# Lecture 8: Regression: model complexity and generalization
Modeling Social Data, Spring 2019
Columbia University

March 15, 2019

# Notes from ce2388

## 1 Introduction

Deciding how complex your model should be and how to evaluate it. Phrase a linear model as an optimization problem. Today we are discussing the Art of Modeling the claim is that for most social science problems you dont need fancy methods. When the features are really clear (age, income, etc.) linear models are usually good enough. If you understand everything you can do with a linear model then more complex models are just a step away.

## 2 notes

Check Lecture 7 notebook

A few things to keep in mind when exploring this data set: Looking at a histogram of page views on a log-scale there is no place to put a zero. The really really high page view counts are probably bots or something. If we did not have a log-scale we would have a very low resolution. Another key aspect of modeling is the readability of your plots in order to visualize the data you are manipulating. So splitting the data into different groups can help.

The users: (age, gender, daily views, median daily views) = 225,000 rows. The shaded area around the line, when using geom_smooth, is capturing the uncertainty. Now, instead of having ggplot2 do the modeling for us, let's do it ourselves.

```
model <- lm(log10(daily.views) ~ age, datamodel = data)
summary(model)
```

The standard error is if you got different samples of the data how much would the coefficient vary. $X \to \bar{X}_n \quad \sigma \to \sigma_{se} = \frac{\sigma}{\sqrt{n}}$

Bootstrap.

1. Sample from data you have
2. Estimate coefficient
3. look at standard deviation of distribution of estimates

So you do this a whole bunch of times and you'll build up some distribution where $\hat{w}$ is our estimate and we look at the width of the distribution so the standard deviation of this distribution is the standard error on the coefficient estimate. If you have a small sample the distribution will be wider than if you had a large sample of data. If you have enough data you'll get a really tight distribution and still a small standard error.

There are a bunch of tidyverse functions

```
tidy(model)
glance(model)
```

```
tidy(model) %>%
    ggplot(aes(x = term, ye =estimate) +
    geum_pointrange(aes(ymin = estimate - stderror ...)) +
    facet_wrap(~ term, scale = "free_y")
```

$\hat{y} = w_0 + w_1 x_1 + w_2 ...$

$\hat{y} = w_0 + w_1 \text{age} + w_2 \text{age}^2$ It's linear in the coefficients. But not necessarily the features themselves (the original features).

```
M <- model.matrix(log10(daily.views) ~ age + I(age^2), model_data)
## '+' means use this set of variables so 'I' forces a new feature which is age^2
head(M)
```

This is the actual set of numbers that go into our $X$ matrix. ($\hat{W} = (X^T X)^{-1} X^T y$). So we are just offloading all this onto a linear algebra library, because we are lazy and don't really have time for that. When we go into a non-linear world our ability to interpret the tables falls apart, but it's useful for plots. Adding gender specific features means that the other gender is only explained by the unspecified features. While the machine can do all these things it's up to you to figure out what to put in. There's a lot of creativity when constructing the model.

### Model evaluation

The more features the more crowded the plot and the harder it is to interpret or visualize.

```
ggplot(plot_data, aes(x = pred, y = geom_mean_daily_views, color = age)) +
  geom_point() +
  geom_abline(linetype = "dashed") +
  xlab('Predicted') +
  ylab('Actual') +
  facet_wrap(~ gender, scale = "free")
```

We don't have that many points because we are taking the averages.

