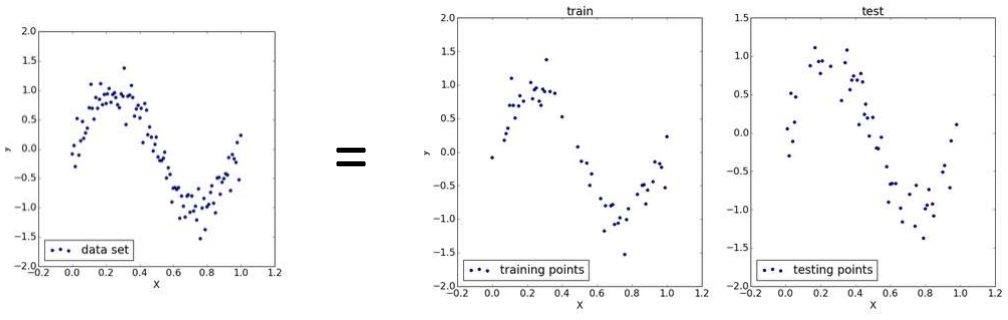
model selection and validation

unseen external data

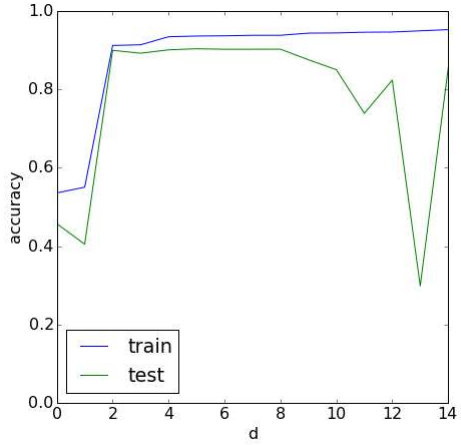
(data not seen during training)

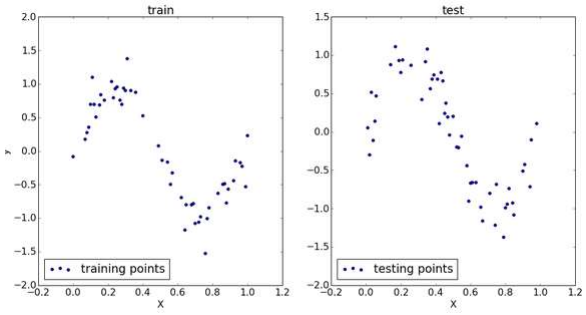
For instance, when we augment the features in a data set by polynomial features of a certain degree *d* we need to set *d* such that it allows for training a model that performs best on all unseen external data.

train-test split



model selection and validation



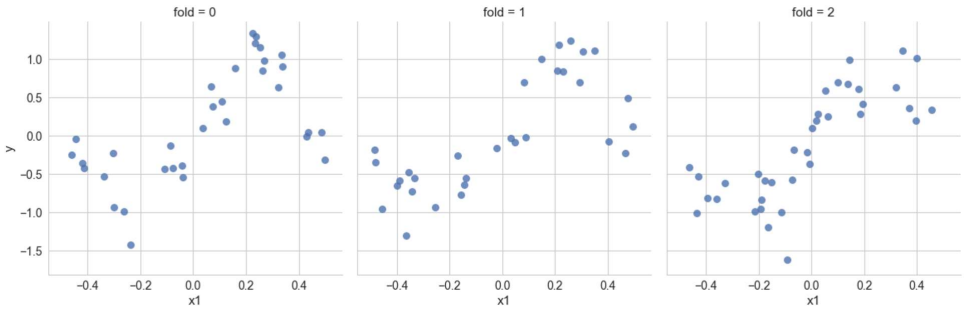


- *d* > 8 overfitting
- *d* = 1 underfitting
- generalization performance?
- validation set

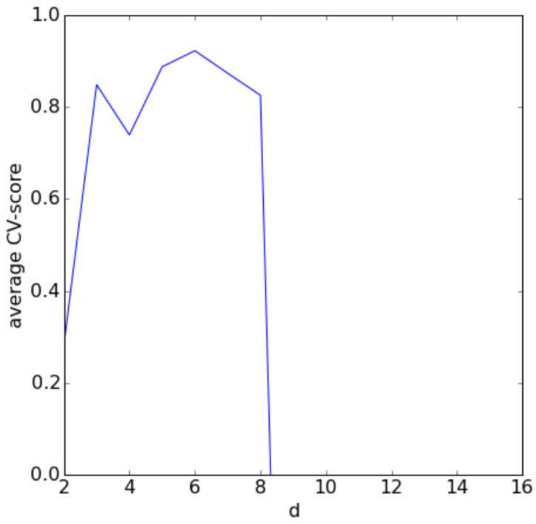
k-fold cross-validation (CV)

- fewer data points for training
- performance results can depend strongly on a particular random choice of the data set splits
- *k*-fold CV:
  - partition data set into *k* smaller sets (folds)
  - For each fold D_i ($i = 1 \ldots k$) do
    1. train a model m_i on the data points in folds D_j with $j \neq i$,
    2. use m_i to compute predictions for D_i.

3-fold cross-validation (CV)



3-fold cross-validation (CV)



```
from sklearn.linear_model import LinearRegression

model = LinearRegression(fit_intercept=True)

X = dataset.copy()
y = X.pop('y')
folds = X.pop('folds')

train_scores = [0] #no features (d=0)
test_scores = [0]

model.fit(X[folds!=1],y[folds!=1])
train_scores.append(model.score(X[folds!=1],y[folds!=1])) # d=1
test_scores.append(model.score(X[folds==1],y[folds==1]))

for degree in range(2,18,1):
    X['x1^'+str(degree)] = X['x1']**degree
    model.fit(X[folds!=1],y[folds!=1])
    train_scores.append(model.score(X[folds!=1],y[folds!=1]))
    test_scores.append(model.score(X[folds==1],y[folds==1]))

tmp = pd.DataFrame()
tmp['train-scores'] = train_scores
tmp['test-scores'] = test_scores
tmp.plot()
plt.show()
```

```
def my_cross_val_predict(model,X,y,cv=5):
    predictions = np.empty(len(y))
    folds = np.random.randint(0,cv,size=len(y))
    for i in range(cv):
        Xtest = X[folds==i]
        ytest = y[folds==i]
        Xtrain = X[folds!=i]
        ytrain = y[folds!=i]
        model.fit(Xtrain,ytrain)
        p = model.predict(Xtest)
        counter = 0
        for j in range(len(folds)):
            if folds[j] == i:
                predictions[j] = p[counter]
                counter += 1
    return predictions
```

```
from sklearn.cross_validation import cross_val_predict

cv_predictions = cross_val_predict(model,X,y,cv=5)
print cv_predictions
```

```
print metrics.r2_score(y,cv_predictions)
```

```
0.412207855062
```