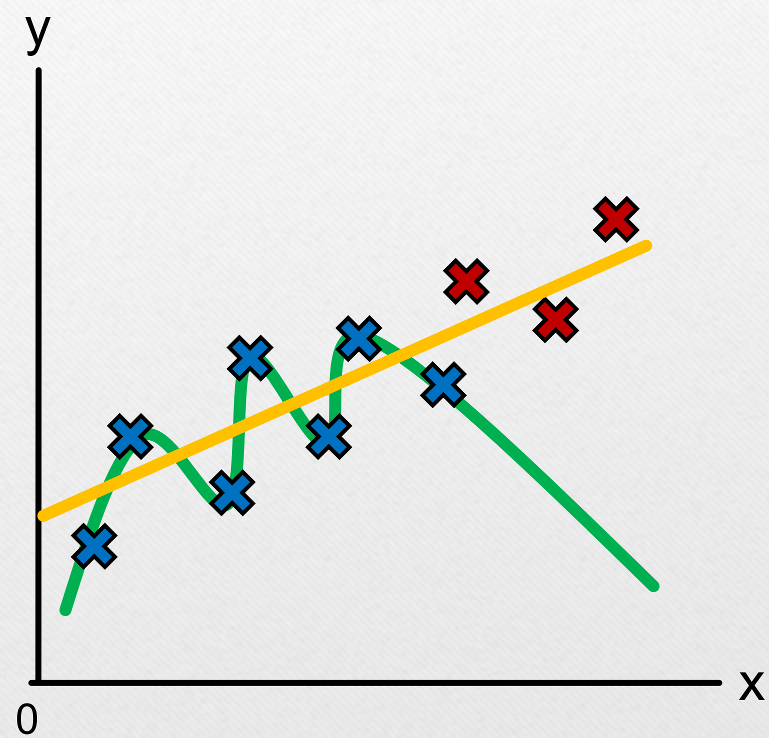
Cross Validation

ECON 4810

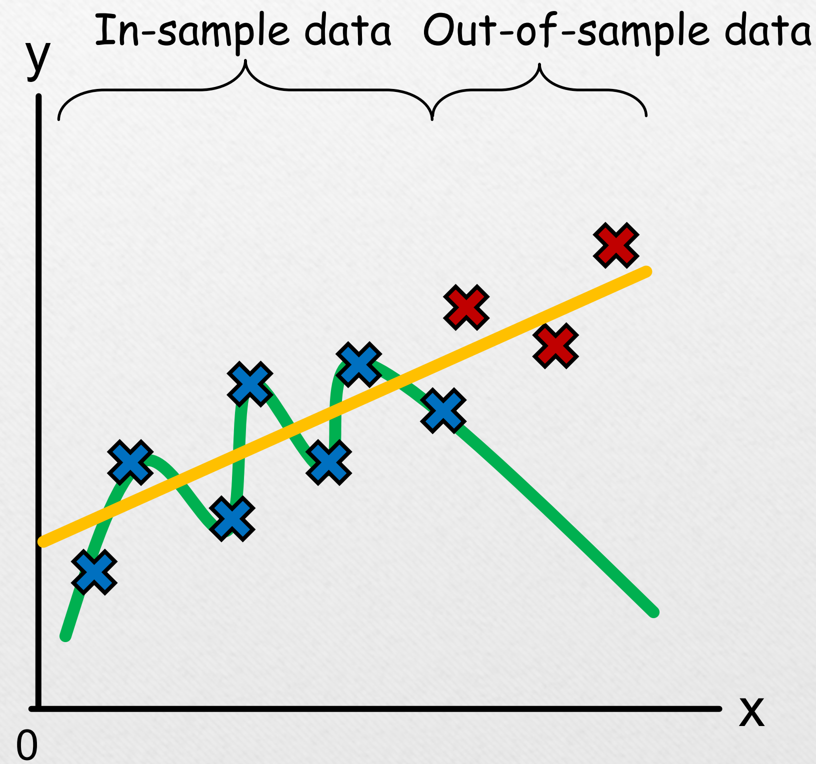
Out-of-Sample Test

- We want accurate prediction
- Given enough complexity, a model will always do well with data it has seen
- We want to know if the model does well with data it has **not yet seen**



