Bias Variance

Import stuff

We'll need: pandas, numpy, matplotlib, sklearn.

In [143]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import neighbors
```

Get the data

Dataset URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data)"

In [121]:
```
a = pd.read_csv('iris.csv')
c = a['species']
a
```

Out[121]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
22	4.6	3.6	1.0	0.2	setosa
23	5.1	3.3	1.7	0.5	setosa
24	4.8	3.4	1.9	0.2	setosa
25	5.0	3.0	1.6	0.2	setosa
26	5.0	3.4	1.6	0.4	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
29	4.7	3.2	1.6	0.2	setosa
...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	virginica

	sepal_length	sepal_width	petal_length	petal_width	species
121	5.6	2.8	4.9	2.0	virginica
122	7.7	2.8	6.7	2.0	virginica
123	6.3	2.7	4.9	1.8	virginica
124	6.7	3.3	5.7	2.1	virginica
125	7.2	3.2	6.0	1.8	virginica
126	6.2	2.8	4.8	1.8	virginica
127	6.1	3.0	4.9	1.8	virginica
128	6.4	2.8	5.6	2.1	virginica
129	7.2	3.0	5.8	1.6	virginica
130	7.4	2.8	6.1	1.9	virginica
131	7.9	3.8	6.4	2.0	virginica
132	6.4	2.8	5.6	2.2	virginica
133	6.3	2.8	5.1	1.5	virginica
134	6.1	2.6	5.6	1.4	virginica
135	7.7	3.0	6.1	2.3	virginica
136	6.3	3.4	5.6	2.4	virginica
137	6.4	3.1	5.5	1.8	virginica
138	6.0	3.0	4.8	1.8	virginica
139	6.9	3.1	5.4	2.1	virginica
140	6.7	3.1	5.6	2.4	virginica
141	6.9	3.1	5.1	2.3	virginica
142	5.8	2.7	5.1	1.9	virginica
143	6.8	3.2	5.9	2.3	virginica
144	6.7	3.3	5.7	2.5	virginica
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Import stuff for Train-Test Split

We'll need: train_test_split

In [122]:
```
from sklearn.model_selection import train_test_split
```

Determine X_train, X_test, y_train, y_test using sklearn.

In [123]:
```
# determine X and y
X = a.drop(['species'],axis=1)
y = a['species']

# split into train and test
X_train,X_test,y_train,y_test = train_test_split(X,y)
print(np.shape(X_train))
```

```
(112, 4)
```

Import stuff for kNN

We'll need: KNeighborsClassifier, accuracy_score.

In [124]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

High Bias

Let's start by training a K Nearest Neighbors classifier with high bias.

In [126]:
```
classifier = KNeighborsClassifier(n_neighbors=110)
classifier.fit(X_train,y_train)
y_predicted = classifier.predict(X_test)

accuracy_score(y_test, y_predicted)
```

Out[126]:
```
0.5
```

What happened here?

With a super high number of n_numbers (110 in this case), the classifier has a very high bias. Therefore, the overall accuracy is very low, or 47.3%. Too much bias results in underfitting.

Now let's train a K Nearest Neighbors classifier with high variance.

In [129]:
```
classifier = KNeighborsClassifier(n_neighbors=1)
classifier.fit(X_train,y_train)
y_predicted = classifier.predict(X_test)

accuracy_score(y_test, y_predicted)
```

Out[129]: 0.9736842105263158

What happened here?

Since the n_neighbors value is super low, the variance is very high. However, the data is still quite accurate, with an accuracy score of 92%. Optimizing on only 1 nearest point means that the noise is modeled very high. Shuffling the data will greatly change the model every time.

Can we do better? Try experimenting with the value of K.

In [144]:
```
for k in range(1,110):
    classifier = KNeighborsClassifier(n_neighbors=k)
    classifier.fit(X_train,y_train)
    y_predicted = classifier.predict(X_test)
    a.append([k,accuracy_score(y_test,y_predicted)])
    y.append(accuracy_score(y_test,y_predicted))


x = np.reshape(a, (109,2))

plt.plot(range(1,110),y)

#the maximum accuracy is when k=2; look at matrix "x"

classifier = KNeighborsClassifier(n_neighbors=2)
classifier.fit(X_train,y_train)
y_predicted = classifier.predict(X_test)

accuracy_score(y_test, y_predicted)
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/indexes/api.py:107:
RuntimeWarning: '<' not supported between instances of 'str' and 'int',
sort order is undefined for incomparable objects
  result = result.union(other)

------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-144-2cc271a8c7ab> in <module>()
      4     y_predicted = classifier.predict(X_test)
      5     a.append([k,accuracy_score(y_test,y_predicted)])
----> 6     y.append(accuracy_score(y_test,y_predicted))
      7
      8

