Models, Assessment, Cross-Validation

5 February 2018

Statistical models: Defintion

We observe data $Z_1, Z_2, \ldots, Z_n$ generated by some probability distribution P. We want to use the data to learn about P.

A **statistical model** is a set of possible distributions $\mathbb{P}$.

Some examples:

1. $\mathbb{P}_1 = \{0 < p < 1 : P(z = 1) = p,\ P(z = 0) = 1 - p\}$.
2. $\mathbb{P}_2 = \{\mu \in \mathbb{R}, \sigma > 0 : Y \sim N(\mu, \sigma^2)\}$
3. $\mathbb{P}_3 = \{\underline{\beta} \in \mathbb{R}^p, \sigma > 0 : Y \sim N(\underline{X}^\top \beta, \sigma^2),\ \underline{X} \text{ fixed}\}$.
4. $\mathbb{P}_4 = \{g \in C_\infty, \underline{\theta} \in \mathbb{R}^p, , \sigma > 0 : Y \sim N(g(\underline{X}; \underline{\theta}, \sigma^2)\}$
5. $\mathbb{P}_5 = \{Y \sim F, \text{where F is a CDF}\}$
6. And so on...

What does it mean to specify a statistical model?

When we observe data, we assume that the data come from some specific distribution P within a set of possible distributions $\mathbb{P}$.

Statistical Models: Bernoulli

$\mathbb{P}_1$ is about as simple as a model as we can have. This is simply the Bernoulli distribution.

$Z_1, \ldots, Z_n$ is just a series of 0's and 1's observed from a distribution $P \in \mathbb{P}$

$$\mathbb{P} = \{0 < p < 1 : P(z = 1) = p,\ P(z = 0) = 1 - p\}$$

- To completely characterize P, we just need to estimate p.
- Need to assume that $P \in \mathbb{P}$.
- This assumption is relative simple: **need independence, and only two outcomes are possible.**
- Suppose we are flipping a "fair" coin. Does reality actually follow this model?
- See Dynamical Bias In The Coin Toss

Statistical Models: The Normal Linear Model

We observe data $Z_i = (Y_i, X_i)$ generated by some probability distribution P. We want to use the data to learn about P.

$$\mathbb{P} = \{\underline{\beta} \in \mathbb{R}^p, \sigma > 0 : Y \sim N\underline{X}^\top \underline{\beta}, \sigma^2),\ \underline{X} \text{ fixed}\}.$$

- To completely characterize P, we "just" need values β and σ.
- Need to assume that $P \in \mathbb{P}$. -This time, I have to assume a lot more: **Linearity, independence, Gaussian noise, we have the correct predictors, no ignored variables, no collinearity, etc.**

Estimating Parameters: Convergence

If there are parameters to estimate in a model, we have to know that we can even estimate the parameters.

Why do we assume that we take sample data and estimate the parameters? Why can we do inference?

Let $X_1, X_2, \ldots$ be a sequence of random variables, and let X be another random variable with distribution P. Let F_n be the cdf of X_n and let F be the cdf of X.

1. X_n converges **in probability** to X, $X_n \xrightarrow{P} X$, if for every $\epsilon > 0$,

$$\lim_{n\to\infty} P\left(|X_n - X| > \epsilon\right) = 0.$$

2. X_n converges **in distribution** to X, $X_n \xrightarrow{D} X$, if for all t,

$$\lim_{n\to\infty} F_n(t) = F(t)$$

Convergence of the Sample Mean: An Ideal

Suppose $X_1, X_2, \ldots$ are independent random variables, each with mean μ and variance σ^2. Let $\overline{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$.

1. **Weak law of large numbers**

$$\overline{X}_n \xrightarrow{P} \mu$$

The law of large numbers tell us that the probability mass of an average of random variables "piles up" near its expectation.

2. **Central limit theorem**

$$\frac{\sqrt{n}(\overline{X}_n - \mu)}{\sigma} \xrightarrow{D} N(0,1).$$

The CLT tells us about the shape of the "piling", when appropriately normalized.

Evaluation

Once I choose some way to "learn" a statistical model, I need to decide if I'm doing a good job.

How do I decide if I'm doing anything good?

Lots of ways to evaluate estimators, $\widehat{\mu}$ of parameters μ.

(from last time)

- Consistency: $\widehat{\mu} \xrightarrow{P} \mu$.
- Asymptotic Normality: $\widehat{\mu} \xrightarrow{D} N(\mu, \Sigma)$
- Efficiency: how large is Σ
- Unbiased: $\mathbb{E}[\widehat{\mu}] \stackrel{?}{=} \mu$
- etc.

None of these things make sense unless **your model is correct**. But...

Your model is wrong!

Mis-specified models

What happens when your model is wrong?

None of those evaluation criteria really hold. The parameters no longer have a direct connection to reality.

All we can really hope for is that our approximation is close to an approximation of reality.

It is approximation all the way down...

Risk: Estimation and Prediction

Prediction is easier: your model may not actually represent the true state of nature, but it may still predict well.

> Over a 13-year period, [David Leinweber] found [that] annual **butter production** in Bangladesh "explained" 75% of the variation in the annual returns of the Standard & Poor's 500-stock index.
>
> By tossing in **U.S. cheese production** and the **total population of sheep** in both Bangladesh and the U.S., Leinweber was able to "predict" past U.S. stock returns with 99% accuracy.

This is why we don't use R^2 to measure prediction accuracy.

The issue here is that the models created estimated their accuracy within the dataset used to create the models.

The General Idea of "Modeling"

Let's start with the general form of the approximation that is usually used.

- We have a quantitative response and p different predictors: Y and $\underline{X}^\top = (X_1, X_2, \ldots, X_p)$
- $Y = f(\underline{X}) + \epsilon$
- As we have covered before, there is a deterministic and a random portion to this model.
- Our approximation of $f(\underline{X})$ is $\widehat{f}(\underline{X})$

What is the usual procedure for getting $\widehat{f}(\underline{X})$?

