GENERAL ASSEMBLY

Scikit-learn, Evaluating Models and Cross Validation

Zack Peterson

LEARNING OBJECTIVES

- What is Scikit-learn
- Model Evaluation
- Cross Validation

Getting to know Scikit-learn

Look it up in google using the abbreviation sklearn.

Or...http://scikit-learn.org/stable/

Split Data

Importance of Splitting and random data shuffle

Why is it important to do?

Using sklearn, find a method to split your data!

Evaluation Metrics

Model Evaluation

How you can tell if your machine learning algorithm is getting better and how you are doing overall?

Classification and Regression

Classification: ?

Regression: ?

Classification Metrics

Models that make discrete predictions

Accuracy

The most basic and common classification metric is accuracy. Accuracy here is described as the proportion of items classified or labeled correctly.

Accuracy is the default metric used in the `.score()` method for classifiers in sklearn. You can read more in the documentation http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score

Quiz!

Open Quiz1 code!

Quiz

Accuracy: What are the shortcomings?

Picking the most suitable metric

There are many cases where you care about certain outcomes more than others.

Confusion Matrix

Quiz!

And code!

Precision and Recall

Recall: True Positive / (True Positive + False Negative). Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

Precision: True Positive / (True Positive + False Positive). Out of all the items labeled as positive, how many truly belong to the positive class.

Quiz Precision and Recall

Open up Precision and Recall quiz

F1 score

Now that you've seen precision and recall, another metric you might consider using is the *F1 score*. F1 score combines precision and recall relative to a specific positive class.

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

For more information about F1 score how to use it in sklearn, check out the documentation
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score

Check out F1 code

Regression Metrics

As mentioned earlier, for regression problems we are dealing with a model that makes continuous predictions. In this case we care about how close the prediction is.

For example, with height or weight predictions it is unreasonable to expect a model to 100% accurately predict someone's weight down to a fraction of a pound! But we do care how consistently the model can make a close prediction — perhaps within 3 to 4 pounds.

We'll discuss a few measurements of accuracy for regression tasks. Since the Titanic dataset is a classification task, we'll try out another one of sklearn's pre-loaded datasets, the diabetes dataset, for the rest of the lesson.

Quiz Code Mean Absolute error

Mean Squared Error

Mean squared is the most common metric to measure model performance. In contrast with absolute error, the residual error (the difference between predicted and the true value) is squared.

Some benefits of squaring the residual error is that error terms are positive, it emphasizes larger errors over smaller errors, and is differentiable. Being differentiable allows us to use calculus to find minimum or maximum values, often resulting in being more computationally efficient.

For more information about mean squared error and how to use it in sklearn, check out the documentation
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html#sklearn.metrics.mean_squared_error

Quiz Mean Squared Error

R squared and explained variance score

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.explained_variance_score.html#sklearn.metrics.explained_variance_score

