Regression methods

Generic supervised learning algorithms

- These are classical methods for classification with both qualitative and quantitative outcomes
- We will study:
  - Linear regression
  - Regularized methods: ridge and LASSO regression
  - Logistic regression
  - Next session:
    - SVM
    - Decision trees
    - K nearest neighbors
    - Naive Bayes classifier
    - ...

Overview: Linear Regression

- Linear regression is a simple approach to supervised learning, as it assumes that the dependence of Y on $X_1, X_2,\ldots,X_p$ is linear
- Most modern machine learning approaches can be seen as generalizations or extensions of linear regression
- When augmented with kernels or other forms of basis function expansion (which replace X with some non-linear function of the inputs), it can also model non-linear relationships

- Goal: predict Y from X by $f(X)$

Hypothesis

- **Linear Assumption**. Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).
- **Remove Noise**. Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.
- **Remove Collinearity**. Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated or use other approaches
- **Gaussian Distributions**. Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on your variables to make their distribution more Gaussian looking.
- **Rescale Inputs**: Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

Parameter Estimation

- In practice, we often seek to select a distribution (model) corresponding to our data
- If the model is parameterized by some set of values, then this problem is that of parameter estimation
- In general, we typically use maximum likelihood estimation (MLE) to obtain parameter estimates

Linear Regression Model

- Input vector: $X^T = (X_1, X_2, \ldots, X_p)$
- Output Y is real-valued (quantitative response)
- We want to predict Y from X
- Before we actually do the prediction, we have to *train* the function $f(X)$
- By the end of training, we have a function $f(X)$ to map every X into an estimated Y

Linear Regression Model

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

- This is a linear combination of the measurements that are used to make predictions, plus a constant.
- No matter the source of the X_j, the model is linear in the parameters.
- β_0 is the intercept and β_j is the slope for the *j*th variable X_j, which is the **average** increase in Y when X_j is increased by one unit and all other X's are held constant.

Ordinary Least Squares Estimation

- Typically we have a set of *training data* $(X_1, Y_1) \ldots (X_n, Y_n)$ from which to estimate the parameters *beta*
- Each X_i is a vector of feature measurements for the *i*th case.
- RSS stands for *residual sum of squares*:

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} (Y_i - f(X_i))^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2$$

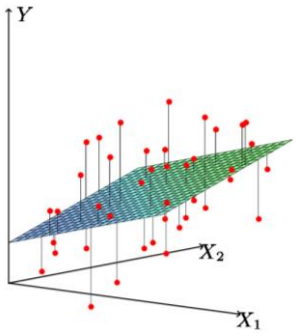
- The RSS is also called the *sum of squared errors* (SSE), where

$$MSE = \frac{SSE}{n} = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2$$

- We see that the MLE for $\boldsymbol{\beta}$ is the one that minimizes the RSS. Thus, we estimate the parameters using *ordinary least squares* (OLS), which is identical to the MLE, to choose $\hat{\beta}_0$ through $\hat{\beta}_p$ as to minimize the RSS.

OLS Estimation

- We illustrate the geometry of OLS fitting, where we seek the linear function of *X* that minimizes the sum of squared residuals from *Y*.



- The predictor function corresponds to a plane (hyper plane) in the 3D space.
- For accurate prediction, hopefully the data will lie close to this hyper plane, but they won't lie exactly in the hyper plane.

OLS Estimation

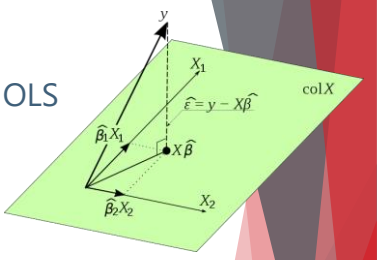
- Let $\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_p \end{bmatrix}$, so the $RSS(\boldsymbol{\beta}) = (\boldsymbol{Y} - \boldsymbol{X\beta})^T\ (\boldsymbol{Y} - \boldsymbol{X\beta})$
- So, we must solve the following quadratic minimization problem:

$$\widehat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} RSS(\boldsymbol{\beta})$$

- This minimization problem has a unique solution, provided that $\boldsymbol{X}$ has full column rank (i.e. the *p* columns of $\boldsymbol{X}$ are linearly independent), given by solving the normal equations:

$$(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{Y}$$

OLS Estimation (cont.)



- The fitted values at the training inputs (i.e. vector of the OLS predictions) are:

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

- In geometric representation, this corresponds to an orthogonal projection of $\boldsymbol{Y}$ onto the column space of $\boldsymbol{X}$.
  - The matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ is the projection matrix, which is called the *hat* matrix because it puts a hat on $\boldsymbol{Y}$.

Accuracy of Coefficient Estimates

- Let's consider a simple linear regression model with $\hat{\beta}_0$ and $\hat{\beta}_1$. How close are $\hat{\beta}_0$ and $\hat{\beta}_1$ to the true values β_0 and β_1, respectively?
- We can answer this by computing the standard errors associated with $\hat{\beta}_0$ and $\hat{\beta}_1$
- These SEs can be used to compute confidence intervals (CIs), prediction intervals (PIs), and perform hypothesis tests on the coefficients.