We get our data: $(\underline{x}_1, y_1), (\underline{x}_2, y_2), ..., (\underline{x}_n, y_n)$. We then choose $\widehat{f}(\underline{X})$ so that it minimizes Mean Squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{f}(\underline{x}_i)\right)^2$$

Optimization of "Risk"

In statistical theory, we are trying to estimate the **Risk** of our predictor/estimator $\widehat{f}(\underline{X})$

We want to predict a new value of Y using our function $\widehat{f}(\underline{X})$:

$$\widehat{Y} = \widehat{f}(\underline{X})$$

And for new observations of Y we want our predicted value based on $\widehat{f}(\underline{X})$ to be as close as possible.

Least squares is based off of the minimization of MSE, but lets get more general.

Evaluating predictions

Of course, both Y and $\widehat{Y}$ are **random**

I want to know how well I can predict **on average**

Let $\widehat{f}$ be some way of making predictions $\widehat{Y}$ of Y using covariates X

In fact, suppose I observe a dataset $\mathcal{D}_n = \{(Y_1, X_1,), \ldots, (Y_n, X_n)\}$.

Then I want to **choose** some $\widehat{f}$ using $\mathcal{D}_n$.

Is $\widehat{f}$ good on average?

Evaluating predictions

Choose some **loss function** that measures prediction quality: $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. We predict Y with $\widehat{Y}$

Examples:

- **Squared-error:**

$$\ell(y, \widehat{y}) = (y - \widehat{y})^2$$

- **Absolute-error:**

$$\ell(y, \widehat{y}) = |y - \widehat{y}|$$

- **Zero-One:**

$$\ell(y, \widehat{y}) = I(y \neq \widehat{y}) = \begin{cases} 0 & y = \widehat{y} \\ 1 & \text{else} \end{cases}$$

Can be generalized to Y in arbitrary spaces.

Prediction risk

Prediction risk

$$R_n(\widehat{f}) = \mathbb{E}[\ell(Y, \widehat{f}(X))]$$

where the expectation is taken over the new data point (Y, X) and $\mathcal{D}_n$ (everything that is random).

For **regression** applications, we will use squared-error loss:

$$R_n(\widehat{f}) = \mathbb{E}[(Y - \widehat{f}(X))^2]$$

For **classification** applications, we will use zero-one loss:

$$R_n(\widehat{f}) = \mathbb{E}[I(Y \neq \widehat{f}(X))]$$

Example 1: Estimating the mean

Suppose we know that we want to predict a quantity Y, where $\mathbb{E}[Y] = \mu \in \mathbb{R}$ and $\mathbb{V}[Y] = 1$. That is, $Y \sim P \in \mathbb{P}$, where

$$\mathbb{P} = \{P : \mathbb{E}[Y] = \mu \text{ and } \mathbb{V}[Y] = 1\}.$$

Our data is $\mathcal{D}_n = \{Y_1, \ldots, Y_n\}$ such that $Y_i \overset{i.i.d.}{\sim} P$, and we want to estimate μ (and hence P).

Estimating the mean

- Let $\widehat{Y} = \overline{Y}_n$ be the sample mean.
- We can ask about the **estimation risk** (since we're estimating μ):

$$\begin{aligned} R_n(\overline{Y}_n; \mu) &= \mathbb{E}[(\overline{Y}_n - \mu)^2] \\ &= \mathbb{E}[\overline{Y}_n^2] - 2\mu\mathbb{E}[\overline{Y}_n] + \mu^2 \\ &= \mu^2 + \frac{1}{n} - 2\mu^2 + \mu^2 \\ &= \frac{1}{n} \end{aligned}$$

Predicting new Y's

- Let $\widehat{Y} = \overline{Y}_n$ be the sample mean.
- What is the **prediction risk** of $\overline{Y}$?

$$\begin{aligned} R_n(\overline{Y}_n) &= \mathbb{E}[(\overline{Y}_n - Y)^2] \\ &= \mathbb{E}[\overline{Y}_n^2] - 2\mathbb{E}[\overline{Y}_n Y] + \mathbb{E}[Y^2] \\ &= \mu^2 + \frac{1}{n} - 2\mu^2 + \mu^2 + 1 \\ &= 1 + \frac{1}{n} \end{aligned}$$

Predicting new Y's

- What is the prediction risk of guessing $Y = 0$?
- You can probably guess that this is a stupid idea.
- Let's show why it's stupid.

$$\begin{aligned} R_n(0) &= \mathbb{E}[(0 - Y)^2] \\ &= 1 + \mu^2 \end{aligned}$$

Predicting new Y's

What is the prediction risk of guessing $Y = \mu$?

This is a great idea, but we don't know μ.

Let's see what happens anyway.

$$\begin{aligned} R_n(\mu) &= \mathbb{E}[(Y - \mu)^2] \\ &= 1 \end{aligned}$$

Estimating the mean

- Prediction risk: $R(\overline{Y}_n) = 1 + \frac{1}{n}$
- Estimation risk: $R(\overline{Y}_n; \mu) = \frac{1}{n}$
- There is actually a nice interpretation here:
  1. The common $1/n$ term is $\mathbb{V}\left[\overline{Y}_n\right]$
  2. The extra factor of 1 in the prediction risk is **irreducible error**
     - Y is a random variable, and hence noisy.
     - We can never eliminate it's intrinsic variance.
     - In other words, even if we knew μ, we could never get closer than 1, on average.
- Intuitively, $\overline{Y}_n$ is the obvious thing to do.

