DATA1030: Hands-on Data Science

Intro to ML

Mud card

Admin

- **I was wondering how broad our options are for the data we choose for our final project, and what our final goal can be.**
  - any topic is fine
  - the dataset should one of these three properties:
    - contains missing values
    - not i.i.d. (it is time series or one object is described by multiple datapoints)
    - the dataset is large
  - use your dataset to answer a regression or a classification question
- **I also want to know the grade criterion of our individual final project. What kinds of aspect decide the grade? For example, the fluency, completeness and logicality of our essay, or the data we are going to use, etc.**
  - it's a mix of all of those aspects
  - we will grade based on the technical correctness of your work but we will also take into account the quality of the reports and the presentations
  - rubrics will be sent out ahead of time
- **I struggle with how to open the GitHub documents with Jupyter Notebook.**
  - you will have problems by the time you submit the first problem set :)
- **What is the difference between this class and other CSCI classes offered like Machine Learning?**
  - CSCI1420 and DATA2060 are equivalent classes
  - both are much more theoretical than DATA1030
  - we learn about the math and numerics behind ML algorithms and the students implement it using python and numpy using OOP
- **How much background coding experience is needed in Python for this class?**
  - if you know basic container types, control flow, and know how to write functions, you will be fine
- **For the individual semester project, is the project individual only for the writing of the code? Or everything needs to be completely individual?**
  - completely individual
  - not sure how a group of people would give one joint presentation if everyone works on a different dataset :)
- **Why we're not doing deep learning.**
  - this is a mandatory course for the DSI master's students and they are required to take the deep learning course CSCI2470
- **Will there be examples of projects from previous semesters that we will be able to look at before getting started on our own?**
  - yes, they are already in the course's github repo

Supervised ML

- **Is y a true label for one feature or a group if features. E.g if you have a structure dataset with house #bed, #bath, price, and sq. Ft, what feature would Y be the true label for?**
  - y would be the sale price of the house
  - one price per house and each house is described by a number of features
- **Difference between classification and regression problems.**
  - we will cover this much more
  - classification: your target variable is categorical (e.g., yes-no, grades like A, B, C, NC)
  - regression: the target variable is continuous (e.g., sale price, people's age)
- **The different types of variables and are they all considered the same weight.**
  - initially yes because you might not know or you might be wrong about the best way to weight them
  - some ML algorithms will assign weights
- **I think you mentioned hyper-parameters? Not really sure what that is**
  - that's OK, give it 4-5 weeks :)
- **The amount of data needed to achieve high performance (before there is a plateau)? It seemed that there was some way of finding this out and I would be interested to know how to test this in practice**

    - yep, it's called the learning curve and we will cover it during the second half of the term

Other ML areas

- **I just got a little bit confused about the first sentence "only the feature matrix X is available, there is no target variable" in the part of Other Areas of ML. What does it mean?**
    - there is no target variable to predict in unsupervised ML
- **The muddiest part of the lecture was the mechanisms of unsupervised learning and the instances of when it would be helpful. I understand in theory when and why one would want to find groups and cluster data, but I would love to see an example of how those groups may be formed and its uses.**
    - we might cover some of this towards the end but unsupervised ML is not the main topic of this lecture
    - do some online reading if you'd like to learn more

Learning objectives

By the end of the lecture, you will be able to

- describe the main goals of the ML pipeline
- list the main steps of the ML pipeline
- explain the bias-variance trade off

Learning objectives

By the end of the lecture, you will be able to

- **describe the main goals of the ML pipeline**
- list the main steps of the ML pipeline
- explain the bias-variance trade off

An ML example

- let's assume you just moved to an island and you never had papayas before but it is common on the island
- what do you do?
- sample some papayas and collect some info
    - for each papaya you try, you collect color, firmness, and whether it tasted good or not
    - classification problem with two features
- once you have enough data, you can train a machine learning model to predict if a new previously unseen papaya is tasty or not based on its color and firmness

Let's define this problem!

- **the learner's input**
    - Domain set $\mathcal{X}$ - a set of objects we wish to label. In the papaya example: the set of all papayas. $\mathcal{X}$ can be in infinite set or a set that's too large to handle on any computer (e.g., all possible 640x480 images with 3 color channels and 256 possible pixel values)
        - domain points are represented by a vector of features e.g., (color, firmness)
        - domain points are also called instances, and $\mathcal{X}$ is also called the instance space
    - Label set $\mathcal{Y}$ - a set of possible labels. In the papaya example: we restrict our label set to {0,1}, 0 meaning the papaya tastes bad, 1 meaning the papaya tastes good.
        - such a label set is categorical, i.e., we have a classification problem at hand
        - the label set can be continuous too, e.g., the real number between 0 and 1, meaning that 0.5 is an OK tasting papaya.
        - the label set can also be probabilistic
            - i.e., two papayas with the same color and firmness can sometimes be tasty and sometimes bad
            - this is quite normal, the features you collect usually do not uniquely determine the label
    - Training data $S = ((x_1, y_1), \ldots, (x_m, y_m))$ - a finite sequence of pairs from $\mathcal{X}$, $\mathcal{Y}$. This is what the learner has access to.
        - S is also called the training set, and examples in S are also called training examples
        - $X = (x_1, \ldots, x_m)$ is the feature matrix which is usually a 2D matrix, and $Y = (y_1, \ldots, y_m)$ is the target variable which is a vector.
- **the learner's output**

