CS 5785 – Applied Machine Learning – Lec. 8

Prof. Nathan Kallus, Cornell Tech

September 18, 2018

Basics of Inference for the Linear Model

Inference isn't the core of this class (the core is making predictive models), however it is a useful tool.

Supervised learning algorithm can be thought of as a black box that maps data to a prediction rule (f^*).

OLS is one such algorithm that proceeds as follows:

$$\hat{f}(X) = \hat{\beta}^T X$$
$$\text{where} \quad \hat{\beta} = (X^T X)^{-1} X^T Y.$$

Inference opens up the black box (data $\rightarrow \hat{f}$) and seeks to understand what is going on inside, and what exactly can be learned and generalized. It asks questions like "How does a particular supervised learning algorithm depend on the data, and once we have that understanding, what can we learn from this that generalizes, beyond just the data itself".

For example some of the questions we can answer are as follows:

- How much variance/uncertainty is there in $\hat{\beta}$?
- How much variance/uncertainty is there in $\hat{f}$(x)?
- How different is $\hat{f}$ from f^*?
- We know $\hat{f} \neq f^*$, but maybe we can say something about f^* nonetheless? (i.e. "f^* is like this" and "to what extent is f^* like this?")
- How sure are we that a certain variable matters, or does not matter?

For this lecture we will focus on OLS, and to answer these questions, we must first understand how OLS depends on data. In **Figure ??**, 40 points of data were drawn using random, uniformly distributed x and $y = x + 1$ with noise of

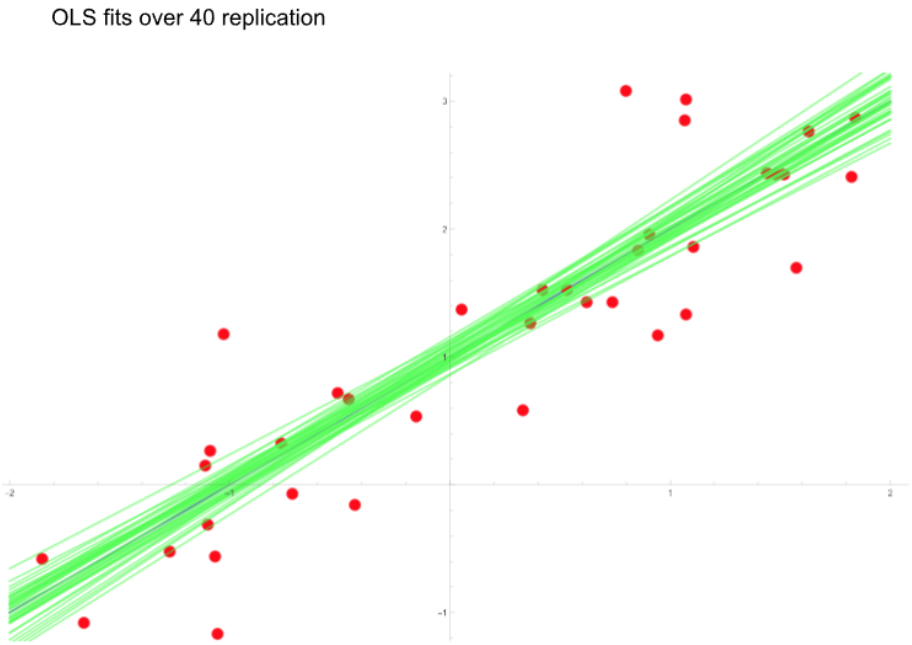

Figure 1: 40 OLS lines fit to 40 random draws of data

a univariate Gaussian distribution with mean 0 and variance of 0.5.

Data was drawn randomly 40 times and a line was fit to the data using OLS each time. In particular, since the data is drawn at random, $\hat{\beta}$ is also random because it depends on the data. Drawing a new set of data is like drawing a new random $\hat{\beta}$.

Figure ?? shows a histogram of the slopes (values of β_1) over 1000 replications.

As we know, the correct slope of f^* is 1, and the data is centered on 1, but we have some error. If I get a model with a slope, for example, $\hat{\beta} = 0.9$, then what can I say about β^*? Using inference we can make such a statement.