Predicting new Y's

- Let's try one more: $\widehat{Y}_a = a\overline{Y}_n$ for some $a \in (0, 1]$.

$$R_n(\widehat{Y}_a) = \mathbb{E}[(\widehat{Y}_a - Y)^2] = (1 - a)^2\mu^2 + \frac{a^2}{n} + 1$$

- We can minimize this in a to get the best possible prediction risk for an estimator of the form $\widehat{Y}_a$:

$$\arg\min_a R_n(\widehat{Y}_a) = \left(\frac{\mu^2}{\mu^2 + 1/n}\right)$$

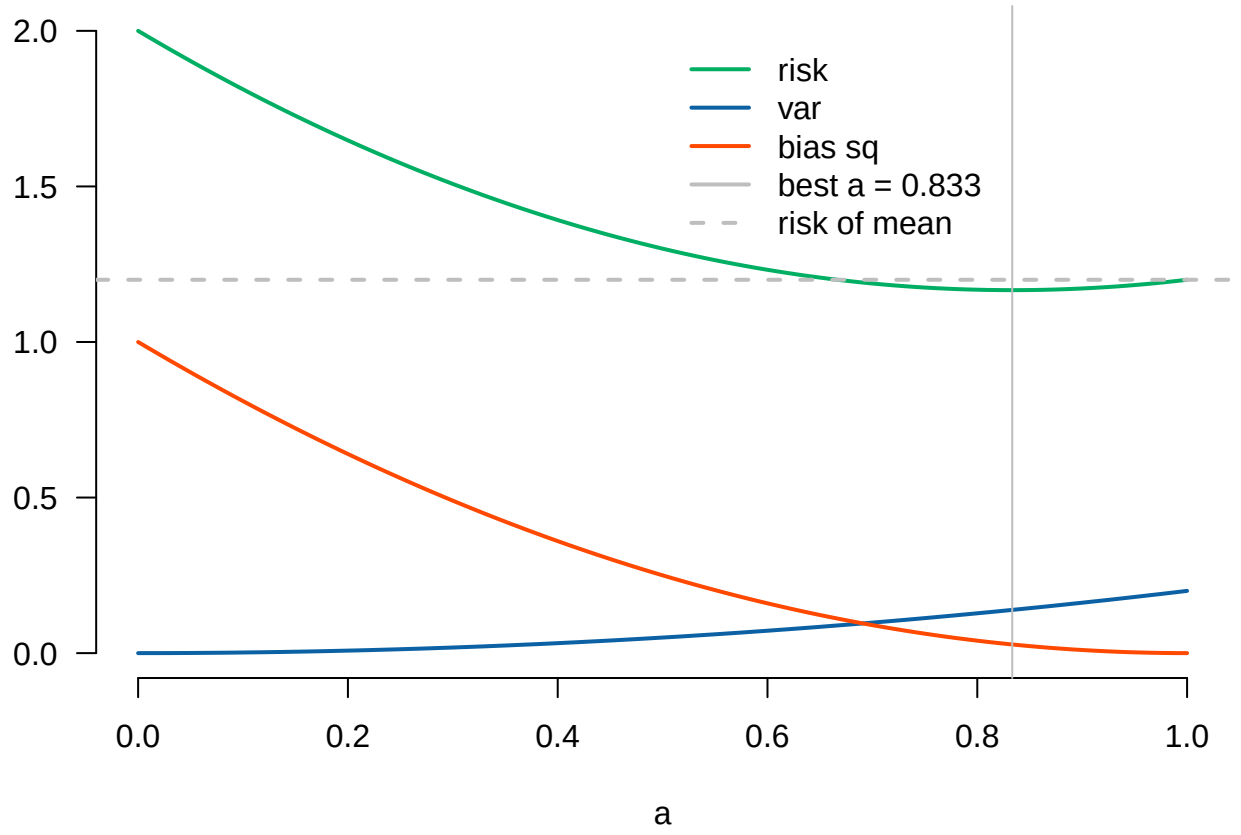
What happens if $|\mu| \ll 1$?

Wait a minute! You're saying there is a **better** estimator than $\overline{Y}_n$?

Bias-variance tradeoff: Estimating the mean

$$R(a) = R_n(\widehat{Y}_a) = (a - 1)^2\mu^2 + \frac{a^2}{n} + \sigma^2$$

```
mu=1; n=5; sig2=1
```



What?

Just to restate:

- If $\mu = 1$ and $n = 5$ then it is better to predict with 0.83 $\overline{Y}_n$ than with $\overline{Y}_n$ itself.
- In this case

1. $R(a) = R_1(a\overline{Y}_n) = 1.17$
2. $R(\overline{Y}_n) = 1.2$

Prediction risk

$$R_n(f) = \mathbb{E}[\ell(Y, f(X))]$$

Why care about $R_n(f)$?

- (+) Measures predictive accuracy on average.
- (+) How much confidence should you have in f's predictions.
- (+) Compare with other models.
- (-) **This is hard:**
  - Don't know P (if I knew the truth, this would be easy)

Risk for general models

We just saw that when you know the true model, and you have a nice estimator, the prediction risk has a nice decomposition

(this generalizes to much more complicated situations)

- Suppose we have a class of prediction functions $\mathcal{F}$,

$$\text{e.g. } \mathcal{F} = \left\{\beta : f(x) = x^\top \beta\right\}$$

- We use the data to choose some $\widehat{f} \in \mathcal{F}$ and set $\widehat{Y} = \widehat{f}(X)$
- The **true** model is g (not necessarily in $\mathcal{F}$). Then:

$$R_n(\widehat{f}) = \int \left[\text{bias}^2(\widehat{f}(x)) + \text{var}(\widehat{f}(x))\right] p(x)dx + \sigma^2$$

where $X \sim p$ and

$$\begin{aligned}
\text{bias}(\widehat{f}(x)) &= \mathbb{E}[\widehat{f}(x)] - g(x) \\
\text{var}(\widehat{f}(x)) &= \mathbb{E}[(\widehat{f}(x) - \mathbb{E}\widehat{f}(x))^2] \\
\sigma^2 &= \mathbb{E}[(Y - g(X))^2]
\end{aligned}$$

Bias-variance decomposition

So,

1. prediction risk = bias^2 + variance + irreducible error
2. estimation risk = bias^2 + variance

What is $R(a)$ for our estimator $\widehat{Y}_a = a\overline{Y}_n$?

