Introduction to Predictive Models

Mladen Kolar (mkolar@chicagobooth.edu)

Supervised learning

Training experience: a set of labeled examples (samples, observations) of the form $(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_n, y_n)$ where $x_i = (1, x_{i1}, \ldots, x_{ip})$ are vectors of input variables (covariates, predictors) and y is the output

This implies the existence of a "teacher" who knows the right answers

What to learn: A function $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_p \mapsto \mathcal{Y}$ which maps the input variables into the output domain

Goal: minimize the error (loss) function

- Ideally, we would like to minimize error on all possible instances
- But we only have access to a limited set of data ...

Supervised learning: Regression

Simply put, the goal is to predict a target variable Y with input variables X!

A useful way to think about it is that Y and X are related as

$$y_i = f(x_i) + \epsilon_i$$

and the goal is to learn f from n examples

- $f(x)$: the part of Y you learn from X, the signal.
- ϵ: the part of Y you do not learn from X, the noise.

Regression in business

Optimal portfolio choice:

- Predict the future joint distribution of asset returns
- Construct an optimal portfolio (choose weights)

Determining price and marketing strategy:

- Estimate the effect of price and advertisement on sales
- Decide what is optimal price and ad campaign

Credit scoring model:

- Predict the future probability of default using known characteristics of borrower
- Decide whether or not to lend (and if so, how much)

Regression in everything

Straight prediction questions:

- What price should I charge for my car?
- What will the interest rates be next month?
- Will this person like that movie?

Explanation and understanding:

- Does your income increase if you get an MBA?
- Will tax incentives change purchasing behavior?
- Is my advertising campaign working?

Objectives

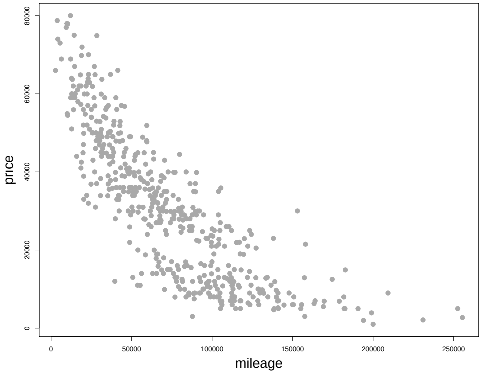
On the basis of the training data we would like to:

- Accurately predict unseen test cases.
- Understand which inputs affect the outcome, and how.
- Assess the quality of our predictions and inferences.

Example: Predicting Price of Used Cars

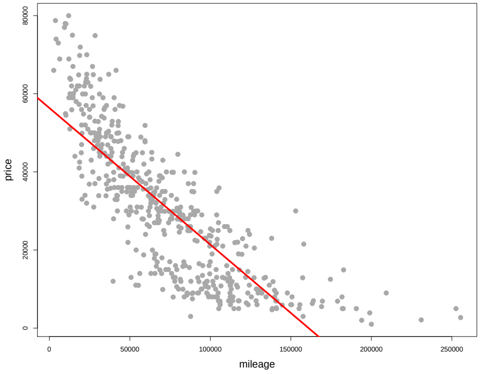
We might be interested in predicting the price of a used car as a function of its characteristics. Each observation corresponds to a used car at a dealer.

- price: US dollars
- mileage: how many miles does a car have.



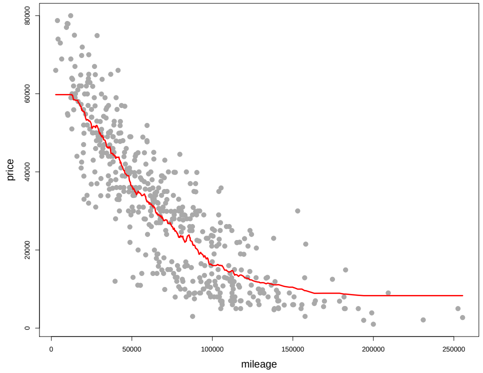
What should $f(\cdot)$ be?

How about this. . .



If *mileage* $= 50,000$ what is the prediction for *price*?

or this?



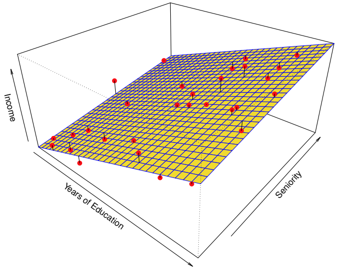
If *mileage* $= 50,000$ what is the prediction for *price*?

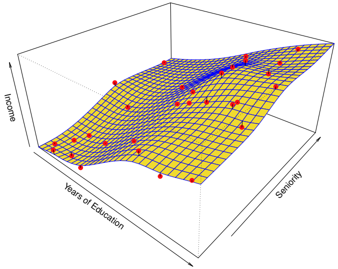
How do we estimate $f(\cdot)$?

- Using *training data*:

$$\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$$

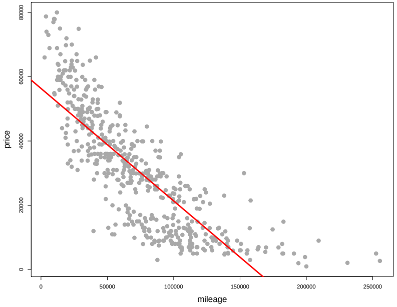
- We use a machine learning method to *estimate* the function $f(\cdot)$
- Two general methodological strategies:
  1. simple parametric models (restricted assumptions about $f(\cdot)$)
  2. non-parametric models (flexibility in defining $f(\cdot)$)



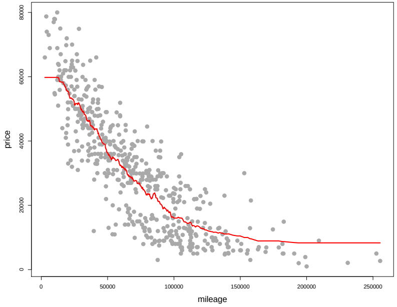


Back to Used Cars

Parametric Model
($Y = \alpha + \beta x + \epsilon$)