But now we can see how terrible this model is

```
In [15]: pred_actual <- model_data %>%
           add_predictions(model) %>%
           mutate(actual = log10(daily.views))

         ggplot(pred_actual, aes(x = 10^pred, y = 10^actual)) +
           geom_point(alpha = 0.1) +
           geom_abline(linetype = "dashed") +
           scale_x_log10(label = comma, breaks = seq(0,100,by=10)) +
           scale_y_log10(label = comma) +
           xlab('Predicted') +
           ylab('Actual')
```

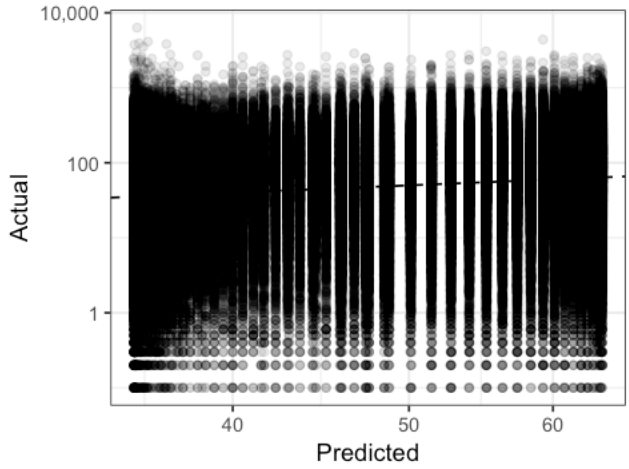



Figure 1: The variance is crazy and this model is good for nothing.

We can also quantify how well this does. MSE-model $= \frac{1}{n}\sum_{i-1}^{n}(y_i - \hat{y}_i)^2$, RMSE-model $= \sqrt{\text{MSE}_{\text{model}}}$, $(var(y))$ = MSE-baseline(mean) $= \frac{1}{n}\sum_{i=1}(y_i - \bar{y})^2$. $\frac{MSE-baseline-MSE-model}{MSE-baseline} = R^2$ which is the fraction of variance explained.

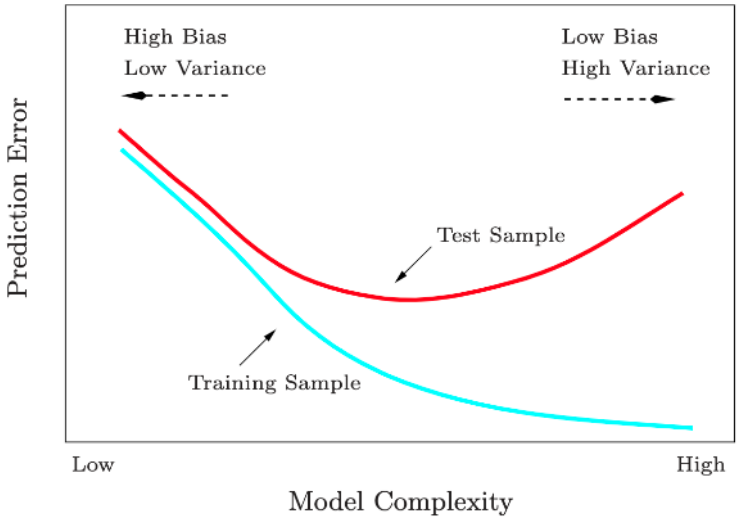
In R:

```
rmse(model, model_data)
rsquare(model, model_data)
```

Check Lecture 8

Bias-Variance tradeoff. A linear model is biased because it can only find linear trends. But it does not vary a lot with different samples of the data. High bias, low variance. As the model gets more complicated you can have a better fit. At some point there's a sweet spot with enough complexity and a good balance of bias-variance.

## Bias-variance tradeoff




Simple models may be "wrong" (high bias), but fits don't vary a lot with different samples of training data (low variance)

Cross-validation. You can train it one one subset and test the model on another, never seen, subset of the data.

Complexity control

$\frac{1}{n}\sum_{i=1}^{n}(y_i - wx_i)^2 (which\ is\ the\ fit) + \lambda||w||^2 (size\ of\ coefficients,\ tradeoff\ between\ fit\ and\ complexity)$

So we have our hyperparameter $\lambda$ which is what we need to determine to find the best model. We can do this through various testing techniques. Into to glmnet what glmnet does is figure out the paremeters in our fit to the data above So even if you give it something crazy like a 10th degree polynomial it will return something normal.

