Prediction performance metrics

Prediction error

- how well does a predictor g work on a data set $x^1, \ldots, x^n$, $y^1, \ldots, y^n$?
- that is, how close are the predictions $\hat{y}^i = g(x^i)$ to the actual outcomes y^i?
- a *performance metric* is a scalar measure of how large the prediction errors are
- usually the smaller the metric, the better the prediction performance
- prediction performance metric is sometimes called the *prediction error*

Prediction performance metrics

- *mean square error*: $\frac{1}{n}\sum_{i=1}^{n}||\hat{y}^i - y^i||_2^2$ (for scalar y, $\frac{1}{n}\sum_{i=1}^{n}(\hat{y}^i - y^i)^2$)
- *root mean square (RMS) error*:

$$\left(\frac{1}{n}\sum_{i=1}^{n}||\hat{y}^i - y^i||_2^2\right)^{1/2}$$

- *mean absolute error* (MAE) (for scalar y): $\frac{1}{n}\sum_{i=1}^{n}|\hat{y}^i - y^i|$
- *mean fractional error* (for scalar, positive $y, \hat{y}$):

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}^i - y^i|}{\min\{\hat{y}^i, y^i\}}$$

- and many others, *e.g.*, *median error* (for scalar y), median of $|\hat{y}^i - y^i|$, $i = 1, \ldots, n$

Comparing predictors using a performance metric

- prediction performance metric allows us to compare different predictors on a given data set
- example conclusions (on a common data set)
  - 'k-NN with $k = 7$ does better than k-NN with $k = 12$'
  - 'my neural network does slightly better than your linear model'
- conclusions like these depend on the performance metric, so choose it thoughtfully

Generalization

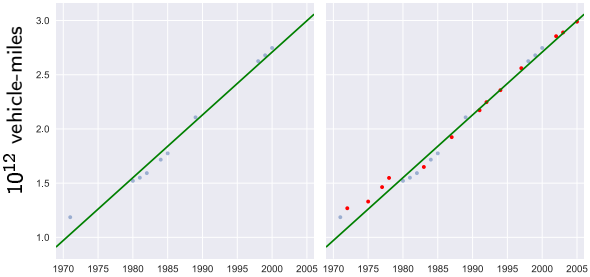
Generalization

- *generalization* is the ability of a predictor to perform well on unseen data
- 'unseen' means the data was not used to create the prediction model
- can analyze mathematically after making some probabilistic assumptions (which we won't discuss in this course)
- instead we'll see some practical methods for assessing generalizability

In-sample and out-of-sample data

- we construct a predictor based on *training data* or *in-sample data*
- we'd like it to work well on *out-of-sample data*, *i.e.*, unseen data
- if it does, we say the predictor *generalizes*, *i.e.*, makes good predictions on data it has never seen
- if it doesn't we say it *fails to generalize* or is *over-fit*

Example: Vehicle-miles traveled



- we predict $y =$ vehicle-miles traveled from feature $x =$ year
- we use 'straight-line' predictor, $\hat{y} = \theta_1 + \theta_2 x$, parameters chosen using least squares
- we *train* predictor using the 12 (in-sample) blue points, MSE 0.0047
- we use this to *predict* y for the 14 (out-of-sample) red points, MSE 0.0051
- so, this predictor generalizes

Out-of-sample validation

Out-of-sample validation

- a method to simulate how the predictor will perform on unseen data
- key idea: divide the data you have into two sets, *train* and *test*
- division of data into training/test sets is often *random* (80/20 or 90/10 are common splits)

- use the *training set* data to choose ('train') the predictor
- use the *test set* or *validation set* data to evaluate the predictor, using your performance metric

- this is an honest simulation of how the predictor works on unseen data
- we *hope* that the predictor will work in a similar way on new unseen data
- this hope is founded on the assumption that future data 'looks like' test data

Out-of-sample validation

- the *test set performance* is what matters
- the *training set performance* does not matter (but we'd expect it to be good)
- we usually expect the test performance to be a little worse than the training performance
- sometimes the test performance is OK, but much worse than the training performance, which is just fine
- example: training error for 1-NN predictor is zero, but it still can make useful predictions