Non-Parametric Model
(k-nearest neighbors)



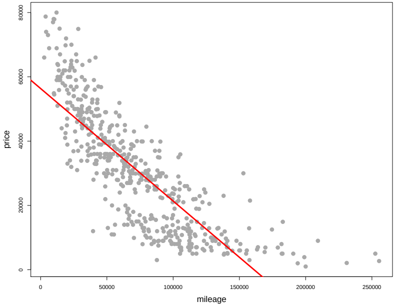
Simple parametric model:

$$Y_i = \alpha + \beta\, x_i + \epsilon_i$$

Using the training data, we estimate $f(x)$ as

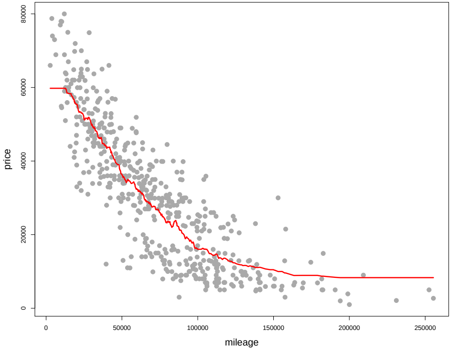
$$\hat{f}(x) = \hat{\alpha} + \hat{\beta}\, x$$

where $\hat{\alpha}$ and $\hat{\beta}$ are the linear regression estimates.

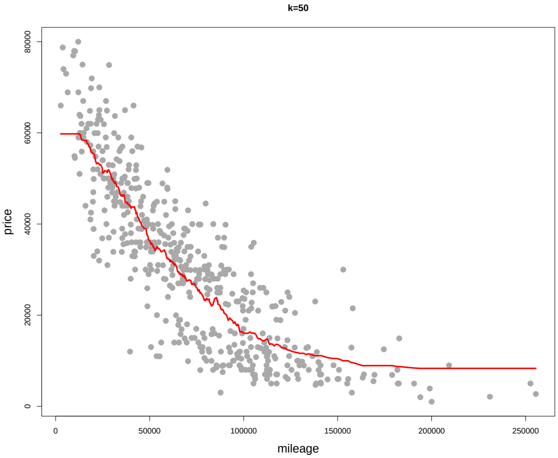


To get this estimate we used *kNN*
- k-nearest neighbors.

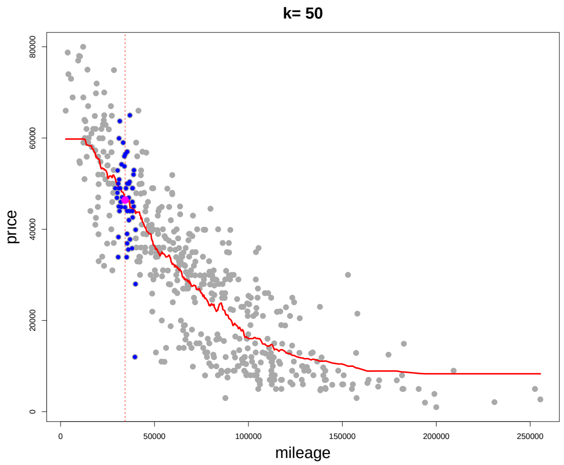
To estimate $f(x_f)$, average the y values for the k training observations with x *closest* to x_f.

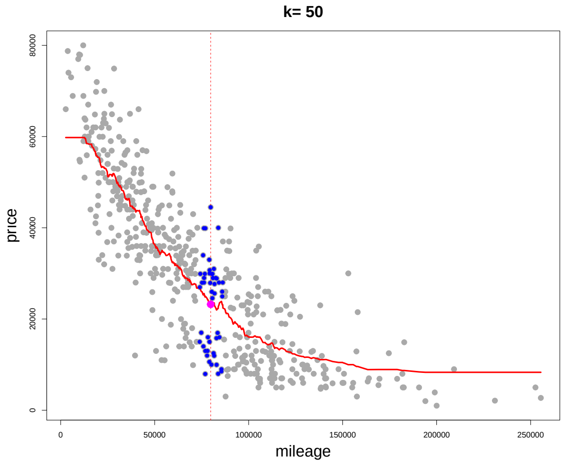


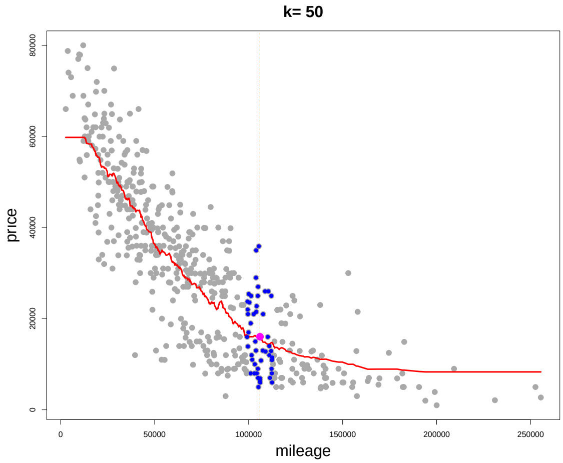
What do I mean by closest?
We will choose the k=50 points that are closest to the X value at which we are trying to predict.