Wow

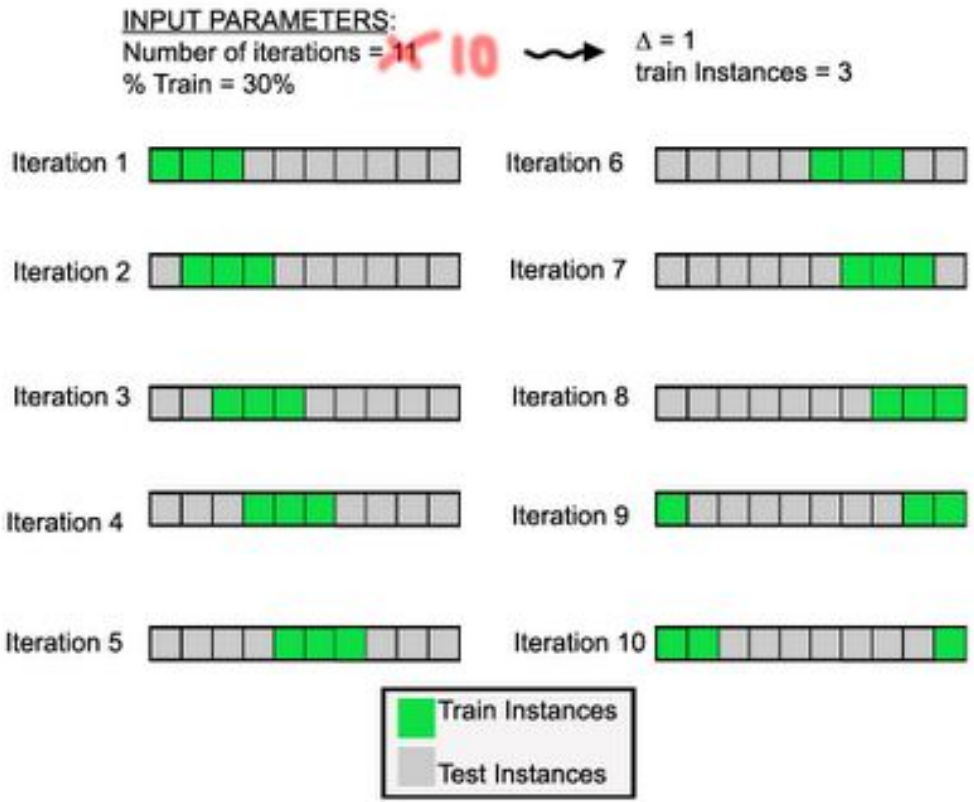
We’re done with model evaluation for now!

On to Cross Validation

CROSS VALIDATION

- Cross validation can help account for bias.

CROSS VALIDATION



GUIDED PRACTICE

CROSS VALIDATION WITH LINEAR REGRESSION

ACTIVITY: CROSS VALIDATION WITH LINEAR REGRESSION

DIRECTIONS (20 minutes)

If we were to continue increasing the number of folds in cross validation, would error increase or decrease?

1. Using the previous code example, perform k-fold cross validation for all even numbers between 2 and 50.
2. Answer the following questions:
   a. What does `shuffle=True` do?
   b. At what point does cross validation no longer seem to help the model?
3. Hint: `range(2, 51, 2)` produces a list of even numbers from 2 to 50

DELIVERABLE

Answers to questions

QUICK CHECK

- We are working with the bikeshare data to predict riders over hours/days with a few features.

- Does it make sense to use a ridge regression or a lasso regression?

UNDERSTANDING REGULARIZATION EFFECTS

- Let's test a variety of alpha weights for Ridge Regression on the bikeshare data.

```
alphas = np.logspace(-10, 10, 21)
for a in alphas:
    print 'Alpha:', a
    lm = linear_model.Ridge(alpha=a)
    lm.fit(modeldata, y)
    print lm.coef_
    print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- What happens to the weights of the coefficients as alpha increases? What happens to the error as alpha increases?

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- Grid search exhaustively searches through all given options to find the best solution. Grid search will try all combos given in `param_grid`.

```
param_ grid = {
    'intercept': [True, False],
    'alpha': [1, 2, 3],
}
```

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- This param grid has six different options:
  - intercept True, alpha 1
  - intercept True, alpha 2
  - intercept True, alpha 3
  - intercept False, alpha 1
  - intercept False, alpha 2
  - intercept False, alpha 3

```
param_ grid = {
    'intercept': [True, False],
    'alpha': [1, 2, 3],
}
```

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- This is an incredibly powerful, automated machine learning tool!

```
from sklearn import grid_search

alphas = np.logspace(-10, 10, 21)
gs = grid_search.GridSearchCV(
    estimator=linear_model.Ridge(),
    param_grid={'alpha': alphas},
    scoring='mean_squared_error')
```

WE CAN MAKE THIS EASIER WITH GRID SEARCH!