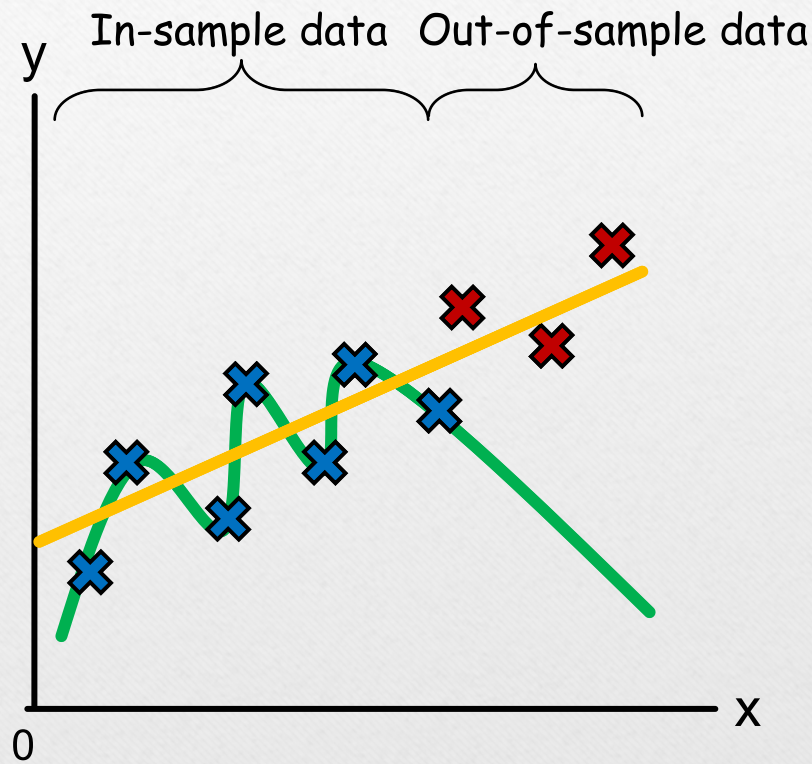
Out-of-Sample Test

- We will intentionally keep part of the data from the model training process
- This reserved data is not seen by the model during training, allowing us to conduct an **out-of-sample test**
- We pick the model (or model parameters) that has the highest out-of-sample prediction accuracy



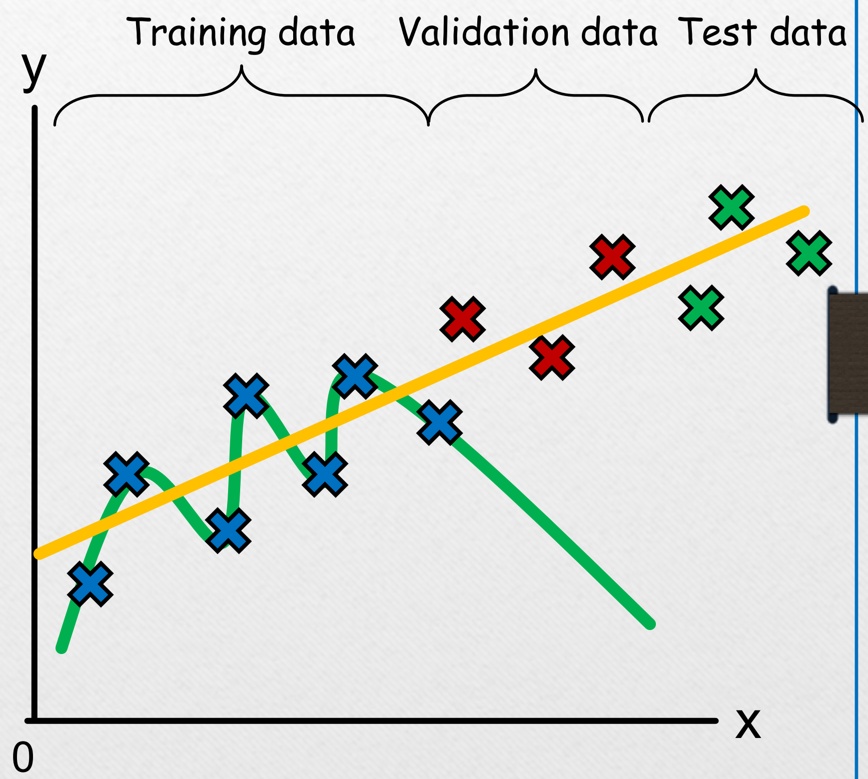
How to Find the Best α?