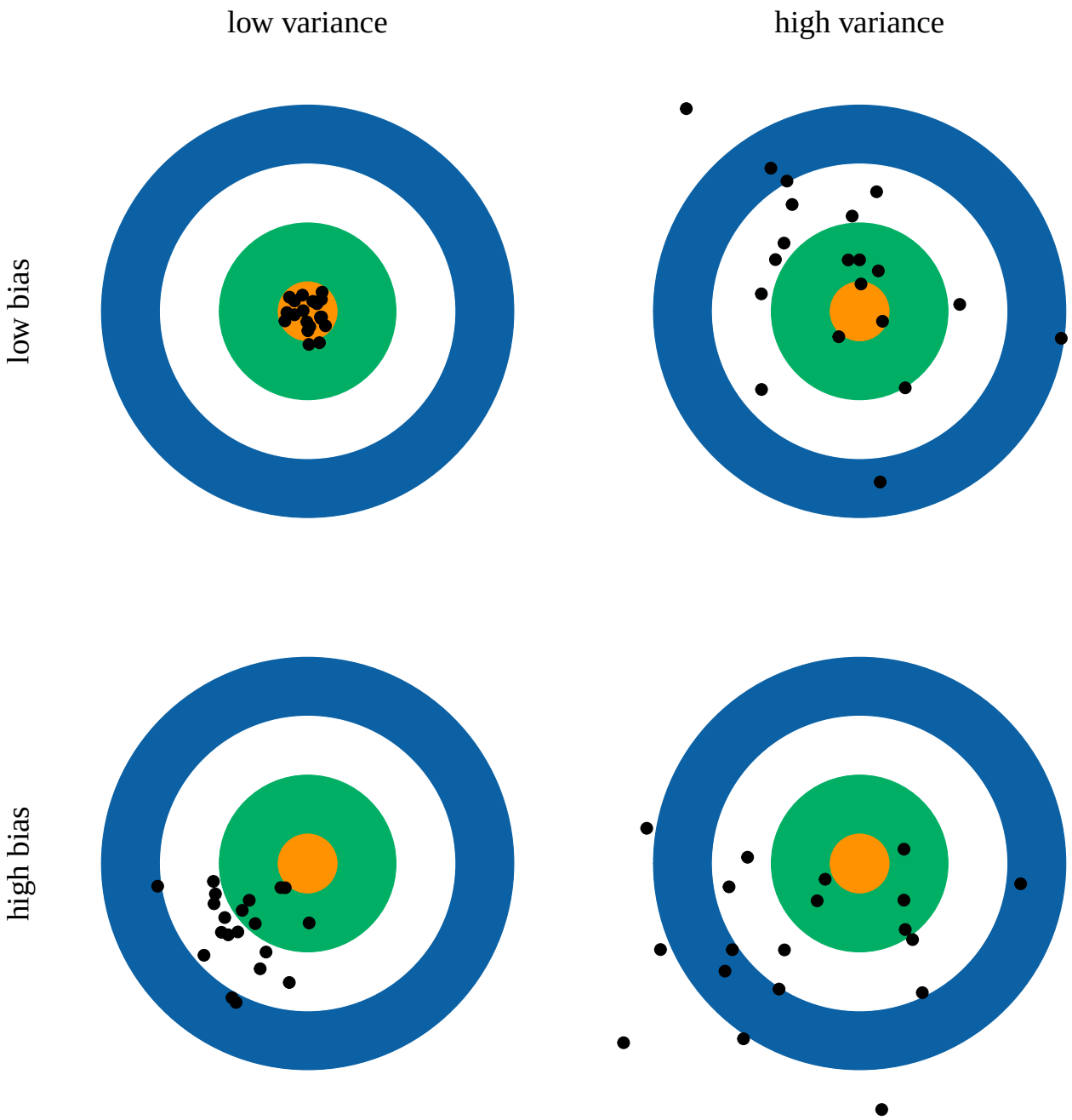
$$\begin{aligned}
\text{bias}(\widehat{Y}_a) &= \mathbb{E}[a\overline{Y}_n] - \mu = (a-1)\mu \\
\text{var}(\widehat{f}(x)) &= \mathbb{E}[(a\overline{Y}_n - \mathbb{E}[a\overline{Y}_n])^2] = a^2\mathbb{E}[(\overline{Y}_n - \mu)^2] = \frac{a^2}{n} \\
\sigma^2 &= \mathbb{E}[(Y-\mu)^2] = 1
\end{aligned}$$

$$\left(\text{That is: } R_n(\widehat{Y}_a) = (a-1)^2\mu^2 + \frac{a^2}{n} + 1\right)$$

Bias-variance decomposition

Important implication: prediction risk is proportional to estimation risk. However, defining estimation risk requires stronger assumptions.

In order to make good predictions, we want our prediction risk to be small. This means that we want to "balance" the bias and variance.



Bias-variance tradeoff: Overview

- **bias:** how well does $\widehat{f}$ approximate the truth g
- more complicated $\mathcal{F}$, lower bias. Flexibility $\Rightarrow$ Parsimony
- more flexibility $\Rightarrow$ larger variance
- complicated models are hard to estimate precisely for fixed n
- irreducible error

Model selection

In practicing regression, we are taught to do the following:

- Take a potentially large set of predictors.
- Whittle down these predictors to some smaller set
- We choose which predictors we want to use by putting some in the model and pulling some out
- We may try transformations of predictors and the response
- We keep messing around until the model fits best according to some criterion.

So we have transformations, multiple predictors, multiple transformations, multiple interactions, etc., that we use to select a model.

We are selecting a model M from a typically huge set of hypothetical models $\mathcal{M}$

$$M \subset \mathcal{M}$$

We throw around tons of models:

$$M_1, M_2, \ldots, M_k$$

Potentially each with different sets predictors, transformations, etc.

