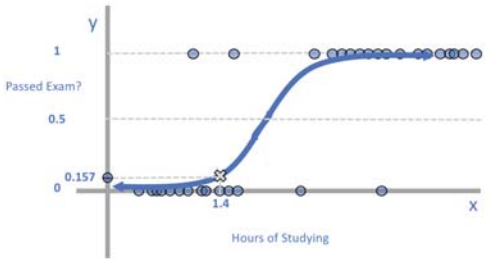
HPSI Human Performance Systems, Inc.

Logistic Regression Classification

- If a student only prepares 1.4 hours for the test
  - $study\_hours$=1.4
  - $\beta_0 = -2.54$
  - $\beta_1 = 0.614$

$$p(pass) = \frac{(e^{\beta_0 + \beta_1 * study_{hours}})}{(e^{\beta_0 + \beta_1 * study_{hours}}) + 1}$$

$$p(pass) = \frac{(e^{-2.54 + 0.614 * 1.4})}{(e^{-2.54 + 0.614 * 1.4}) + 1} = 0.157$$



- Our model predicts a probability of passing of 15.7%
- This falls below the 0.5 threshold and results in a prediction of failing the exam

Logistic Regression Example

Logistic Regression Summary

Exercise Model Comparison

Day 2 Recap

References

1. *Locally Weighted Learning* by Atkeson, Moore, Schaal
2. *Tuning Locally Weighted Learning* by Schaal, Atkeson, Moore
3. Kruschke, J. K. (2014). Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. 2nd Edition.
4. Machine Learning, Neural and Statistical Classification, Editors: D. Michie, D.J. Spiegelhalter, C.C. Taylor

Decision Trees and Ensemble Methods: Random Forest

HPSI Human Performance Systems, Inc.

Main Sources

- Hands-on Machine Learning with Scikit-Learn & Tensor Flow – Aurelien Geron
- Data Science and Analytics with Python – Jesus Rogel-Salazar
- Machine Learning in Python – Michael Bowles
- Python Machine Learning – Sepastian Raschka & Vahid Mirjalili

HPSI Human Performance Systems, Inc.

Outline

- Decision Trees
  - Basics
  - Theoretical Background
  - Advantages
  - Limitations
  - Mathematical Approaches
  - Pruning
  - Evaluation
  - Case Study
- Ensemble
  - Random Forest
  - Example
  - Practice

HPSI Human Performance Systems, Inc.

Decision Trees: Basics

- Decision trees are a hierarchical technique
  - Meaning that a series of decisions are made until a predetermined metric is met
    - Model is built such that a sequence of ordered decisions concerning values of data features results in assigning class labels
- Nonparametric
  - Due to the number of parameters is not pre-determined as is the case with linear models that have pre-determine parameters thus limiting their **degrees of freedom**
  - No assumptions need to be met concerning parameters or distributions
- Best recognized through graphs produced
  - Type of Acyclic graph - are used to model probabilities, connectivity, and causality. A "graph" in this sense means a structure made from nodes and edges
  - Trees consist of nodes and edges defined by decisions rules applied to the data features

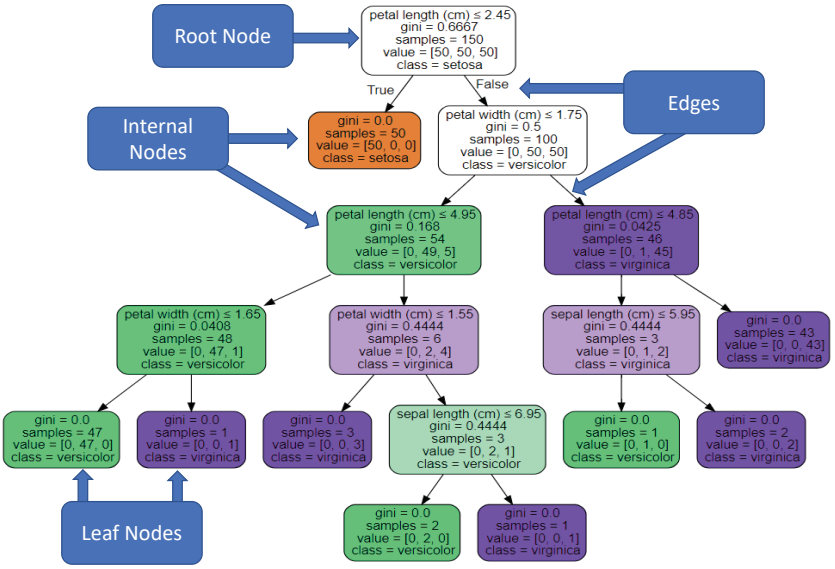
HPSI Human Performance Systems, Inc.

Decision Trees: Basics

- Tree begins with a **Root Node** that has no incoming edges and two or more out going edges
- **Internal Node** – Has one incoming edge and two or more outgoing and represent test conditions at every given level
- **Leaf Node** – One incoming edge and no outgoing edges

HPSI Human Performance Systems, Inc.

Decision Trees: Graph Example



HPSI Human Perfo

Decision Trees: Theoretical Background

- One possible approach that is commonly referenced for building out decision trees is Hunt's algorithm (Hunt, E.B., J. Marin, & P.J. Stone, 1966). Hunt's includes the basic steps below.
  1. If node *n* is pure, such that all the data instances N*n* belong to class *A,* the process is complete and we've generated a leaf node
  2. If node *n* is not pure, we need to keep splitting. This necessitates as split criteria or test condition that allows for the partition. This results in a internal node (child node).
  3. The test condition is then run and we assign each of the instances N*n* to one of the two child nodes created from node *n*
  4. These steps are applied repeatedly to each child node
- This process requires a test condition, which can take several forms
  - Gini coef, error rate, entropy

HPSI Human Performance Systems, Inc.

Decision Trees: Theoretical Background: CART Algorithm (cost function)

- Builds off of the Hunt's paradigm, Classification and Regression Tree (CART)
- Can produce either classification or regression based trees and is used by both sklearn in Python and the CART package in R
  - Can be used on numerical or categorical data
- Default for classification is Gini index for split identification
- First splits the training data in two subsets using a single feature k and a threshold t_k
  - Searches through all possible pairs (k, t_k) to identify the split that produces the purest subsets, based on weighted average of information gain.
- Stops once it cannot find a split that reduces impurity or by a pre-determine hyperparameter (max_depth).

HPSI Human Performance Systems, Inc.

Decision Trees: Theoretical Background

- Uses **recursive binary splitting** - Considering every possible partition of space is computationally infeasible, a **greedy** approach is used to divide the space, called recursive binary splitting.
- **Greedy algorithm** because at each step of the tree building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in the future.
- Trees can be regression or classification based, but in both instances will use **recursive binary splitting**
  - The difference is that in regression based trees we are predicting the actual class whereas in classification we are generating the probability of class inclusion as the determinate of the splitting
  - This probability measure that drives the splitting for classification comes in two forms Gini Index or Entropy

HPSI Human Performance Systems, Inc.

Decision Trees: Advantages

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
- The cost of using the tree (i.e., predicting data) is **logarithmic** in the number of data points used to train the tree.
  - Inverse operation of exponential

Decision Trees: Advantages

- Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model. ROC/Hit Rate/Error Rate

Decision Trees: Limitations

- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called **overfitting**.
  - Compare terminal modes to data points, use the depth for each depth 6^2 equals 64 terminal nodes, if you have 100 data points that's a lot of single data terminal nodes
- Mechanisms such as pruning setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an **ensemble** or Random Forest.

Decision Trees: Limitations

- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts.
  - Consequently, practical decision-tree learning algorithms are based on **heuristic algorithms** such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.

- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to **balance** the dataset prior to fitting with the decision tree

Decision Trees: Definitions

- Overfitting – model becomes overly complex and as a result is predicting noise or the space between features (random error) instead of the true relationship. It is in theory possible to create a leaf node for every data point.

- Ensemble methods – Process of running numerous models and codifying them using a decision rule to choose the optimal model result – example is majority vote on feature inclusion

- Heuristic algorithms – approaches designed for operational efficiency generating a approximation to the ideal result but does not guarantee the best model

Decision Trees: Balancing Dataset

- Imbalanced classes often considers imbalanced to mean a minority class of 10% to 20%. In reality, datasets can get far more imbalanced than this, examples:

1. About 2% of credit card accounts are defrauded per year[1]. (Most fraud detection domains are heavily imbalanced.)
2. Medical screening for a condition is usually performed on a large population of people without the condition, to detect a small minority with it (e.g., HIV prevalence in the USA is ~0.4%).
3. The conversion rates of online ads has been estimated to lie between 10^{-3} to 10^{-6}.
4. Factory production defect rates typically run about 0.1%.
5. Intrusion or threat detection, .0001% of network traffic

Decision Trees: Balancing Dataset

- Working on real world problems will almost certainly include unbalanced datasets. Making standard algorithms function inefficiently.
  - Solution is to **oversample** the minority target or **under-sample** the majority to create a more balanced training dataset
- **Oversampling** – Most commonly used, can result in a synthetic reduction in the variance associated with the variable but as a positive it essentially duplicates the number of errors, i.e. if there's a single false positive and it's included five times you get four additional errors.

HPSI Human Performance Systems, Inc.

Decision Trees: Node split mode

- Decision Trees can use several different types of node split criteria depending on the data or data scientist's preference
  - Regression/MSE – Continuous data
  - Entropy – Binary data splits
  - Gini Coefficient – Most common approach
- Let's take a look at each approach

Both Entropy and Gini Coefficient use Information Gain to determine variable split criteria

HPSI Human Performance Systems, Inc.

Mathematical Approaches: Regression/MSE

- Works to identify the split point in the data set that minimizes **mean squared error (MSE)** point
- The average of each of the groups is the term that minimizes the mean squared error
  - MSE – is the average of the difference between the prediction and actual values

$$\sqrt{\frac{\sum_{t=1}^{n} (Y_t - \hat{Y}_t)^2}{n}}$$

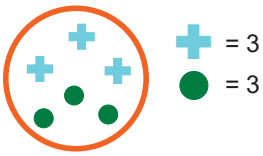
- The decision tree algorithm searches through all variables and all possible split points to identify the point that minimizes error

HPSI Human Performance Systems, Inc.

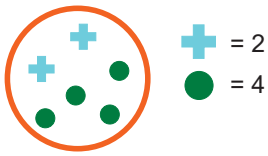
Decision Trees: Mathematical Approaches: Classification, Entropy

- The formula for entropy is below, where Pi is the probability that a random selection would have a state i

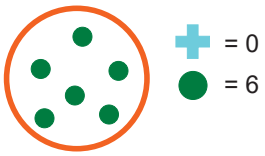
$$\text{Entropy} = \text{sum}(-P_i * \log_2 P_i)$$



$(3/6 \log_2 3/6) - (3/6 \log_2 3/6) = 1$



$(2/6 \log_2 2/6) - (4/6 \log_2 4/6) = 0.92$



$(6/6 \log_2 6/6) = 0$

Decision Trees: Mathematical Approaches: Classification, Entropy

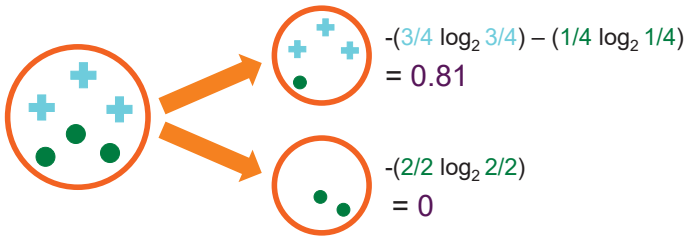
- Information gain helps us understand how important an attribute is in the data
- We can use it to decide how to order the nodes of the decision tree

Information gain = entropy (parent) – average entropy (children)

entropy (parent)

$-(3/6 \log_2 3/6) - (3/6 \log_2 3/6)$
$= 1$

average entropy (children)



$-(3/4 \log_2 3/4) - (1/4 \log_2 1/4)$
$= 0.81$

$-(2/2 \log_2 2/2)$
$= 0$

Decision Trees: Mathematical Approaches: Classification, Entropy

- In order to calculate the average entropy for the split, we need to weigh the split by the number of data points in each node. So we create a weighted average of the entropy of the children nodes.

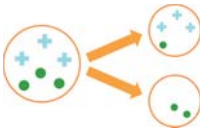
Information gain = entropy (parent) – average entropy (children)

entropy (parent)

$= 1$

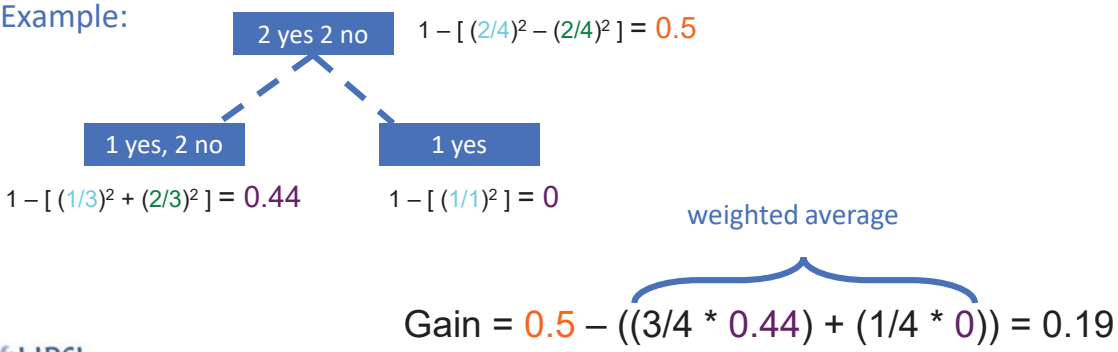
Weighted average entropy (children)

$= (2/6 * 0) + (4/6 * 0.81)$
$= 0.54$



Decision Trees: Mathematical Approaches: Classification, Gini Coefficient

- Gini
  - Gini Impurity = $1 - \text{sum}[(Pi)^2]$
  - Pi – Represents the probability that a random selection would have state i.
  - Same mathematical process as entropy
- Example:



Decision Trees: Code example: Entropy

```
#Importing packages
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
from sklearn import tree
import sklearn as sk
from IPython.display import Image
import pydotplus
import graphviz
```

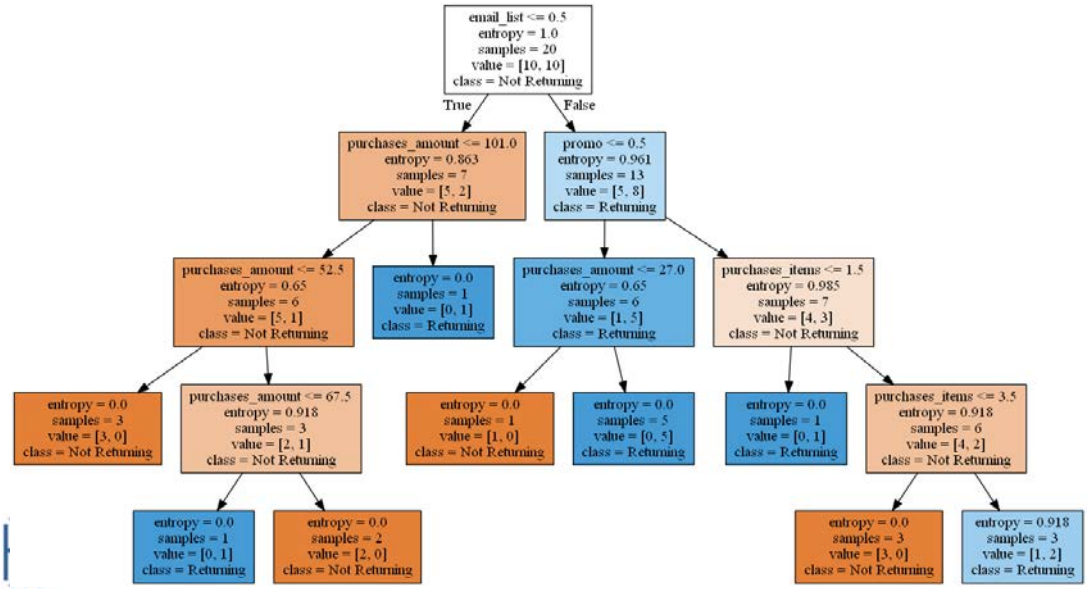

Decision Trees: Code example: Entropy

```
#Creating Data
customers = pd.DataFrame()
customers['purchases_amount'] = [105, 65, 89, 99, 149, 102, 34, 120, 129, 39,
                                 20, 30, 109, 40, 55, 100, 23, 20, 70, 10]
customers['purchases_items'] = [1, 4, 5, 4, 7, 1, 2, 10, 6, 5,
                                1, 3, 2, 1, 5, 10, 3, 3, 1, 1]
customers['promo'] = [1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
                      1, 1, 1, 0, 1, 1, 1, 0, 1, 1]
customers['email_list'] = [1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
                           0, 1, 1, 0, 1, 0, 1, 1, 0, 0]
customers['checkouts'] = [1, 5, 3, 3, 1, 2, 4, 4, 1, 1,
                          1, 1, 2, 4, 1, 1, 2, 1, 1, 1]
repeat_customers = pd.DataFrame()
repeat_customers['repeat'] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                              0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Decision Trees: Code example: Entropy

```
#Launch the Tree and fit the tree
decision_tree = tree.DecisionTreeClassifier(
    criterion='entropy',   --- Setting the split criteria
    max_features=1, --- number of features used to split per node
    max_depth=4, --- depth of the tree
    random_state = 1000 --- normally wouldn't use but simply creates the same tree
for everyone for comparison purposes
)
decision_tree.fit(customers, repeat_customers)  --- fits the tree
sk.tree.export_graphviz(decision_tree) --- produces the results
```


Decision Trees: Example and Practice



Decision Trees: Code example: Exercise and Practice

- Using the same data remove the random state and compare results
  - Are they the same?
  - If not why?
- Adjust the depth of the tree and compare results

Decision Trees: Overfitting and Hyper-parameters

- Decision trees are often prone to overfitting one solution is utilize the hyper-parameters to control the depth of the tree or limit the expansion of leaf nodes
- Another option is to use a ensemble method via bagging or what's known as Random Forest
  - Will discuss further later in the day

HPSI Human Performance Systems, Inc.

Decision Trees: Hyper-parameter tuning (Pruning)

- **Minimum samples for a node split** Minimum number of samples (or observations) which are required in a node to be considered for splitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample. It should be tuned using cross validation.
- **Minimum samples for a terminal node (leaf)** The minimum number of samples (or observations) required in a terminal node or leaf. For imbalanced class problems, a lower value should be used since regions dominant with samples belonging to minority class will be much smaller in number.
- **Maximum depth of tree (vertical depth)** The maximum depth of trees, lower values prevent a model from learning relations which might be highly specific to the particular sample. It should be tuned using cross validation.

HPSI Human Performance Systems, Inc.

Decision Trees: Hyper-parameter tuning (Pruning)

- **Maximum number of terminal nodes** Also referred as *number of leaves*. Since binary trees are created, a depth of *n* would produce a maximum of 2^n leaves.
- **Maximum features to consider for split** The number of features to consider (selected randomly) while searching for a best split. A typical value is the square root of total number of available features. A higher number typically leads to over-fitting but is dependent on the problem as well.
- There's actually many more, the documentation is here:
  - http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

HPSI Human Performance Systems, Inc.

Decision Trees: Pruning Code Example

- Go to Spyder

Decision Trees: Prediction

- You can use the built in prediction function in sklearn to implement our models
  - clf.predict([[5,1.5,2,2]])
  - tree_clf_reg_full.predict([[5,1.5,2,2]])
- Seems like we have different results, let's look at model evaluation next.
- Quick Exercise
  - Use different ranges of each of the variables to generate different predictions
    - Petal Length
    - Petal Width
    - Sepal Length
    - Sepal Width

Break 15 Minutes

Decision Trees: Evaluation: Receiver Operating Curve or Area Under the Curve

- ROC and AUC are two very common approaches for measuring the performance of classification models
- Essentially both these measure the misclassification error rate associated with your model
- A Confusion Matrix is a good tool for understanding how accurate you model is classifying and can be used to build ROC

HPSI Human Performance Systems, Inc.

Decision Trees: Evaluation: Confusion Matrix and threshold

- Let's use intruder detection as a example
- Say we have 135 emails entering our system and we are trying to detect whether they are fraudulent or not
  - We use lots of criteria – source, subject, if they came from a prince…
- Generate probability measures as a result of our tree algorithm to determine the likelihood that any one of these emails fraudulent
- The cutoff point that is predetermined in the tree at 50% but can be modified as a input the confusion_matrix() function by adjusting the sample_weight parameter

	Predictive Class		
Actual Class		Positive Fraud	Negative Not Fraud
	Positive Fraud	True Positive 10	False Negative 7
	Negative Not Fraud	False Positive 22	True Negative 96

HPSI Human Performance Systems, Inc.

Decision Trees: Evaluation: Confusion Matrix and threshold

- Let's consider the extremes: what if we set the threshold to 0?
  - Means that everything is captured as Fraud and no ever gets a email again!

	Predictive Class		
Actual Class		Positive Fraud	Negative Not Fraud
	Positive Fraud	True Positive 17	False Negative 0
	Negative Not Fraud	False Positive 118	True Negative 0

- Let's consider the extremes: what if we set the threshold to 100?
  - Means nothing is fraud and now everyone is getting rich off of Arabian princes

HPSI Human Performance Systems, Inc.

Decision Trees: Evaluation: Confusion Matrix and threshold

- Let's consider the extremes: what if we set the threshold to 100?
  - Means nothing is fraud and now everyone is getting rich off of Arabian princes

	Predictive Class		
Actual Class		Positive Fraud	Negative Not Fraud
	Positive Fraud	True Positive 0	False Negative 17
	Negative Not Fraud	False Positive 0	True Negative 118

Decision Trees: Evaluation: Confusion Matrix and threshold

- We can also predict the accuracy of our model using this information:
  - True Positive Rate (TPR) = 10/(10+22) = .45
  - False Positive Rate (FDR) = 7/(7+96) = .06

	Predictive Class		
Actual Class		Positive Fraud	Negative Not Fraud
	Positive Fraud	True Positive 10	False Negative 7
	Negative Not Fraud	False Positive 22	True Negative 96

- These two data points can then be used to begin to develop a Receiver Operating Characteristic Curve or ROC curve
  - True Positive Rate (TPR) = 10/(10+22) = .45 = y-axis
  - False Positive Rate (FDR) = 7/(7+96) = .06 = x-axis

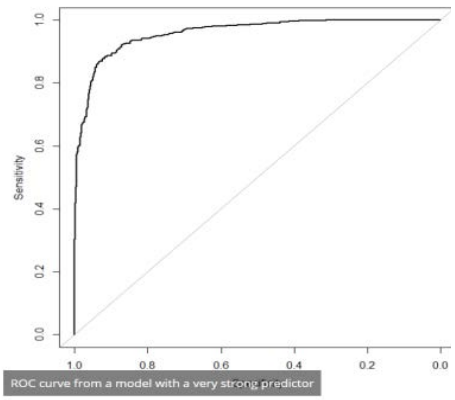
Decision Trees: Evaluation: Confusion Matrix: Code Example

- Below we are using the two decision trees we developed to generate confusion matrix to evaluate our models.
- We first use the original data to predict then compare the results to the original data

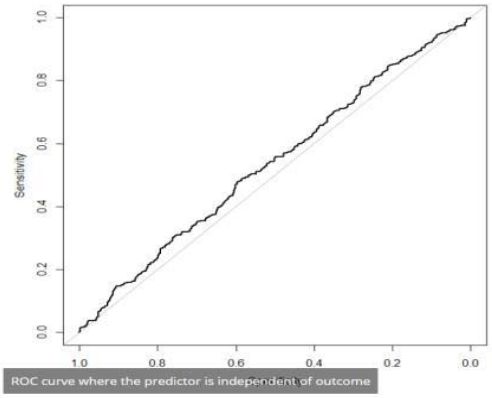
```
clfpred = clf.predict(iris.data)
sk.metrics.confusion_matrix(y,clfpred)

regpred = tree_clf_reg_full.predict(iris.data)
sk.metrics.confusion_matrix(y,regpred)
```

Decision Trees: Evaluation: ROC

- ROC curve is essentially a graphical representation of the adjusted threshold values of the confusion matrix, below are two examples



ROC curve from a model with a very strong predictor



ROC curve where the predictor is independent of outcome

Decision Trees: Evaluation: Code ROC