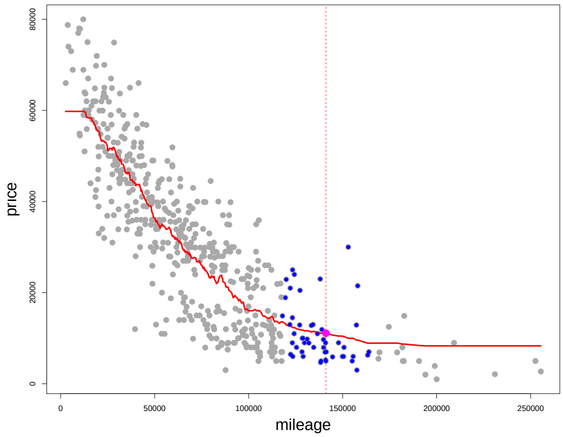
k=50
price
80000
60000
40000
20000
0
0
50000
100000
150000
200000
250000
mileage


k= 50
price
80000
60000
40000
20000
0
0
50000
100000
150000
200000
250000
mileage

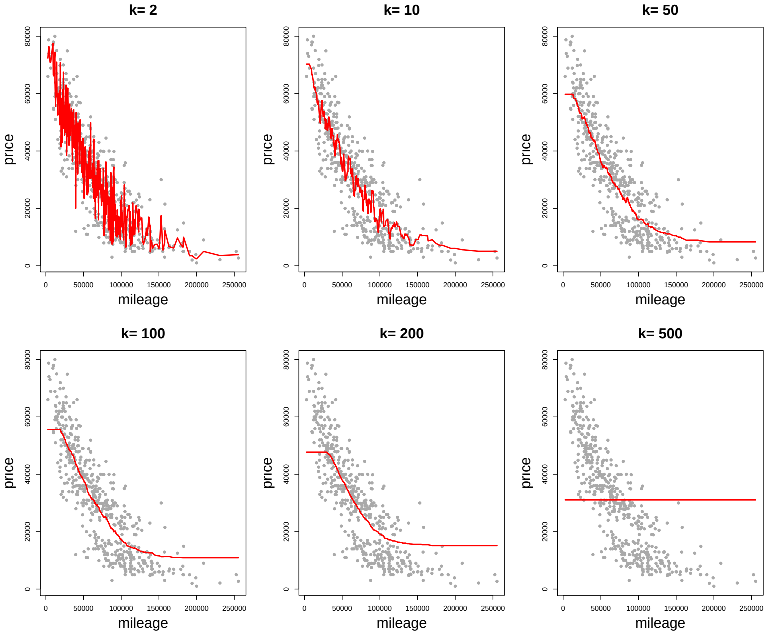

k= 50
price
mileage
0
20000
40000
60000
80000
0
50000
100000
150000
200000
250000


k= 50
price
mileage
0
20000
40000
60000
80000
0
50000
100000
150000
200000
250000

k= 50


price
80000
60000
40000
20000
0
0
50000
100000
150000
200000
250000
mileage

Seems sensible, what about other choices of k?



for k-NN:

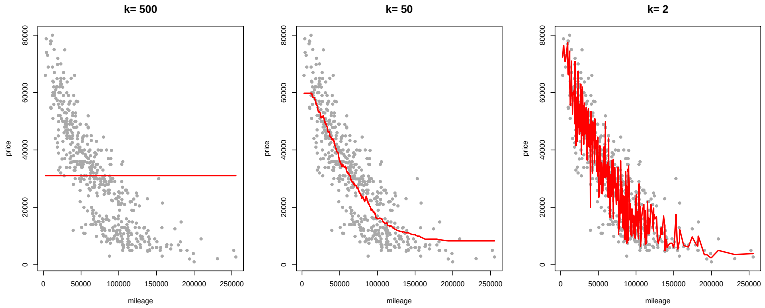
A big k gives us a simple looking function.

A small k can give us a more complex, flexible looking function.

Complexity, Generalization and Interpretation

- As we have seen in the examples above, there are lots of options in estimating $f(X)$.
- Some methods are very flexible some are not... *why would we ever choose a less flexible model?*
  1. Simple, more restrictive methods are usually easier to interpret
  2. More importantly, it is often the case that simpler models are more accurate in making future predictions.

Not too simple, but not too complex!



Measuring Accuracy

Measuring Accuracy

How accurate are each of these models?

We can measure the fit of our model (our fuction estimate) using the mean squared error (MSE)

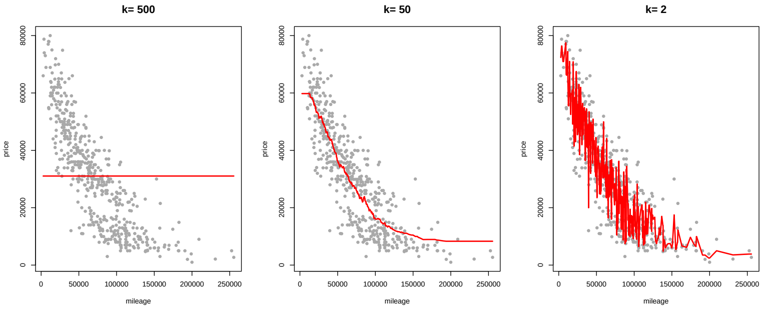
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \hat{f}(X_i)\right]^2$$

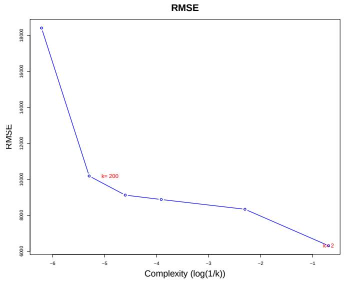
This measures, on average, how large the **mistakes** (errors) made by the model are...

You will also see root mean squared error (RMSE) used

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \hat{f}(X_i)\right]^2}$$

Example: Used Cars (again)





So, I guess we should just go with the most complex model, i.e., $k = 2$, right?

Out-of-Sample Predictions

But, do we really care about explaining what we have already seen?

Key Idea: what really matters is our prediction accuracy out-of-sample!!!

Suppose we have m additional observations (X_i^o, Y_i^o), for $i = 1, \ldots, m$, that we did not use to fit the model.

Let's call this dataset the validation set (also known as **hold-out set**).

Let's look at the out-of-sample MSE:

$$MSE^o = \frac{1}{m} \sum_{i=1}^{m} \left[Y_i^o - \hat{f}(X_i^o) \right]^2$$

(1)

Use the in-sample (training data) $(X_i, Y_i), i = 1, 2, \ldots, n$ to estimate f.

