Validation

Jong-Han Kim

EE787 Machine learning
Kyung Hee University

Generalization

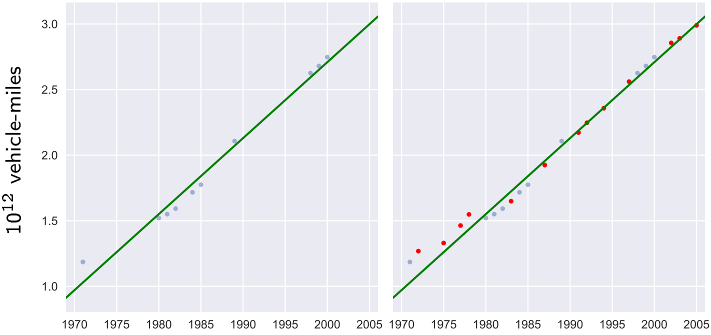
Generalization

- we would like to *learn* from a dataset
- would like learned properties to hold on unseen data
- *generalization* is the ability of a predictor to perform well on unseen data
- can be mathematically analyzed by making 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*
- if it doesn't we say it *fails to generalize*

Example: Vehicle-miles traveled



- we *train* straight-line 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 into two sets, *train* and *test*

- use the *training set* data to choose ('train') the predictor
- use the *test set* or *validation set* data to evaluate the predictor

- 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 founded on the assumption that future data 'looks like' test data

Out-of-sample validation

- the *test set error* (empirical risk on test data set) is what matters
- the *training set error* (empirical risk on training data set) does not matter
- selection of data for the training/test sets is often *random*
  (80/20 or 90/10 are common splits)
- we expect the test error to be a little bigger than the training error
- if the test error is much greater than training error, the predictor is overfit
  (but if the test error is acceptable, this can still be useful)

Interpreting validation results

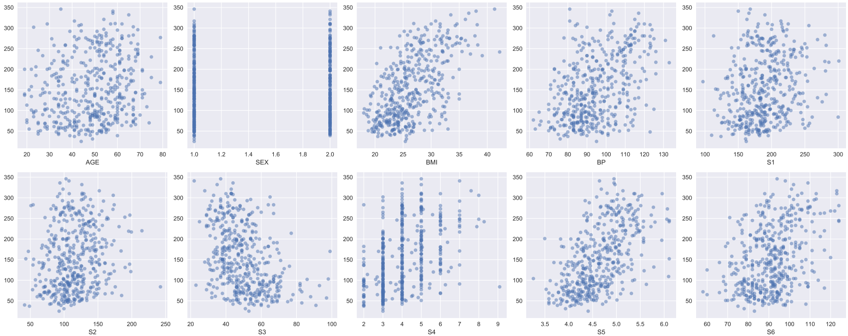
	small training error	large training error
small test error	generalizes, performs well	possible (luck, or fraud?)
large test error	fails to generalize	generalizes, but performs poorly

Choosing a predictor

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: Diabetes



- 10 explanatory variables (age, bmi,. . .)
- data from 442 individuals
- use half for training, half for validation (50-50 split)

Example: Diabetes

features	train loss	test loss
all	2640	3224
S5 and BMI	3004	3453
S5	3869	4227
BMI	3540	4277
S4 and S3	4251	5302
S4	4278	5409
S3	4607	5419
none	5524	6352

- test loss gives a method of selecting features
- data indicates that using only 2 features, S5 and BMI, would predict diabetes almost as well as using all 10 features
- combining S4 and S3 doesn't buy much; combining S5 and BMI much better

Overfitting

Overfitting

- we have a family of predictors
- we might choose a predictor that fits the training data very closely
- but often this leads to poorly fitting the test data
- called *overfitting*

Example: Polynomial fit

- raw data is scalar $u \in \mathbf{R}$
- we use *polynomial features*

$$x = \phi(u) = \begin{bmatrix} 1 \\ u \\ u^2 \\ \vdots \\ u^{d-1} \end{bmatrix}$$

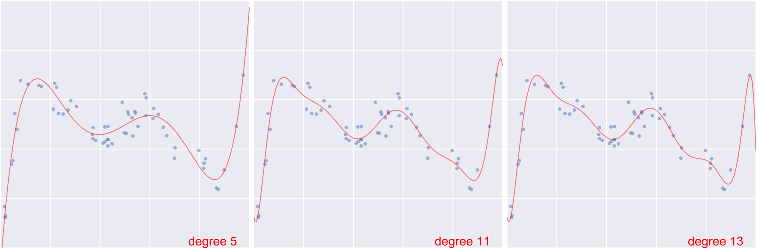
  and linear predictor $g(x) = \theta^T x$

- predictor is polynomial of u of degree $d-1$:

$$\hat{y} = g(x) = \theta_1 + \theta_2 u + \cdots + \theta_d u^{d-1}$$

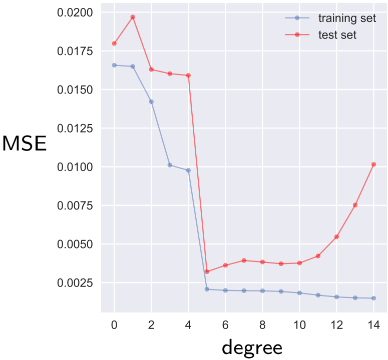
- choose θ by ERM with square loss $l^{\mathsf{sqr}}(\hat{y}, y) = (\hat{y} - y)^2$

Example: Polynomial fit



- $n = 60$ data points
- predictor for $d = 6$, $d = 12$, $d = 14$

Choosing degree by validation



- split 60 data points into 48 train and 12 test points
- plot suggests best choice of degree is 5
- can now use degree 5 fit on all data

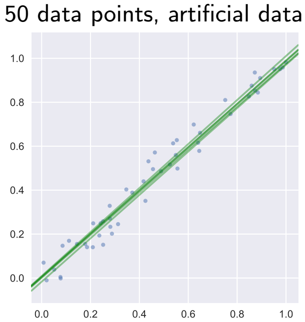
Cross validation

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

- 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

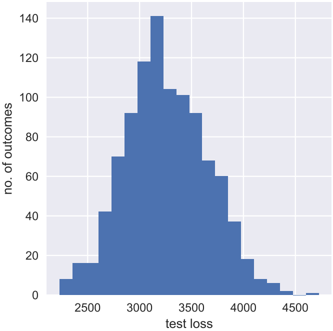


fold	training loss	test loss	θ_1	θ_2
1	0.028	0.030	-0.016810	0.9874
2	0.026	0.036	0.005917	0.9822
3	0.030	0.023	0.008961	1.0010
4	0.028	0.031	0.004135	0.9859
5	0.028	0.029	0.000844	0.9742

And to be even more confident . . .

- split data into train:test (say, 80:20) *randomly*
- develop predictor from 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*

Example: Repeated train/validation



- 1000 experiments
- diabetes data, with BMI and S5 features
- mean loss: 3258