```
gs.fit(modeldata, y)

print -gs.best_score_ # mean squared error here comes in negative, so let's make it positive.
print gs.best_estimator_ # explains which grid_search setup worked best
print gs.grid_scores_ # shows all the grid pairings and their performances.
```

GUIDED PRACTICE

GRID SEARCH CV, SOLVING FOR ALPHA

ACTIVITY: GRID SEARCH CV, SOLVING FOR ALPHA

DIRECTIONS (25 minutes)

1. Modify the previous code to do the following:
   a. Introduce cross validation into the grid search. This is accessible from the cv argument.
   b. Add fit_intercept = True and False to the param_grid dictionary.
   c. Re-investigate the best score, best estimator, and grid score attributes as a result of the grid search.

DELIVERABLE

New code and output that meets above requirements

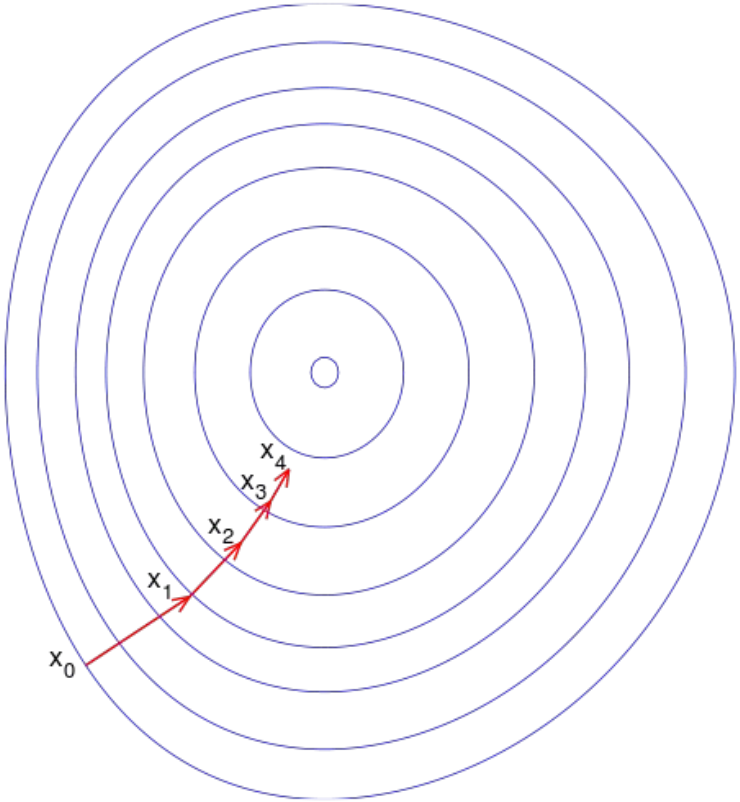
MINIMIZING LOSS THROUGH GRADIENT DESCENT

GRADIENT DESCENT

- Gradient Descent can also help us minimize error.

- How Gradient Descent works:
  - A random linear solution is provided as a starting point
  - The solver attempts to find a next “step”: take a step in any direction and measure the performance.
  - If the solver finds a better solution (i.e. lower MSE), this is the new starting point.
  - Repeat these steps until the performance is optimized and no “next steps” perform better. The size of steps will shrink over time.

GRADIENT DESCENT



A CODE EXAMPLE OF GRADIENT DESCENT