```
#The ROC is best used in a 2-demensional format so we will pull forward a
previous example focused on customer prediction

customerpred = decision_tree.predict(customers)

#Here we see the confusion matrix
sk.metrics.confusion_matrix(repeat_customers, customerpred)

#The roc_curve has three outputs fpr:false positive rate, tpr: true positive
rate and thresholds, below we are developing output for each

false_positive_rate, true_positive_rate, thresholds =
roc_curve(repeat_customers, customerpred)

Here we are develop the area under the curve

roc_auc = auc(false_positive_rate, true_positive_rate)
```


Decision Trees: Evaluation: Code ROC

- Here we use the variables we created in the previous slide to display the output graphically

```
#The code below plots these outputs
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

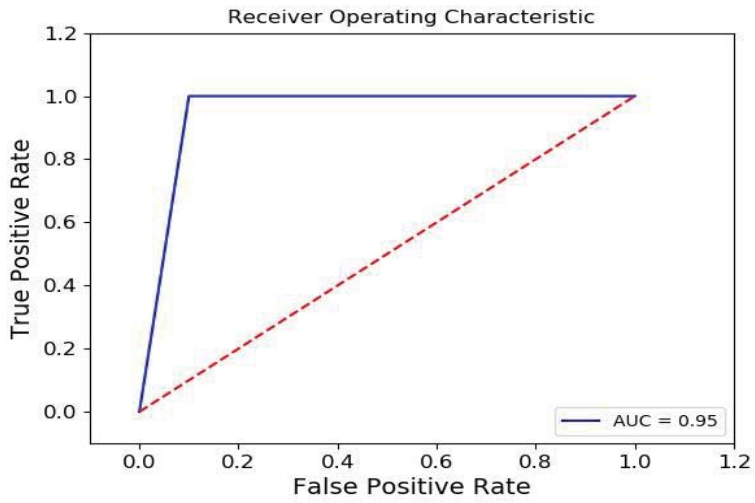
Decision Trees: Evaluation: Code ROC

- Here we use the variables we created in the previous slide to display the output graphically

```
#The code below plots these outputs
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

HPSI Human Performance Systems, Inc.

Decision Trees: Evaluation: Code ROC



HPSI Human Performance Systems, Inc.

Break: 15 Minutes

HPSI Human Performance Systems, Inc.

Case Study Example

- Review Case Study and Discuss (time permitting)

HPSI Human Performance Systems, Inc.

Outline

- Decision Trees
  - Basics
  - Theoretical Background
  - Advantages
  - Limitations
  - Mathematical Approaches
  - Pruning
  - Evaluation
  - Case Study
- Ensemble
  - Random Forest
  - Example
  - Practice

HPSI Human Performance Systems, Inc.

Ensemble Methods

HPSI Human Performance Systems, Inc.

Ensemble Methods

- Ensemble methods are more or less aggregated predictions of many different algorithms in order to increase predictive accuracy.
- Often in solving a machine learning problem building a ensemble model comes at the end of the process after trying several different approaches you can combine them into one all knowing predictor!
- We will focus on Random Forrest, a ensemble of decision tress but also discuss bagging and boosting

HPSI Human Performance Systems, Inc.

Ensemble Methods

- Example: suppose we have developed several different classifiers a KNN model, Logistic Regression, an SVM and a Decision Tree.
- We could use the majority vote process to build the final classification of our data points, based on below what would be the outcome for this single data point?

Model	KNN	Logistic Regression	Support Vector Machine	Decision Tree
Prediction	1	0	1	1

- This is called **hard voting,** another approach is **soft voting (discuss later)**
- This process often works better than using single **weak learners –** or algorithms that predict only slightly better than random guessing

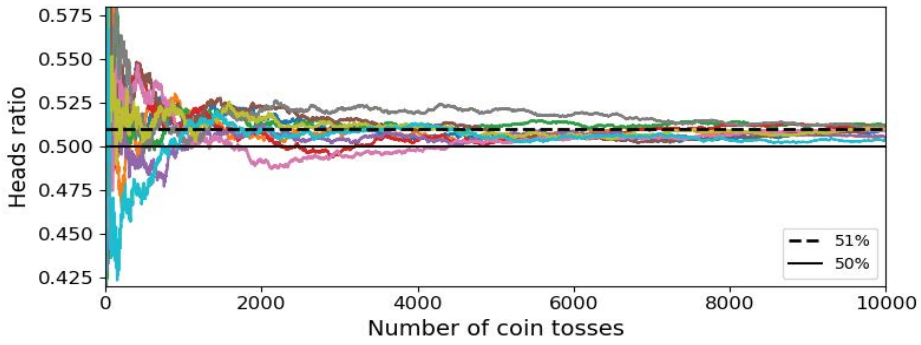
HPSI Human Performance Systems, Inc.

Ensemble Methods

- Why do the models work better together?
  - Essentially scale increases the probability of finding a majority vote.
  - As an example if we have a bias coin flip the gives us 51% chance of heads and 49% chance of tails the more we toss the coin the higher the probability of getting a majority vote for heads.
    - 1,000 = 75% chance
    - 10,000 = 97% chance
- Works the same way for model building, the reliability of the results simple increase with scale.
  - Works best if the models are perfectly independent, meaning the error terms don't correlation which is hard when using one approach on a single dataset, that's why combining approaches can sometimes result in better outcomes

HPSI Human Performance Systems, Inc.

Ensemble Methods: Probabilities Converge (Code Available)



HPSI Human Performance Systems, Inc.

Ensemble Methods: Ensemble

- **Soft voting** is used for algorithms that produce probabilistic outputs or those that have a hyperparameter that can be modified to support probability outputs (such as SVM, probability hyperparameter to TRUE).
  - The class is generated using the highest class probability as the metric
  - Replacing voting = hard to voting = soft in the VotingClassifier() function

HPSI Human Performance Systems, Inc.

Ensemble Method: Code Example

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons

#Load in the models we will be using
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier

#Use the train test function on the moon dataset
X, y = make_moons(n_samples=500, noise=0.30, random_state=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=1)

#Next we are establishing the variables for each of our models
log_clf = LogisticRegression(random_state=1)
rnd_clf = RandomForestClassifier(random_state=1)
svm_clf = SVC(random_state=1)
```

HPSI Human Performance Systems, Inc.

Ensemble Method: Code Example

```
#Next we are just developing a voting classifier as we discussed that
uses hard voting to develop an ensemble method
voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
voting_clf.fit(X_train, y_train)


#Now let's see how we did
from sklearn.metrics import accuracy_score # This metric calculates
the error rate for each of our models

for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

HPSI Human Performance Systems, Inc.

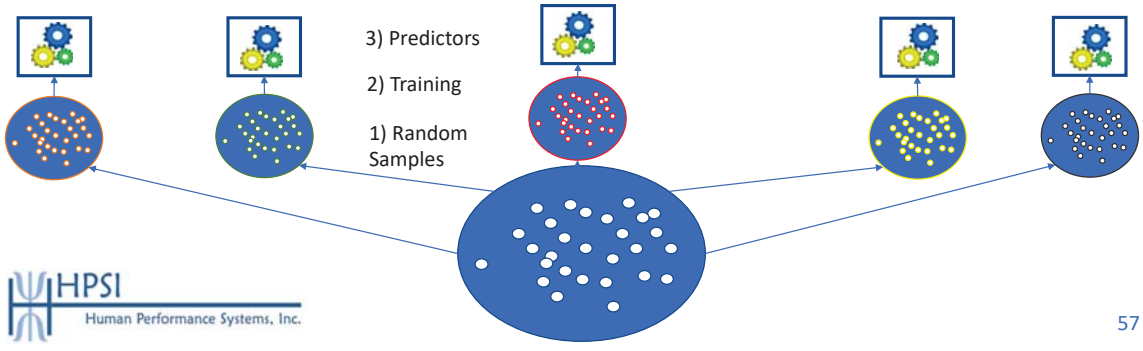
Ensemble Methods: Random Forest – power in numbers

- Ensemble methods – Essentially instead of building a single tree we are going to build a whole bunch
- We can limit the growth of the trees but don't have to use any of the hyper-parameters
- Set the number of trees grown and track the error classification rate of our algorithm
- Can be used for again for both Regression or Classification
- Random Forest uses **bagging**
  - **Bagging** – Bootstrap Aggregation – selecting subsets of the data with replacement to generate unique trees
  - **Boosting** – Another form of dataset selection that is commonly used in ensemble methods

HPSI Human Performance Systems, Inc.

Ensemble Methods: Bagging

- As discussed on method is to use several methods and combine them to produce a more powerful outcome
- You can also you the same technic but re-sample the data numerous times to produce different results, one such method is **Bagging**
- **Bagging –** is sampling with replacement, meaning that the entire dataset is available for every sub-sample. (without replacement is often called **pasting)**



HPSI Human Performance Systems, Inc.

Ensemble Methods: Random Forest

Sampling with replacement

Sampling without replacement

Obs	X1	X2	Y1	Y2
1	2.5	3.6	4.8	3.7
2	2.8	4.7	-2.8	7.1
3	5.8	9.7	9.1	13

Obs	X1	X2	Y1	Y2
1	2.5	3.6	4.8	3.7
2	2.8	4.7	-2.8	7.1
1	2.5	3.6	4.8	3.7

Obs	X1	X2	Y1	Y2
2	2.8	4.7	-2.8	7.1
1	2.5	3.6	4.8	3.7
3	5.8	9.7	9.1	13

HPSI Human Performance Systems, Inc.

Ensemble Methods: Random Forest

- If we keep building decision trees on the same dataset, we would essentially get the same decision trees every time

Obs	X1	X2	Y1	Y2
1	2.5	3.6	4.8	3.7
2	2.8	4.7	-2.8	7.1
3	5.8	9.7	9.1	13

Obs	X1	X2	Y1	Y2
2	2.8	4.7	-2.8	7.1
1	2.5	3.6	4.8	3.7
3	5.8	9.7	9.1	13

Obs	X1	X2	Y1	Y2
3	5.8	9.7	9.1	13
1	2.5	3.6	4.8	3.7
2	2.8	4.7	-2.8	7.1

HPSI Human Performance Systems, Inc.

Ensemble Methods: Random Forest

- Use a subset of attributes generated by bagging to build original data sets to make decision trees

Obs	X1	X2	Y1	Y2
1	2.5	3.6	4.8	3.7
1	2.5	3.6	4.8	3.7
3	5.8	9.7	9.1	13

Obs	X1	X2	Y1	Y2
2	2.8	4.7	-2.8	7.1
1	2.5	3.6	4.8	3.7
2	2.8	4.7	-2.8	7.1

Obs	X1	X2	Y1	Y2
3	5.8	9.7	9.1	13
3	5.8	9.7	9.1	13
2	2.8	4.7	-2.8	7.1

HPSI Human Performance Systems, Inc.

Ensemble Methods: Bagging

- Typically once the sub-samples have been gathered the new classes are generated through hard voting for classification or the average for regression
  - However if the algorithm produces a probabilistic output, like decision trees, soft voting is used.

Ensemble Methods: Coding Example

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

#This is for classification if we want to run the same process on regression
it's BaggingRegressor
bag_clf = BaggingClassifier(DecisionTreeClassifier(),
n_estimators=500,max_samples=100, bootstrap=True,n_jobs=-1)
#n_estimators = number of tree classifiers
#max_samples= number of times each of those classifiers is trained using
random sampling with replacement
#bootstrap = tells the baggingclassifier to use with replacement
#n_jobs = tells the classifier the number of CPUs to use for the classifier,
-1 tells to use all available
bag_clf.fit(X_train, y_train) #No we run the data from the moon dataset
y_pred=bag_clf.predict(X_test) #Generate predictions using the test dataset
print(accuracy_score(y_test, y_pred)) #Calculate the accuracy score
```


Ensemble Methods: Random Forest

- Instead of building the ensemble from the ground up using the bagging method we can also just use the Random Forrest classifier
- Typically all the hyperparameter are the same as the DecisionTreeClassifier and the hyperparameters of the baggingclassifier
- Extra Randomness is also included as the Random Forest classifier as it searches for the best node split among all the random subsets of features
  - This creates greater tree diversity but trades higher **variance** for lower **bias**, which is often the case when increasing the complexity of the method
- **Bias** – is underfitting a model, meaning you're missing important relationships between variables.
- **Variance –** is overfitting the model, meaning the model is measuring the spaces between data points or the **noise** versus

Ensemble Methods: Random Forest: Code Example

```
from sklearn.ensemble import RandomForestClassifier

rnd_clf=RandomForestClassifier(n_estimators=500,
max_leaf_nodes=16,n_jobs=-1)

#n_estimators = number of tree classifiers
#max_leaf_nodes = complexity of the model, limits the terminal leaf
nodes

rnd_clf.fit(X_train, y_train)

y_pred_rf=rnd_clf.predict(X_test)
```


Ensemble Methods: Random Forest: Variable Importance

- One of the really nice features associated with Random Forrest is it's ability to select variables that are most "important" to training the model.
- This is done by determining which variables are closest to the root of the tree and then moving outward
- Random Forrest does this at scale and the output is generated as a part of the training process
  - Sklearn calculates the average depth at which each feature is utilized and ranks them accordingly

Ensemble Methods: Random Forest: Variable Importance: Code Example

```
iris = datasets.load_iris()
# Fit the model using a different random forest classifier
ext = ExtraTreesClassifier()
ext.fit(iris.data, iris.target)

# display the relative importance of each attribute
print(ext.feature_importances_)

# Below will align the feature importance with the variable being used and
print the result

for name, importance in zip(iris["feature_names"], ext.feature_importances_):
    print(name, "=", importance)
```

Neural Network Lecture

Day 4 and Day 5

Amir Jafari

HPSI
Human Performance Systems, Inc.

What is Machine learning?

- What is Machine Learning (ML)?
- In a technical term, ML is to program computers so that they can learn from input data.
- Learning is the process of acquisition of knowledge or skills through experience.
- The input to this learning process is training data and the output is some expertise that can perform some task.
- Since computers were invented, we wanted to know, computers might be made to learn. How to program them to learn and improve automatically.

HPSI
Human Performance Systems, Inc.

Machine Learning in Our Life

- Imagine:
- Computers learn from medical records which treatments are most effective for new diseases
- Computers learn from houses experience to optimize energy costs based on patterns of occupants usage.
- Computers learn from evolving interests of their users in order to highlight especially relevant stories from the online morning newspaper.
- Computer cannot learn nearly as well as people learn yet. However, algorithms have been invented that are effective for certain types of learning tasks, and a theoretical understanding of learning is beginning to emerge.

HPSI
Human Performance Systems, Inc.

Machine Learning Applications

- Aerospace
  - High performance aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations, aircraft component fault detectors
- Automotive
  - Automobile automatic guidance systems, fuel injector control, automatic braking systems, misfire detection, virtual emission sensors, warranty activity analyzers
- Banking
  - Check and other document readers, credit application evaluators, cash forecasting, firm classification, exchange rate forecasting, predicting loan recovery rates, measuring credit risk

HPSI Human Performance Systems, Inc.

More Machine Learning Applications

- Medical
  - Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, emergency room test advisement
- Defense
  - Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression.
- Speech
  - Speech recognition, speech compression, vowel classification, text to speech synthesis
- Financial
  - Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit line use analysis.

HPSI Human Performance Systems, Inc.

What is Deep Learning?

- ”Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, partially supervised or unsupervised.”

HPSI Human Performance Systems, Inc.

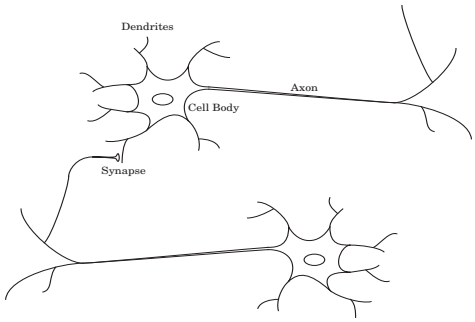
Deep Learning Applications

- 8 Inspirational Applications of Deep Learning
- 10 Deep Learning Applications
- 18 DEEP LEARNING STARTUPS
- GPU Application

HPSI Human Performance Systems, Inc.

Brain Function

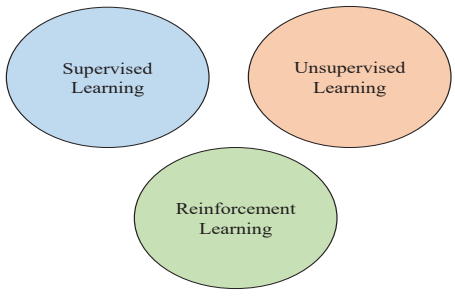
- The brain consists of a large number (approximately 10^{11}) of highly connected elements (approximately 10^4 connections per element) called neurons.



HPSI Human Performance Systems, Inc.

Machine Learning Basics

- The three different types of machine learning.



HPSI Human Performance Systems, Inc.

Supervised Learning

Supervised learning is a method to:

- Approximate a function from a training data set.
- Training data set includes both input and target.
- For each input data there is a target (desired results) associated with.
- Analyzing the training data set to approximate the function, it is called a classifier or pattern recognition (output is discrete) or a regression function (output is continuous).

HPSI
Human Performance Systems, Inc.

My note

HPSI
Human Performance Systems, Inc.

Unsupervised Learning

Unsupervised learning is a method to:

- Infer a function from a training data set.
- Training data set includes just the inputs.
- For each input data there is no target.
- Since there is no target there is no error to evaluate the model.
- This distinguishes unsupervised learning from supervised learning.
- Mostly is used in the clustering applications.

My note

Reinforcement Learning

Reinforcement learning is a method to:

- How to map situations to actions.
- It is based on penalty for making mistakes and rewards for any success.
- For each input data there is no target.
- The learning happens based on the trail and error.
- It is inspired by behaviorist psychology.
- Mostly is used game theory, control theory and multi agent systems and so on.

My note

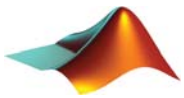
Problem Types

- **Fitting** (nonlinear regression). Map between a set of inputs and a corresponding set of targets. (e.g., estimate home prices from tax rate, estimate emission levels from fuel consumption and speed, predict body fat level from body measurements.)
- **Pattern recognition** (*classification*). Classify inputs into a set of target categories. (e.g., classify a tumor as benign or malignant, from uniformity of cell size.)
- **Clustering** (*segmentation*) Group data by similarity. (e.g., group customers according to buying patterns, group genes with related expression patterns.)
- **Prediction** (time series analysis, system identification, filtering or dynamic modeling). Predict the future value of some time series. (e.g., predict the future value of some stock.)

Softwares

We are going to use:

- Matlab (licensed). This is just for demos and you do not need to install it.
- Python (Open source), widely used in machine learning field.



Introduction

- Machine learning combines statistics and computer science fields.
- Statistics, probability, estimation and confidence intervals are some of main topics in machine learning in the statistical part.
- Linear algebra is another mathematical skill which is highly necessary in machine learning.
- Optimization theory and calculus are widely used in machine learning algorithms.

Why we need math?

- There are so many machine learning codes out there, and they are fairly simple to run.
- You need to download the packages and library to run the machine learning algorithms.
- However, to get some useful results and meaningful performance, you need to have a good mathematical background.
- After this lecture, you will get a glimpse of what types of mathematical skills you need to practice.
- In this lecture, we go over the main topics and further materials will be provided to you in order to strengthen you mathematical skills.

HPSI Human Performance Systems, Inc.

Definition of Probability?

- Relative Frequency.
- Subjective Probability.
- Axiomatic Probability.

HPSI Human Performance Systems, Inc.

Notation

- $a \in A$ a is member of A.
- $\cup$ is union, $\cap$ is intersection.
- $\sum$ is summation.
- $\int$ is integral.
- R is set of real numbers.
- **a,b,c** vector.
- **A,B,C** Matrix.
- $\frac{\sigma}{\sigma x} f(x)$ is a function.
- $y = f(x)$ is a function.
- $\frac{d}{dx} f(x)$ is a function.
- $||A||$ is norm A.
- a,b,c is set.
- $\emptyset$ is empty set.
- $\subset$ is subset.
- $y = f(\mathbf{x})$

HPSI Human Performance Systems, Inc.