Sometimes this is done without much regard for reality. Crappy models will get selected and some criterion gets cited as for why the model is good.

Some Model Selection Criterion

In past applications, especially from academic settings, various methods are used for selected

- R^2 or its adjusted form: a "measure of variability accounted for by our predictors" and its not good.
- Other more complex ones:
  - $C_p = \frac{1}{n}(SSE + 2p\widehat{\sigma}^2)$
  - $AIC = \frac{1}{n\widehat{\sigma}^2}(SSE + 2p\widehat{\sigma}^2)$
  - $BIC = \frac{1}{n\widehat{\sigma}^2}(SSE + log(n)p\widehat{\sigma}^2)$

For soe reason, someone should use stepwise regression based off of these criterion (See ISLR, p)

Model Selection Example: $p = 3$

Consider the linear model situation with just three predictors, X_1, X_2, X_3 and we are trying to model (i.e., predict) some quantitative response Y.

Potential variations on model:

- Lets make this easy and make X_1 the intercept. Then $X_1 = 1$ and we're done.
- What about X_2? Lets say X_2 and X_3 are continuous:
  - We could try polynomial terms: \$X_2^2, $X_2^3, \ldots$
  - We could try transformations: $\log(X_2), \sqrt{X_2}, e^{X_2}, \ldots$
  - Interaction with X_3
- Polynomial terms and transformations apply to X_3
- We can also perform transformations on the respone Y

Each of these situations creates a different model, M_i and we check if this model is optimal. What are the consequences of that?

Inference after model selection.

Try running this code... It may represent a worst case scenario... But there's a lesson here!

```
n = 1000; p = 100
data <- matrix(rnorm(n*(p+1)), ncol = (p+1))
df <- data.frame(y = data[,1], data[,-1])
model <- lm(y ~ ., data=df)
summary(model)
best <- step(model, trace = F)
summary(best)
```

Training Data and Validation Data

Many issues with model selection stem from the calculation of the MSE, i.e., our estimate of prediction risk

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{f}(\underline{x}_i)\right)^2$$

This calcuation can be referred to as "optimistic".

- $\widehat{f}(\underline{x}_i)$ is chosen so that this is minimized.
- This minimization is done on the data we are given.
- Ideally, we should minimize based off new data, i.e., we want to minimize

  $$(y_0 - \widehat{f}(\underline{x}_0))^2$$

  where x_0 and y_0 are NEW data points.

Data Splitting

Data Splitting is way of trying to simulate having new data when we don't actually have any. - The basic idea is to split our data into two pieces: 1. Training Data: The data we use to estimate our model. 2. Test/Validation Data: The data we calculate the estimate risk on.

Recap of the theory.:

$$R_n(\widehat{f}) = \mathbb{E}[\ell(Y, \widehat{f}(X))]$$

where the expectation is taken over the new data point (Y, X) and $\mathcal{D}_n$ (everything that is random).

We saw one estimator of R_n:

$$\widehat{R}_n(\widehat{f}) = \sum_{i=1}^{n}\ell(Y_i, \widehat{f}(X_i)).$$

This is the training error. It is a **BAD** estimator because it is often optimistic.

Intuition for CV

- One reason that $\widehat{R}_n(\widehat{f})$ is bad is that we are using the same data to pick $\widehat{f}$ **AND** to estimate R_n.
- Notice that R_n is an expected value over a **NEW** observation (Y, X).
- We don't have new data.

Wait a minute...

...or do we?

- What if we set aside one observation, say the first one (Y_1, X_1).
- We estimate $\widehat{f}^{(1)}$ without using the first observation.
- Then we test our prediction:

$$\widetilde{R}_1(\widehat{f}^{(1)}) = \ell(Y_1, \widehat{f}^{(1)}(X_1)).$$

- But that was only one data point (Y_1, X_1). Why stop there?
- Do the same with (Y_2, X_2)! Get an estimate $\widehat{f}^{(2)}$ without using it, then

$$\widetilde{R}_2(\widehat{f}^{(2)}) = \ell(Y_2, \widehat{f}^{(2)}(X_2)).$$

Keep going

- We can keep doing this until we try it for every data point.
- And then average them! (Averages are good)
- In the end we get

$$\text{LOO-CV} = \sum_{i=1}^{n} \widetilde{R}_i(\widehat{f}^{(i)}) = \sum_{i=1}^{n} \ell(Y_i - \widehat{f}^{(i)}(X_i))$$

- This is leave-one-out cross validation

Problems with LOO-CV

1. Each held out set is small ($n = 1$). Therefore, the variance of my predictions is high.
2. Since each held out set is small, the training sets overlap. This is bad.
   - Usually, averaging reduces variance:

$$\mathbb{V}\left[\overline{X}\right] = \frac{1}{n^2}\sum_{i=1}^{n}\mathbb{V}\left[X_i\right] = \frac{1}{n}\mathbb{V}\left[X_1\right].$$

   - But only if the variables are independent. If not, then

$$\begin{aligned}\mathbb{V}\left[\overline{X}\right] &= \frac{1}{n^2}\mathbb{V}\left[\sum_{i=1}^{n} X_i\right] \\ &= \frac{1}{n}\mathbb{V}\left[X_1\right] + \frac{1}{n^2}\sum_{i\neq j}\text{Cov}\left[X_i,\ X_j\right].\end{aligned}$$

   - Since the training sets overlap a lot, that covariance can be pretty big.
3. We have to estimate this model n times.
   - There is an exception to this one. More on that in a minute.

K-fold CV

- To aleviate some of these problems, people usually use K-fold cross validation.
- The idea of K-fold is

1. Divide the data into K groups.
2. Leave a group out and estimate with the rest.
3. Test on the held-out group. Calculate an average risk over these $\sim n/K$ data.
4. Repeat for all K groups.
5. Average the average risks.