Suppose the true model is linear $f^*(x) = \mathbb{E}[Y|X = x] = {\beta^*}^T X$
And that the errors (deviation of $\hat{Y}$) from for $Y = f^*(x)$ are Gaussian,

$$\begin{aligned} \epsilon &= y - f^*(x) \sim N(0, \sigma^2), \\ y &= f^*(x) + \epsilon \\ Y &= f^*(X) + E \end{aligned}$$

Then:

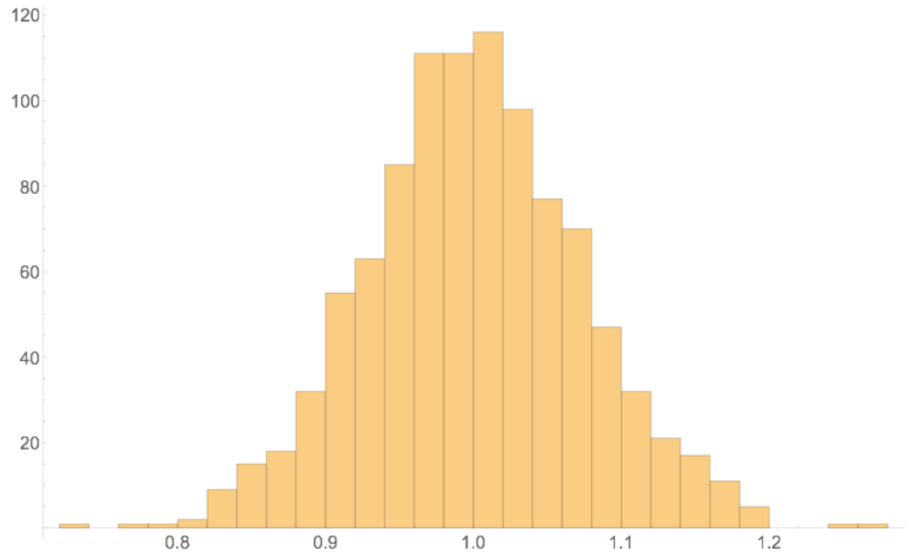

Figure 2: Histogram of β_1 values over 1000 replications

$$\begin{aligned}
\hat{\beta} &= (X^T X)^{-1} X^T Y \\
&= (X^T X)^{-1}(X^T X)\beta^* + (X^T X)^{-1} E \\
&= \beta^* + (X^T X)^{-1} E \\
\hat{\beta} - \beta^* &= (X^T X)^{-1} E
\end{aligned}$$

where

$$E = (\epsilon_1, \epsilon_2, \cdots, \epsilon_n)^T.$$

Note: The second term in line 3 is the error in $\hat{\beta}$.

It turns out that $\hat{\beta}$ follows the Multivariate Gaussian Distribution,

$$\hat{\beta} \sim N(\beta^*, \sigma^2 (X^T X)^{-1}),$$

and $\forall j, k = 0, 1, 2, \cdots, p, j \neq k$,

$$\begin{aligned}
&\mathbb{E}(\hat{\beta}_j) = \beta_j^*, \\
&\text{Var}(\hat{\beta}_j) = \sigma^2 (X^T X)^{-1}_{jj}, \\
&\text{Cov}(\hat{\beta}_j, \hat{\beta}_k) = \sigma^2 (X^T X)^{-1}_{jk}.
\end{aligned}$$

In particular,

$$\hat{\beta}_j = N(\beta_j^*, \sigma^2 (X^T X)^{-1}_{jj}).$$

As the number of draws increases (approaching infinity), the distribution approaches Gaussian. If we did $\sim 1,000$ replications we could in fact develop this, unfortunately we don't often have that number of data sets. Regardless, we

know that this will happen, so we do not need $1,000$ replications, however we don't know the mean and standard deviation of this distribution.

So what? First of all, we now know how much uncertainty we have in $\hat{\beta}$:

$$\sigma_{\hat{\beta}_j} = \sigma\sqrt{(X^T X)^{-1}_{jj}}$$

actually we almost know, but we need

$$\begin{aligned}\sigma &= \mathbb{E}[\epsilon^2] \\ &= \mathbb{E}[(Y - \beta^{*T}x)^2]\end{aligned}$$

We can estimate σ in p-dimensions as:

$$\hat{\sigma} = \frac{1}{n-p-2}\sum_{i=1}^{n}(Y_i - \hat{\beta}^T x_i)^2$$

Then, β_j has a standard deviation of approximately $\hat{\sigma}\sqrt{(X^T X)^{-1}_{jj}}$
Similarly, the prediction at $\hat{F}(x) = \beta^T x$ has a standard deviation of approximately:

$$\hat{\sigma}\sqrt{x^T(X^T X)^{-1}x}$$

This tells us how uncertain we are in our prediction of $\beta^{*T}x$

What does this uncertainty mean?
If we were to repeatedly draw a new training dataset of size n, and fit OLS to it, the collection of the results we get for the j^{th} coefficient would have such a standard deviation.