    - a prediction rule $h: \mathcal{X} \rightarrow \mathcal{Y}$ - this is also called the predictor, a hypothesis, or in the papaya example a classifier. It would be a regressor if $\mathcal{Y}$ was continuous. In the papaya example, the predictor is the rule that our learner will employ to predict if a papaya will be tasty based on color and firmness as they examine it e.g., in the farmer's market or before picking the fruit from the tree.
    - this prediction rule is generated based on S so $h: X \rightarrow Y$ is more appropriate
    - once the prediction rule is determined, we can use it to predict the label to previously unseen data

How is S used?

- in ML, you only use part of S to train the model
- you hold out some fraction of S to calculate what's called the **generalization error**
- it measures how well the model is expected to perform on previously unseen data
- it helps to avoid models that overfit or underfit
    - overfit: model is too complex, it performs very well on the training set but it doesn't generalize to previously unseen data
    - underfit: the model is too simple, it performs poorly on te training set and on previously unseen data as well

Recap the goals:

- use the training data (X and y) to develop a model which can accurately predict the target variable (y_new') for previously unseen data (X_new)
    - model performance or 'accuracy' is a metric you need to choose to measure model performance and objectively compare various models
- measure the generalization error: measure how well the model is expected to perform on previously unseen data

Quiz

Learning objectives

By the end of the lecture, you will be able to

- describe the main goals of the ML pipeline
- **list the main steps of the ML pipeline**
- explain the bias-variance trade off

The steps

1. Exploratory Data Analysis (EDA): you need to understand your data and verify that it doesn't contain errors

- do as much EDA as you can!

2. Split the data into different sets: most often the sets are train, validation, and test (or holdout)

- practitioners often make errors in this step!
- you can split the data randomly, based on groups, based on time, or any other non-standard way if necessary to answer your ML question

3. Preprocess the data: ML models only work if X and Y are numbers! Some ML models additionally require each feature to have 0 mean and 1 standard deviation (standardized features)

- often the original features you get contain strings (for example a gender feature would contain 'male', 'female', 'non-binary', 'unknown') which needs to be transformed into numbers
- often the features are not standardized (e.g., age is between 0 and 100) but it needs to be standardized

4. Choose an evaluation metric: depends on the priorities of the stakeholders

- often requires quite a bit of thinking and ethical considerations

5. Choose one or more ML techniques: it is highly recommended that you try multiple models

- start with simple models like linear or logistic regression
- try also more complex models like nearest neighbors, support vector machines, random forest, etc.

6. Tune the hyperparameters of your ML models (aka cross-validation)

- ML techniques have hyperparameters that you need to optimize to achieve best performance
- for each ML model, decide which parameters to tune and what values to try
- loop through each parameter combination
  - train one model for each parameter combination
  - evaluate how well the model performs on the validation set
- take the parameter combo that gives the best validation score
- evaluate that model on the test set to report how well the model is expected to perform on previously unseen data

7. Interpret your model: black boxes are often not useful

- check if your model uses features that make sense (excellent tool for debugging)
- often model predictions are not enough, you need to be able to explain how the model arrived to a particular prediction (e.g., in health care)

Quiz

Learning objectives

By the end of the lecture, you will be able to

- describe the main goals of the ML pipeline
- list the main steps of the ML pipeline
- **explain the bias-variance trade off**

Bias-variance tradeoff illustrated through a simple ML pipeline

In [ ]:
```
# import packages

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from matplotlib import pylab as plt
import matplotlib
from matplotlib.colors import ListedColormap
%matplotlib inline

# scikit-learn code is reproducable is the random seed is fixed.
np.random.seed(2)

# read in the data
# our toy dataset, we don't know how it was generated.
df = pd.read_csv('data/toy_data.csv')

X = df[['x1','x2']].values
y = df['y'].values

print(np.shape(X))
print(np.shape(y))
print(np.unique(y,return_counts=True))
```

1. Exploratory Data Analysis (EDA)

In [ ]:
```
plt.hist(X[y==0,0],alpha=0.4,color='r',label='x1 for class 0')
plt.hist(X[y==1,0],alpha=0.4,color='b',label='x1 for class 1')
plt.xlabel('x1')
plt.ylabel('y')
plt.title('toy dataset')
plt.legend()
plt.show()
```

In [ ]:
```
plt.hist(X[y==0,1],alpha=0.4,color='r',label='x2 for class 0')
plt.hist(X[y==1,1],alpha=0.4,color='b',label='x2 for class 1')
plt.xlabel('x2')
plt.ylabel('y')
plt.title('toy dataset')
plt.legend()
plt.show()
```