Now we have $\hat{f}$.

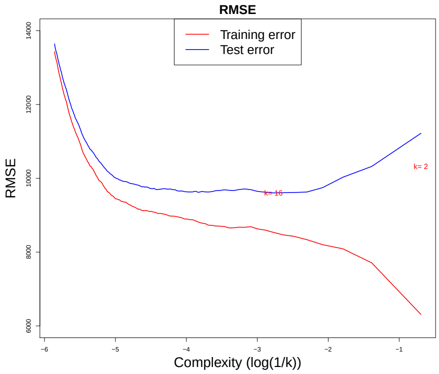
(2)

Evaluate predictive performance on the test data (X_i^o, Y_i^o), for $i = 1, \ldots, m$,

$$MSE^o = \frac{1}{m} \sum_{i=1}^{m} \left[Y_i^o - \hat{f}(X_i^o) \right]^2$$

Out-of-Sample Predictions

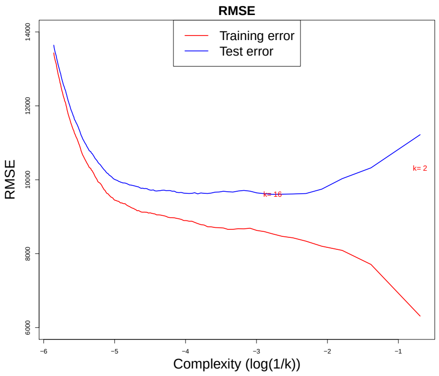
In our Used Cars example, I have an additional 500 observations that I will use as a validation set. That is, I will fit a k-NN models on the original 500 observations and then predict on the other 500 observations.



Now, the model where $k = 35$ looks like the most accurate choice!!

Not too simple but not too complex!!!

Overfitting



The training error decreases as k gets smaller.

The testing error, measured on independent data, decreases at first, then starts increasing.

Overfitting

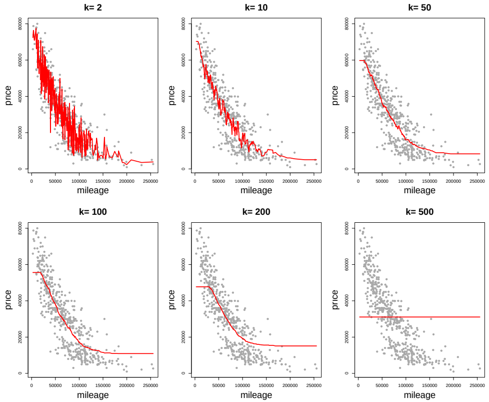
A general, hugely important problem for all machine learning algorithms.

We can find a function $\hat{f}$ that predicts perfectly the training data but does not generalize well to new data.

We are seeing an instance here: if we have a lot of parameters, the function $\hat{f}$ “memorizes” the data points, but is wild everywhere else.

What we really care is how well we can predict on new examples.

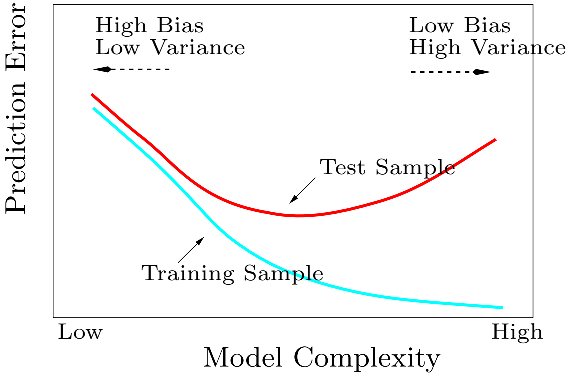
Overfitting and underfitting



The smaller the *k* is, the more flexible model is (has more degrees of freedom). These estimators have high variance.

Typical overfitting means that error on the training data is very low, but error on new instances is high.

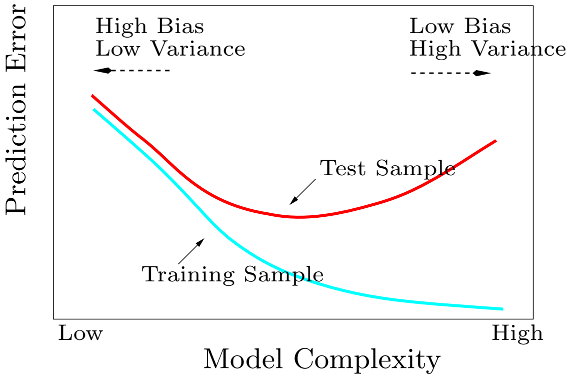
The Key Idea in Machine Learning!!



Complex enough to find the signal, but not so complex that you chase the noise in the training data, only the signal will help you predict new Y given new X.

Bias-Variance Trade-Off

Training- versus Validation-Set Performance



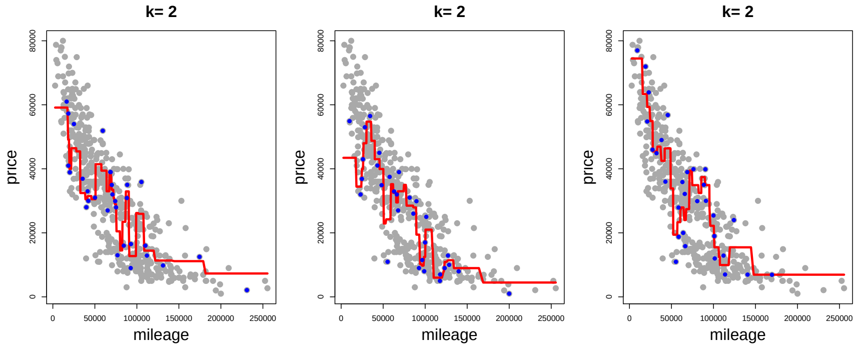
Bias-Variance Trade-Off

Why do complex models behave poorly in making predictions?

Let's start with an example...