"Weird R things with the dots, it's just terrible" - Jake Hofman

# Notes from cg3111

## 1 Regression and Model Evaluation

Regression is a statistical analysis procedure that can summarize the current data, predict future outcomes and describe the associations between the predictors and the outcomes. In today's class, we used the file from last lecture with information on 225,000 anonymous Nielsen panelists, containing their age, gender, and the number of pages ( distinct urls) they typically access on their web browser each day, filtering out users with zero pageviews.

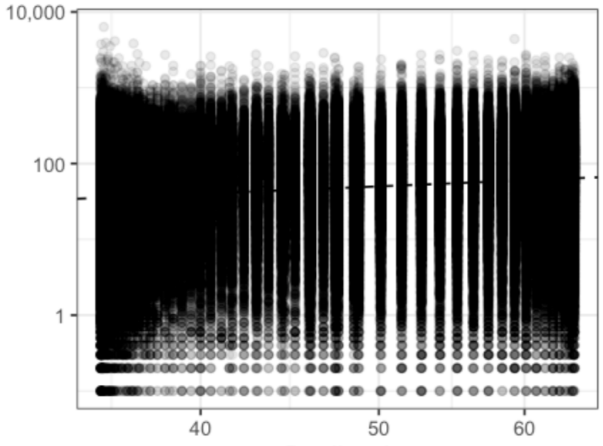
### 1.1 Plotting predicted vs. actual values

We fit the predicted and actual values based on the model and differentiate them with respect to their gender. Each dot on the plot corresponding to an 'age' value is computed by taking the median of all page views in that age. The size of the dots is proportional to the number of people with that age. We aim to find the best fit for data by visual interpretation.

- Too many features makes it difficult to understand the plot.so we first start by plotting the predicted vs.actual values along with a diagonal line to indicate how a perfect model would fit. The points are binned such that each dot represents the average actual outcomes of all the observations that were predicted to be in a small range.
- We then differentiate the data based on gender. We can notice that both the actual and predicted outcomes are higher in female on average.
- We can also choose to differentiate with respect to a continuous attribute such as age.

### 1.2 Model Variability

Until now, the model seems to do a good job. However, once the binning is removed and all the 225,000 points are plotted, we can notice much variation within the individual observations. The variance between the output is greater than the individual variance as shown in the figure below. Thus, viewing the average data is not same as viewing the entire data.




### 1.3 Quantifying the model

Now, it is evident that variance is an important information about the data.We need to account for variation of the data along with the trend or correlation in the data.

One useful metric of evaluation is the RMSE (Root Mean Square Error). It represents the standard deviation between the differences of predicted value and average value, given as,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \tag{1}$$

we have,

$$MSE_{\text{baseline}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{2}$$

Thus, we define $R^2$, the fraction of variance explained, as,

$$R^2 = \frac{MSE_{\text{baseline}} - MSE_{\text{model}}}{MSE_{\text{baseline}}} \tag{3}$$

For the previous plot, the RMSE is very high with a value of about 165. This implies that the prediction model is almost same as randomly guessing the outcome.