My note - Set examples

My note - Venn Diagram

My note - Complement and DeMorgan’s Law

Axiomatic Probability

- $P(A) >= 0$
- $P(s) = 1$
- $P(A \cup B) = P(A) + P(B)$ A and B are disjoint.
- $A \cap B = \emptyset$ is disjoint.
- $A \cap B = P(A) \times P(B)$ if A and B are independent.

My note - Example rolling a die

Conditional Probability

- $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- $P(B|A) = \frac{P(A \cap B)}{P(A)}$
- $P(A \cap B) = P(B|A) \times P(A)$
- $P(A \cap B) = P(A|B) \times P(B)$
- $P(B|A) = \frac{P(A|B) \times P(B)}{P(A)}$ BAYES RULE.

HPSI
Human Performance Systems, Inc.

Random variable

- A random variable is a mapping from sample space to the real line.
- $F_x(\lambda) = P(s : x(s) <= \lambda)$ Distribution function.
- $f_x(\lambda) = \frac{d}{d\lambda} F_x(\lambda)$ Probability Density function.
- $F_x(\lambda) = \int_{-\infty}^{\lambda} f_x(\mu) d\mu$
- $E[x] = \int_{-\infty}^{\infty} \lambda f_x(\lambda) d\lambda$

HPSI
Human Performance Systems, Inc.

My note - Random variable

Random Variable

MATLAB:

```
gaussian_numbers =randn(1000)
hist(gaussian_numbers,20)
```

R:

```
BMI<-rnorm(n=1000, m=24.2, sd=2.2)
hist(BMI)
```

Python:

```
import matplotlib.pyplot as plt
import numpy as np
import plotly.plotly as py
gaussian_numbers = np.random.randn(1000)
plt.hist(gaussian_numbers)
plt.title("Gaussian Histogram")
```

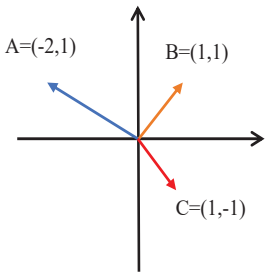

Linear algebra

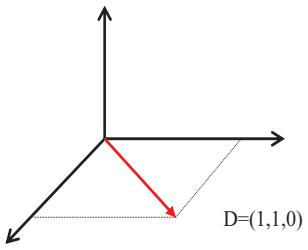
- Operations on or between vectors and matrices
- Linear regression, dimensionality reduction and solution of linear systems of equations are some application of linear algebra.
- Most common form of data in machine learning is in a vector form. The 2D array where rows represent samples represent columns attributes.
- A vector is a n-tuple of values usually real numbers where n is the the dimension of the vector, n can be any positive number.
- Vector is written in a column form.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Vectors

- Vector is a point in space or a line with magnitude and direction.





Vector arithmetic

- Addition of the vectors. **x =a+b**
- Scalar multiplication. $\mathbf{x} = a\mathbf{x}$
- Dot product of vector. $a = \mathbf{x} \cdot \mathbf{y} = \sum_i^n x_i y_i$ or $a = \mathbf{x} \cdot \mathbf{y} = ||\mathbf{x}||||\mathbf{y}||cos(\theta)$

Matrix

- Matrix is mapping. It is an $m \times n$ two dimensional array.
- A vector is the bases set of a matrix. These bases set creates the matrix.
- A vector and matrix can be transposed. Transpose is the swap of rows and columns.
- Matrix is identified by two subscript. First element in subscript shows row number and the second element shows column number.
- A_{21} shows indicates to second row and first column.

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Matrix arithmetic

- **C = A+B**, two matrix with the same dimension can be added ($c_{ij} = a_{ij} + b_{ij}$), the results has the same size.
- $\mathbf{A} = c.\mathbf{B}$, an scalar can be multiplied by each element of a matrix. ($A_{ij} = c + B_{ij}$), the results has the same size.
- Matrix multiplication.
- **A . (B . C) = (A . B) . C**, associative.
- **A . B** $\neq$ **B. A**, commutative.
- $(\mathbf{A} \cdot \mathbf{B})^T = \mathbf{B}^T.\mathbf{A}^T$, transposition.
- $\mathbf{A}\mathbf{A}^{-1} = I$, the inverse of an n-by-m matrix **A** is denoted $\mathbf{A}^{-1}$.

Matrix arithmetic

- **C = A+B**, two matrix with the same dimension can be added ($c_{ij} = a_{ij} + b_{ij}$), the results has the same size.
- $\mathbf{A} = c.\mathbf{B}$, an scalar can be multiplied by each element of a matrix. ($A_{ij} = c + B_{ij}$), the results has the same size.
- **A . (B . C) = (A . B) . C**, associative, **A . B** $\neq$ **B. A**, commutative and $(\mathbf{A} \,.\, \mathbf{B})^T = \mathbf{B}^T.\mathbf{A}^T$, transposition.
- In any chain of matrix multiplications, the column dimension of one matrix in the chain must match the row dimension of the following matrix in the chain.
- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$, $(k\mathbf{A}^{-1}) = k^{-1}\mathbf{A}^{-1}$, $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$, $(\mathbf{AB}^{-1}) = \mathbf{B}^{-1}\mathbf{A}^{-1}$

My note - Linear algebra

- Assume: **A** is (3x5), **B** is (5x5) and **C** is (3x1).
- **A . B . A**, **A . B .** $\mathbf{A}^T$, $\mathbf{C}^T$.**A . B**, **C. A . B**

HPSI
Human Performance Systems, Inc.

Eigenvalues and Eigenvectors

- Eigenvalues and eigenvectors play a prominent role in the study of ODE and Linear Algebra and in many applications in the physical sciences.
- Let A be an $n \times n$ matrix. The value λ is an eigenvalue of A if there exists a non-zero vector v such that $Av = \lambda v$
- In this case, vector v is called an eigenvector of A corresponding to.

HPSI
Human Performance Systems, Inc.

Eigenvalues

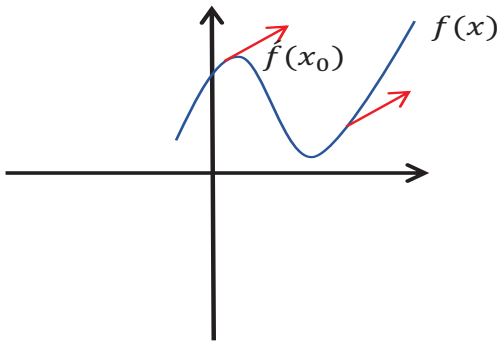
HPSI
Human Performance Systems, Inc.

Eigenvectors

HPSI
Human Performance Systems, Inc.

Calculus derivative

- A derivative, $\frac{d}{dx}f(x)$ is defined as the slope of a function $f(x)$. This is sometimes also written as $f'(x)$.

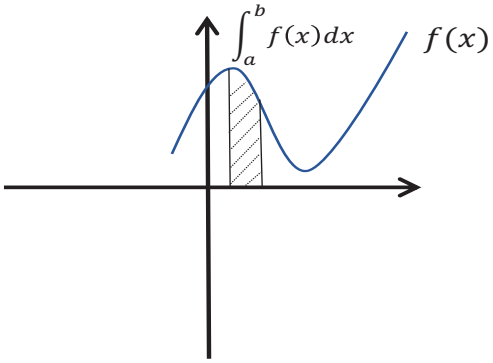


HPSI
Human Performance Systems, Inc.

HPSI
Human Performance Systems, Inc.

Calculus integral

- Integrals are an inverse operation of the derivative (plus a constant).



HPSI
Human Performance Systems, Inc.

HPSI
Human Performance Systems, Inc.

Vector calculus

- Assume vector $\mathbf{x} = (x_0, x_1, ..., x_{n-1})^T$ and a function $f(x)$

$$\frac{\sigma f(x)}{\sigma x} = \begin{pmatrix} \frac{\sigma f(x)}{\sigma x_0} \\ \frac{\sigma f(x)}{\sigma x_1} \\ \vdots \\ \frac{\sigma f(x)}{\sigma x_{n-1}} \end{pmatrix}$$

- This is called the gradient of the function with respect to vector $\mathbf{x}$, written as $\nabla f(x)$ or ∇f

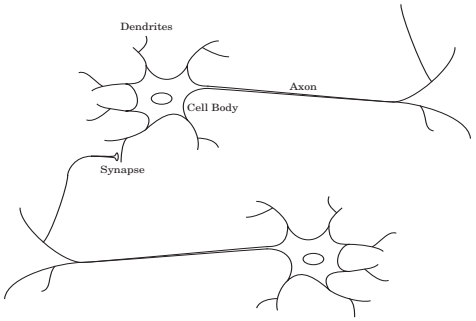
My note - Vector calculus

Summary

- We were touching the basic concepts of probability in this lecture.
- Linear algebra is very crucial in machine learning algorithms.
- This lecture just briefly introduces the main and basic concepts of math which we needed for basic machine learning algorithms.
- Check khan academy again if you need to review or refresh your mind.
- We are going to use these mathematical concepts through out the course and implement in into the computer programmes.

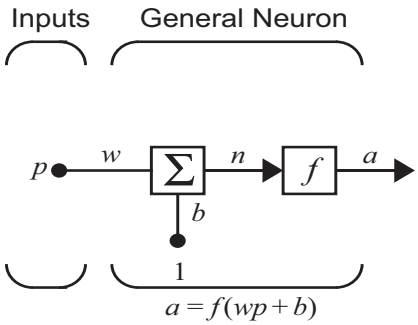
Brain Function

- Neurons Respond Slowly
  - 10^{-3} s compared to to 10^{-9} s for electrical circuits.
- The brain uses massively parallel comptation
  - $\approx 10^{11}$ neurons in the brain.
  - $\approx 10^{4}$ connections per neurons.



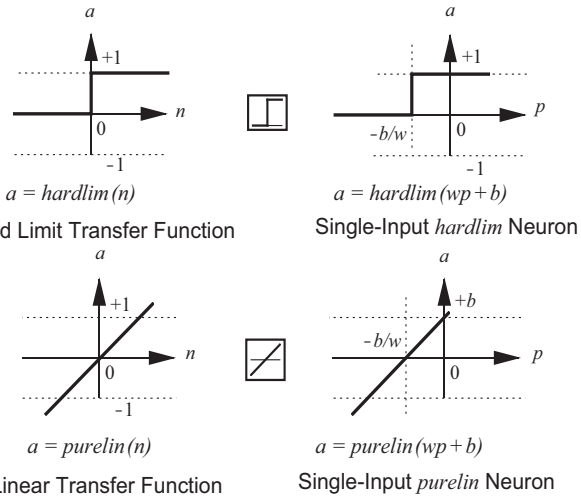
HPSI
Human Performance Systems, Inc.

Single input model



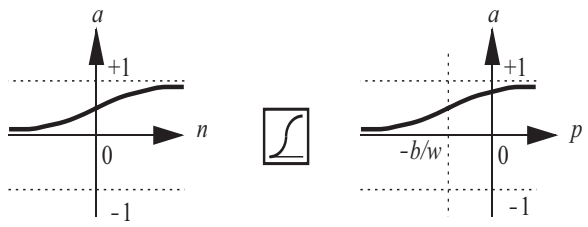
HPSI
Human Performance Systems, Inc.

Transfer function



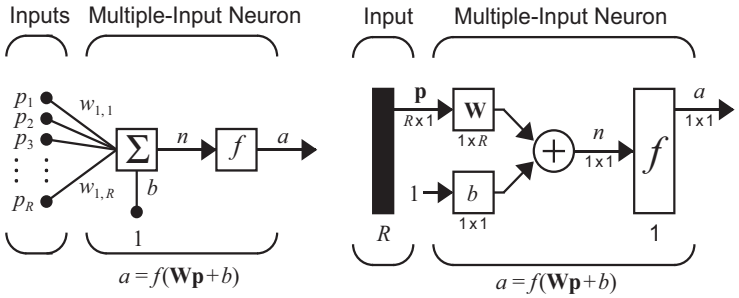
HPSI
Human Performance Systems, Inc.

Transfer function



HPSI
Human Performance Systems, Inc.

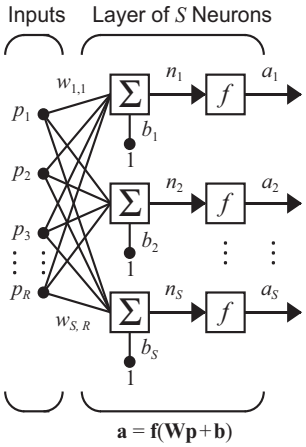
Multiple input neuron



Abreviated Notation

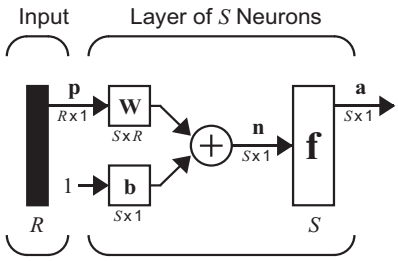
HPSI
Human Performance Systems, Inc.

Layer of neurons



HPSI
Human Performance Systems, Inc.

Abbreviated notation



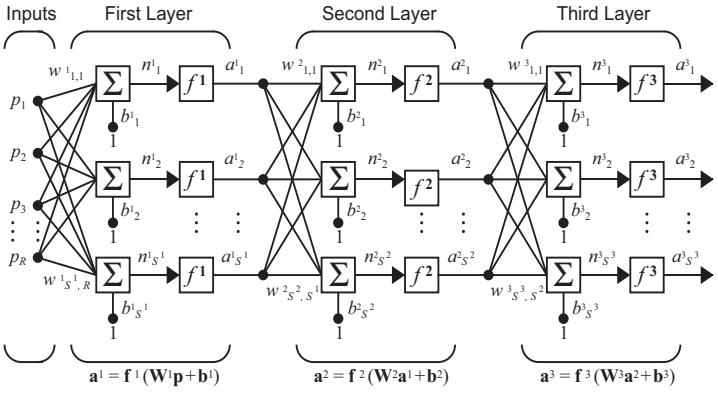
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p}+\mathbf{b})$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}$$

HPSI
Human Performance Systems, Inc.

Multilayer network

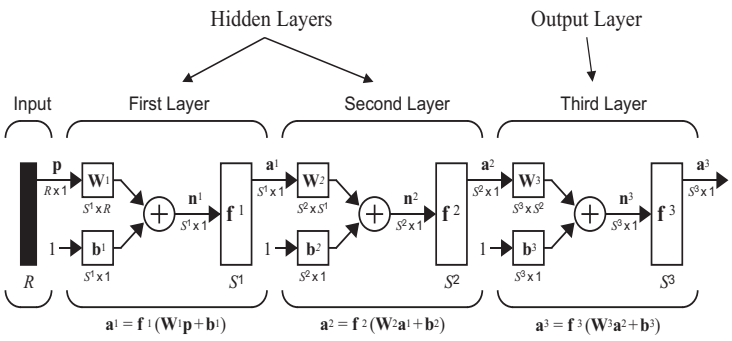


$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

HPSI
Human Performance Systems, Inc.

Abbreviated notation

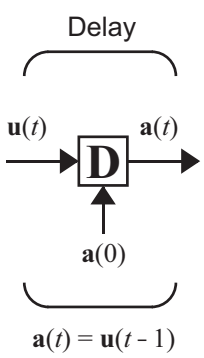


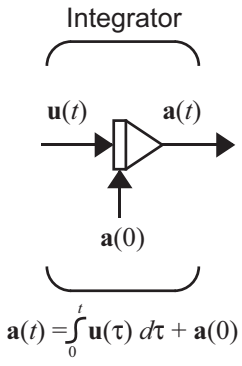
$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

HPSI
Human Performance Systems, Inc.

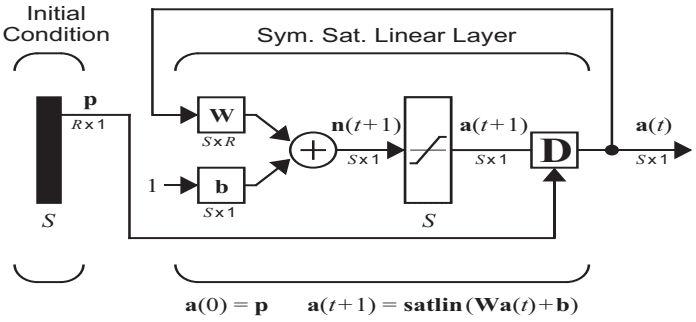
Delays and integrators





HPSI Human Performance Systems, Inc.

Recurrent network

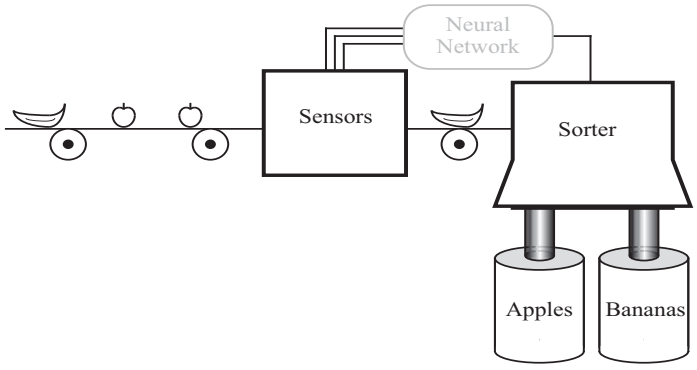


$$\mathbf{a}(1) = \mathbf{satlins}(\mathbf{Wa}(0) + \mathbf{b}) = \mathbf{satlins}(\mathbf{Wp} + \mathbf{b})$$

$$\mathbf{a}(2) = \mathbf{satlins}(\mathbf{Wa}(1) + \mathbf{b})$$

HPSI Human Performance Systems, Inc.

Apple banana sorter



HPSI Human Performance Systems, Inc.

Prototype vectors

Measurement Vector

$$\mathbf{p} = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix}$$

Shape: {1 : round ; -1 : eliptical}
Texture: {1 : smooth ; -1 : rough}
Weight: {1 : > 1 lb. ; -1 : < 1 lb.}

Prototype Banana

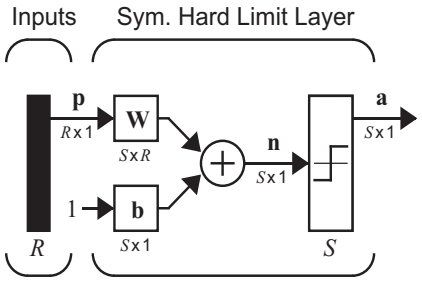
$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

Prototype Apple

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

HPSI Human Performance Systems, Inc.

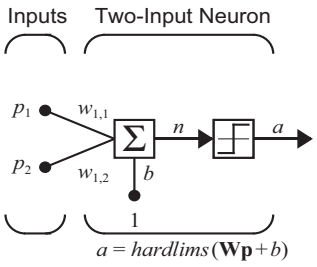
Perceptron



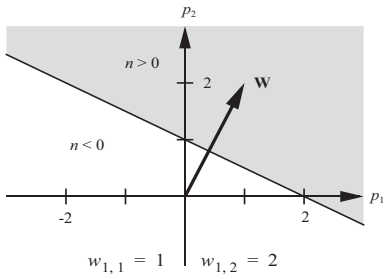
$$\mathbf{a} = \mathbf{hardlims}(\mathbf{Wp} + \mathbf{b})$$

HPSI Human Performance Systems, Inc.

Two input case



$$a = hardlims(\mathbf{Wp} + b)$$



$$a = hardlims(n) = hardlims(\begin{bmatrix} 1 & 2 \end{bmatrix}\mathbf{p} + (-2))$$

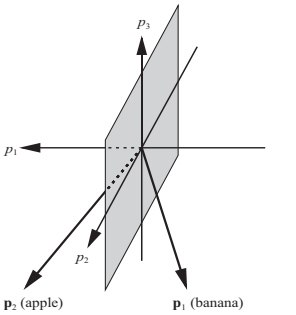
Decision Boundary

$$\mathbf{Wp} + b = 0 \qquad \begin{bmatrix} 1 & 2 \end{bmatrix}\mathbf{p} + (-2) = 0$$

HPSI Human Performance Systems, Inc.

Apple banana example

$$a = hardlims\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b\right)$$



- The decision boundary should separate the prototype vectors
- $p_1 = 0$
- The weight vector should be orthogonal to the decision boundary, and should point in the direction of the vector which should produce an output of 1. The bias determines the position of the boundary.
- $\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$

HPSI Human Performance Systems, Inc.

Testing the network

Banana:

$$a = hardlims\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0\right) = 1(\text{banana})$$

Apple:

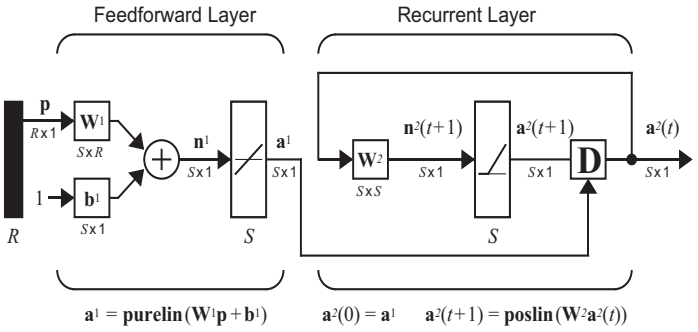
$$a = hardlims\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0\right) = -1\ (\text{apple})$$

"Rough" Banana:

$$a = hardlims\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0\right) = 1(\text{banana})$$

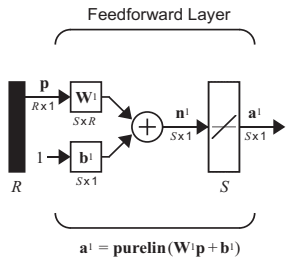
HPSI Human Performance Systems, Inc.

Hamming Network



HPSI Human Performance Systems, Inc.