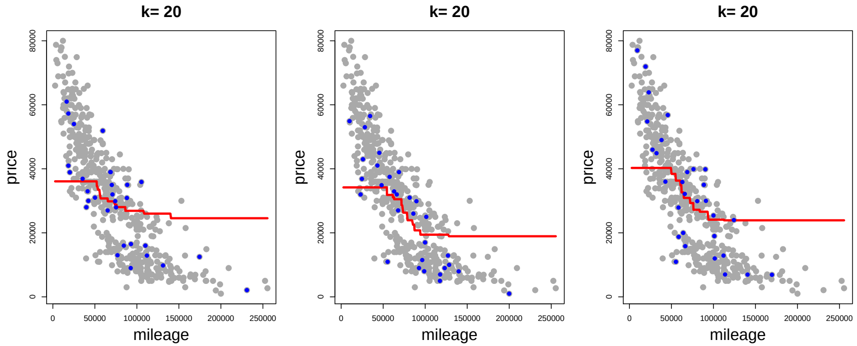
- In the Used Cars example, I will randomly choose 30 observations to be in the training set 3 different times...
- for each training set I will estimate $f(\cdot)$ using the k-nearest neighbors idea... first with $k = 2$ and them with $k = 20$

$k = 2$ High variability...



(blue points are the training data used)

$k = 20$ Low variability ... but BIAS!!



(blue points are the training data used)

What did we see here?

- When $k = 2$, it seems that the estimate of $f(\cdot)$ varies a lot between training sets...
- When $k = 20$ the estimates look a lot more stable...

Now, imagine that you are trying to predict *price* when $mileage = 50,000\ldots$

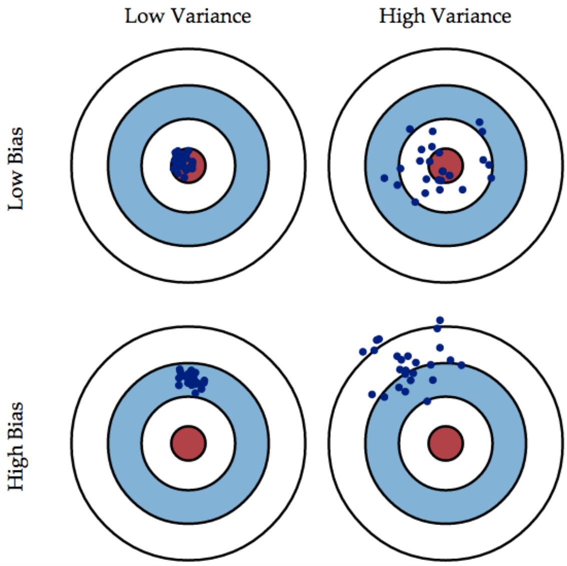
compare the changes in the predictions made by the different training sets under $k = 2$ and $k = 20\ldots$

What do you see?

Bias-Variance Trade-Off

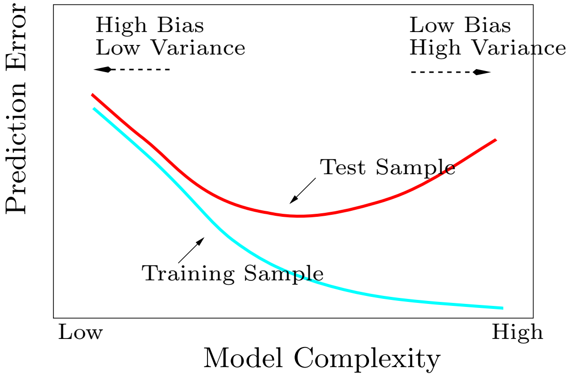
- see bias-variance-illustration.R
- This is an illustration of what is called the *bias-variance trade-off*.
- In general, simple models are trying to explain a complex, real problem with not a lot of flexibility so it introduces bias... on the other hand, by being simple the estimates tend to have low variance
- On the other hand, complex models are able to quickly adapt to the real situation and hence lead to small bias... however, if *too adaptable*, it tends to vary a lot, i.e., high variance.

Bias-Variance Trade-Off



Bias-Variance Trade-Off

Once again, the key idea that we need to understand!!



Bias-Variance Trade-Off

Let's get back to our original representation of the problem... it helps us understand what is going on...

$$Y = f(X) + \epsilon$$

- We need flexible enough models to find $f(\cdot)$ without imposing bias...
- ... but, too flexible models will "chase" non-existing patterns in ϵ leading to unwanted variability

Data Splitting and Cross Validation

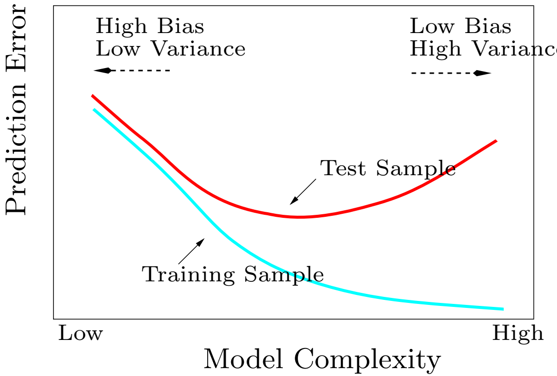
Training Error versus Test error

The **test error** is the average error that results from using a machine learning model to predict the response on a new observation, one that was not used in training the model.

The **training error** can be easily calculated by applying the statistical learning method to the observations used in its training.

The training error rate often is quite different from the test error rate, and in particular the former can *dramatically underestimate* the latter. **Why?**

Training- versus Validation-Set Performance



Methods to Assess Predictive Performance

Best solution: a large designated test set. Often not available.

Can we come up with a way to estimate how the models would have performed on new data using only currently available data?

- We consider a class of methods that estimate the test error by holding out a subset of the training observations from the fitting process, and then applying the learned model to those held out observations.