/anaconda3/lib/python3.6/site-packages/pandas/core/series.py in append
(self, to_append, ignore_index, verify_integrity)
   2152             to_concat = [self, to_append]
   2153         return concat(to_concat, ignore_index=ignore_index,
-> 2154                       verify_integrity=verify_integrity)
   2155
   2156     def _binop(self, other, func, level=None, fill_value=None):

/anaconda3/lib/python3.6/site-packages/pandas/core/reshape/concat.py in
 concat(objs, axis, join, join_axes, ignore_index, keys, levels, names,
 verify_integrity, sort, copy)
    223                        keys=keys, levels=levels, names=names,
    224                        verify_integrity=verify_integrity,
--> 225                        copy=copy, sort=sort)
    226     return op.get_result()
    227

/anaconda3/lib/python3.6/site-packages/pandas/core/reshape/concat.py in
 __init__(self, objs, axis, join, join_axes, keys, levels, names, ignor
e_index, verify_integrity, copy, sort)
    284                        ' only pd.Series, pd.DataFrame, and pd.P
anel'
    285                        ' (deprecated) objs are valid'.format(ty
pe(obj)))
--> 286                 raise TypeError(msg)
    287
    288             # consolidate

TypeError: cannot concatenate object of type "<class 'numpy.float64'>";
 only pd.Series, pd.DataFrame, and pd.Panel (deprecated) objs are valid
```

What happened here?

The maximum accuracy_score was recieved when k=2.

Explain the Bias-Variance tradeoff.

This tradeoff represents the struggle to find a value that minimizes both sources of error - bias and variance. With a high variance, the bias is near 0, which means that we aren't doing a good job of predicting the data values in the test data. With a very high bias, we aren't correctly finding trends in the data and are making a super general model.

How do bias and variance affect training and testing error?

High variance means that the data is overfitted; it doesn't fit well on a cross-validation set. A high bias would imply underfitting, where the model is not fit well to basically any data. High variance implies a low training error, but high testing error. Bias is vice-cersa.

K-Fold Cross Validation

What is K-Fold Cross Validation?

KFCV splits a group into k groups. Cross-validation is used to estimate a model's skill on unseen data. There's always a tradeoff when you want more test sets, which would mean less training sets.

If you have 200 data points, split it into 10 bins, so 20 points per bin. Run k separate learning experiments. Split each 20 into test and training sets. Average out the test results from the k experiments.

How can we use K-Fold Cross Validation to determine the optimal value of K to use in kNN?

First find a value that lowers the variance. Don't choose a very large K otherwise that would limit the number of iterations that are possible. A larger K means less bias towards overestimating but a higher variance (and less efficient).

Import stuff for K-Fold Cross Validation

We'll need: cross_val_score.

In [132]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

Now let's use cross validation to tune the hyperparameter K (the number of neighbors to consider). We want to choose the value of K that minimizes the test error.

In [148]:
```
# list of possible K-values for KNN
k_values = [i for i in range(1, 50)]

X_train,X_test,y_train,y_test = train_test_split(X,y)
```

Plot the accuracy of the classifier for each value of K.

In [154]:
```
e = []
f = []
average = []
for i in k_values:
    KNC = neighbors.KNeighborsClassifier(n_neighbors = i)
    KNC.fit(X_train,y_train)
    e.append(i)
    f.append(KNC.score(X_test,y_test))
    best = cross_val_score(KNC,X_test,y_test)
    average.append(np.mean(best))

plt.plot(e, average)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-154-504f3f0fa5b5> in <module>()
      7     e.append(i)
      8     f.append(KNC.score(X_test,y_test))
----> 9     best = cross_val_score(KNC,X_test,y_test)
     10     average.append(np.mean(best))
     11 

/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_validation.py in cross_val_score(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch)
    340                                 n_jobs=n_jobs, verbose=verbose,
    341                                 fit_params=fit_params,
--> 342                                 pre_dispatch=pre_dispatch)
    343     return cv_results['test_score']
    344 

/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_validation.py in cross_validate(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch, return_train_score)
    204             fit_params, return_train_score=return_train_score,
    205             return_times=True)
--> 206         for train, test in cv.split(X, y, groups))
    207 
    208     if return_train_score:

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/parallel.py in __call__(self, iterable)
    777             # was dispatched. In particular this covers the edge
    778             # case of Parallel used with an exhausted iterator.
--> 779             while self.dispatch_one_batch(iterator):
    780                 self._iterating = True
    781             else:

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/parallel.py in dispatch_one_batch(self, iterator)
    623                 return False
    624             else:
--> 625                 self._dispatch(tasks)
    626                 return True
    627 