This quantifies how much uncertainty varies from an algorithm's dependence on the random data.

$$\widehat{\beta}_j \sim \mathcal{N}\Bigg(\beta_j^*, \underbrace{\sigma^2(\mathbb{X}^T\mathbb{X})^{-1}_{jj}}_{\sigma^2(\widehat{\beta}_j)}\Bigg)$$

Where does $\widehat{\beta}_j$ fall under the bell curve?
Usually in the middle.

$$\begin{aligned}&\mathbb{P}\Big(\beta_j^* - z\sigma(\beta_j^*) \leq \widehat{\beta}_j \leq \beta_j^* + z\sigma(\beta_j^*)\Big) \\ &= \mathbb{P}\Big(-z \leq \underbrace{\frac{\widehat{\beta}_j}{\sigma(\widehat{\beta}_j)}}_{\text{z-score of } \widehat{\beta}_j} - \frac{\beta_j^*}{\sigma(\widehat{\beta}_j)} \leq z\Big) \\ &= \mathbb{P}\Big(-z \leq \mathcal{N}(0,1) \leq z\Big)\end{aligned}$$

For $z = 1.96 \approx 2$, we get that

$$\beta_j^* - z\sigma(\widehat{\beta}_j) \leq \widehat{\beta}_j \leq \beta_j^* + z\sigma(\widehat{\beta}_j)$$

happens with probability 95%

Add $-\beta_j^* - \widehat{\beta}_j$ and negate

$$\widehat{\beta}_j - z\sigma(\widehat{\beta}_j) \leq \beta_j^* \leq \widehat{\beta}_j + z\sigma(\widehat{\beta}_j)$$

with probability 95%, that is

$$\beta_j^* \in \underbrace{\left[\widehat{\beta}_j - z\sigma(\beta_j^*), \widehat{\beta}_j + z\sigma(\beta_j^*)\right]}_{\text{confidence interval for } \beta_j^*}$$

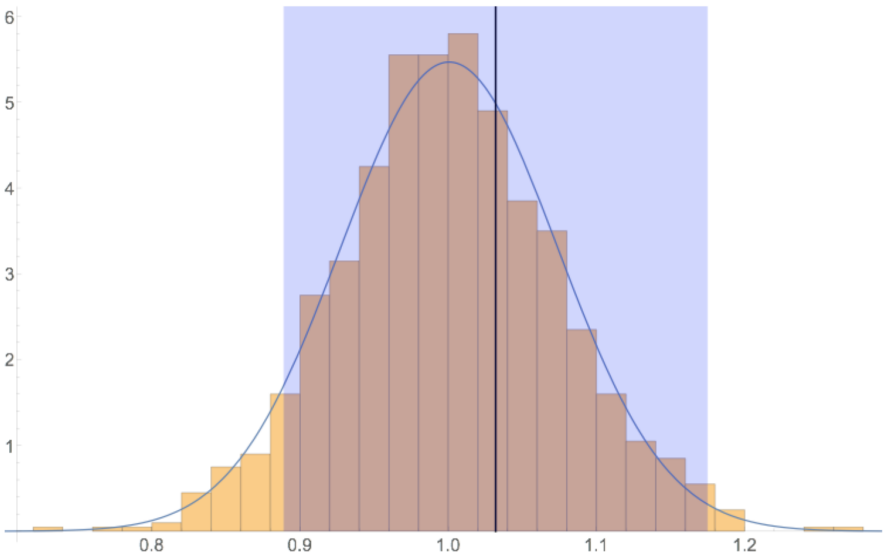

Figure 3: Confidence interval representation for $\widehat{\beta}_j$

If we knew $\sigma(\beta_j^*)$, then we know that this interval will contain the unknown β_j^* with high probability. We don't know $\sigma(\beta_j^*)$, so we estimate it as

$$\widehat{\sigma}(\beta_j^*) = \widehat{\sigma}\sqrt{(\mathbb{X}^T\mathbb{X})_{jj}^{-1}}$$

$\frac{\widehat{\beta}_j}{\widehat{\sigma}(\beta_j^*)}$ known as the t-score of $\widehat{\beta}_j$

Note that going from $\sigma(\beta_j^*)$ to $\widehat{\sigma}(\beta_j^*)$ makes our interval a little to small to cover 95%. Therefore we need to expand it a bit by using Student T distribution. When n is large, this is about the same, however.

p-value

The p-value is the largest $p \in [0, 1]$ such that the $(1-p) - CI$ contains 0.
A way to score how far $\widehat{\beta}_j$ is from 0.