Validation-set approach

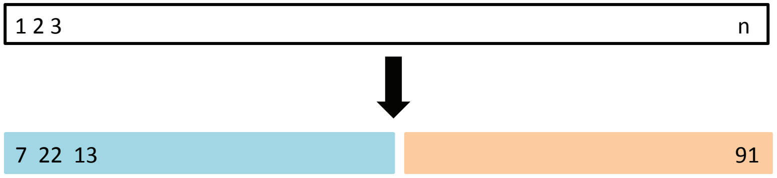
Randomly divide the available set of samples into two parts: a training set and a validation or hold-out set.

The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.

The resulting validation-set error provides an estimate of the test error.

- Typically assessed using RMSE/MSE in the case of regression.
- This metric is commonly referred to as **out-of-sample** RMSE/MSE.

Validation-set approach



A random splitting into two parts: left part is training set, right part is validation set

The model is fitted on the training set, and the fitted model is used to predict the responses for the observations in the validation set.

Size of Validation Set

How should one split the data into a training set and a validation set?

- More data in the validation set means less data for constructing models.
- Less data in the validation set may not give good estimate of predictive performance.

Generally, no more than 30% of the complete data needs to be held out as a validation set.

- Not an issue if the data set is large.
- If size of complete data is small, may want to drop to closer to 10%.

Data splitting (refinement)

If we have a lot of data, we can split it into three parts:

- train set (50%) — used for fitting models
- validation set (25%) — used for model selection
- test set (25%) — assessment of the selected model

Data splitting (refinement)

If we have a lot of data, we can split it into three parts:

- train set (50%) — used for fitting models
- validation set (25%) — used for model selection
- test set (25%) — assessment of the selected model

Be careful!!!

Test set only unbiased if you never never ever ever do any any any any learning on the test data.

For example, if you use the test set to select
the k in k-NN. . . . no longer unbiased!!!

Drawbacks of validation set approach

The validation estimate of the test error can be highly variable.

- depends on precisely which observations are included in the training set and which observations are included in the validation set.

In the validation approach, only a subset of the observations—those that are included in the training set rather than in the validation set—are used to fit the model.

This suggests that the validation set error may tend to **overestimate** the test error for the model fit on the entire data set. *Why*?

Cross-validation

Using a validation-set to evaluate the performance of competing models has two potential drawbacks:

1. the results can be highly dependent on the choice of the validation set... what samples? how many?
2. by leaving aside a subset of data for validation we end up estimating the models with less information. It is harder to learn with fewer samples and this might lead to an overestimation of errors.

Cross-Validation is a refinement of the validation strategy that help address both of these issues.

K-Fold Cross-Validation

An improvement from validation-set approach to reduce variability.

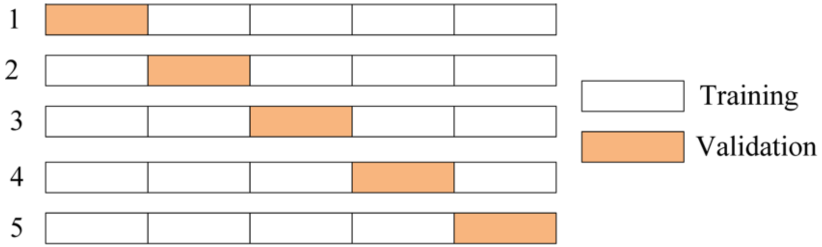
- Can be used to compare and select best model.
- Can be used to give an idea of the test error of the final chosen model.

Algorithm

1. Randomly divide the data into K equal-sized parts.
2. Leave out part k, fit the model to the other $K-1$ parts (combined), and then obtain predictions for the left-out kth part.
3. Repeat for $k = 1, 2, \dots, K$ and then results are combined.

K-Fold Cross-Validation

Divide data into K roughly equal-sized parts ($K = 5$ here)



K-Fold Cross-Validation in Detail

Let the K parts be $C_1, C_2, \ldots, C_K$, where C_k contains the indices of the observations in part k.
There are n_k observations in part k: roughly, $n_k = n/K$.

Compute the cross-validation score

$$\mathrm{CV} = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{MSE}_k \left(\approx \frac{1}{K} \sum_{k=1}^{K} \mathrm{MSE}_k \right)$$

where

$$\mathrm{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i^{(k)})^2$$

- $\hat{y}_i^{(k)}$ is the fit for observation i, obtained from the data with part k removed.

Extreme K-Fold: LOOCV

The extreme case of K-Fold CV is when we set the number of folds K equal to the number of data points n.

This is known as *leave-one-out cross-validation* (**LOOCV**).

- Build a model on $n-1$ data points and predict on the nth.
- Very expensive – need to build n models.

LOOCV is sometimes useful, but typically doesn't *shake up* the data enough. The estimates from each fold are highly correlated and hence their average can have high variance.

How Many Folds?

A common question is “how many folds should be used for cross-validation?”

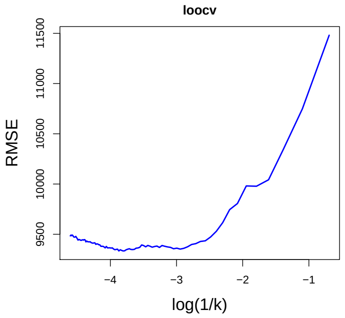
Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward.

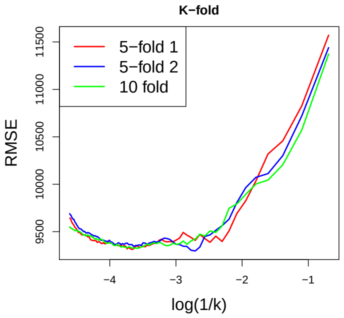
This bias is minimized when $K = n$ (LOOCV), but this estimate has high variance, as noted earlier.