```
num_to_approach, start, steps, optimized = 6.2, 0., [-1, 1], False
while not optimized:
    current_distance = num_to_approach - start
    got_better = False
    next_steps = [start + i for i in steps]
    for n in next_steps:
        distance = np.abs(num_to_approach - n)
        if distance < current_distance:
            got_better = True
            print distance, 'is better than', current_distance
            current_distance = distance
            start = n
```

A CODE EXAMPLE OF GRADIENT DESCENT

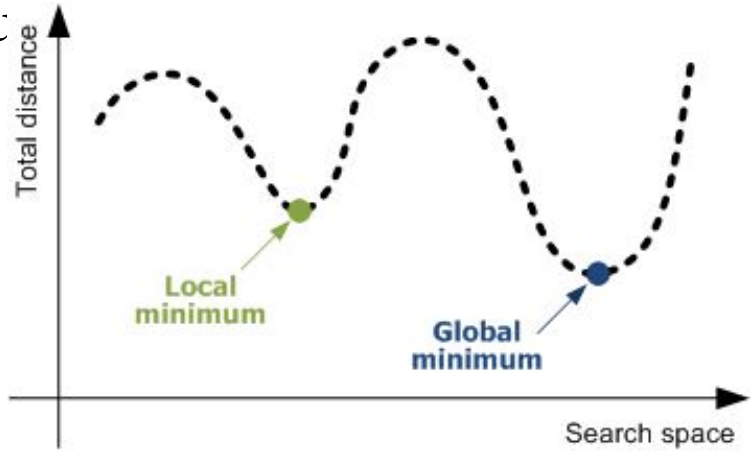
```
    if got_better:
        print 'found better solution! using', current_distance
        a += 1
    else:
        optimized = True
        print start, 'is closest to', num_to_approach
```

- What is the code doing? What could go wrong?

GLOBAL VS LOCAL MINIMUMS

- Gradient Descent could solve for a *local* minimum instead of a *global* minimum.

- A *local* minimum is confined to a very specific subset of solutions. The *global* minimum considers all solutions. These could be equal, but that's not always tru



DEMO

APPLICATION OF GRADIENT DESCENT

APPLICATION OF GRADIENT DESCENT

- Gradient Descent works best when:
  - We are working with a large dataset. Smaller datasets are more prone to error.
  - Data is cleaned up and normalized.
- Gradient Descent is significantly faster than OLS. This becomes important as data gets bigger.

APPLICATION OF GRADIENT DESCENT

- We can easily run a Gradient Descent regression.

- Note: The verbose argument can be set to 1 to see the optimization steps.

```
lm = linear_model.SGDRegressor()
lm.fit(modeldata, y)
print lm.score(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- Untuned, how well did gradient descent perform compared to OLS?

APPLICATION OF GRADIENT DESCENT

- Gradient Descent can be tuned with
  - the learning rate: how aggressively we solve the problem
  - epsilon: at what point do we say the error margin is acceptable
  - iterations: when should be we stop no matter what

INDEPENDENT PRACTICE

ON YOUR OWN

ACTIVITY: ON YOUR OWN

EXERCISE

DIRECTIONS (30 minutes)

There are tons of ways to approach a regression problem.

1. Implement the Gradient Descent approach to our bikeshare modeling problem.
2. Show how Gradient Descent solves and optimizes the solution.
3. Demonstrate the grid_search module.
4. Use a model you evaluated last class or the simpler one from today. Implement param_grid in grid search to answer the following questions:
    a. With a set of values between 10^-10 and 10^-1, how does MSE change?
    b. Our data suggests we use L1 regularization. Using a grid search with l1_ratios between 0 and 1, increasing every 0.05, does this statement hold true? If not, did gradient descent have enough iterations to work properly?
    c. How do these results change when you alter the learning rate?

DELIVERABLE

Gradient Descent approach and answered questions

ACTIVITY: ON YOUR OWN

EXERCISE

Starter Code

```
params = {} # put your gradient descent parameters here
gs = grid_search.GridSearchCV(
    estimator=linear_model.SGDRegressor(),
    cv=cross_validation.KFold(len(modeldata), n_folds=5, shuffle=True),
    param_grid=params,
    scoring='mean_squared_error',
    )

gs.fit(modeldata, y)

print 'BEST ESTIMATOR'
print -gs.best_score_
print gs.best_estimator_
print 'ALL ESTIMATORS'
print gs.grid_scores_
```