What does a confidence interval mean?
The probability 95% is over random new draws of data.
I.e. given the data & interval, it's not that β_j^* is random and may be in the interval with probability 95%.
Rather, it's $\widehat{\beta}_j$ that's random.

How to implement in code?

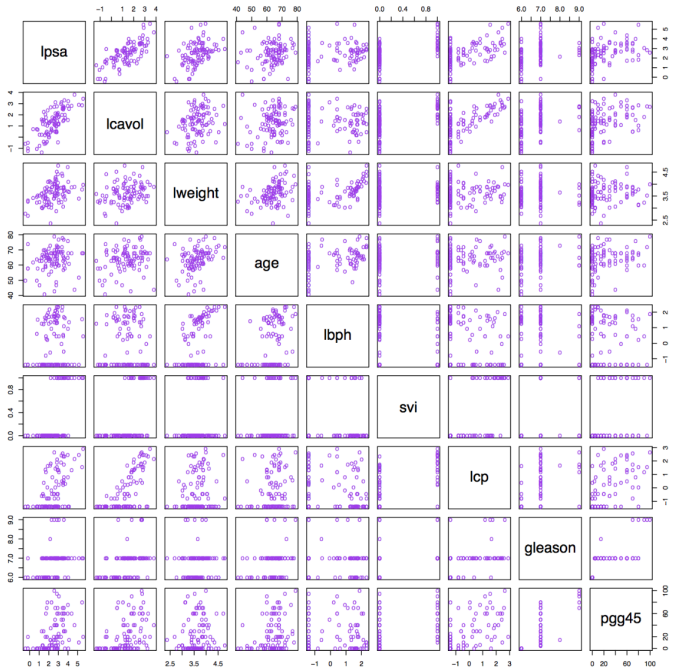

FIGURE 1.1. *Scatterplot matrix of the prostate cancer data. The first row shows the response against each of the predictors in turn. Two of the predictors,* `svi` *and* `gleason`*, are categorical.*

Figure 4: Scatterplot Matrix

```
In [40]: X = prostate[['lcavol','lweight','age','lbph','svi','lcp','gleason','pgg45']
         Y = prostate['lpsa']
         print X.shape, Y.shape

         (97, 8) (97,)

In [39]: import sklearn
         import sklearn.linear_model
         lm=sklearn.linear_model.LinearRegression().fit(X,Y)
         print lm.intercept_
         print lm.coef_

         0.18156084546895768
         [ 0.56434128  0.62201979 -0.02124819  0.09671252  0.7616734  -0.10605094
           0.04922793  0.00445751]
```

Figure 5: Code example implementation

```
In [37]: import statsmodels.api as sm
         sm.OLS(Y,sm.add_constant(X)).fit().summary()
```

Out[37]:

OLS Regression Results

Dep. Variable:	lpsa	**R-squared:**	0.663
Model:	OLS	**Adj. R-squared:**	0.633
Method:	Least Squares	**F-statistic:**	21.68
Date:	Mon, 17 Sep 2018	**Prob (F-statistic):**	7.65e-18
Time:	10:23:26	**Log-Likelihood:**	-98.248
No. Observations:	97	**AIC:**	214.5
Df Residuals:	88	**BIC:**	237.7
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1816	1.321	0.137	0.891	-2.443	2.806
lcavol	0.5643	0.088	6.425	0.000	0.390	0.739
lweight	0.6220	0.201	3.096	0.003	0.223	1.021
age	-0.0212	0.011	-1.917	0.058	-0.043	0.001
lbph	0.0967	0.058	1.670	0.098	-0.018	0.212
svi	0.7617	0.241	3.158	0.002	0.282	1.241
lcp	-0.1061	0.090	-1.180	0.241	-0.285	0.073
gleason	0.0492	0.155	0.317	0.752	-0.259	0.358
pgg45	0.0045	0.004	1.021	0.310	-0.004	0.013

Figure 6: Statistic summary

Beyond OLS

We are only covering the basics of inference - just enough to understand confidence intervals.
A few things to be aware of (mostly without explanation):

- What if ϵ is not normally distributed?
  Mostly OK, as long as n is moderately sized relatively to p. We shouldn't worry about this.
- What if f^* is not linear?
  The inference is really on f^* best linear approximation. It makes sense if f^* is almost linear.

- What about linear regression?
  Proceeds mostly the same. Most packages for linear regression (sklearn excluded) will report p-values and confidence intervals for coefficients.
- What about shrinkage and subset selection?
  Won't work out of the box (for inference on coefficients)... Advanced topic.
- What about kNN, neural networks etc?
  Can make inference on predictions $\widehat{f}(x)$. Hard to do on the model itself.