Why K-fold better?

1. Less overlap, smaller covariance.
2. Larger hold-out sets, smaller variance.
3. Less computations (only need to estimate K times)

Why might it be worse?

1. LOO-CV is (nearly) unbiased.
2. The risk depends on how much data you use to estimate the model.
3. LOO-CV uses almost the same amount of data.

Viualizing CV

CV code

```
cv.lm <- function(data, formulae, nfolds = 5) {
  data <- na.omit(data)
  formulae <- sapply(formulae, as.formula)
  responses <- sapply(formulae, function(form) all.vars(form)[1])
  names(responses) <- as.character(formulae)
  n <- nrow(data)
  fold.labels <- sample(rep(1:nfolds, length.out = n))
  mses <- matrix(NA, nrow = nfolds, ncol = length(formulae))
  colnames <- as.character(formulae)
  for (fold in 1:nfolds) {
    test.rows <- which(fold.labels == fold)
    train <- data[-test.rows, ]
    test <- data[test.rows, ]
    for (form in 1:length(formulae)) {
      current.model <- lm(formula = formulae[[form]], data = train)
      predictions <- predict(current.model, newdata = test)
      test.responses <- test[, responses[form]]
      test.errors <- test.responses - predictions
      mses[fold, form] <- mean(test.errors^2)
    }
  }
  return(colMeans(mses))
}
```

Over-fitting vs. Under-fitting

Over-fitting means estimating a really complicated function when you don't have enough data.

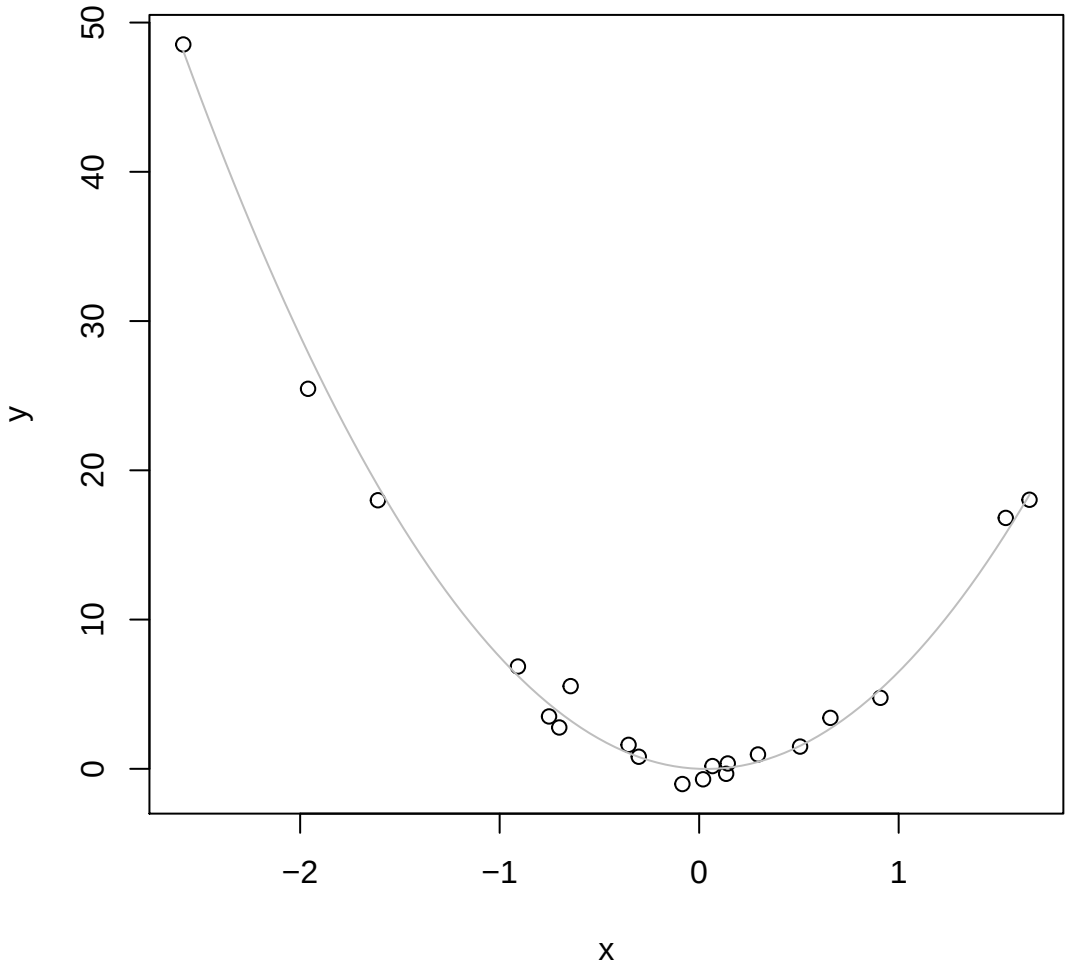
- This is likely a low-bias/high-variance situation.

Under-fitting means estimating a really simple function when you have lots of data.