In [ ]:
```
plt.scatter(X[y==0,0],X[y==0,1],color='r',label='class 0',alpha=0.4)
plt.scatter(X[y==1,0],X[y==1,1],color='b',label='class 1',alpha=0.4)
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('toy dataset')
plt.gca().set_aspect('equal')
plt.legend()
plt.show()
```

2. Split the data into different sets

In [ ]:
```
help(train_test_split)
```

In [ ]:
```
X_train, X_other, y_train, y_other = train_test_split(X,y,test_size=0.4)
print(np.shape(X_other),np.shape(y_other))
print('train:',np.shape(X_train),np.shape(y_train))

X_val, X_test, y_val, y_test = train_test_split(X_other,y_other,test_size=0.5)
print('val:',np.shape(X_val),np.shape(y_val))
print('test:',np.shape(X_test),np.shape(y_test))
```

3. Preprocess the data

In [ ]:
```
help(StandardScaler)
```

In [ ]:
```
scaler = StandardScaler().fit(X_train)
# the scaler object contains the feature means and variations in the training set
print(scaler.mean_)
print(scaler.var_)

# the scaler is used to transform the sets
X_train_prep = scaler.transform(X_train)
X_val_prep = scaler.transform(X_val)
X_test_prep = scaler.transform(X_test)
```

4. Choose an evaluation metric

In [ ]:
```
help(accuracy_score)
```

Quiz

5. Choose one or more ML techniques

In [ ]:
```
help(SVC)
```

6. Tune the hyperparameters of your ML models (aka cross-validation)

In [ ]:
```
Cs = np.logspace(-1,3,13)
print(Cs)
train_scores = []
validation_scores = []
models = []
for C in Cs:
    classifier = SVC(kernel='rbf',C = C, probability=True) # this is our classifier
    classifier.fit(X_train_prep,y_train) # the model is fitted to the training data

    y_train_pred = classifier.predict(X_train_prep)
    train_accuracy = accuracy_score(y_train,y_train_pred) # calculate the validation accuracy
    train_scores.append(train_accuracy)

    y_val_pred = classifier.predict(X_val_prep) # predict the validation set
    validation_accuracy = accuracy_score(y_val,y_val_pred) # calculate the validation accuracy
    validation_scores.append(validation_accuracy)

    models.append(classifier)
    print(C, train_accuracy, validation_accuracy)
```

The bias - variance tradeoff

```
In [ ]: plt.plot(Cs,train_scores,label='train score')
        plt.plot(Cs,validation_scores,label='validation score')
        plt.semilogx()
        plt.legend()
        plt.xlabel('C parameter')
        plt.ylabel('accuracy')
        plt.show()
```

- **high bias model** (aka underfitting)
  - it performs poorly on the train and validation sets
  - small C values in the example above
- **high variance model** (aka overfitting)
  - it performs very well on the training set but it performs poorly on the validation set
  - high C
- the goal of the parameter tuning is to find the balance between bias and variance
  - usually the best model is the one with the best validation score
  - C = 46 in our case

How does the best model perform on the test set?

- this score tells us how well the model generalizes to previously unseen data because the test set was not touched before
- usually it is close to the best validation score

```
In [ ]: y_test_pred = models[-5].predict(X_test_prep)
        print(accuracy_score(y_test, y_test_pred))
```

7. Interpret your model

- with two features, this is easy
- plot the decision boundary and probabilities

```
In [ ]: # Plot the decision boundary. For that, we will assign a color to each
        # point in the mesh [x_min, m_max]x[y_min, y_max].

        cm = plt.cm.RdBu
        cm_bright = ListedColormap(['#FF0000', '#0000FF'])
        h = .02  # step size in the mesh
        x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
        y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                             np.arange(y_min, y_max, h))

        # use the best model with C = 46
        classifier = models[-5]
        # scale the data before predicting! this is very important!
        Z = classifier.predict_proba(scaler.transform(np.c_[xx.ravel(), yy.ravel()]))[:, 1]

        # Put the result into a color plot
        Z = Z.reshape(xx.shape)
        plt.contour(xx, yy, Z,vmin=0,vmax=1,levels=[0.5],colors=['k'])
        plt.contourf(xx, yy, Z, cmap=cm, alpha=.8,vmin=0,vmax=1,levels=np.arange(0,1.02,0.02))
        plt.colorbar(ticks=[0,0.2,0.4,0.6,0.8,1],label='predicted probability')
        plt.scatter(X_train[y_train==0,0],X_train[y_train==0,1],color='r',label='class 0')
        plt.scatter(X_train[y_train==1,0],X_train[y_train==1,1],color='b',label='class 1')
        plt.xlabel('x1')
        plt.ylabel('x2')
        plt.title('the decision boundary')
        plt.gca().set_aspect('equal')
        plt.savefig('figures/decision_boundary.jpg',dpi=150)
        plt.show()
```

Learning objectives

Hopefully you can now

- describe the main goals of the ML pipeline
- list the main steps of the ML pipeline
- explain the bias-variance trade off

Mud card

```
In [ ]:
```