Feedforward Layer



For Banana/Apple Recognition

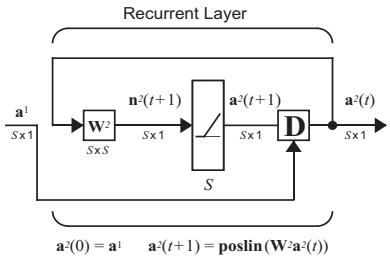
$$S = 2$$

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{b}^1 = \begin{bmatrix} R \\ R \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + 3 \\ \mathbf{p}_2^T\mathbf{p} + 3 \end{bmatrix}$$

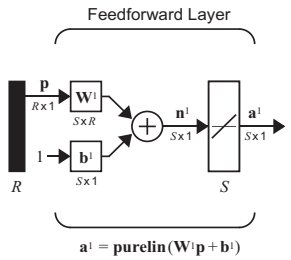
Recurrent Layer



$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} \qquad \varepsilon < \frac{1}{S-1}$$

$$\mathbf{a}^2(t+1) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}\mathbf{a}^2(t)\right) = \mathbf{poslin}\left(\begin{bmatrix} a_1^2(t) - \varepsilon a_2^2(t) \\ a_2^2(t) - \varepsilon a_1^2(t) \end{bmatrix}\right)$$

Hamming Operation



For Banana/Apple Recognition

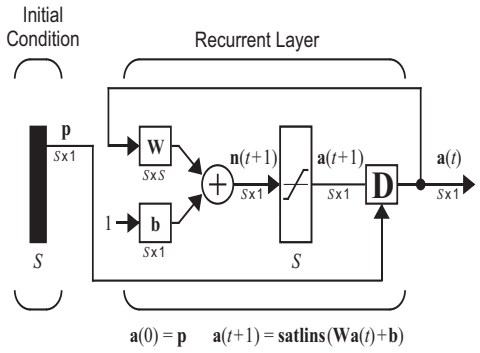
$$S = 2$$

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{b}^1 = \begin{bmatrix} R \\ R \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + 3 \\ \mathbf{p}_2^T\mathbf{p} + 3 \end{bmatrix}$$

Hopfield Network



HPSI Human Performance Systems, Inc.

Apple/Banana Problem

$$\mathbf{W} = \begin{bmatrix} 1.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 0.9 \\ -0.9 \end{bmatrix}$$

$$a_1(t+1) = satlins(1.2a_1(t))$$

$$a_2(t+1) = satlins(0.2a_2(t) + 0.9)$$

$$a_3(t+1) = \text{satlins}(0.2a_3(t) - 0.9)$$

Test: “Rough” Banana

$$\mathbf{a}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \quad \mathbf{a}(1) = \begin{bmatrix} -1 \\ 0.7 \\ -1 \end{bmatrix} \quad \mathbf{a}(2) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \quad \mathbf{a}(3) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \text{ (Banana)}$$

HPSI Human Performance Systems, Inc.

Summary

- Perceptron
  - Feedforward Network
  - Linear Decision Boundary
  - One neuron for each decision
- Hamming Network
  - Competitive Network
  - First Layer - Pattern Matching (Inner Product)
  - Second Layer - Competition (Winner - Take - All)
  - # Neurons = # Prototype Patterns
- Hopfield Network
  - Dynamic Associative Memory Network
  - Network Output Converges to Prototype Pattern
  - # Neurons = # Elements is each Prototype Pattern

HPSI Human Performance Systems, Inc.

Learning rules

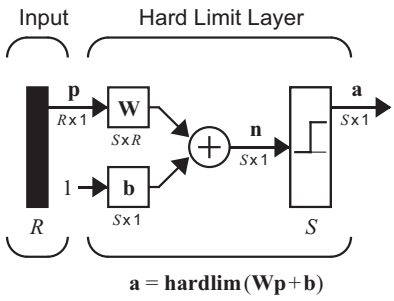
- Supervised learning: Network is provided with a set of examples of proper network behavior (inputs and targets)
  $\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\}$

- Reinforcement learning: Network is only provided with a grade, or score, which indicates network performance.

- Unsupervised learning: Only network inputs are available to the learning algorithm. Network learns to categorize (cluster) the inputs.

HPSI
Human Performance Systems, Inc.

Perceptron architecture



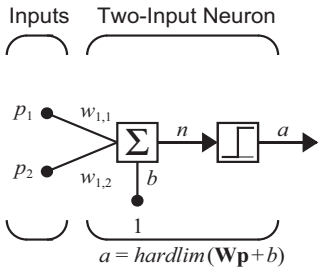
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$$_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

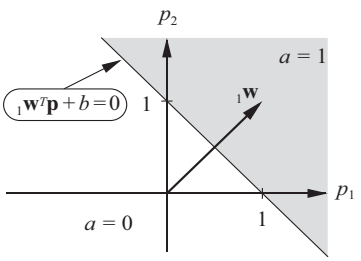
$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

HPSI
Human Performance Systems, Inc.

Single neuron perceptron



$$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$$

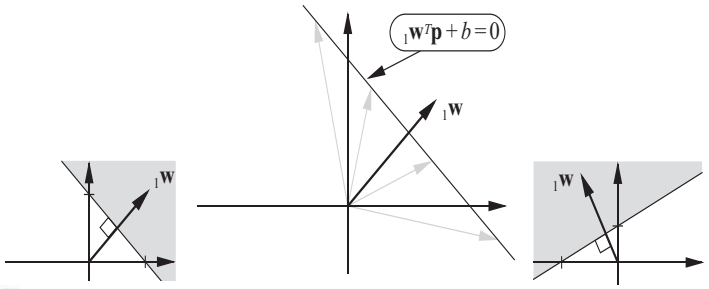


$$a = hardlim({}_1\mathbf{w}^T\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

HPSI
Human Performance Systems, Inc.

Decision boundary

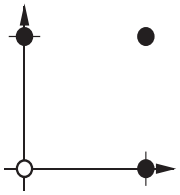
- ${}_1\mathbf{w}^T\mathbf{p} + b = 0,\ {}_1\mathbf{w}^T\mathbf{p} = b$
- All points on the decision boundary have the same inner product with the weight vector.
- Therefore they have the same projection onto the weight vector, and they must lie on a line orthogonal tot he weight vector



HPSI Human Performance Systems, Inc.

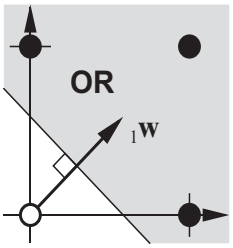
Example OR

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \quad \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 1\right\}$$



HPSI Human Performance Systems, Inc.

OR solution



- Weight vector should be orthogonal to the decision boundary.
- ${}_i\mathbf{w} = \begin{bmatrix}0.5\\0.5\end{bmatrix}$
- Pick a point on the decision boundary to find the bias.
- ${}_i\mathbf{w}\mathbf{p} + b = [0.50.5]\begin{bmatrix}0\\0.5\end{bmatrix} + b = 0.25 + b = 0 \Rightarrow b = -0.25$

HPSI Human Performance Systems, Inc.

Multiple neuron perceptron

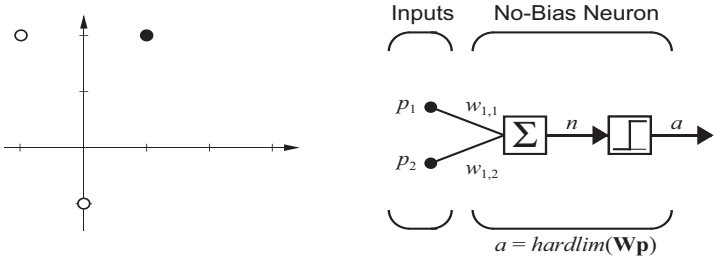
- Each neuron will have its own decision boundary.
- ${}_i\mathbf{w}^T\mathbf{p} + b_i = 0$
- A single neuron can classify input vectors into two categories.
- A multi neuron perceptron can classify input vectors into 2^S categories.

HPSI Human Performance Systems, Inc.

Learning rule test problem

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$

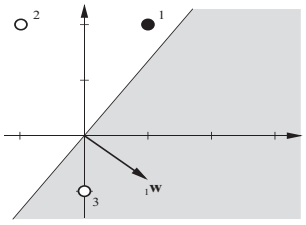


HPSI Human Performance Systems, Inc.

Starting point

Random initial weight:

$${}_1\mathbf{w} = \begin{bmatrix}1.0\\-0.8\end{bmatrix}$$



Present $\mathbf{p}_1$ to the network:

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix}1.0 & -0.8\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}\right)$$

$$a = hardlim(-0.6) = 0$$

Incorrect Classification.

HPSI Human Performance Systems, Inc.

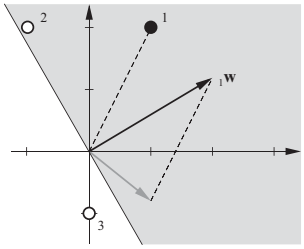
Tentative learning rule

Set ${}_1\mathbf{w}$ to $\mathbf{p}_1$ – Not stable ×

Add $\mathbf{p}_1$ to ${}_1\mathbf{w}$ √

Tentative Rule: If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



HPSI Human Performance Systems, Inc.

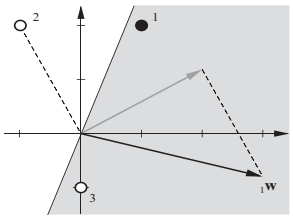
Second input vector

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_2) = hardlim\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right)$$

$$a = hardlim(0.4) = 1 \quad \text{(Incorrect Classification)}$$

Modification to Rule: If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$



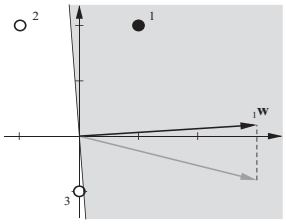
HPSI Human Performance Systems, Inc.

Third input vector

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_3) = hardlim\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix}\right)$$

$$\mathrm{a} = \mathrm{hardlim}(0.8) = 1 \quad \text{(Incorrect Classification)}$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$



Patterns are now correctly classified.

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

HPSI Human Performance Systems, Inc.

Unified learning rule

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p} $$

$$ b^{new} = b^{old} + e $$

A bias is a weight with an input of 1.

Multiple neuron perceptrons

To update the ith row of the weight matrix:

$$ {}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p} $$

$$ b_i^{new} = b_i^{old} + e_i $$

Matrix form:

$$ \mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T $$

$$ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e} $$

Apple banana example

Training Set

$$ \left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, t_1 = \begin{bmatrix}1\end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, t_2 = \begin{bmatrix}0\end{bmatrix}\right\} $$

Initial Weights

$$ \mathbf{W} = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} \qquad b = 0.5 $$

First Iteration

$$ a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right) $$

$$ a = hardlim(-0.5) = 0 \qquad e = t_1 - a = 1 - 0 = 1 $$

$$ \mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} + (1)\begin{bmatrix}-1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix} $$

$$ b^{new} = b^{old} + e = 0.5 + (1) = 1.5 $$

Second iteration

$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + (1.5)\right)$$

$$a = hardlim(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix} + (-1)\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

HPSI Human Performance Systems, Inc.

Check

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(1.5) = 1 = t_1$$

$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(-1.5) = 0 = t_2$$

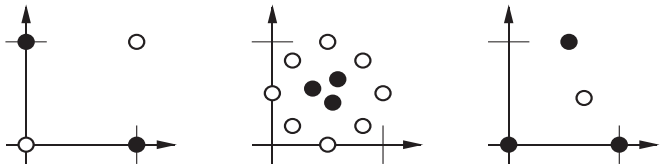
HPSI Human Performance Systems, Inc.

Perceptron limitations

Linear Decision Boundary

$${}_1\mathbf{w}^T\mathbf{p} + b = 0$$

Linearly Inseparable Problems



HPSI Human Performance Systems, Inc.

Python implementation

- Some library has to be imported to implement the perceptron learning rule.

```
>>>import numpy as np
>>>import pandas as pd
>>>import matplotlib.pyplot as plt
>>>ppn = Perceptron(eta=0.1, n_iter=10)
>>>ppn.fit(X, y)
```

HPSI Human Performance Systems, Inc.

Summary

- We were touching the main concepts behind neural network in this lecture.
- Neural network is the back bone of machine learning algorithms.
- This lecture just briefly introduces the main and basic concepts of neural network.
- Read chapter 1 through 4 again and look at the solved problems from the following book, Neural Network Design.
- We are going to use these concepts in this lecture for the first mini project.

HPSI Human Performance Systems, Inc.

Taylor series expansion

$$F(x) = F(x^*) + \frac{d}{dx}F(x)\Big|_{x=x^*}(x-x^*) + \frac{1}{2}\frac{d^2}{dx^2}F(x)\Big|_{x=x^*}(x-x^*)^2 + \cdots + \frac{1}{n!}\frac{d^n}{dx^n}F(x)\Big|_{x=x^*}(x-x^*)^n + \cdots$$

HPSI Human Performance Systems, Inc.

Example

$$F(x) = e^{-x}$$

Taylor series of $F(x)$ about $x^* = 0$:

$$F(x) = e^{-x} = e^{-0} - e^{-0}(x-0) + \frac{1}{2}e^{-0}(x-0)^2 - \frac{1}{6}e^{-0}(x-0)^3 + \ldots$$

$$F(x) = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \ldots$$

Taylor series approximations:

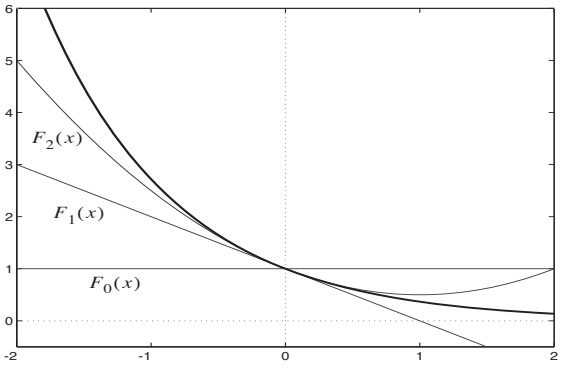
$$F(x) \approx F_0(x) = 1$$

$$F(x) \approx F_1(x) = 1 - x$$

$$F(x) \approx F_2(x) = 1 - x + \frac{1}{2}x^2$$

HPSI
Human Performance Systems, Inc.

Plot of approximations



HPSI
Human Performance Systems, Inc.

Vector case

$$F(\mathbf{x}) = F(x_1, x_2, \ldots, x_n)$$

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \frac{\partial}{\partial x_1}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_1 - x_1^*) + \frac{\partial}{\partial x_2}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_2 - x_2^*)$$

$$+ \cdots + \frac{\partial}{\partial x_n}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_n - x_n^*) + \frac{1}{2}\frac{\partial^2}{\partial x_1^2}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_1 - x_1^*)^2$$

$$+ \frac{1}{2}\frac{\partial^2}{\partial x_1 \partial x_2}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_1 - x_1^*)(x_2 - x_2^*) + \cdots$$

HPSI
Human Performance Systems, Inc.

Matrix form

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \cdots$$

Gradient

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n}F(\mathbf{x}) \end{bmatrix}$$

Hessian

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n}F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix}$$

HPSI Human Performance Systems, Inc.

Directional derivatives

- First derivative (slope) of $F(\mathbf{x})$ along x_i axis: $\frac{\sigma F(\mathbf{x})}{\sigma x_i}$ - (i th element of gradient).
- Second derivative (curvature) of $F(\mathbf{x})$ along x_i axis: $\frac{\sigma^2 F(\mathbf{x})}{\sigma x_i^2}$ - (i, i th element of Hesian)
- First derivative (slope) of $F(\mathbf{x})$ along $\mathbf{p}$: $\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{||\mathbf{p}}$
- Second derivative (curvature) of $F(\mathbf{x})$ along $\mathbf{p}$: $\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x})}{||\mathbf{p}^2}$

HPSI Human Performance Systems, Inc.

Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2$$

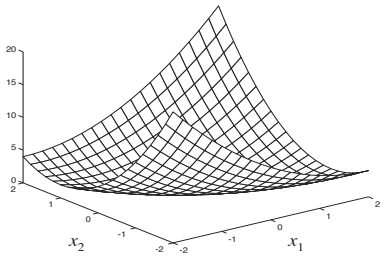
$$\mathbf{x}^* = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \qquad \mathbf{p} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

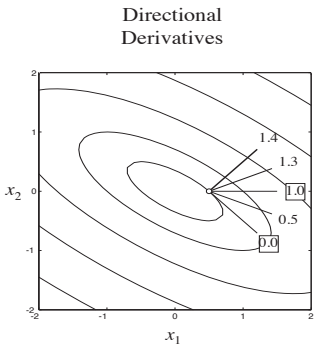
$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \end{bmatrix}\Bigg|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 2x_1 + 2x_2 \\ 2x_1 + 4x_2 \end{bmatrix}\Bigg|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|} = \frac{\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}}{\left\|\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\|} = \frac{\begin{bmatrix} 0 \end{bmatrix}}{\sqrt{2}} = 0$$

HPSI Human Performance Systems, Inc.

Plots





Minima

Strong Minimum:

- The point $\mathbf{x}^*$ is a strong minimum of $F(\mathbf{x})$ if a scalar $\delta > 0$ exists, such that $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$ such that $\delta > ||\Delta\mathbf{x}|| > 0$.

Global Minimum:

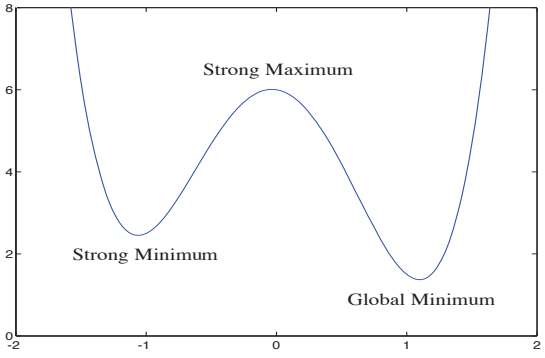
- The point $\mathbf{x}^*$ is a unique global minimum of $F(\mathbf{x})$ if $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$ for all $\Delta\mathbf{x} \neq 0$.

Weak Minimum:

- The point $\mathbf{x}^*$ is a weak minimum of $F(\mathbf{x})$ if it is not a strong minimum, and scalar $\delta > 0$ exists such that $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $\delta > ||\Delta\mathbf{x}|| > 0$.

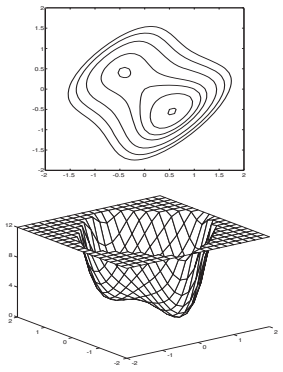
Scalar example

$$F(x) = 3x^4 - 7x^2 - \frac{1}{2}x + 6$$

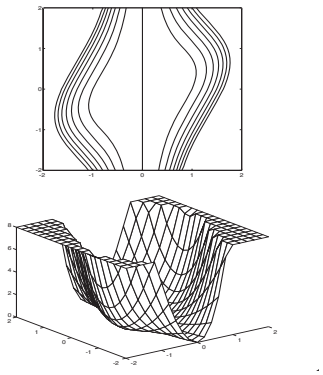


Vector Example

$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$



$F(\mathbf{x}) = (x_1^2 - 1.5x_1x_2 + 2x_2^2)x_1^2$



HPSI
Human Performance Systems, Inc.

Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x}^* = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

(Not a function of $\mathbf{x}$ in this case.)

To test the definiteness, check the eigenvalues of the Hessian. If the eigenvalues are all greater than zero, the Hessian is positive definite.

$$|\nabla^2 F(\mathbf{x}) - \lambda \mathbf{I}| = \left| \begin{bmatrix} 2-\lambda & 2 \\ 2 & 4-\lambda \end{bmatrix} \right| = \lambda^2 - 6\lambda + 4 = (\lambda - 0.76)(\lambda - 5.24)$$

$\lambda = 0.76, 5.24$ Both eigenvalues are positive, therefore strong minimum.

HPSI
Human Performance Systems, Inc.

Quadratic functions

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c \qquad \text{(Symmetric } \mathbf{A}\text{)}$$

Gradient and Hessian:

Useful properties of gradients:

$$\nabla(\mathbf{h}^T\mathbf{x}) = \nabla(\mathbf{x}^T\mathbf{h}) = \mathbf{h}$$

$$\nabla\mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{Q}^T\mathbf{x} = 2\mathbf{Q}\mathbf{x} \text{ (for symmetric } \mathbf{Q}\text{)}$$

Gradient of Quadratic Function:

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}$$

Hessian of Quadratic Function:

$$\nabla^2 F(\mathbf{x}) = \mathbf{A}$$

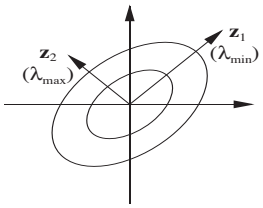
HPSI
Human Performance Systems, Inc.

Eigenvector

$$\mathbf{p} = \mathbf{z}_{max} \qquad \mathbf{c} = \mathbf{B}^T\mathbf{p} = \mathbf{B}^T\mathbf{z}_{max} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\frac{\mathbf{z}_{max}^T \mathbf{A}\mathbf{z}_{max}}{\|\mathbf{z}_{max}\|^2} = \frac{\sum_{i=1}^{n} \lambda_i c_i^2}{\sum_{i=1}^{n} c_i^2} = \lambda_{max}$$

The eigenvalues represent curvature (second derivatives) along the eigenvectors (the principal axes).



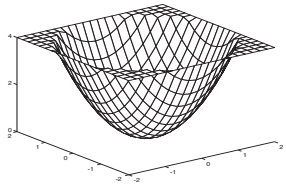
HPSI
Human Performance Systems, Inc.

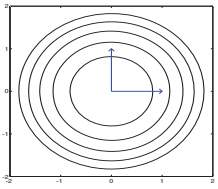
Circular hollow

$$F(\mathbf{x}) = x_1^2 + x_2^2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{x}$$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad \lambda_1 = 2 \qquad \mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \lambda_2 = 2 \qquad \mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

(Any two independent vectors in the plane would work.)



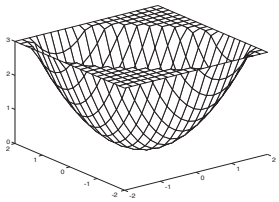


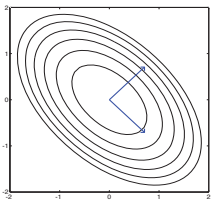
HPSI
Human Performance Systems, Inc.

Elliptical hollow

$$F(\mathbf{x}) = x_1^2 + x_1 x_2 + x_2^2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad \lambda_1 = 1 \qquad \mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \lambda_2 = 3 \qquad \mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



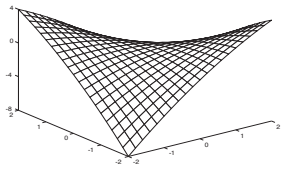


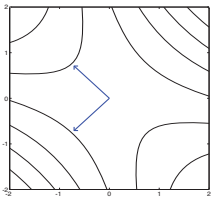
HPSI
Human Performance Systems, Inc.