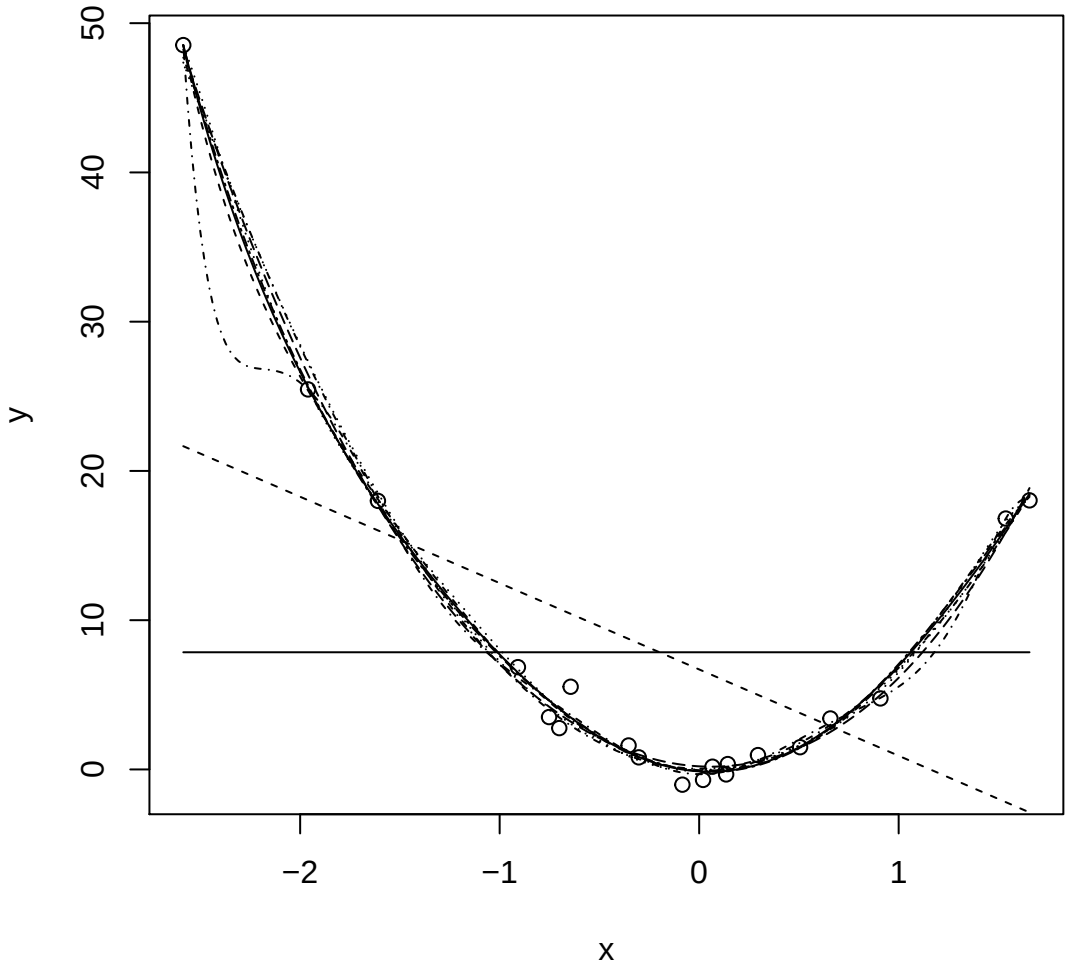
- This is likely a high-bias/low-variance situation.
- Both of these outcomes are bad (they have high risk).
- The best way to avoid them is to use a reasonable estimate of **prediction risk** to choose how complicated your model should be.

Example

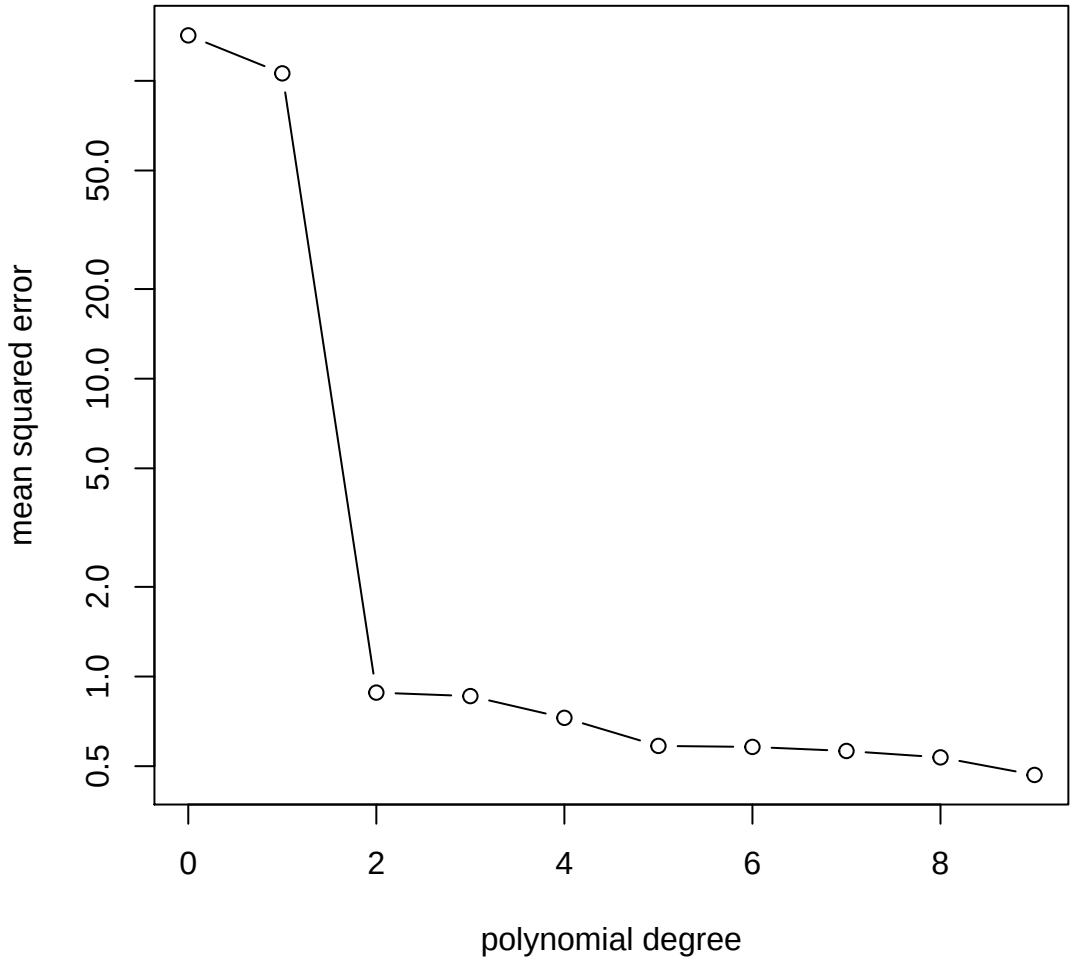
```
x = rnorm(20)
y = 7 * x^2 - 0.5 * x + rnorm(20)
plot(x, y)
curve(7 * x^2 - 0.5 * x, col = "grey", add = TRUE)
```