Interpreting validation results

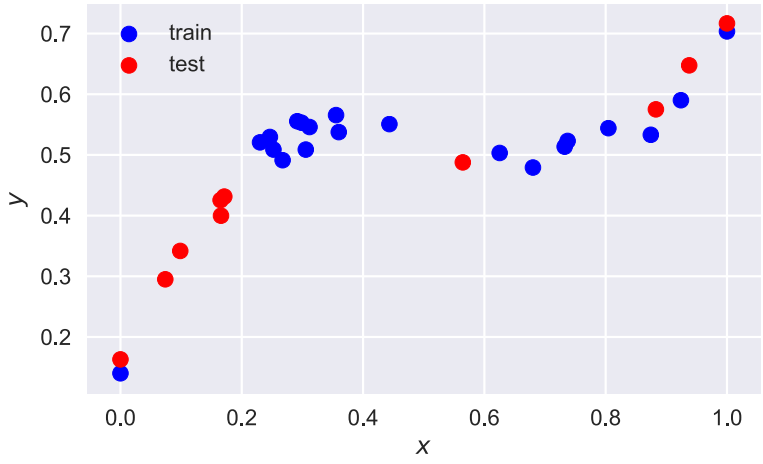
- the *test set performance* is what matters
- the *training set performance* does not matter
- top row in the table below are good prediction models

	small train error	large train error
small test error	generalizes, performs well	possible (luck, or fraud?)
large test error	fails to generalize, overfit	generalizes, but performs poorly

Choosing among candidate predictors

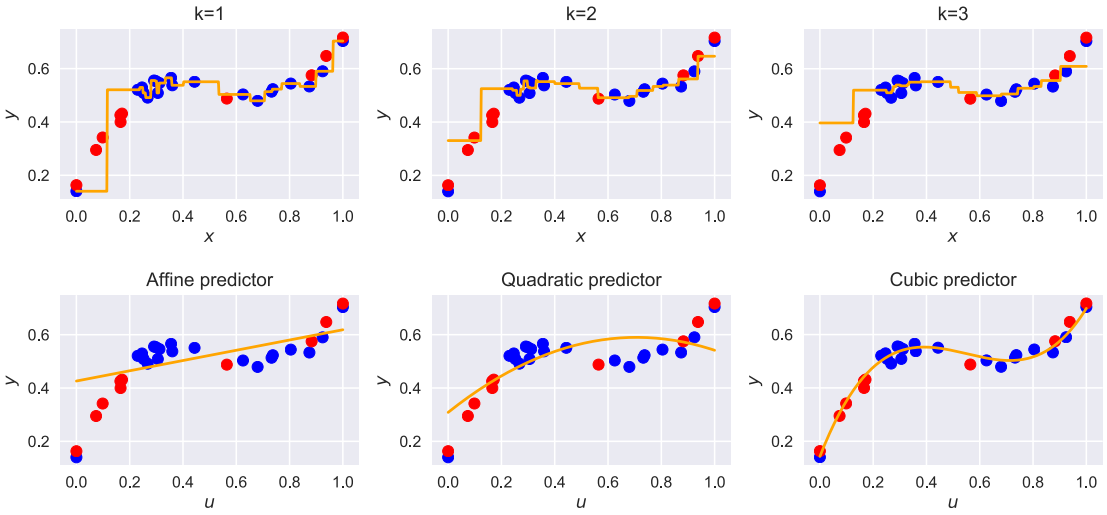
- validation is a good method to choose among candidate predictors
- typically we choose predictor among candidates with smallest test error
- in some cases, might accept a bit larger test error in favor of a 'simpler' predictor (more on this later)

Example — train and test data



- data (x^i, y^i) with $x^i \in \mathbf{R}$, training data set size 20, test set size 10

Example — k-NN and polynomial models



Example — comparison of models

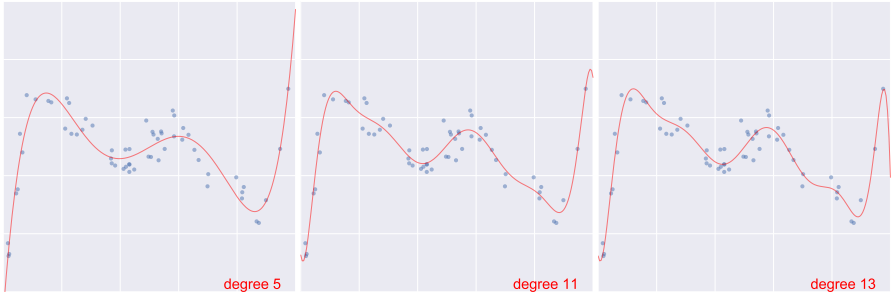
- we use RMS error as performance metric
- which is the best prediction model?

	k-NN			polynomial		
RMS error	$k=1$	$k=2$	$k=3$	affine	quadr.	cubic
train	0	0.046	0.062	0.082	0.073	0.017
test	0.101	0.083	0.106	0.110	0.086	0.025