- Idea: we pick the model (or model **hyperparameters** such as α) that has the highest out-of-sample prediction accuracy
- Problem: If we pick α this way, the model has seen the reserved data, so it is no longer out of sample



How to Find the Best Model

- Solution: split the data into three parts:
  1. **Training set** for training the model
  2. **Validation set** for choosing models and hyperparameters
  3. **Test set** for reporting out-of-sample performance

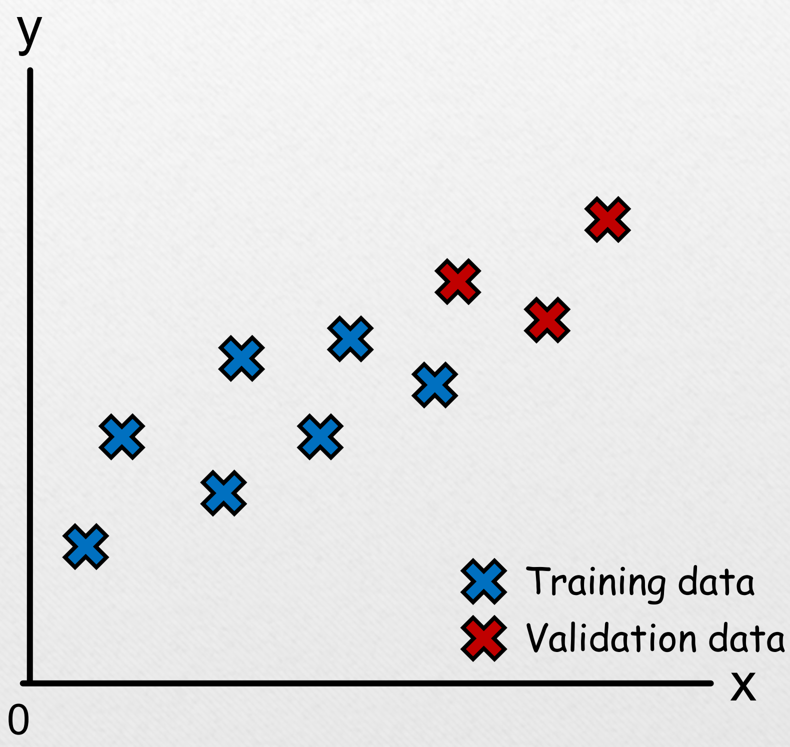


How to Find the Best Model

- To ensure that each set of data is representative, we generally want to split the data randomly rather than sequentially
- The exception is time series data. We must split such data sequentially to avoid hindsight bias