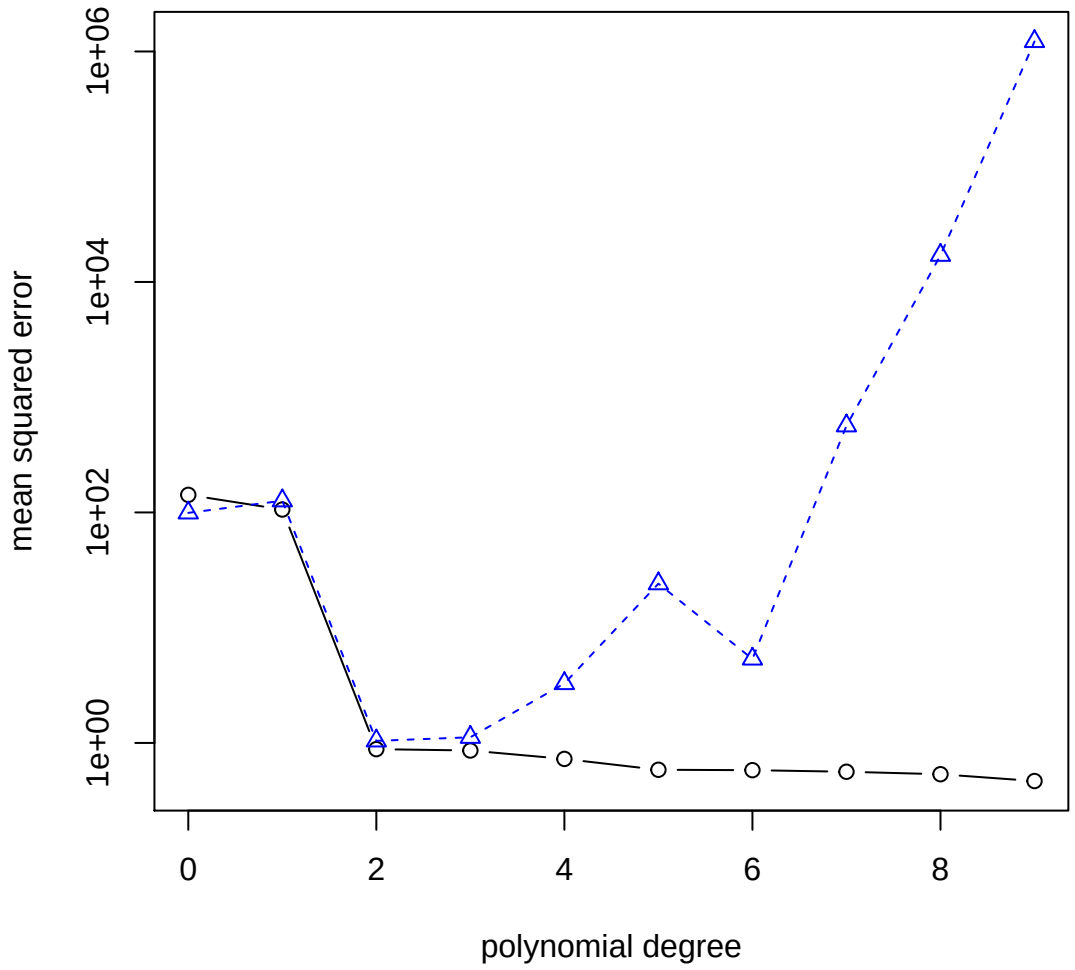
```r
plot(x, y)
poly.formulae <- c("y~1", paste("y ~ poly(x,", 1:9, ")", sep = ""))
poly.formulae <- sapply(poly.formulae, as.formula)
df.plot <- data.frame(x = seq(min(x), max(x), length.out = 200))
fitted.models <- list(length = length(poly.formulae))
for (model_index in 1:length(poly.formulae)) {
fm <- lm(formula = poly.formulae[[model_index]])
lines(df.plot$x, predict(fm, newdata = df.plot), lty = model_index)
fitted.models[[model_index]] <- fm
}
```



```r
mse.q <- sapply(fitted.models, function(mdl) {
mean(residuals(mdl)^2)
})
plot(0:9, mse.q, type = "b", xlab = "polynomial degree", ylab = "mean squared error",
log = "y")
```



```
x.new = rnorm(20000)
y.new = 7 * x.new^2 - 0.5 * x.new + rnorm(20000)
gmse <- function(mdl) {
mean((y.new - predict(mdl, data.frame(x = x.new)))^2)
}
gmse.q <- sapply(fitted.models, gmse)
plot(0:9, mse.q, type = "b", xlab = "polynomial degree", ylab = "mean squared error",
log = "y", ylim = c(min(mse.q), max(gmse.q)))
lines(0:9, gmse.q, lty = 2, col = "blue")
points(0:9, gmse.q, pch = 24, col = "blue")
```



```
little.df <- data.frame(x = x, y = y)
cv.q <- cv.lm(little.df, poly.formulae)
plot(0:9, mse.q, type = "b", xlab = "polynomial degree", ylab = "mean squared error",
log = "y", ylim = c(min(mse.q), max(gmse.q)))
lines(0:9, gmse.q, lty = 2, col = "blue", type = "b", pch = 2)
lines(0:9, cv.q, lty = 3, col = "red", type = "b", pch = 3)
legend("topleft", legend = c("In-sample", "Generalization", "CV"), col = c("black",
"blue", "red"), lty = 1:3, pch = 1:3)
```