$K = 5$ or 10 provides a good compromise for this bias-variance trade-off

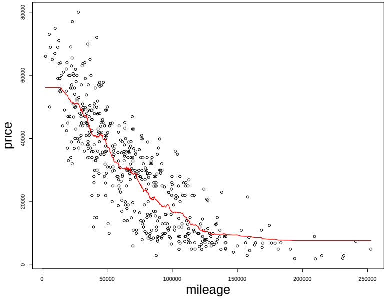
- industry standard

Example: Used Cars





Min is at about
$k = 35$.



Assumptions of Split Sample Approaches

Are we making any assumptions when using validation-set approach or cross-validation?

Independence—we are randomly sampling the data when building partitions and assuming no structure in the rows.

- This approach doesn't work for time series.
- Hold out last piece of data (most recent dates).

Recap: Out-of-Sample Validation

Validation-set approach

K-**fold cross validation**

Recap: Out-of-Sample Validation

Validation-set approach **K-fold cross validation**

- Give an estimate of the true test error. Tend to overestimate the error.
- Can be used to choose model complexity and to compare different models.

Recap: Out-of-Sample Validation

Validation-set approach	**K-fold cross validation**

- Give an estimate of the true test error. Tend to overestimate the error.
- Can be used to choose model complexity and to compare different models.

Validation-set approach	K-fold cross validation
Highly variable for different splits	Less variable for different splits
Sensitive to split proportion	Less sensitive to choice of K
Computationally less expensive	Computationally more expensive

More on k-NN

More on k-Nearest Neighbors, $p > 1$

We have looked at simple examples of kNN (with one x!!).

In this section we look at kNN more carefully, in particular, how do you use kNN when x has p variables??!!

An important advantage of kNN is that it is feasible for *BIG DATA*, big n **and** big p.

The k-nearest neighbors algorithm will try to **predict** based on similar (close) records on the **training dataset**.

Remember, the problem is to guess a future value Y_f given new values of the covariates $X_f = (x_{1f}, x_{2f}, x_{3f}, \ldots, x_{pf})$.

kNN:

What do the Y's look like in the region around X_f?

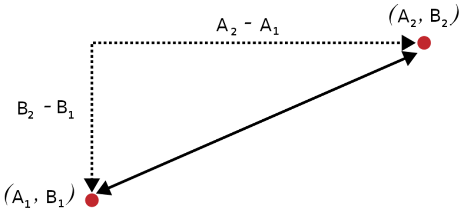
We need to find the k observations in the training dataset that are close to X_f. How? "Nearness" to the i^{th} neighbor can be defined by (Euclidean distance):

$$d_i = \sqrt{\sum_{j=1}^{p} (x_{jf} - x_{ji})^2}, \quad x_i \text{ in training data}$$

Prediction:

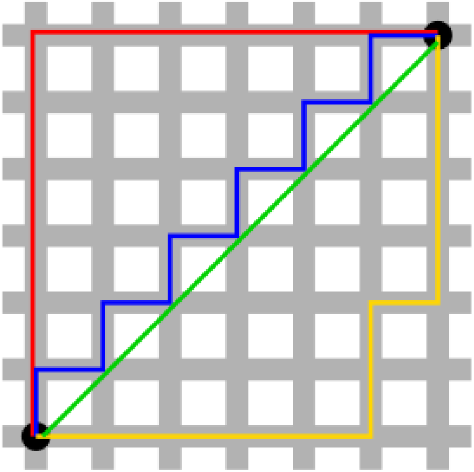
Take the average of the $Y's$ in the k-nearest neighborhood.
Average y_i corresponding to k smallest d_i.

Euclidean distance



$$\sqrt{(A_1 - A_2)^2 + (B_1 - B_2)^2}$$

Another distance – Manhattan



In general:

$$|A_1 - A_2| + |B_1 - B_2| + \ldots + |Z_1 - Z_2|$$

Note:

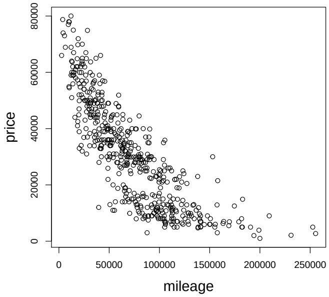
- The distance metric used above is only valid for numerical values of X. When $X's$ are categorical we need to think about a different distance metric or perform some manipulation of the information.
- The scale of X also will have an impact. In general it is a good idea put the $X's$ in the same scale before running kNN.

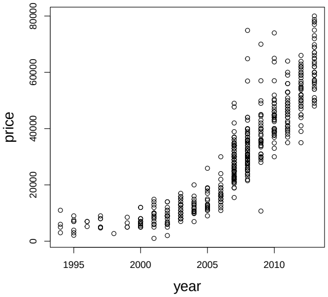
What do we mean by scale?

If weight is in pounds you get one distance, if weight is in kilograms you get a different number!!

To see how this works, let us also look at `year`:

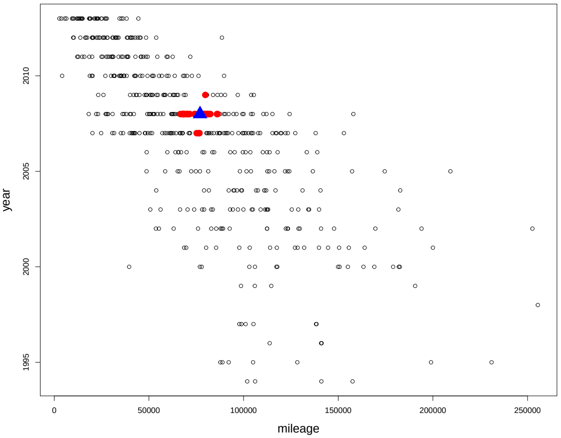
x_1=mileage
x_2=year



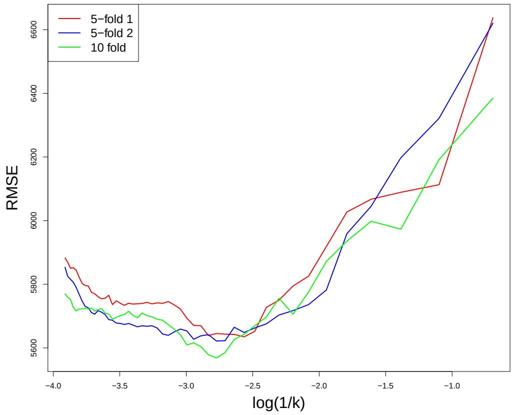


Hmm. how is y=price related to x_2=year ?