Example: Polynomial fit

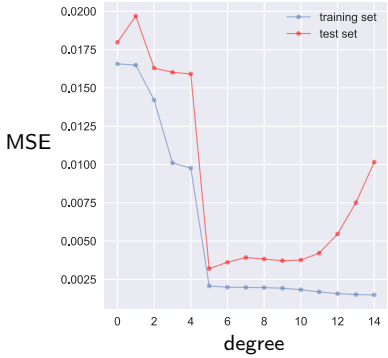
- raw data is scalar $u \in \mathbf{R}$, scalar $v = y$
- we use feature mapping $x = \phi(u) = (1, u, \ldots, u^{d-1})$ and linear predictor $g(x) = \theta^{\mathsf{T}} x$
- predictor is polynomial of u of degree $d - 1$: $\hat{y} = g(x) = \theta_1 + \theta_2 u + \cdots + \theta_d u^{d-1}$
- choose θ by least squares

Example: Polynomial fit



- $n = 60$ data points
- predictor for $d = 6$, $d = 12$, $d = 14$
- which predictor is best?
- degree 13 predictor has smallest training RMS error

Choosing degree by validation



- split 60 data points into 48 train and 12 test points
- evaluate RMS error of each predictor on train and tests sets
- RMS error on training data set decreases with increasing degree
- but plot of test error suggests best choice of degree is 5

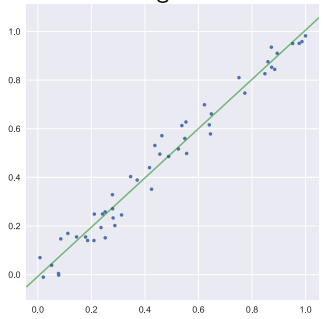
Cross validation

Cross validation

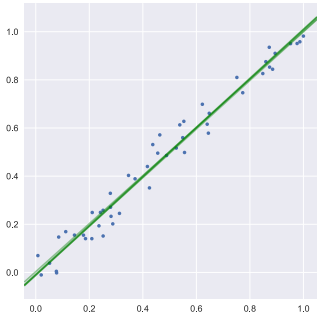
- an extension of out-of-sample validation
  - divide the data into k *folds*
  - for each i, fit predictor on all data but fold i
  - evaluate predictor on fold i
  - use average test error, across the folds, to judge the method
  - standard deviation of fold test error gives idea of how well model generalizes across folds
- can give some idea of the variability of the test error
- can assess *stability* of the modeling method by looking at predictor parameters found in each fold (are they similar? very different?)

Example: Cross validation

trained using all the data



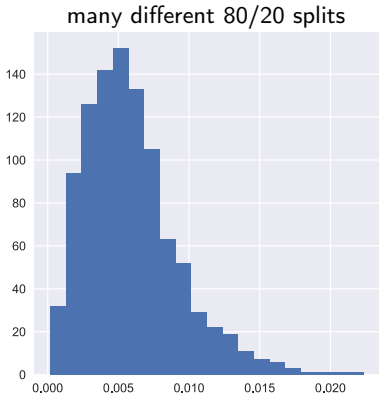
5-fold validation



fold	training loss	test loss	θ_1	θ_2
1	0.0027	0.0027	0.00334	0.998
2	0.0069	0.0071	-0.01095	1.010
3	0.0070	0.0058	-0.01248	1.021
4	0.0054	0.0047	-0.00959	1.017
5	0.0052	0.0066	-0.00691	1.018

And to be even more confident . . .

- split data into train:test (say, 80:20) *randomly*
- train predictor using training data
- evaluate on test data
- repeat above for many different random splits into train:test
- look at histogram of test errors to judge the method
- called *repeated train/validation*
- plot shows RMS test error for data from previous slide



Once you've chosen a predictor

Train / validate / test

- if you evaluate too many models on the test set, you're beginning to learn it, and it's no longer a good simulation of how the model will do on data you've never seen
- to avoid this, split original data into 3 data sets
  - *training data set*, used to fit multiple candidate models
  - *validation data set*, used to evaluate performance of models
  - *test data set*, a pristine, untouched data set reserved to evaluate the model you choose in validation

  (unfortunately, some people reverse the meanings of 'test' and 'validation' here)
- some practitioners do this; others don't
- in this course, we'll just use out-of-sample or 5-fold cross-validation

The final predictor

- you're now satisfied, possibly using train / validation / test, with your choice of predictor
- one option is to just use that predictor, which was trained on only the training data
- another option is to re-train your chosen predictor on the *whole* original data set, including data previously reserved as test and / or validation
- both practices are common
- example:
  - you train k-NN predictors for various values of k
  - validation suggests that $k = 6$ is a good choice
  - the final predictor you use is k-NN with $k = 6$, using *all the original data*