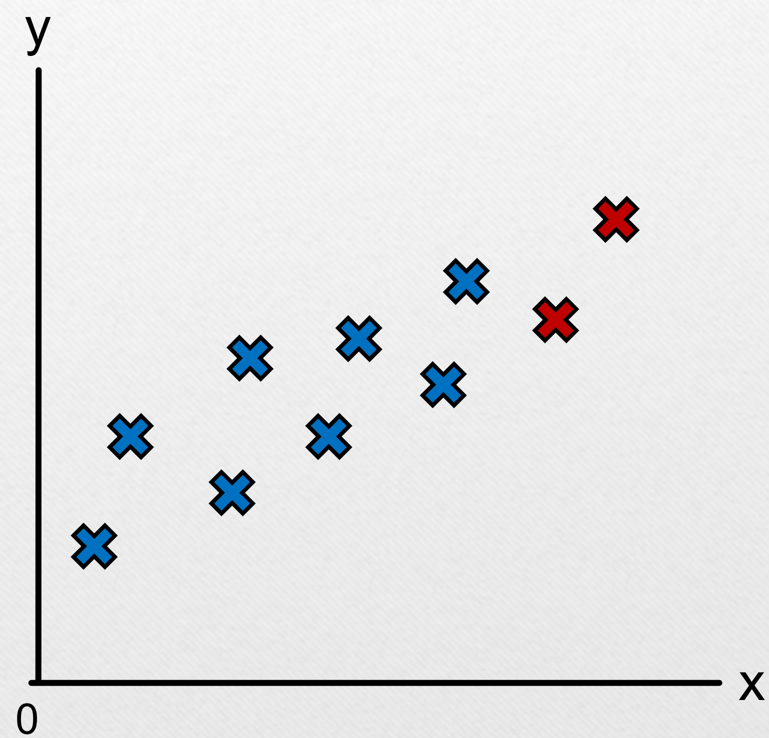
K-Fold Cross Validation

- We might question the representativeness of the validation set, particularly when the sample size is small
- At the same time, we might be unwilling to increase its size since that makes the training set smaller
- What to do in this case?



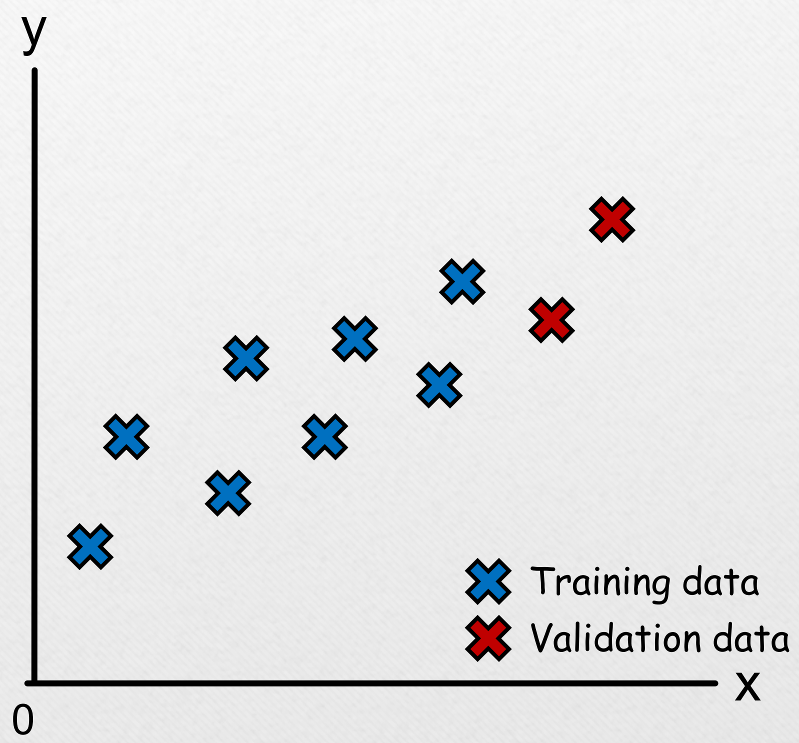
K-Fold Cross Validation

- We can handle this problem by using ***k*-fold cross validation**
- We first divide the data we intend to use for training and validation into *k* equal-sized folds. Five is a common choice
- We repeat the training process for *k* times. Each time we use one fold for validation and the rest for training