To predict y at the blue triangle, we average the y values corresponding to the red points.

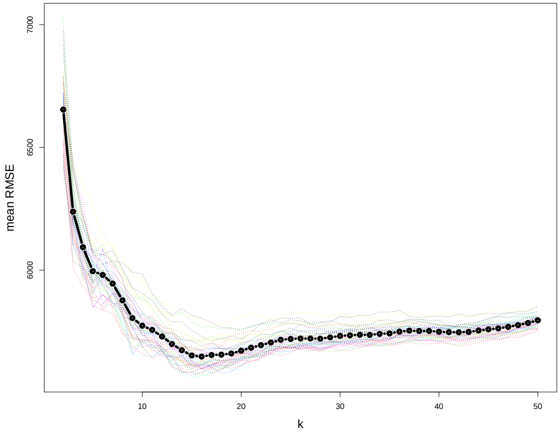


Let's do the CV and see what works out of sample.



Seems to indicate a much smaller k (than when we just used `mileage`), but it is very noisy.

Let's average many CV's.

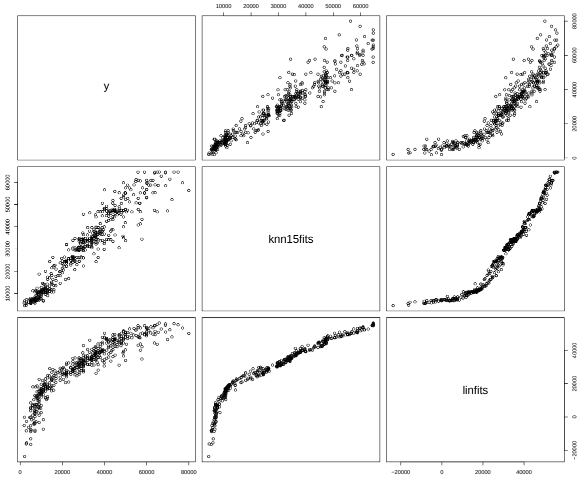


Indicates a *k* of about 15, which is much smaller than when we just had `mileage`. Minimum MSE is smaller than when we just had `mileage`.

What do we do next, after having used cross-validation to choose a model (or tuning parameter)?

It may be an obvious point, but worth being clear: we now fit our estimator to the entire training set (x_i, y_i), $i = 1, \ldots, n$, using the selected model.

Refit using all the data and $k = 15$.



Matching

A lot of data-mining methods work by matching.

- Which ones are most like you ??
- Which training x are most like x_f ??
- Shoppers **like you** have bought . . .

"Like" means a choice of distance, and getting the distance right in high dimensions can be very hard.

Summary of kNN

kNN is a powerfull, widely used, intutitive technique.

In principle, we can use kNN for large n and p.

We have used it to illustrate the Bias-Variance tradeoff which is **a fundamental concept**.

Note:

- Choosing the distance can be tricky.
- Using distance in high dimensions can be tricky.
- Rescaling all the (numeric) x's is common.

Doing CV with a Bigger n

The Used Cars data we have been using as an example only has $n = 1000$ observations.

While the big ideas are the same, some things will work out differently with larger n.

Let's do $n = 20,000$ to illustrate.

The key differences will be:

- We won't have to rerun the CV many times and average.
  With the larger sample size, you will get much less variation in the CV results.

- We will start by leaving out a **test** data set.

We will use CV on the remaining data to make modeling decisions, and then apply our data to the test data to see you well we predict out of sample.

When we refit using all the Used Cars data, we did not have any true out-of-sample data !!

kNN: California Housing

Data: Median home values in census tract plus the following information

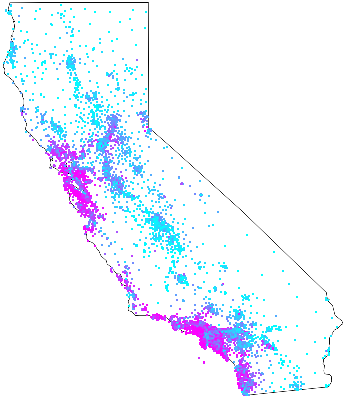
- Location (latitude, longitude)
- Demographic information: population, income, etc...
- Average room/bedroom number, home age
- Let's start using just location as our $X's$... euclidean distance is quite natural here, right?

Goal: Predict $log(MedianValue)$

There are 20,640 observations and 8 x's.

We should spend a long time plotting the data, but let's suppose we are in a hurry.

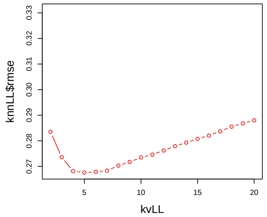
y=logMedVal
vs longitude and latitude.

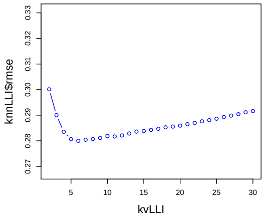


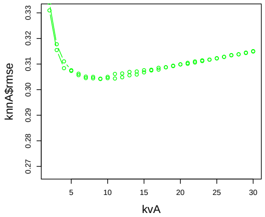
10,000 in train. Rest in test. Standardize each x: (x-m)/s.

Do 5-fold cross-validation on train.

Red: longitude and latitude
Blue: longitude and latitude and Income
Green: all 8 x's (2 runs).







Pretty small k values.
All x is worst.

And, here are the rmse's on the test data.

```
rmse test, long,lat: 0.2533981
rmse test, long,lat,income: 0.2628331
rmse test, all: 0.2897788
```

location, location, location. . . . . .