Elongated saddle

$$F(\mathbf{x}) = -\frac{1}{4}x_1^2 - \frac{3}{2}x_1x_2 - \frac{1}{4}x_2^2 = \frac{1}{2}\mathbf{x}^T\begin{bmatrix}-0.5 & -1.5\\ -1.5 & -0.5\end{bmatrix}\mathbf{x}$$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix}-0.5 & -1.5\\ -1.5 & -0.5\end{bmatrix} \quad \lambda_1 = 1 \quad \mathbf{z}_1 = \begin{bmatrix}-1\\ 1\end{bmatrix} \quad \lambda_2 = -2 \quad \mathbf{z}_2 = \begin{bmatrix}-1\\ -1\end{bmatrix}$$



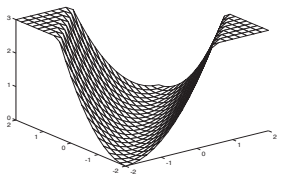


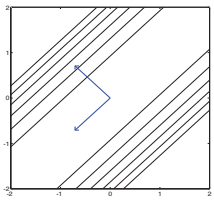
HPSI
Human Performance Systems, Inc.

Stationary Valley

$$F(\mathbf{x}) = \frac{1}{2}x_1^2 - x_1x_2 + \frac{1}{2}x_2^2 = \frac{1}{2}\mathbf{x}^T\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\mathbf{x}$$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} \quad \lambda_1 = 1 \quad \mathbf{z}_1 = \begin{bmatrix}-1\\ 1\end{bmatrix} \quad \lambda_2 = 0 \quad \mathbf{z}_2 = \begin{bmatrix}-1\\ -1\end{bmatrix}$$





HPSI
Human Performance Systems, Inc.

OR solution

- If the eigenvalues of the Hessian matrix are all positive, the function will have a single strong minimum.
- If the eigenvalues are all negative, the function will have a single strong maximum.
- If some eigenvalues are positive and other eigenvalues are negative, the function will have a single saddle point.
- If the eigenvalues are all nonnegative, but some eigenvalues are zero, then the function will either have a weak minimum or will have no stationary point.
- If the eigenvalues are all nonpositive, but some eigenvalues are zero, then the function will either have a weak maximum or will have no stationary point.
- Stationary point: $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{d}$

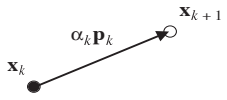
HPSI
Human Performance Systems, Inc.

Basic optimization algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

or

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$



$\mathbf{p}_k$ - Search Direction

α_k - Learning Rate

HPSI
Human Performance Systems, Inc.

Steepest descent

Choose the next step so that the function decreases:

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$$

For small changes in $\mathbf{x}$ we can approximate $F(\mathbf{x})$:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta \mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta \mathbf{x}_k$$

where

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x})\big|_{\mathbf{x} = \mathbf{x}_k}$$

If we want the function to decrease:

$$\mathbf{g}_k^T \Delta \mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$$

We can maximize the decrease by choosing:

$$\mathbf{p}_k = -\mathbf{g}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$$

HPSI
Human Performance Systems, Inc.

Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \qquad \alpha = 0.1$$

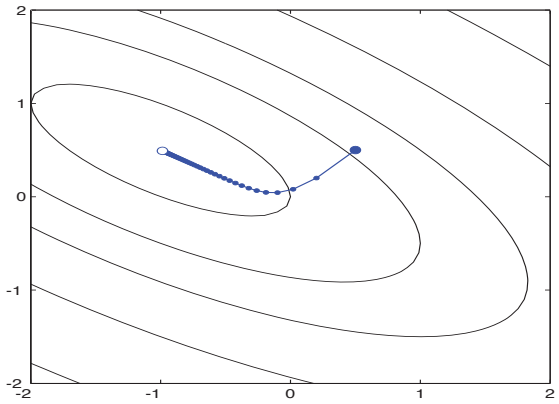
$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \qquad \mathbf{g}_0 = \nabla F(\mathbf{x})\big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} - 0.1 \begin{bmatrix} 1.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$

HPSI
Human Performance Systems, Inc.

Plot



HPSI
Human Performance Systems, Inc.

Newton's method

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{x}_k^T \mathbf{A}_k \Delta\mathbf{x}_k$$

Take the gradient of this second-order approximation and set it equal to zero to find the stationary point:

$$\mathbf{g}_k + \mathbf{A}_k \Delta\mathbf{x}_k = \mathbf{0}$$

$$\Delta\mathbf{x}_k = -\mathbf{A}_k^{-1}\mathbf{g}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k$$

HPSI
Human Performance Systems, Inc.

Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

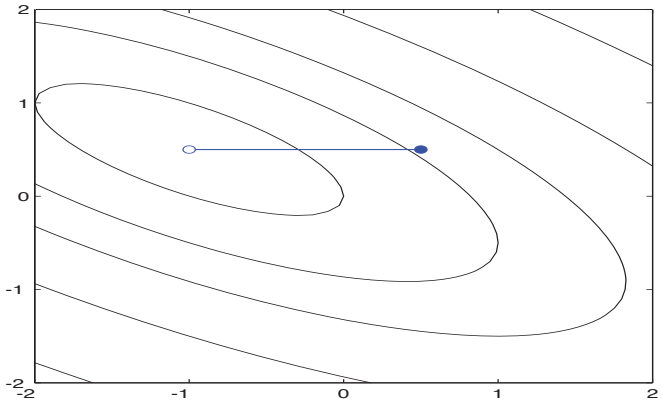
$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \qquad \mathbf{g}_0 = \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\mathbf{x}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 1 & -0.5 \\ -0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

HPSI
Human Performance Systems, Inc.

Plot

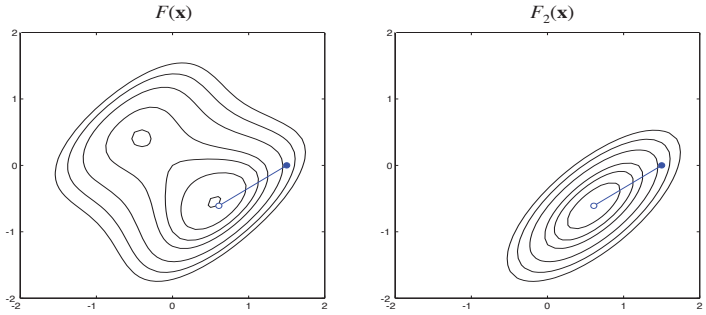


HPSI
Human Performance Systems, Inc.

Non-Quadratic Example

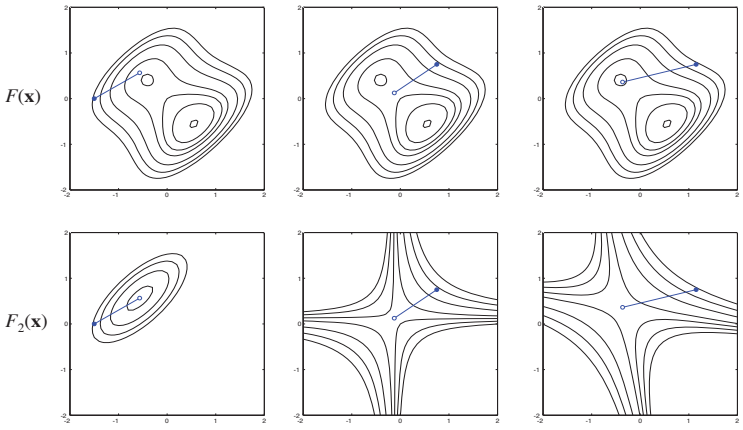
$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$

Stationary Points: $\mathbf{x}^1 = \begin{bmatrix} -0.42 \\ 0.42 \end{bmatrix}$ $\mathbf{x}^2 = \begin{bmatrix} -0.13 \\ 0.13 \end{bmatrix}$ $\mathbf{x}^3 = \begin{bmatrix} 0.55 \\ -0.55 \end{bmatrix}$



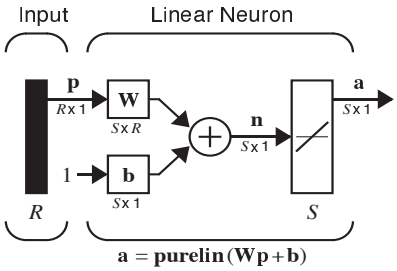
HPSI
Human Performance Systems, Inc.

Different Initial Conditions



HPSI
Human Performance Systems, Inc.

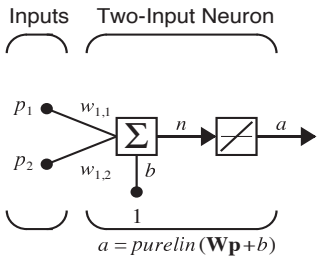
ADALINE Network

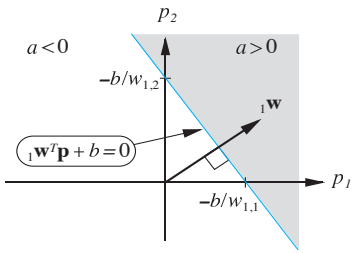


$$\mathbf{a} = \mathbf{purelin}(\mathbf{Wp}+\mathbf{b}) = \mathbf{Wp}+\mathbf{b}$$

$$a_i = purelin(n_i) = purelin({}_i\mathbf{w}^T\mathbf{p}+b_i) = {}_i\mathbf{w}^T\mathbf{p}+b_i \qquad {}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

Two input ADALINE





$$a = purelin(n) = purelin({}_1\mathbf{w}^T\mathbf{p}+b) = {}_1\mathbf{w}^T\mathbf{p}+b$$

$$a = {}_1\mathbf{w}^T\mathbf{p}+b = w_{1,1}p_1 + w_{1,2}p_2 + b$$

Mean square error

Training Set:

$$\{\mathbf{p}_1,\mathbf{t}_1\},\{\mathbf{p}_2,\mathbf{t}_2\},\ldots,\{\mathbf{p}_Q,\mathbf{t}_Q\}$$

Input: $\mathbf{p}_q$ Target: $\mathbf{t}_q$

Notation:

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \qquad a = {}_1\mathbf{w}^T\mathbf{p}+b \Rightarrow a = \mathbf{x}^T\mathbf{z}$$

Mean Square Error:

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

Error analysis

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$$F(\mathbf{x}) = E[t^2 - 2t\mathbf{x}^T\mathbf{z} + \mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x}]$$

$$F(\mathbf{x}) = E[t^2] - 2\mathbf{x}^T E[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z}\mathbf{z}^T]\mathbf{x}$$

$$\boxed{F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}}$$

$$c = E[t^2] \qquad \mathbf{h} = E[t\mathbf{z}] \qquad \mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$$

The mean square error for the ADALINE Network is a quadratic function:

$$F(\mathbf{x}) = c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$$

$$\mathbf{d} = -2\mathbf{h} \qquad \mathbf{A} = 2\mathbf{R}$$

Approximate Steepest Descent

Approximate mean square error (one sample):

$$\hat{F}(\mathbf{x}) = (t(k) - a(k))^2 = e^2(k)$$

Approximate (stochastic) gradient:

$$\hat{\nabla}F(\mathbf{x}) = \nabla e^2(k)$$

$$[\nabla e^2(k)]_j = \frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} \qquad j = 1, 2, \ldots, R$$

$$[\nabla e^2(k)]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}$$

Approximate Gradient Calculation

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial [t(k) - a(k)]}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}}[t(k) - ({}_1\mathbf{w}^T\mathbf{p}(k) + b)]$$

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}}\left[t(k) - \left(\sum_{i=1}^{R} w_{1,i}p_i(k) + b\right)\right]$$

$$\frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k) \qquad \frac{\partial e(k)}{\partial b} = -1$$

$$\hat{\nabla}F(\mathbf{x}) = \nabla e^2(k) = -2e(k)\mathbf{z}(k)$$

LMS Algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$$

$${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k)$$

$$b(k+1) = b(k) + 2\alpha e(k)$$

Multiple-Neuron Case

$${}_i\mathbf{w}(k+1) = {}_i\mathbf{w}(k) + 2\alpha e_i(k)\mathbf{p}(k)$$

$$b_i(k+1) = b_i(k) + 2\alpha e_i(k)$$

Matrix Form:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha \mathbf{e}(k)\mathbf{p}^T(k)$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha \mathbf{e}(k)$$

Example

Banana $\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, \mathbf{t}_1 = [-1]\right\}$ Apple $\left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, \mathbf{t}_2 = [1]\right\}$

$$\mathbf{R} = E[\mathbf{p}\mathbf{p}^T] = \frac{1}{2}\mathbf{p}_1\mathbf{p}_1^T + \frac{1}{2}\mathbf{p}_2\mathbf{p}_2^T$$

$$\mathbf{R} = \frac{1}{2}\begin{bmatrix}-1\\1\\-1\end{bmatrix}\begin{bmatrix}-1 & 1 & -1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1\\1\\-1\end{bmatrix}\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\0 & 1 & -1\\0 & -1 & 1\end{bmatrix}$$

$$\lambda_1 = 1.0, \qquad \lambda_2 = 0.0, \qquad \lambda_3 = 2.0$$

$$\alpha < \frac{1}{\lambda_{max}} = \frac{1}{2.0} = 0.5$$

Iteration One

Banana $a(0) = \mathbf{W}(0)\mathbf{p}(0) = \mathbf{W}(0)\mathbf{p}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = 0$

$$e(0) = t(0) - a(0) = t_1 - a(0) = -1 - 0 = -1$$

$$\mathbf{W}(1) = \mathbf{W}(0) + 2\alpha e(0)\mathbf{p}^T(0)$$

$$\mathbf{W}(1) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} + 2(0.2)(-1) \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}^T = \begin{bmatrix} 0.4 & -0.4 & 0.4 \end{bmatrix}$$

Iteration Two

Apple $a(1) = \mathbf{W}(1)\mathbf{p}(1) = \mathbf{W}(1)\mathbf{p}_2 = \begin{bmatrix} 0.4 & -0.4 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = -0.4$

$$e(1) = t(1) - a(1) = t_2 - a(1) = 1 - (-0.4) = 1.4$$

$$\mathbf{W}(2) = \begin{bmatrix} 0.4 & -0.4 & 0.4 \end{bmatrix} + 2(0.2)(1.4) \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}^T = \begin{bmatrix} 0.96 & 0.16 & -0.16 \end{bmatrix}$$

Iteration three

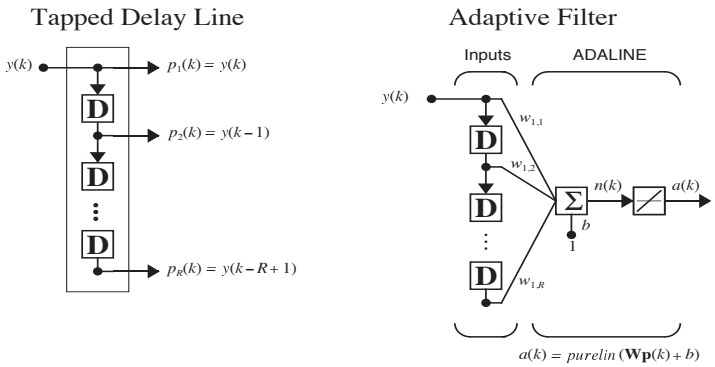
$$a(2) = \mathbf{W}(2)\mathbf{p}(2) = \mathbf{W}(2)\mathbf{p}_1 = \begin{bmatrix} 0.96 & 0.16 & -0.16 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = -0.64$$

$$e(2) = t(2) - a(2) = t_1 - a(2) = -1 - (-0.64) = -0.36$$

$$\mathbf{W}(3) = \mathbf{W}(2) + 2\alpha e(2)\mathbf{p}^T(2) = \begin{bmatrix} 1.1040 & 0.0160 & -0.0160 \end{bmatrix}$$

$$\mathbf{W}(\infty) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

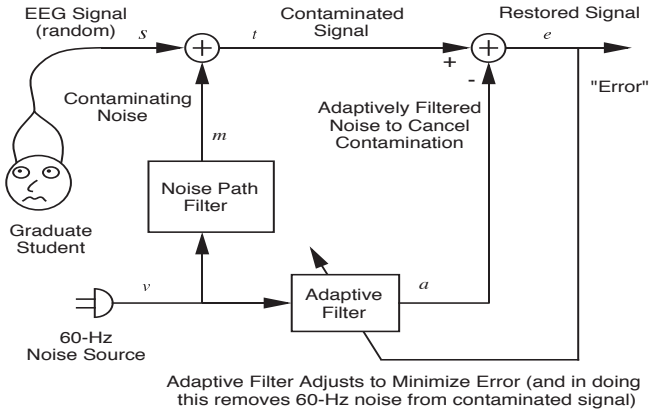
Adaptive Filtering



$$a(k) = purelin(\mathbf{W}\mathbf{p} + b) = \sum_{i=1}^{R} w_{1,i} y(k - i + 1) + b$$

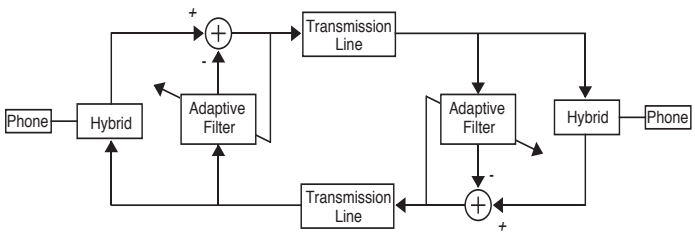
HPSI Human Performance Systems, Inc.

Example: Noise Cancellation



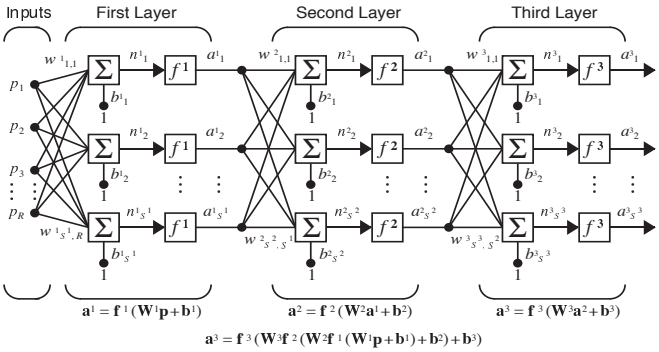
HPSI Human Performance Systems, Inc.

Echo Cancellation



HPSI Human Performance Systems, Inc.

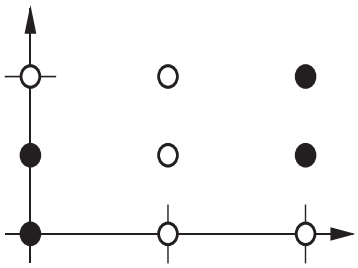
Multilayer Perceptron



$R - S^1 - S^2 - S^3$ Network

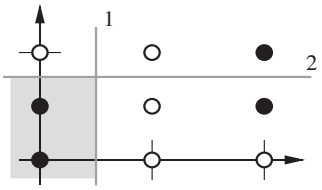
HPSI Human Performance Systems, Inc.

Example



HPSI Human Performance Systems, Inc.

Elementary Decision Boundaries



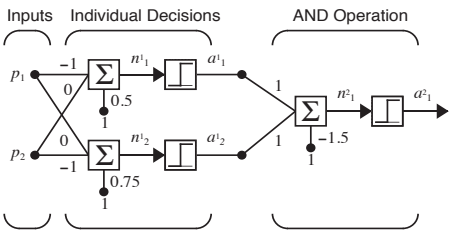
First Boundary:

$$a_1^1 = hardlim(\begin{bmatrix}-1 & 0\end{bmatrix}\mathbf{p} + 0.5)$$

Second Boundary:

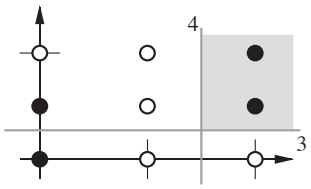
$$a_2^1 = hardlim(\begin{bmatrix}0 & -1\end{bmatrix}\mathbf{p} + 0.75)$$

First Subnetwork



HPSI Human Performance Systems, Inc.