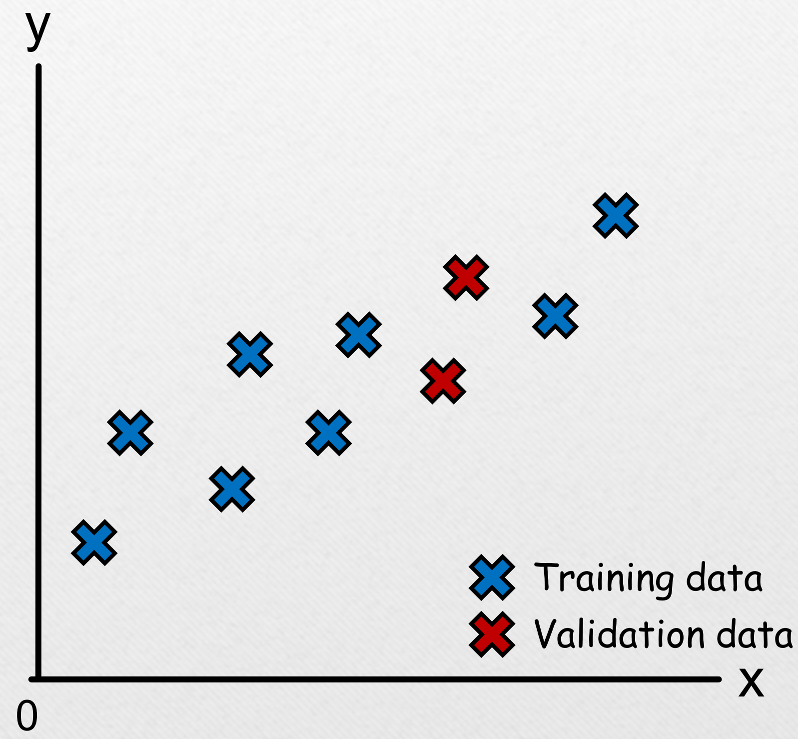
K-Fold Cross Validation

- Iteration 1



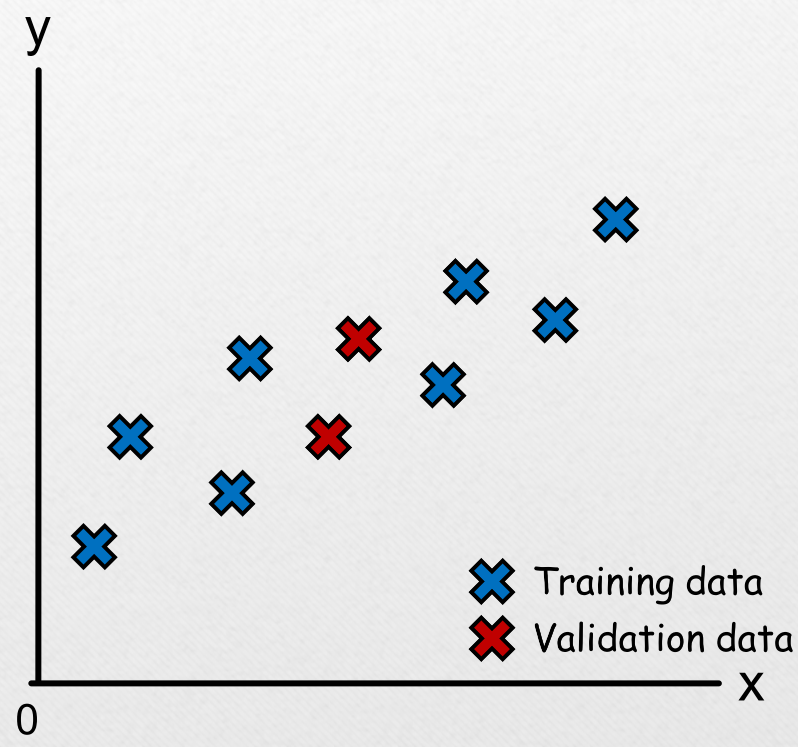
K-Fold Cross Validation

- Iteration 2



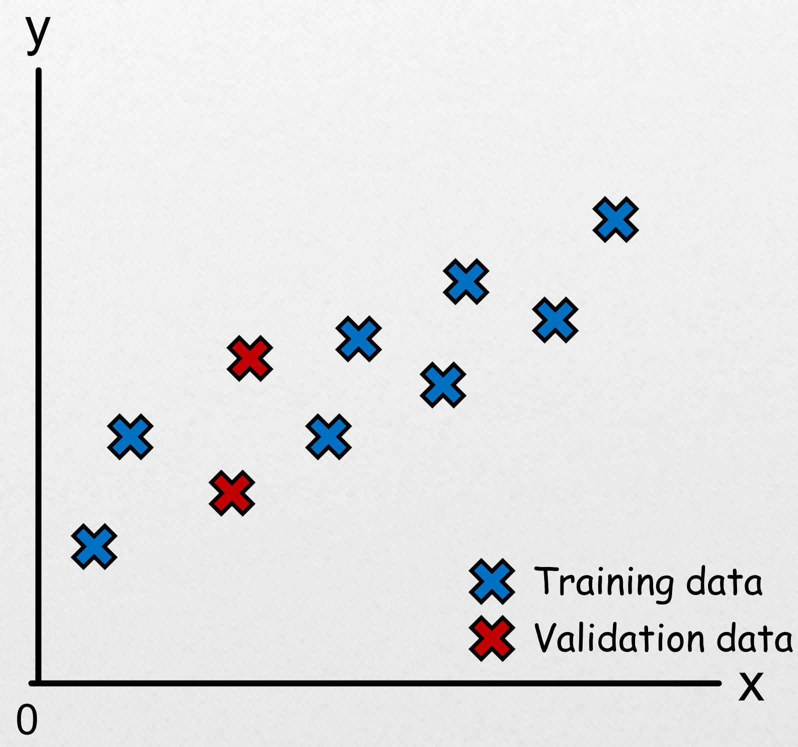
K-Fold Cross Validation

- Iteration 3



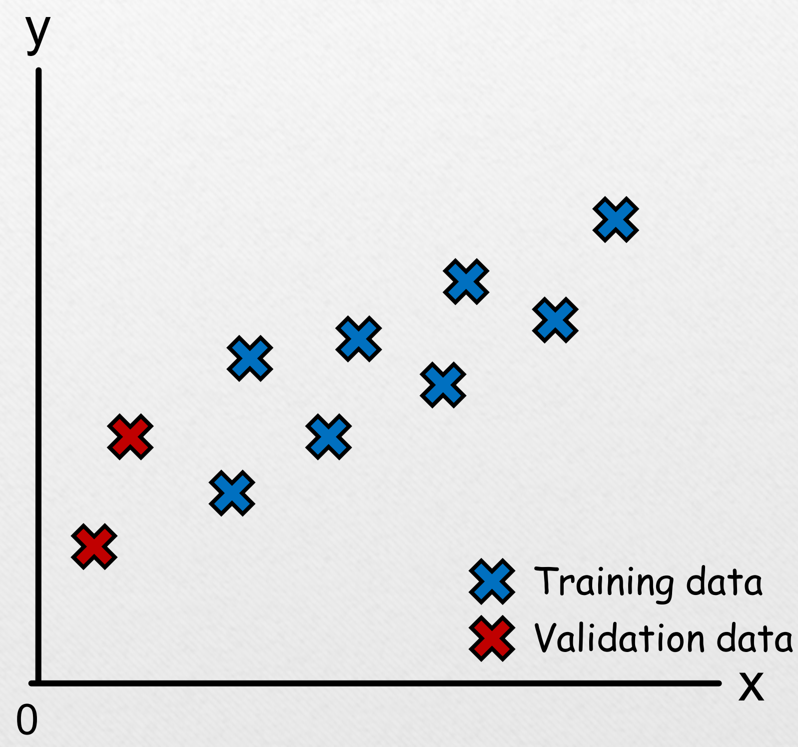
K-Fold Cross Validation

- Iteration 4



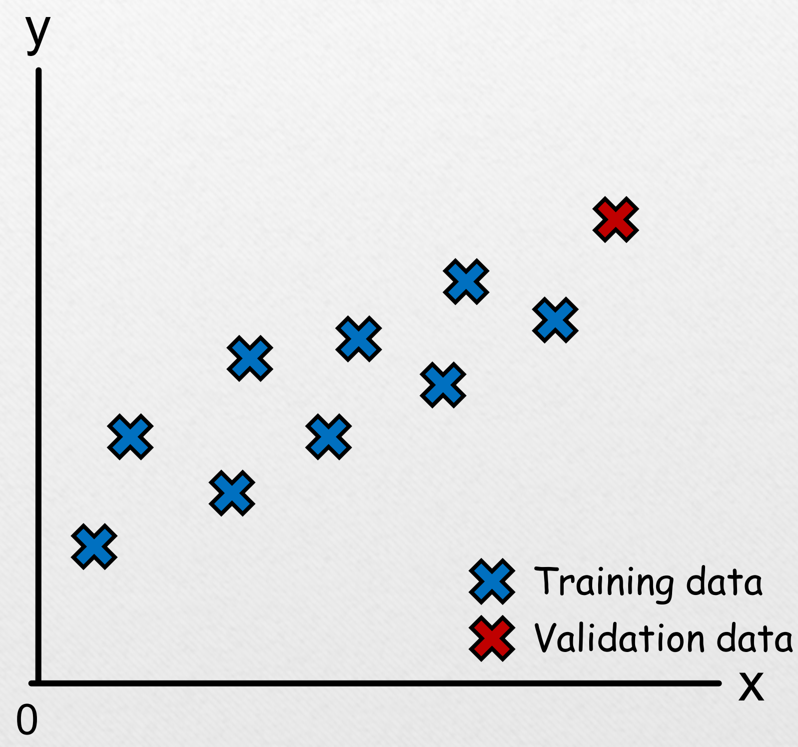
K-Fold Cross Validation

- Iteration 5



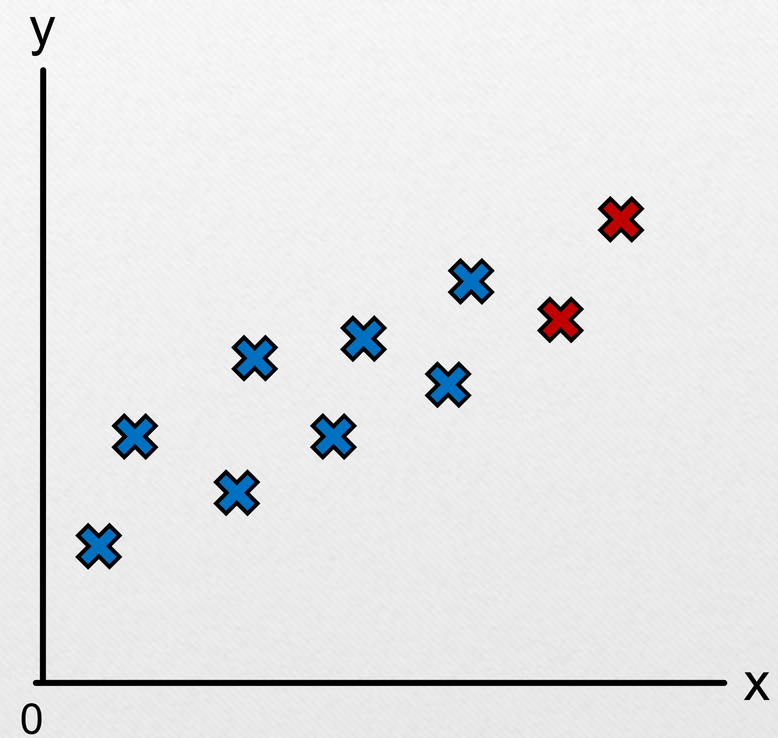
Leave-One-Out Validation

- If each fold only has one sample, the process is called leave-one-out validation (LOOV)



K-Fold Cross Validation