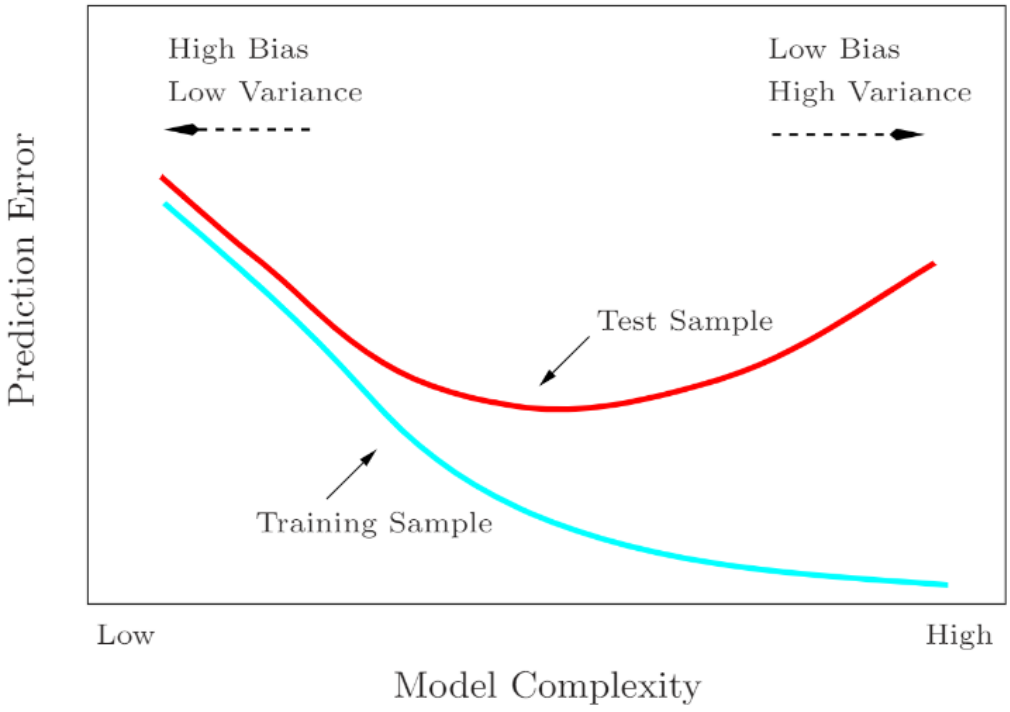
# 2 Overfitting

A sufficiently complex model can fit any data and attain almost zero error on the data. However, it should be noted that the models should be complex enough to explain the past but simple enough to generalize to the future. This can be achieved by splitting the data into 3 sets.

- Training dataset - The model is trained using this data
- Validation dataset - This dataset is used to test the performance of the model/classifier.
- Testing dataset - The dataset gives an unbiased estimate of the accuracy of the model since the model has never seen this data before.

# 3 Bias and Variance

Bias is the assumption that the model has about the data. It is commonly termed as, "the model is 'biased' towards a wrong type of data". Variance is a measure of how much the model varies with the data used for training. A model with high variance may lead to overfitting as it will change significantly depending on the training data. A model with high bias may results in underfitting as it makes strong and often incorrect assumptions about the data. The prediction with respect to bias and variance is as shown below.




# 4 Cross Validation

Cross Validation is one technique to ensure effciency of the model. As discussed before, the data at hand is split into 3 separate sets.

- The training data is fed to the model for it to learn and build a classifier based on the input.
- Another Held-out data set is used to frequently test the performance of the model. Cross validation mainly deals with how data is split into training and validation datasets for better model verification.
- The testing dataset is used to provide the final model accuracy on a totally new data. This becomes a good estimate for the model's performance

One way achieve this is to randomly split data into 3 separate dataset and used them for training, validation and testing respectively. One backdraw in this technique is when there's very little data. It is important to have sufficient data at hand to train the model and equal important to have data for validation.

## 4.1 K-fold cross validation

One technique to follow is the K-fold Cross Validation. In this approach, we split the data into k sets, choose k-1 sets for training and 1 set for validations, then measure the error in prediction. Repeat this process k times with each of the k sets serving as a validation dataset once. Take average of these errors and choose the model with the least average validation error. +


source: https://medium.com/@sebastiannorena/some-model-tuning-methods-bfef3e6544f0

# 5 Regularization

Regularization is a way to penalize model complexity to allow it to generalize well to the future. One way to achieve this is to shrink the weights. The loss function is as given below.

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - wx_i)^2 + \lambda||w||^2 \tag{4}$$

here, as the $\lambda$ value increases the coefficient decreases. Thus, the weights are brought down for the model to generalize well