Elementary Decision Boundaries



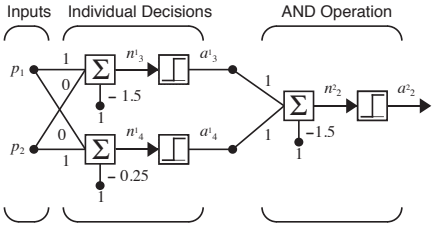
Third Boundary:

$$a_3^1 = hardlim(\begin{bmatrix}1 & 0\end{bmatrix}\mathbf{p} - 1.5)$$

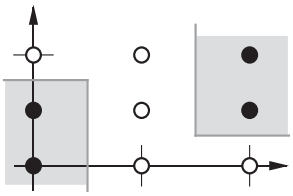
Fourth Boundary:

$$a_4^1 = hardlim(\begin{bmatrix}0 & 1\end{bmatrix}\mathbf{p} - 0.25)$$

Second Subnetwork



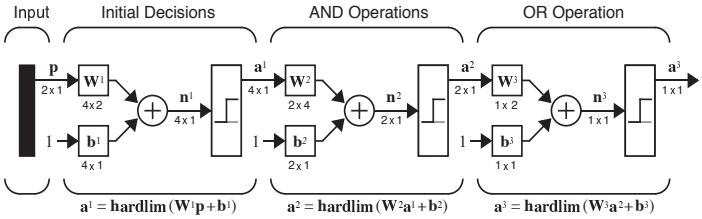
Total Network



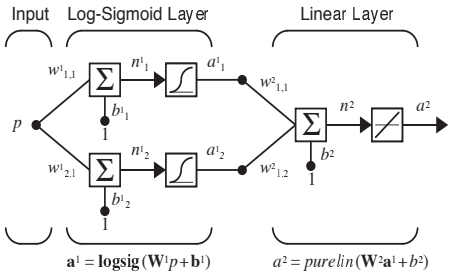
$$\mathbf{W}^1 = \begin{bmatrix}-1 & 0\\ 0 & -1\\ 1 & 0\\ 0 & 1\end{bmatrix} \quad \mathbf{b}^1 = \begin{bmatrix}0.5\\ 0.75\\ -1.5\\ -0.25\end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\end{bmatrix} \quad \mathbf{b}^2 = \begin{bmatrix}-1.5\\ -1.5\end{bmatrix}$$

$$\mathbf{W}^3 = \begin{bmatrix}1 & 1\end{bmatrix} \quad \mathbf{b}^3 = \begin{bmatrix}-0.5\end{bmatrix}$$



Function Approximation Example



$$f^1(n) = \frac{1}{1+e^{-n}}$$

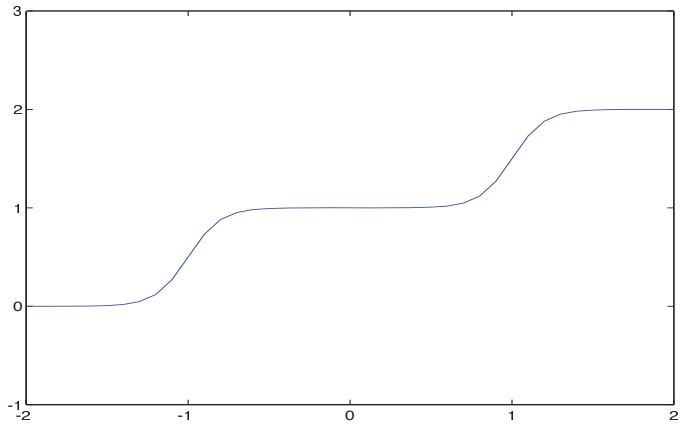
$$f^2(n) = n$$

Nominal Parameter Values

$$w^1_{1,1} = 10 \qquad w^1_{2,1} = 10 \qquad b^1_1 = -10 \qquad b^1_2 = 10$$

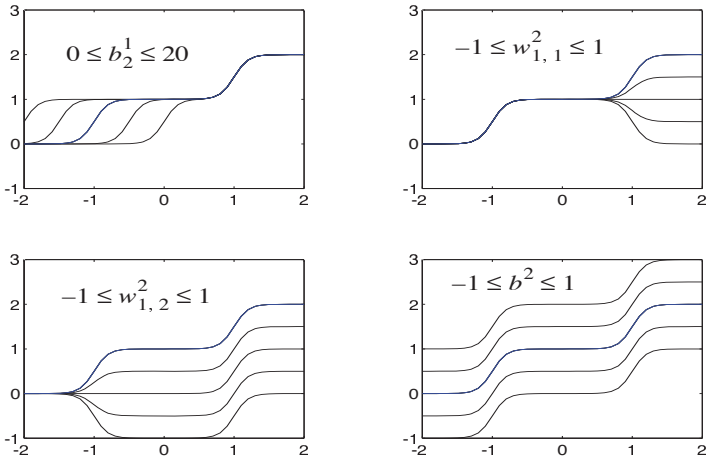
$$w^2_{1,1} = 1 \qquad w^2_{1,2} = 1 \qquad b^2 = 0$$

Nominal Response



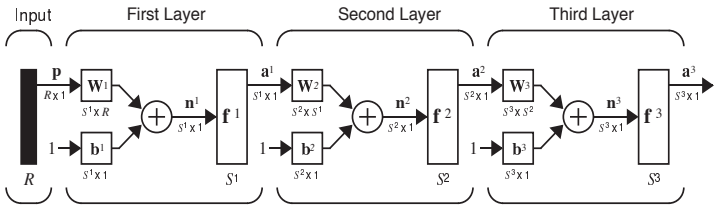
HPSI
Human Performance Systems, Inc.

Parameter Variations



HPSI
Human Performance Systems, Inc.

Multilayer Network



$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}) \qquad m = 0, 2, \ldots, M-1$$

$$\mathbf{a}^{0} = \mathbf{p}$$

$$\mathbf{a} = \mathbf{a}^{M}$$

HPSI
Human Performance Systems, Inc.

Performance Index

Training Set

$$\{\mathbf{p}_1,\mathbf{t}_1\},\{\mathbf{p}_2,\mathbf{t}_2\},\ldots,\{\mathbf{p}_Q,\mathbf{t}_Q\}$$

Mean Square Error

$$F(\mathbf{x})= E[e^2] = E[(t-a)^2]$$

Vector Case

$$F(\mathbf{x})= E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})]$$

Approximate Mean Square Error (Single Sample)

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k)-\mathbf{a}(k))^T(\mathbf{t}(k)-\mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k)$$

Approximate Steepest Descent

$$w^m_{i,j}(k+1) = w^m_{i,j}(k)-\alpha\frac{\partial\hat{F}}{\partial w^m_{i,j}} \qquad b^m_i(k+1) = b^m_i(k)-\alpha\frac{\partial\hat{F}}{\partial b^m_i}$$

Chain Rule

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn}\times\frac{dn(w)}{dw}$$

Example

$$f(n) = \cos(n) \qquad n = e^{2w} \qquad f(n(w)) = \cos(e^{2w})$$

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn}\times\frac{dn(w)}{dw} = (-\sin(n))(2e^{2w}) = (-\sin(e^{2w}))(2e^{2w})$$

Application to Gradient Calculation

$$\frac{\partial\hat{F}}{\partial w^m_{i,j}} = \frac{\partial\hat{F}}{\partial n^m_i}\times\frac{\partial n^m_i}{\partial w^m_{i,j}} \qquad \frac{\partial\hat{F}}{\partial b^m_i} = \frac{\partial\hat{F}}{\partial n^m_i}\times\frac{\partial n^m_i}{\partial b^m_i}$$

Gradient Calculation

$$n^m_i = \sum_{j=1}^{S^{m-1}} w^m_{i,j}a^{m-1}_j + b^m_i$$

$$\frac{\partial n^m_i}{\partial w^m_{i,j}} = a^{m-1}_j \qquad \frac{\partial n^m_i}{\partial b^m_i} = 1$$

Sensitivity

$$s^m_i \equiv \frac{\partial\hat{F}}{\partial n^m_i}$$

Gradient

$$\frac{\partial\hat{F}}{\partial w^m_{i,j}} = s^m_i a^{m-1}_j \qquad \frac{\partial\hat{F}}{\partial b^m_i} = s^m_i$$

Steepest Descent

$$w^m_{i,j}(k+1) = w^m_{i,j}(k) - \alpha s^m_i a^{m-1}_j \qquad b^m_i(k+1) = b^m_i(k) - \alpha s^m_i$$

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \qquad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n^m_1} \\ \frac{\partial \hat{F}}{\partial n^m_2} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n^m_{S^m}} \end{bmatrix}$$

Next Step: Compute the Sensitivities (Backpropagation)

Jacobian Matrix

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \equiv \begin{bmatrix} \frac{\partial n^{m+1}_1}{\partial n^m_1} & \frac{\partial n^{m+1}_1}{\partial n^m_2} & \cdots & \frac{\partial n^{m+1}_1}{\partial n^m_{S^m}} \\ \frac{\partial n^{m+1}_2}{\partial n^m_1} & \frac{\partial n^{m+1}_2}{\partial n^m_2} & \cdots & \frac{\partial n^{m+1}_2}{\partial n^m_{S^m}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n^{m+1}_{S^{m+1}}}{\partial n^m_1} & \frac{\partial n^{m+1}_{S^{m+1}}}{\partial n^m_2} & \cdots & \frac{\partial n^{m+1}_{S^{m+1}}}{\partial n^m_{S^m}} \end{bmatrix}$$

$$\frac{\partial n^{m+1}_i}{\partial n^m_j} = \frac{\partial \left(\sum_{l=1}^{S^m} w^{m+1}_{i,l} a^m_l + b^{m+1}_i \right)}{\partial n^m_j} = w^{m+1}_{i,j} \frac{\partial a^m_j}{\partial n^m_j}$$

$$\frac{\partial n^{m+1}_i}{\partial n^m_j} = w^{m+1}_{i,j} \frac{\partial f^m(n^m_j)}{\partial n^m_j} = w^{m+1}_{i,j} \dot{f}^m(n^m_j)$$

$$\dot{f}^m(n^m_j) = \frac{\partial f^m(n^m_j)}{\partial n^m_j}$$

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \dot{\mathbf{F}}^m(\mathbf{n}^m) \qquad \dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n^m_1) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n^m_2) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n^m_{S^m}) \end{bmatrix}$$

Backpropagation (Sensitivities)

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}}$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

The sensitivities are computed by starting at the last layer, and then propagating backwards through the network to the first layer.

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \ldots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1$$

Initialization (Last Layer)

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M}(t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M}$$

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M)$$

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_i^M)$$

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a})$$

HPSI Human Performance Systems, Inc.

Summary

Forward Propagation

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \qquad m = 0, 2, \ldots, M-1$$

$$\mathbf{a} = \mathbf{a}^M$$

Backpropagation

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a})$$

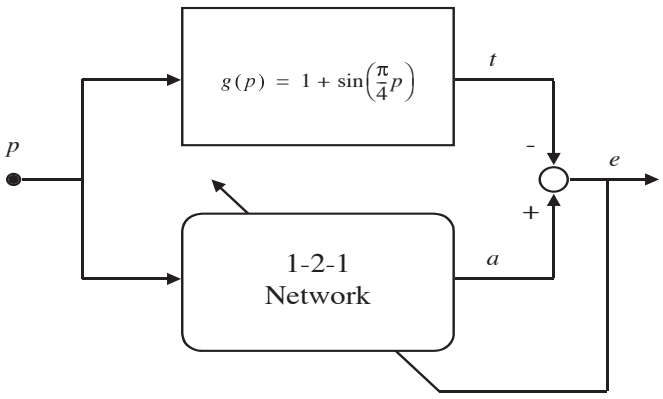
$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1} \qquad m = M-1, \ldots, 2, 1$$

Weight Update

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \qquad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m$$

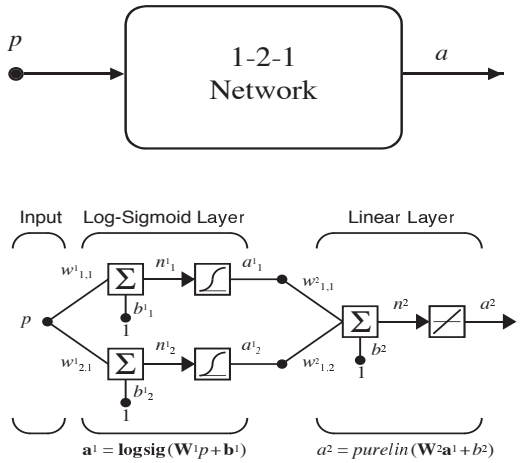
HPSI Human Performance Systems, Inc.

Example: Function Approximation



HPSI Human Performance Systems, Inc.

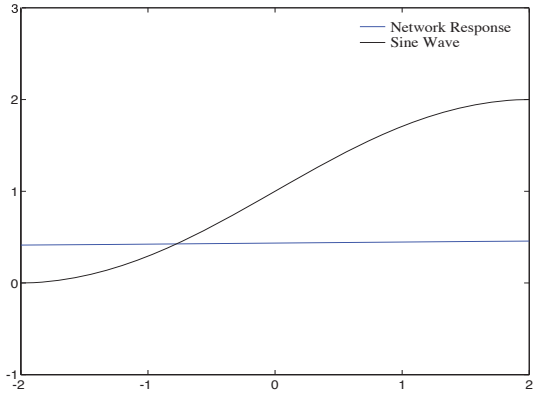
Network



HPSI Human Performance Systems, Inc.

Initial Conditions

$$\mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} \quad \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} \quad \mathbf{W}^2(0) = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix} \quad \mathbf{b}^2(0) = \begin{bmatrix} 0.48 \end{bmatrix}$$



HPSI Human Performance Systems, Inc.

Forward Propagation

$$a^0 = p = 1$$

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1 \mathbf{a}^0 + \mathbf{b}^1) = \mathbf{logsig}\left(\begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix}\begin{bmatrix} 1 \end{bmatrix} + \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}\right) = \mathbf{logsig}\left(\begin{bmatrix} -0.75 \\ -0.54 \end{bmatrix}\right)$$

$$\mathbf{a}^1 = \begin{bmatrix} \frac{1}{1+e^{0.75}} \\ \frac{1}{1+e^{0.54}} \end{bmatrix} = \begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix}$$

$$a^2 = \mathrm{f}^2(\mathbf{W}^2 \mathbf{a}^1 + \mathbf{b}^2) = purelin\left(\begin{bmatrix} 0.09 & -0.17 \end{bmatrix}\begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix} + \begin{bmatrix} 0.48 \end{bmatrix}\right) = \begin{bmatrix} 0.446 \end{bmatrix}$$

$$e = t - a = \left\{1 + \sin\left(\frac{\pi}{4} p\right)\right\} - a^2 = \left\{1 + \sin\left(\frac{\pi}{4} 1\right)\right\} - 0.446 = 1.261$$

HPSI Human Performance Systems, Inc.

Transfer Function Derivatives

$$\dot{f}^1(n) = \frac{d}{dn}\left(\frac{1}{1+e^{-n}}\right) = \frac{e^{-n}}{(1+e^{-n})^2} = \left(1-\frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = (1-a^1)(a^1)$$

$$\dot{f}^2(n) = \frac{d}{dn}(n) = 1$$

Backpropagation

$$\mathbf{s}^2 = -2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t}-\mathbf{a}) = -2\left[\dot{f}^2(n^2)\right](1.261) = -2\left[1\right](1.261) = -2.522$$

$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{W}^2)^T\mathbf{s}^2 = \begin{bmatrix}(1-a_1^1)(a_1^1) & 0\\ 0 & (1-a_2^1)(a_2^1)\end{bmatrix}\begin{bmatrix}0.09\\ -0.17\end{bmatrix}\begin{bmatrix}-2.522\end{bmatrix}$$

$$\mathbf{s}^1 = \begin{bmatrix}(1-0.321)(0.321) & 0\\ 0 & (1-0.368)(0.368)\end{bmatrix}\begin{bmatrix}0.09\\ -0.17\end{bmatrix}\begin{bmatrix}-2.522\end{bmatrix}$$

$$\mathbf{s}^1 = \begin{bmatrix}0.218 & 0\\ 0 & 0.233\end{bmatrix}\begin{bmatrix}-0.227\\ 0.429\end{bmatrix} = \begin{bmatrix}-0.0495\\ 0.0997\end{bmatrix}$$

Weight Update

$$\alpha = 0.1$$

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha\mathbf{s}^2(\mathbf{a}^1)^T = \begin{bmatrix}0.09 & -0.17\end{bmatrix} - 0.1\begin{bmatrix}-2.522\end{bmatrix}\begin{bmatrix}0.321 & 0.368\end{bmatrix}$$

$$\mathbf{W}^2(1) = \begin{bmatrix}0.171 & -0.0772\end{bmatrix}$$

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha\mathbf{s}^2 = \begin{bmatrix}0.48\end{bmatrix} - 0.1\begin{bmatrix}-2.522\end{bmatrix} = \begin{bmatrix}0.732\end{bmatrix}$$

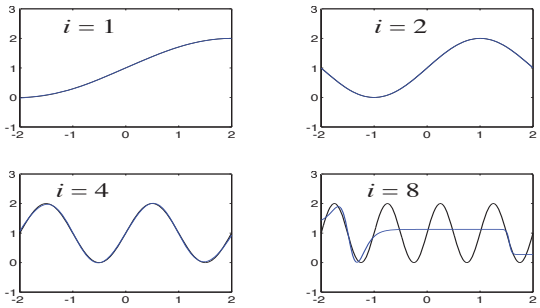
$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha\mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix}-0.27\\ -0.41\end{bmatrix} - 0.1\begin{bmatrix}-0.0495\\ 0.0997\end{bmatrix}\begin{bmatrix}1\end{bmatrix} = \begin{bmatrix}-0.265\\ -0.420\end{bmatrix}$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha\mathbf{s}^1 = \begin{bmatrix}-0.48\\ -0.13\end{bmatrix} - 0.1\begin{bmatrix}-0.0495\\ 0.0997\end{bmatrix} = \begin{bmatrix}-0.475\\ -0.140\end{bmatrix}$$

Choice of Architecture

$$g(p) = 1 + \sin\left(\frac{i\pi}{4}p\right)$$

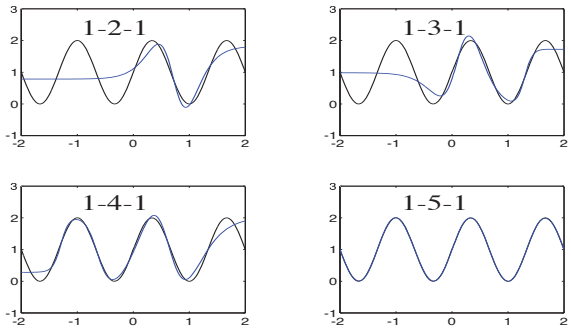
1-3-1 Network



HPSI
Human Performance Systems, Inc.

Choice of Network Architecture

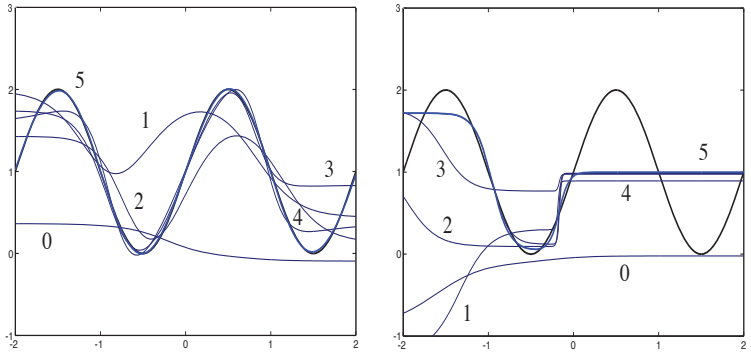
$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right)$$



HPSI
Human Performance Systems, Inc.

Convergence

$$g(p) = 1 + \sin(\pi p)$$

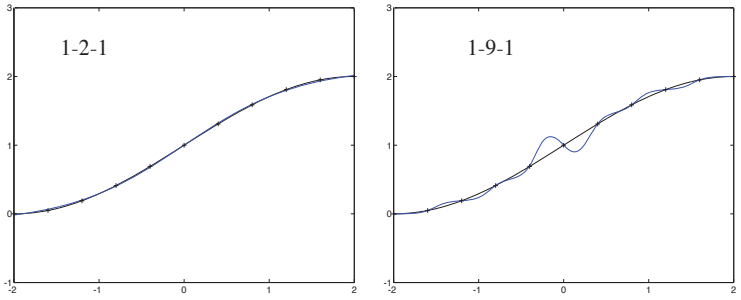


HPSI
Human Performance Systems, Inc.

Generalization

$$\{\mathbf{p}_1,\mathbf{t}_1\}, \{\mathbf{p}_2,\mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q,\mathbf{t}_Q\}$$

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \qquad p = -2, -1.6, -1.2, \ldots, 1.6, 2$$



HPSI Human Performance Systems, Inc.

Generalization

- A cat that once sat on a hot stove will never again sit on a hot stove or on a cold one either.

Mark Twain

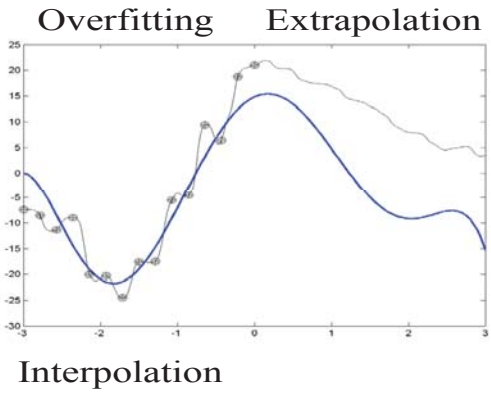
HPSI Human Performance Systems, Inc.

Cause of Overfitting

- The network input-output mapping is accurate for the training data and for test data never seen before.
- The network interpolates well.
- Poor generalization is caused by using a network that is too complex (too many neurons/parameters). To have the best performance we need to find the least complex network that can represent the data (Ockham's Razor).
- Find the simplest model that explains the data.

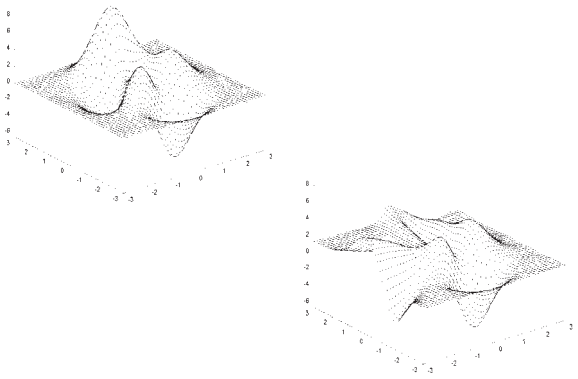
HPSI Human Performance Systems, Inc.

Good Generalization



HPSI
Human Performance Systems, Inc.

Extrapolation in 3-D



HPSI
Human Performance Systems, Inc.

Measuring Generalization

- Part of the available data is set aside during the training process.
- After training, the network error on the test set is used as a measure of generalization ability.
- The test set must never be used in any way to train the network, or even to select one network from a group of candidate networks.
- The test set must be representative of all situations for which the network will be used.

HPSI
Human Performance Systems, Inc.

Methods for Improving Generalization

- Pruning (removing neurons) until the performance is degraded.
- Growing (adding neurons) until the performance is adequate.
- Validation Methods
- Regularization

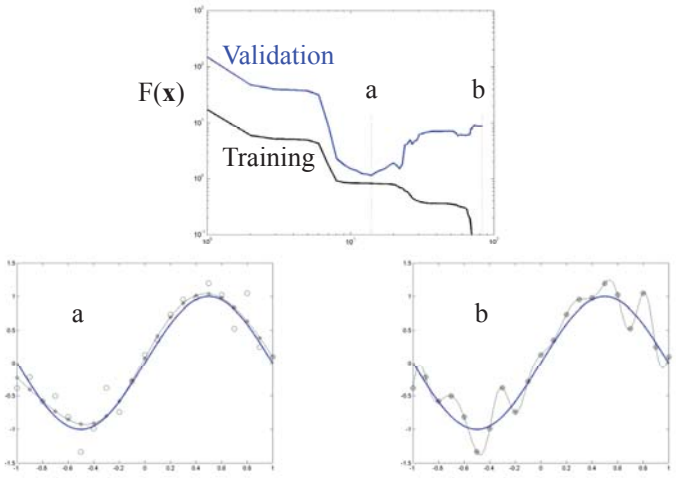
HPSI Human Performance Systems, Inc.