- Performance measures are averaged across folds
- K-fold cross validation trades training time for representativeness of validation data
- Might not be feasible when model is time consuming to train



K-Fold Cross Validation Workflow

1. Prepare data
   `X_in,X_test,y_in,y_test = train_test_split(X,y)`
2. Set the parameter search space
   `p = {'alpha':[1,10,…]}`
3. Create instance of model classes
   ```
   m = Lasso()
   gscv = GridSearchCV(m,p,cv=5)
   ```
4. Fit model
   `gscv.fit(X_in,y_in)`
5. Check model performance
   `gscv.score(X_test,y_test)`
6. Make prediction
   `gscv.predict(X_new)`

Cross Validation Workflow for Time-Consuming Models

1. Prepare data

```
X_in,X_test,y_in,y_test =
train_test_split(X,y)
X_train,X_valid,y_train,y_valid =
train_test_split(X_in,y_in)
```

2. Set the parameter search space

```
alpha=[1,10,…]
```

3. Fit a model for each parameter

```
for a in alpha:
    m = Lasso(alpha=a)
    m.fit(X_train,y_train)
    s = m.score(X_valid,y_valid)
    if s > best_score:
        best_model = m
        best_score = s
```

4. Check model performance

```
best_model.score(X_test,y_test)
```

5. Make prediction

```
best_model.predict(X_new)
```