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/parallel.py in _dispatch(self, batch)
    586         dispatch_timestamp = time.time()
    587         cb = BatchCompletionCallBack(dispatch_timestamp, len(batch), self)
--> 588         job = self._backend.apply_async(batch, callback=cb)
    589         self._jobs.append(job)
    590 

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/_parallel_backends.py in apply_async(self, func, callback)
```

```
    109     def apply_async(self, func, callback=None):
    110         """Schedule a func to be run"""
--> 111         result = ImmediateResult(func)
    112         if callback:
    113             callback(result)

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/_parall
el_backends.py in __init__(self, batch)
    330         # Don't delay the application, to avoid keeping the inp
ut
    331         # arguments in memory
--> 332         self.results = batch()
    333
    334     def get(self):

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/paralle
l.py in __call__(self)
    129
    130     def __call__(self):
--> 131         return [func(*args, **kwargs) for func, args, kwargs in
 self.items]
    132
    133     def __len__(self):

/anaconda3/lib/python3.6/site-packages/sklearn/externals/joblib/paralle
l.py in <listcomp>(.0)
    129
    130     def __call__(self):
--> 131         return [func(*args, **kwargs) for func, args, kwargs in
 self.items]
    132
    133     def __len__(self):

/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_validat
ion.py in _fit_and_score(estimator, X, y, scorer, train, test, verbose,
 parameters, fit_params, return_train_score, return_parameters, return_
n_test_samples, return_times, error_score)
    486         fit_time = time.time() - start_time
    487         # _score will return dict if is_multimetric is True
--> 488         test_scores = _score(estimator, X_test, y_test, scorer,
 is_multimetric)
    489         score_time = time.time() - start_time - fit_time
    490         if return_train_score:

/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_validat
ion.py in _score(estimator, X_test, y_test, scorer, is_multimetric)
    521     """
    522     if is_multimetric:
--> 523         return _multimetric_score(estimator, X_test, y_test, sc
orer)
    524     else:
    525         if y_test is None:

/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_validat
ion.py in _multimetric_score(estimator, X_test, y_test, scorers)
    551             score = scorer(estimator, X_test)
    552         else:
```

```
--> 553             score = scorer(estimator, X_test, y_test)
    554
    555         if hasattr(score, 'item'):

/anaconda3/lib/python3.6/site-packages/sklearn/metrics/scorer.py in _pa
ssthrough_scorer(estimator, *args, **kwargs)
    242 def _passthrough_scorer(estimator, *args, **kwargs):
    243     """Function that wraps estimator.score"""
--> 244     return estimator.score(*args, **kwargs)
    245
    246

/anaconda3/lib/python3.6/site-packages/sklearn/base.py in score(self,
 X, y, sample_weight)
    347         """
    348         from .metrics import accuracy_score
--> 349         return accuracy_score(y, self.predict(X), sample_weight
=sample_weight)
    350
    351

/anaconda3/lib/python3.6/site-packages/sklearn/neighbors/classificatio
n.py in predict(self, X)
    143         X = check_array(X, accept_sparse='csr')
    144
--> 145         neigh_dist, neigh_ind = self.kneighbors(X)
    146
    147         classes_ = self.classes_

/anaconda3/lib/python3.6/site-packages/sklearn/neighbors/base.py in kne
ighbors(self, X, n_neighbors, return_distance)
    345                     "Expected n_neighbors <= n_samples, "
    346                     " but n_samples = %d, n_neighbors = %d" %
--> 347                     (train_size, n_neighbors)
    348                 )
    349         n_samples, _ = X.shape

ValueError: Expected n_neighbors <= n_samples,  but n_samples = 25, n_n
eighbors = 26
```

What is the optimal value of K found using 20-fold cross validation?

In [156]:
```
#10 is generally an optimal value
```

Explain stratified cross validation.

This involves selecting folds so that the mean response value of the folds is approximately equal in all the folds.

Grid Search

What is Grid Search?

Grid Search involves having a set of models with different parameter values, and then evaluating them with a cross-validation to select the most accurate one.

Import stuff for GridSearch

We'll need: GridSearchCV.

In [157]:
```
from sklearn.model_selection import GridSearchCV
from sklearn import svm, datasets
```

Try using GridSearch to determine the optimal value of K.

In [161]:
```
# list of possible k values for KNN
k_values = list(range(1, 50))
parameters = {"n_neighbors":k_values}

clf = GridSearchCV(KNC, parameters)
clf.fit(X_train,y_train)
```

Out[161]:
```
GridSearchCV(cv=None, error_score='raise',
       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30, m
etric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=26, p=2,
           weights='uniform'),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
48, 49]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```