Early Stopping

- Break up data into training, validation, and test sets.
- Use only the training set to compute gradients and determine weight updates.
- Compute the performance on the validation set at each iteration of training.
- Stop training when the performance on the validation set goes up for a specified number of iterations.
- Use the weights which achieved the lowest error on the validation set.

HPSI Human Performance Systems, Inc.

Early Stopping Example



HPSI Human Performance Systems, Inc.

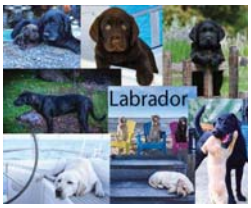
What is deep learning?

- Deep learning is a branch of machine learning involving algorithms that have many nonlinear processing stages.
- In most cases, deep learning refers to training neural networks that have many layers.
- Before approximately 2006, most neural network applications used one or two hidden layers.
- Since that time, the development of more powerful GPUs, and the associated general purpose programming languages, enabled larger neural networks to be tested.
- Larger networks require large data sets to prevent overfitting. The number of large data sets has increased dramatically in recent years.

HPSI
Human Performance Systems, Inc.

Deep Learning Applications

- Google Deepmind – AlphaGO, the first computer GO program to beat a top professional GO player.
- Android operating system speech recognition.
- photosearch for Google+.
- Skype translator – speech recognition.
- Microsoft Cortana digital assistant.
- Facebook – Deep Face, face recognition.
- Apple – Siri



HPSI
Human Performance Systems, Inc.

Course objective

- Learn about the most popular deep learning architectures.
- Learn about the most popular open source deep learning software frameworks.
- Learn how to implement deep networks using deep learning frameworks.

HPSI
Human Performance Systems, Inc.

Course emphasis

- Key emphasis will be on implementation of deep learning concepts on GPUs using open source software frameworks.
- Will not cover basic machine learning concepts.
- Will assume knowledge of key ideas from linear algebra, optimization, probability, machine learning (as covered, for example, in Neural Network Design – hagan.okstate.edu/nnd.html)

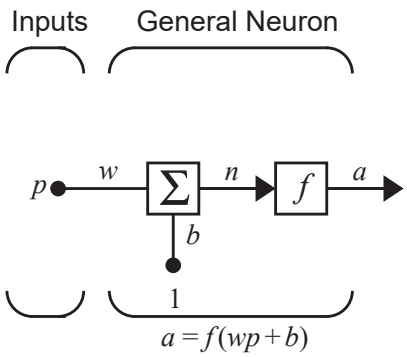
Course schedule

- Introduction
- Multilayer networks
- Training multilayer networks, gradient calculation
- Torch
- Convolution networks
- Training convolution networks, gradient calculation
- Caffe
- Restricted Boltzmann machine
- Deep belief network
- Theano
- Long short term memory
- Training recurrent networks
- TensorFlow

Brief history of deep learning

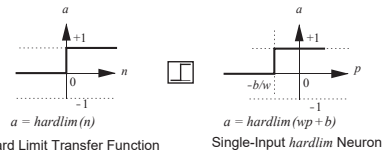
- Backpropagation for multilayer networks discovered, popularized (1974, 1982, 1985, 1986).
- Convolution networks introduced (1989).
- Long Short Term Memory network developed (1997).
- Deep belief network presented (2006).
- NVIDIA unveiled CUDA, a language for general purpose programming of GPUs (2006)

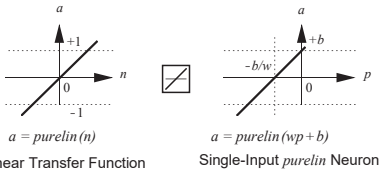
Basic network building block (neuron)



HPSI
Human Performance Systems, Inc.

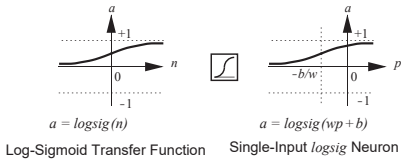
Transfer (activation) functions (1)

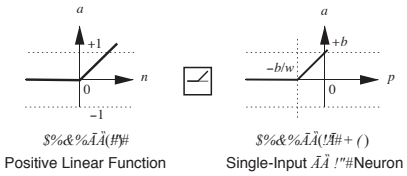




HPSI
Human Performance Systems, Inc.

Transfer (activation) functions (2)





HPSI
Human Performance Systems, Inc.

Transfer (activation) functions (3)

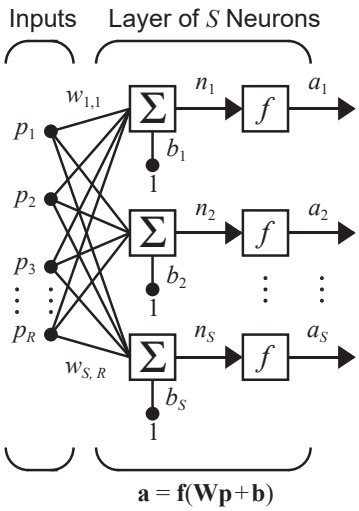
Softmax

$$a_i = f_i(\mathbf{n}) = \frac{e^{n_i}}{\sum_{j=1}^{S} e^{n_j}}$$

Used at the output layer of a pattern recognition network with multiple output neurons.

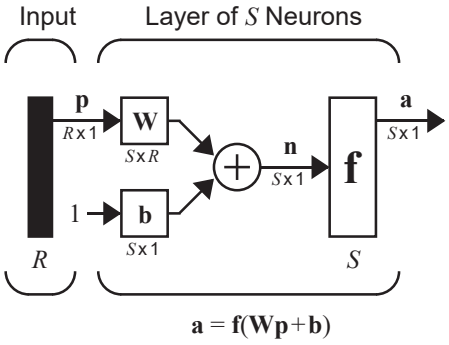
HPSI Human Performance Systems, Inc.

Layer of neurons



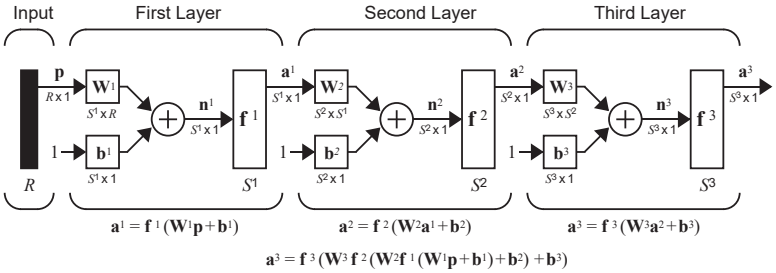
HPSI Human Performance Systems, Inc.

Matrix notation

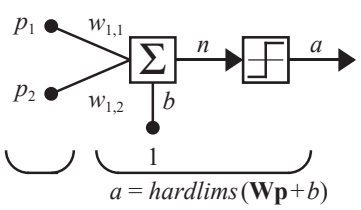


HPSI Human Performance Systems, Inc.

Multiple layer network

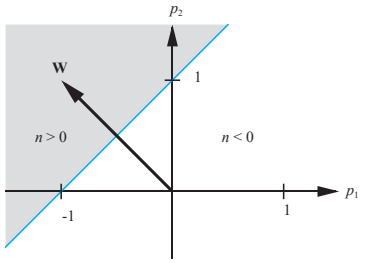


Single layer network decision boundary

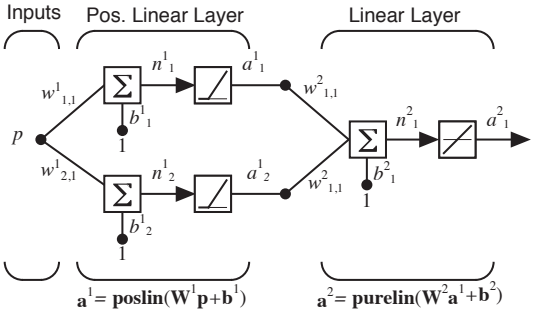


Decision boundary

$$n = \mathbf{W}\mathbf{p} + b == 0$$



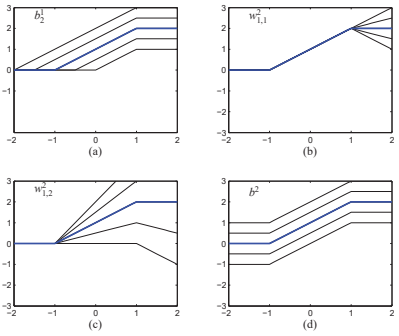
Poslin network

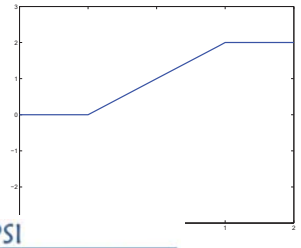


Poslin network function

$$\mathbf{W}^1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T, \mathbf{b}^1 = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$$

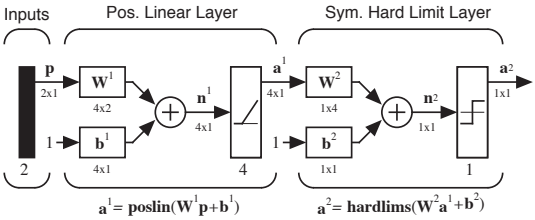
$$\mathbf{W}^2 = \begin{bmatrix} -1 & 1 \end{bmatrix}, \mathbf{b}^2 = [0]$$





HPSI Human Performance Systems, Inc.

2D poslin network

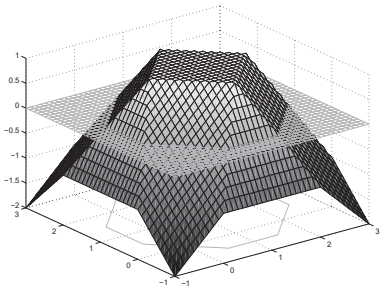


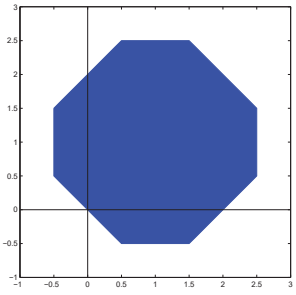
$$\mathbf{W}^1 = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}^T, \mathbf{b}^1 = \begin{bmatrix} -1 & 3 & 1 & 1 \end{bmatrix}^T$$

$$\mathbf{W}^2 = \begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix}, \mathbf{b}^2 = [5]$$

HPSI Human Performance Systems, Inc.

2D Poslin network surface and decision boundary





HPSI Human Performance Systems, Inc.

Supervised learning

Training Set

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, ..., \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

Performance Indices

Mean Square Error

$$F(\mathbf{x}) = \frac{1}{QS^M} \sum_{q=1}^{Q} \sum_{i=1}^{S^M} \left(t_{i,q} - a_{i,q}^M\right)^2$$

Cross Entropy

$$F(\mathbf{x}) = -\sum_{q=1}^{Q} \sum_{i=1}^{S^M} t_{i,q} ln \frac{a_{i,q}^M}{t_{i,q}}$$

Approximate performance index

2nd order Taylor series expansion (quadratic)

$$F(\mathbf{x}) \cong F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T|_{\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \nabla^2 F(\mathbf{x})|_{\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*)$$

Gradient – Direction of increasing $F(\mathbf{x})$

$$\nabla F(\mathbf{x}) = \left[\frac{\partial F(\mathbf{x})}{\partial x_1} \quad \frac{\partial F(\mathbf{x})}{\partial x_2} \quad \cdots \quad \frac{\partial F(\mathbf{x})}{\partial x_n}\right]^T$$

Hessian – Curvature of $F(\mathbf{x})$

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 F(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 F(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 F(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_n^2} \end{bmatrix}$$

Optimization of performance

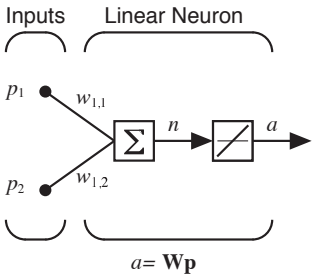
General optimization algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

- The search direction at iteration k is $\mathbf{p}_k$.
- The learning rate is α_k.
- For small learning rates, the largest reduction in $F(\mathbf{x})$ is obtained by setting $\mathbf{p}_k = -\nabla F(\mathbf{x}_k)$, the negative of the gradient direction. This is called the *steepest descent*, or gradient descent algorithm.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x}_k)$$

Example



$$\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\2\end{bmatrix}, \mathbf{t}_1 = [-1]\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix}2\\-1\end{bmatrix}, \mathbf{t}_2 = [-1]\right\}$$
$$\left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, \mathbf{t}_3 = [1]\right\}, \left\{\mathbf{p}_4 = \begin{bmatrix}-1\\0\end{bmatrix}, \mathbf{t}_4 = [1]\right\}$$

HPSI Human Performance Systems, Inc.

Example performance index

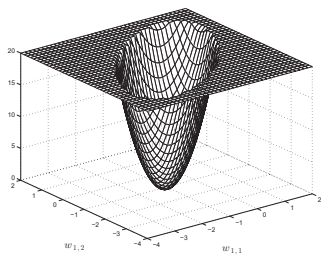
$$\mathbf{U} = \begin{bmatrix}\mathbf{p}_1^T\\\mathbf{p}_2^T\\\mathbf{p}_3^T\\\mathbf{p}_4^T\end{bmatrix}, \mathbf{t} = \begin{bmatrix}t_1\\t_2\\t_3\\t_4\end{bmatrix}, \mathbf{x} = \begin{bmatrix}w_{1,1}\\w_{1,2}\end{bmatrix}$$

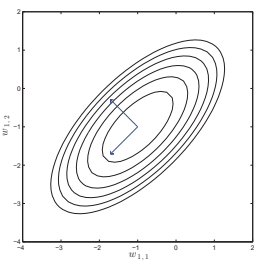
$$\begin{aligned}F(\mathbf{x}) &= \sum_{q=1}^{4}(t_q - a_q)^2 = (\mathbf{t} - \mathbf{U}\mathbf{x})^T(\mathbf{t} - \mathbf{U}\mathbf{x})\\&= \left(\mathbf{t}^T\mathbf{t} - 2\mathbf{t}^T\mathbf{U}\mathbf{x} + \mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{x}\right)\\&= c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}\\\nabla F(\mathbf{x}) &= \mathbf{A}\mathbf{x} + \mathbf{d}, \nabla^2 F(\mathbf{x}) = \mathbf{A}\\c &= \mathbf{t}^T\mathbf{t}, \mathbf{d} = -2\mathbf{U}^T\mathbf{t}, \mathbf{A} = 2\mathbf{U}^T\mathbf{U}\end{aligned}$$

HPSI Human Performance Systems, Inc.

Example performance surface

$$\mathbf{U} = \begin{bmatrix}-1 & 2\\2 & -1\\0 & -1\\-1 & 0\end{bmatrix}, \mathbf{t} = \begin{bmatrix}-1\\-1\\1\\1\end{bmatrix}, \mathbf{A} = \begin{bmatrix}12 & -8\\-8 & 12\end{bmatrix}, \mathbf{d} = \begin{bmatrix}4\\4\end{bmatrix}, c = 4$$





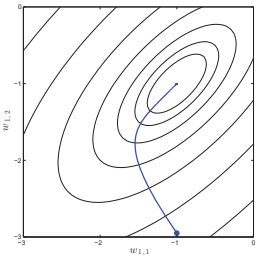
ce surface

Contour plot

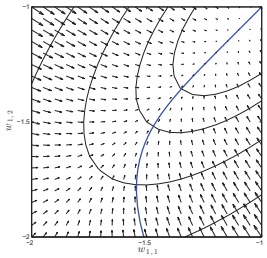
HPSI Human Performance Systems, Inc.

Example steepest descent path

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x}_k) = \mathbf{x}_k - 0.01\,(\mathbf{A}\mathbf{x}_k + \mathbf{d}) = \begin{bmatrix} 0.88 & 0.08 \\ 0.08 & 0.88 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0.04 \\ 0.04 \end{bmatrix}$$



Steepest descent path



Zoomed path with gradients

HPSI Human Performance Systems, Inc.

Stochastic gradient

- Standard steepest descent is a batch algorithm, because the entire batch of data is used to compute the gradient.
- If we compute the gradient for one data point, it is an incremental, or stochastic algorithm.
- Mini-batches can also be used, where the algorithm operates on a subset of data – especially useful for large data sets.

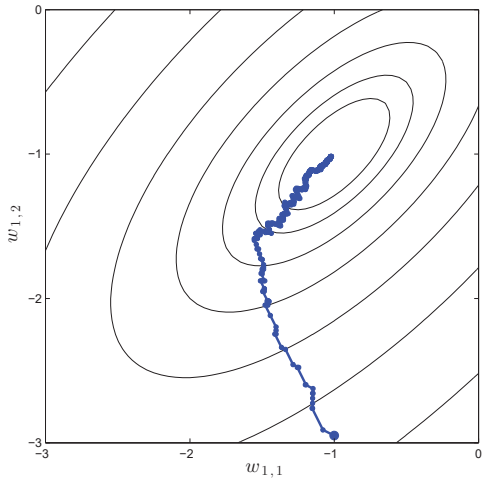
$$\hat{F}(\mathbf{x}) = (t_k - a_k)^2$$
$$\nabla \hat{F}(\mathbf{x}) = -2\,(t_k - a_k)\,\nabla a_k = -2 e_k \mathbf{p}_k$$

Stochastic gradient algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e_k \mathbf{p}_k$$

HPSI Human Performance Systems, Inc.

Stochastic gradient trajectory



HPSI Human Performance Systems, Inc.

Steepest descent in multilayer networks

- For multilayer networks, the error is not a direct function of weights in the hidden layers.
- To compute the necessary gradients we need to use the chain rule.
- The chain rule is implemented one component at a time (e.g., performance function, transfer function, weight function).

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial w_{i,j}^m} = \frac{\partial \hat{F}(\mathbf{x})}{\partial n_i^m}\frac{\partial n_i^m}{\partial w_{i,j}^m} = \frac{\partial \hat{F}(\mathbf{x})}{\partial n_i^m}\frac{\partial(\sum_{l=1}^{S^{m-1}} w_{i,l}^m a_l^{m-1} + b_i^m)}{\partial w_{i,j}^m}$$
$$= \frac{\partial \hat{F}(\mathbf{x})}{\partial n_i^m} a_j^{m-1}$$

HPSI Human Performance Systems, Inc.

Module derivatives

Derivative across transfer function

$$\frac{\partial \mathbf{a}^m}{\partial (\mathbf{n}^m)^T} = \dot{\mathbf{F}}^{\mathbf{m}}(\mathbf{n}^m) = \begin{bmatrix} \frac{\partial f_1^m(\mathbf{n}^m)}{\partial n_1^m} & \frac{\partial f_1^m(\mathbf{n}^m)}{\partial n_2^m} & \cdots & \frac{\partial f_1^m(\mathbf{n}^m)}{\partial n_{S^m}^m} \\ \frac{\partial f_2^m(\mathbf{n}^m)}{\partial n_1^m} & \frac{\partial f_2^m(\mathbf{n}^m)}{\partial n_2^m} & \cdots & \frac{\partial f_2^m(\mathbf{n}^m)}{\partial n_{S^m}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{S^m}^m(\mathbf{n}^m)}{\partial n_1^m} & \frac{\partial f_{S^m}^m(\mathbf{n}^m)}{\partial n_2^m} & \cdots & \frac{\partial f_{S^m}^m(\mathbf{n}^m)}{\partial n_{S^m}^m} \end{bmatrix}$$

Derivative across weight function

$$\frac{\partial \mathbf{n}^{m+1}}{\partial (\mathbf{a}^m)^T} = \frac{\partial [\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}]}{\partial (\mathbf{a}^m)^T} = \mathbf{W}^{m+1}$$

HPSI Human Performance Systems, Inc.

Example transfer function derivatives

Poslin

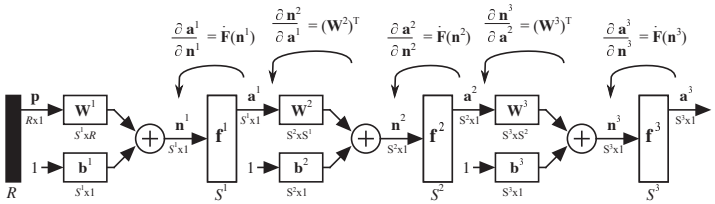
$$\dot{\mathbf{F}}^{\mathbf{m}}(\mathbf{n}^m) = \begin{bmatrix} \text{hardlim}(n_1^m) & 0 & \cdots & 0 \\ 0 & \text{hardlim}(n_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \text{hardlim}(n_{S^m}^m) \end{bmatrix}$$

Softmax

$$\dot{\mathbf{F}}^{\mathbf{m}}(\mathbf{n}^m) = \begin{bmatrix} a_1^m\left(\sum_{i=1}^{S^m} a_i^m - a_1^m\right) & -a_1^m a_2^m & \cdots & -a_1^m a_{S^m}^m \\ -a_2^m a_1^m & a_2^m\left(\sum_{i=1}^{S^m} a_i^m - a_2^m\right) & \cdots & -a_2^m a_{S^m}^m \\ \vdots & \vdots & \ddots & \vdots \\ -a_{S^m}^m a_1^m & -a_{S^m}^m a_2^m & \cdots & a_{S^m}^m\left(\sum_{i=1}^{S^m} a_i^m - a_{S^m}^m\right) \end{bmatrix}$$

HPSI Human Performance Systems, Inc.

Multilayer chain rule (backpropagation)



$$\frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{n}^1} = \frac{\partial \mathbf{a}^1}{\partial \mathbf{n}^1} \times \frac{\partial \mathbf{n}^2}{\partial \mathbf{a}^1} \times \frac{\partial \mathbf{a}^2}{\partial \mathbf{n}^2} \times \frac{\partial \mathbf{n}^3}{\partial \mathbf{a}^2} \times \frac{\partial \mathbf{a}^3}{\partial \mathbf{n}^3} \times \frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{a}^3}$$

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{n}^m} = \frac{\partial \mathbf{a}^m}{\partial \mathbf{n}^m} \times \frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{a}^m} \times \frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{n}^{m+1}}$$

HPSI Human Performance Systems, Inc.

Initializing backpropagation

For Mean Square Error

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial n_i^M} = \sum_{j=1}^{S^M} \frac{\partial \hat{F}(\mathbf{x})}{\partial a_j^M} \frac{\partial a_j^M}{\partial n_i^M} = \frac{1}{S^M} \sum_{j=1}^{S^M} \frac{\partial \left(t_j - a_j^M\right)^2}{\partial a_j^M} \frac{\partial a_j^M}{\partial n_i^M}$$

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial n_i^M} = \frac{-2}{S^M} \sum_{j=1}^{S^M} \left(t_j - a_j^M\right) \frac{\partial a_j^M}{\partial n_i^M}$$

$$= \frac{-2}{S^M} \sum_{j=1}^{S^M} e_j \frac{\partial a_j^M}{\partial n_i^M}$$

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{n}^M} = \frac{\partial (\mathbf{a}^M)^T}{\partial \mathbf{n}^M} \frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{a}^M} = \frac{-2}{S^M} \dot{\mathbf{F}}^{\mathbf{M}}(\mathbf{n}^m)\,\mathbf{e}$$

HPSI Human Performance Systems, Inc.

Summary of multilayer stochastic gradient

If we define the sensitivity to be

$$\mathbf{s}^m \triangleq \frac{\partial \hat{F}(\mathbf{x})}{\partial \mathbf{n}^m}$$

Then the stochastic gradient for mean square error can be computed as

$$\mathbf{s}^M = \frac{-2}{S^M} \dot{\mathbf{F}}^{\mathbf{M}}(\mathbf{n}^m)\,\mathbf{e}$$

$$\mathbf{s}^m = \dot{\mathbf{F}}(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

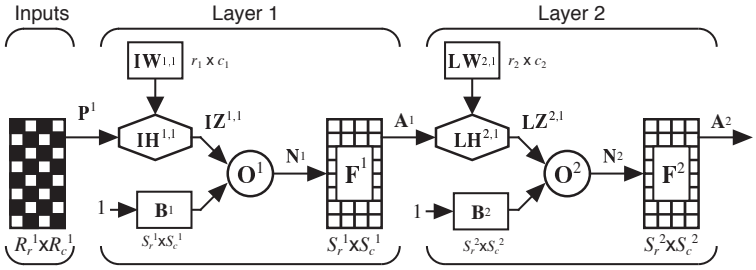
$$\frac{\partial \hat{F}(\mathbf{x})}{\partial w_{i,j}^m} = s_i^m a_j^{m-1}$$

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial b_i^m} = s_i^m$$

HPSI Human Performance Systems, Inc.

Convolution network

- A convolution network is a multilayer feedforward network that has two- or three-dimensional inputs.
- It has weight functions that are not generally viewed as matrix multiplication (or inner product) operations.



HPSI Human Performance Systems, Inc.

2D Convolution

- The principal layer type for convolution networks is the convolution layer.
- Let the input image be represented by the $R_r \times R_c$ matrix $\mathbf{V}$.
- The weight function for this layer performs a convolution operation on the image, using the convolution kernel that is represented by the $r \times c$ matrix $\mathbf{W}$.

$$z_{i,j} = \sum_{k=1}^{r} \sum_{l=1}^{c} w_{k,l} v_{i+k-1,j+l-1}$$

- In matrix form, we will write it as

$$\mathbf{Z} = \mathbf{W} \circledast \mathbf{V}$$

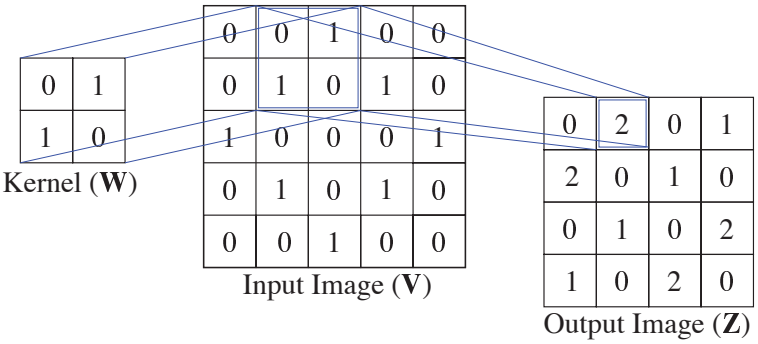
HPSI Human Performance Systems, Inc.

Padding and stride

- Convolution will reduce the number of rows in the image by $r - 1$ and the number of columns by $c - 1$.
- To maintain image size, we can pad the outside of the image with zeros before convolving.
- The width of the zero padding is P^d.
- The output image can be made smaller by taking larger strides, or kernel movements. Normally, the kernel is moved one step at a time when performing the convolution. If the stride is increased to 2, the output image size is reduced by a factor of 2.
- The number of steps for the stride is S^t.

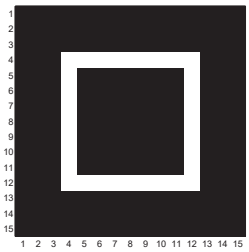
HPSI Human Performance Systems, Inc.

Example convolution

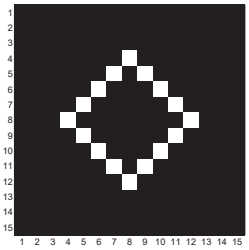


HPSI Human Performance Systems, Inc.

Input Images



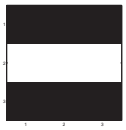
Square



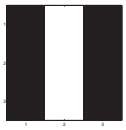
Diamond

HPSI Human Performance Systems, Inc.

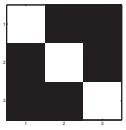
Convolution kernels (filters)



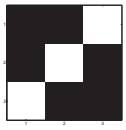
Horizontal kernel



Vertical kernel



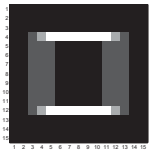
Slash kernel



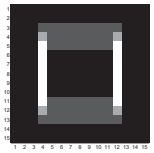
Backslash kernel

HPSI Human Performance Systems, Inc.

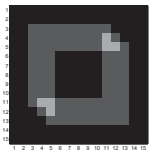
Filtered square



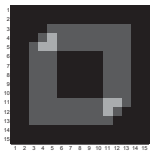
Horizontal filtered square



Vertical filtered square



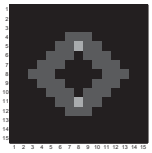
Slash filtered square



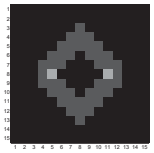
Backslash filtered square

HPSI Human Performance Systems, Inc.

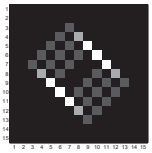
Filtered diamond



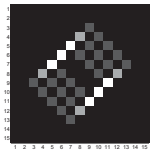
Horizontal filtered diamond



Vertical filtered diamond



Slash filtered diamond



Backslash filtered diamond

HPSI Human Performance Systems, Inc.

Pooling

- A pooling (or subsampling) layer often follows a convolution layer and consolidates $r \times c$ elements in the input image to 1 element in the output image.
- The purpose is to reduce the spatial size of the feature map.
- This reduces the number of parameters in the network.

Average pooling

$$z_{i,j} = \left\{ \sum_{k=1}^{r} \sum_{l=1}^{c} v_{r(i-1)+k,c(j-1)+l} \right\} w$$

Matrix format

$$\mathbf{Z} = w \boxplus_{r,c}^{ave} \mathbf{V}$$

HPSI Human Performance Systems, Inc.

Max pooling

- For max pooling, the maximum of the elements in the consolidation window is used, rather than the sum.
- Recent studies have shown max pooling to be a little more effective in some applications than average pooling.

Max pooling

$$z_{i,j} = max\left\{v_{r(i-1)+k,c(j-1)+l} | k = 1, ..., r; l = 1, ..., c\right\}$$

Matrix format

$$\mathbf{Z} = \boxplus_{r,c}^{max} \mathbf{V}$$

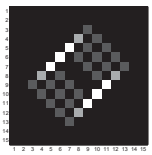
HPSI
Human Performance Systems, Inc.

Pooling: choice of stride and size

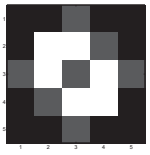
- As in convolution operations, various strides can be used in pooling operations.
- Normally, the consolidation window is square, and the stride is equal to the window size, so there is no overlap in the consolidations.
- The most common choice is $r = 2$, $c = 2$ and $S^t = 2$.

HPSI
Human Performance Systems, Inc.

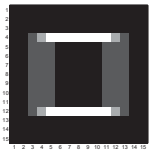
Max pooling (3x3) examples



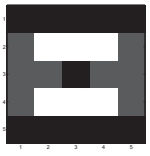
Backslash filtered diamond



Max pooled filtered diamond



Horizontal filtered square

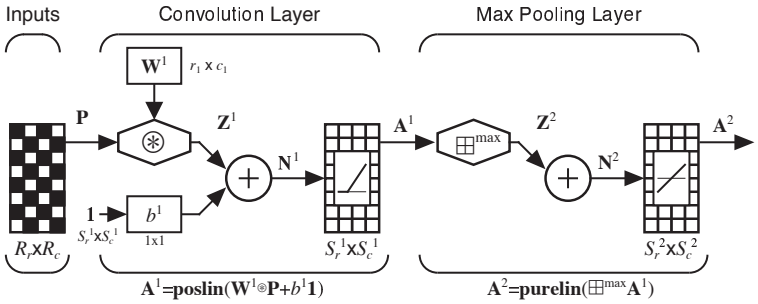


Max pooled filtered square

HPSI
Human Performance Systems, Inc.

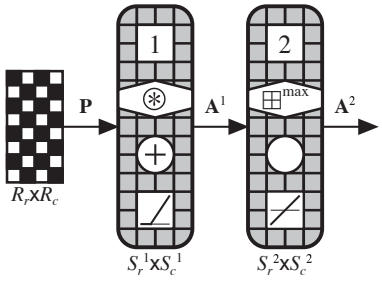
Network diagram notation

The following figure represents the combination of a convolution layer and a pooling layer. Notice that the bias in the convolution layer is a scalar, which is added to each element of $\mathbf{IZ}^{1,1}$.



Simplified notation

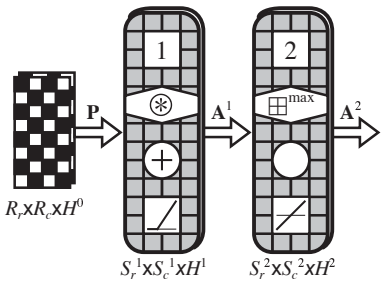
When many layers are involved, it is helpful to have a simplified notation to represent each layer.



Feature maps

- Each convolution kernel (weight) can identify one elemental feature in the input.
- Complex patterns will consist of combinations of many elemental features.
- Multiple kernels can be included in a single convolution layer to extract multiple features.
- LeCun, in his original developement, called the outputs of one kernel operation a feature map (FM).
- This idea can be extended to the input image. For example, a color image consists of red, green and blue planes.
- A convolution layer then takes a set of feature maps as an input, and produces another set of feature maps as an output.

Network diagram with feature maps



HPSI
Human Performance Systems, Inc.

Connection matrix

- Assume that there are H^m FMs in Layer m, and H^{m+1} FMs in layer $m+1$.
- If each FM in Layer m is connected to every FM in Layer $m+1$, then there would be $H^m \times H^{m+1}$ convolution kernels in Layer $m+1$.
- We can reduce the number of kernels by using only a subset of the possible connections.
- The connection matrix between Layer m and Layer $m+1$ will be denoted $\mathbf{C}^{m+1}$.
- Element $c_{i,j}^{m+1}$ will equal 1 when FM j in Layer m is connected to FM i in Layer $m+1$. Otherwise, it will equal 0.

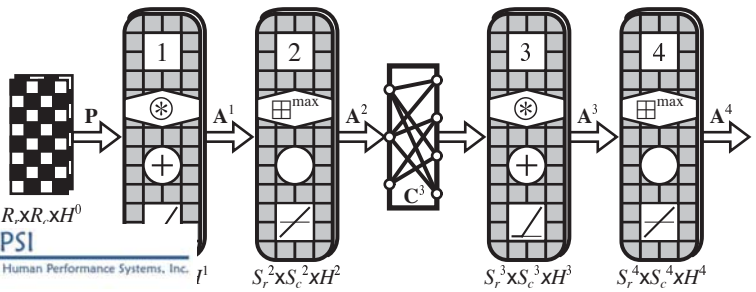
HPSI
Human Performance Systems, Inc.

Notation with connection matrix

$$z_{i,j}^{m,h} = \sum_{l \in C^{m,h}} \sum_{u=1}^{r^m} \sum_{v=1}^{c^m} w_{u,v}^{m,(h,l)} a_{i+u-1,j+v-1}^{m-1,l}$$

$C^{m,h}$ is the set of indices of FMs in layer $m-1$ that connect to FM h in Layer m (from row h of $\mathbf{C}^m$).

$$\mathbf{Z}^{m,h} = \sum_{l \in C^{m,h}} \mathbf{W}^{m,(h,l)} \circledast \mathbf{A}^{m-1,l}$$



HPSI
Human Performance Systems, Inc.

Uses of connection matrices

- Connection matrices are generally located between a pooling layer and the following convolution layer.
- If no connection block appears between a pooling layer and the following convolution layer, the FMs are fully connected.
- The number of FMs in a convolution layer and its following pooling layer are equal, and feature map *h* in the convolution layer is connected only to feature map *h* in the following pooling layer. No connection block is shown.

HPSI
Human Performance Systems, Inc.

Conversion from matrix to vector

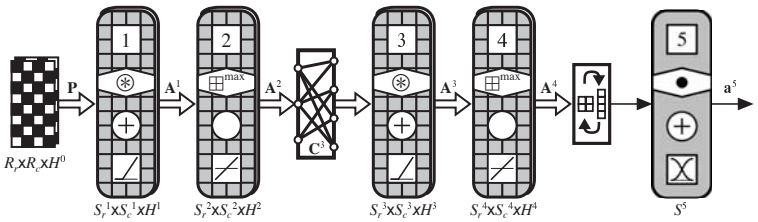
- In addition to the connection block, there is one other block that can appear between layers of a convolution network.
- It converts the output of a layer from a set of FMs to a single vector.
- It stacks the columns of the FMs on top of each other, starting from the first FM to the last. ($\mathbf{a}_i$ indicates the i^{th} column of matrix $\mathbf{A}$.)

$$vec(\mathbf{A}^m) = \left[\left(\mathbf{a}_1^{m,1}\right)^T \left(\mathbf{a}_2^{m,1}\right)^T \cdots \left(\mathbf{a}_{S_c^m}^{m,1}\right)^T \left(\mathbf{a}_1^{m,2}\right)^T \cdots \left(\mathbf{a}_{S_c^m}^{m,H^m}\right)^T \right]^T$$

HPSI
Human Performance Systems, Inc.

Use of matrix to vector conversion

The matrix to vector conversion is normally used before the final layer of a convolution network, which is a standard dot product (matrix multiplication) layer. (The same conversion block can indicate vector to matrix conversion, if needed.)



HPSI
Human Performance Systems, Inc.

Derivative definitions

In order to compute the gradient of the performance with respect to the weights and biases, we need to perform a backpropagation operation. Because the net input is a matrix, we will use a different notation for sensitivity – **dN**. Other derivatives will be defined as:

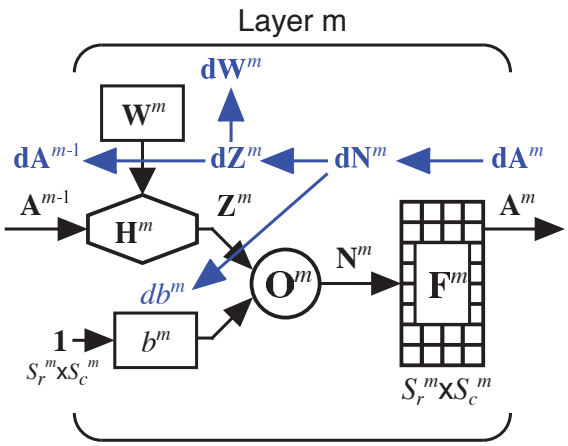
$$\mathbf{dN} \equiv \frac{\partial \hat{F}}{\partial \mathbf{N}}, \mathbf{dA} \equiv \frac{\partial \hat{F}}{\partial \mathbf{A}}, \mathbf{dZ} \equiv \frac{\partial \hat{F}}{\partial \mathbf{Z}}, \mathbf{dW} \equiv \frac{\partial \hat{F}}{\partial \mathbf{W}}, \mathbf{db} \equiv \frac{\partial \hat{F}}{\partial \mathbf{b}}$$

As we backpropagate across Layer m, we will be performing the following operations.

$$\mathbf{dA}^m \rightarrow \mathbf{dN}^m \rightarrow \mathbf{dZ}^m \rightarrow \mathbf{dA}^{m-1}$$

HPSI Human Performance Systems, Inc.

Backpropagating across a layer



HPSI Human Performance Systems, Inc.

Backpropagating across the transfer function

$$dn_{i,j}^{m,h} \equiv \frac{\partial \hat{F}}{\partial n_{i,j}^{m,h}} = \frac{\partial a_{i,j}^{m,h}}{\partial n_{i,j}^{m,h}} \times \frac{\partial \hat{F}}{\partial a_{i,j}^{m,h}}$$

$$\frac{\partial \hat{F}}{\partial a_{i,j}^{m,h}} = da_{i,j}^{m,h}$$

$$dn_{i,j}^{m,h} = \dot{f}^{m,h}(n_{i,j}^{m,h}) \times da_{i,j}^{m,h}$$

Matrix form (**dN**)

$$\mathbf{dN}^{m,h} = \dot{\mathbf{F}}^{m,h} \circ \mathbf{dA}^{m,h}$$

Here $\circ$ is the Hadamard product, which is an element by element matrix multiplication.

HPSI Human Performance Systems, Inc.

Backpropagating across the summation net input function

Scalar form (dz)

$$dz_{i,j}^{m,h} = dn_{i,j}^{m,h}$$

Matrix form (**dZ**)

$$\mathbf{dZ}^{m,h} = \mathbf{dN}^{m,h}$$

Scalar form (*db*)

$$db^{m,h} = \frac{\partial \hat{F}}{\partial b^{m,h}} = \frac{\partial \hat{F}}{\partial n_{i,j}^{m,h}} \times \frac{\partial n_{i,j}^{m,h}}{\partial b^{m,h}}$$

$$db^{m,h} = \sum_{i=1}^{S_r^m} \sum_{j=1}^{S_c^m} dn_{i,j}^{m,h}$$

Matrix form (*db*)

$$db^{m,h} = (\boxplus_{S_r^m, S_c^m} \mathbf{dN}^{m,h}))$$

Backpropagating across the convolution weight function

Scalar form (da)

$$da_{i,j}^{m-1,l} = \frac{\partial \hat{F}}{\partial a_{i,j}^{m,l}} = \frac{\partial \hat{F}}{\partial z_{u,v}^{m,h}} \times \frac{\partial z_{u,v}^{m,h}}{\partial a_{i,j}^{m-1,l}}$$

$$da_{i,j}^{m-1,l} = \sum_{h \epsilon C_b^{m,l}} \sum_{u=1}^{S_r^m} \sum_{v=1}^{S_c^m} dz_{i,j}^{m,h} w_{i-u+1,j-v+1}^{m,(h,l)}$$

$C_b^{m,l}$ – FMs in layer m where lth FM in layer $m-1$ connects.

Matrix form (**dA**)

$$\mathbf{dA}^{m-1,l} = \sum_{h \epsilon C_b^{m,l}} (rot180(\mathbf{W}^{m,(h,l)})) \star (\mathbf{dZ}^{m,h})$$

$rot180$ means matrix is rotated by 180 degrees.

Gradient for convolution weight

Scalar form (dw)

$$dw_{u,v}^{m,h,l} = \frac{\partial \hat{F}}{\partial w_{u,v}^{m,(h,l)}} = \frac{\partial \hat{F}}{\partial z_{i,j}^{m,h}} \times \frac{\partial z_{i,j}^{m,h}}{\partial w_{u,v}^{m,(h,l)}}$$

$$dW_{u,v}^{m,h,l} = \sum_{i=1}^{S_r^m} \sum_{j=1}^{S_c^m} dZ_{i,j}^{m,h} v_{u+i-1,v+j-1}^{m,l}$$

here l is feature map in layer $m-1$.

Matrix form (**dW**)

$$\mathbf{dW}^{m,h,l} = \mathbf{dZ}^{m,h} \star \mathbf{V}^{m,l}$$

Backpropagating across average pooling

Scalar form (da)

$$da^{m-1,h}_{r^{m-1}(i-1)+k,c^{m-1}(j-1)+l} = \frac{\partial \hat{F}}{\partial a^{m-1,h}_{i,j}} = \frac{\partial \hat{F}}{\partial z^{m,h}_{i,j}} \times \frac{\partial z^{m,h}_{i,j}}{\partial a^{m-1,h}_{i,j}}$$

$$= dZ^{m,h}_{i,j} \times w^{m,h}$$

For k=1 to r^m and l=1 to c^m.

Matrix form (**dA**)

$$\mathbf{dA}^{m-1,h} = (w^{m,h}) \boxtimes^{ave}_{r^m,c^m} (\mathbf{dZ}^{m,h})$$

$\boxtimes^{ave}_{j,k}\mathbf{A}$ takes each element of the matrix $\mathbf{A}$ and expands to j rows and k columns (reverse of $\boxplus^{ave}_{j,k}\mathbf{A}$).

Gradient for average pooling weight

Scalar form (dw)

$$dw^{m,h} = \frac{\partial \hat{F}}{\partial w^{m,h}} = \frac{\partial \hat{F}}{\partial z^{m,h}_{i,j}} \times \frac{\partial z^{m,h}_{i,j}}{\partial w^{m,h}}$$

$$dw^{m,h} = \sum_{i=1}^{S^m_r} \sum_{j=1}^{S^m_c} dz^{m,h}_{i,j} \left\{ \sum_{k=1}^{r^m} \sum_{l=1}^{c^m} a^{m-1,h}_{r^m(i-1)+k,c^m(j-1)+l} \right\}$$

Matrix form (dw)

$$dw^{m,h} = \boxplus^{ave}_{S^m_r,S^m_c}(\mathbf{dZ}^{m,h} \circ (\boxplus^{ave}_{r^m,c^m}\mathbf{A}^{m-1,h}))$$

First it performs a reduction operation that produces a matrix of size S^m_r by S^m_c, and then, after a Hadamard matrix multiplication, it performs another reduction to produce the scalar gradient.