Accuracy of the Model: RSE

- The residual standard error (RSE) is an estimate of the standard deviation of ε.
- In other words, RSE is the average amount that the response will deviate from the true regression line:

$$RSE = \sqrt{\frac{RSS}{n-p-1}}$$

where *p* is the number of predictors (slopes) in the regression model (not including the intercept).

Accuracy of the Model: R^2

- The proportion of variability in *Y* that can be explained using *X:*

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where total sum of squares (TSS) measures the total variance in the response *Y*. It is thought of as the amount of variability inherent in the response before the regression is performed.

- Note that RSS measures the amount of variability that is left unexplained after performing the regression.
- Always between 0 (no fit) and 1 (perfect fit)

Is $\beta_j = 0$ or not?

- H_0: There is no relationship between X_j and Y ($\beta_j = 0$).
- H_a: There is some relationship between X_j and Y ($\beta_j \neq 0$).
- Compute the *t-statistic*: $t = \frac{\hat{\beta}_j - 0}{SE(\hat{\beta}_j)}$
- If *t* is large (and the *p-value* is small, typically < α = 0.05), then we reject H_0 and declare that there is relationship.
- We use the Regression Output table to get the beta coefficients, standard errors, t-statistics and p-values.

Are all regression coefficients 0?

- H_0: all slopes equal 0 ($\beta_1 = \beta_2 = \cdots = \beta_p = 0$).
- H_a: at least one slope ≠ 0.
- Compute the *F-statistic*: $F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)} \sim F_{p,n-p-1}$
- We use the ANOVA table to get the *F-statistic* and its corresponding p-value.
- If *p-value* < 0.05, reject H_0. Otherwise, all of the slopes equal 0 and none of the predictors are useful in predicting the response.

Linear regression with python

- The OLS algorithm is available in many libraries
- In `statsmodels` you can find a first way of working with OLS regression

17

Advertising

- We will work on an advertising data set, it includes sales per semester and budget on different medium
- 200 obs. and 4 variables

Import data and train the model.
Predict new values with .predict

- The first model is Sales ~ TV

```
smf.ols(formula='Sales ~ TV', data=data).fit()
```

You can display new parameters:

```
.params, .conf_int(), .rsquared, .predict()
```

18

Advertising (2)

- We now study the full model
- Use `smf.ols` and display results with `.summary()`
- Using `LinearRegression` from `scikit-learn`, you can also run linear regression

Run the model with both libraries.
Plot Y / pred(Y) and Y / res(Y)

19

Deciding on Important Variables – stepwise regression

- *Best Subset Selection*: we compute the OLS fit for all possible subsets of predictors and then choose between them based on some criterion that balances training error with model size.
- There are 2^p possible models, so can't examine them all.
- We use an automated approach that searches through a subset of all the models.
  - Forward Selection
  - Backward Selection

Overview: Forward Selection

- We begin with the *null model*, a model containing an intercept but no predictors.
- We fit p simple linear regressions and add to the null model that variable resulting in the lowest RSS.
- We add to that model the variable that results in the lowest RSS amongst all two-variable models.
- The algorithm continues until some stopping rule is satisfied (i.e. all remaining variables have a *p-value* greater than some threshold).

Overview: Backward Selection

- We begin with all variables in the model.
- We remove the variable with the largest *p-value* (i.e. least statistically significant).
- The new $(p - 1)$-variable model is fit, and the variable with the largest *p-value* is removed.
- The algorithm continues until a stopping rule is reached.

Qualitative Predictors

- Some predictors are not quantitative but are *qualitative*, taking a discrete set of values.
- These are known as categorical variables, which we can code as indicator variables (dummy variables).
- Examples: gender, student status, marital status, ethnicity

Qualitative Predictors

- When a qualitative predictor has more than two levels, a single dummy variable cannot represent all possible variables.
- Thus, there will always be one dummy variables less than the number of levels in the factor.
  - Factor = Ethnicity
  - Levels = Asian, Caucasian, African American
  - # of Dummy Variables = 3 – 1 = 2
- The level with no dummy variable is the *baseline*.

Qualitative Predictors

- Suppose we want to regress the quantitative response variable Y (such as balance) on both a quantitative variable (such as income) and a qualitative variable (such as gender).
- There are two levels of gender: $Gender_i = \begin{cases} 1 \text{ if female} \\ 0 \text{ if male} \end{cases}$
- The regression model (without interaction) is:

$$Balance_i \approx \beta_0 + \beta_1 Income_i + \beta_2 Gender_i$$
$$= \begin{cases} \beta_0 + \beta_1 Income_i + \beta_2 & \text{if female} \\ \beta_0 + \beta_1 Income_i & \text{if male} \end{cases}$$

- β_2 is the average extra balance that females have for a given income level; note that males are *baseline* (coded as 0).

Qualitative Predictors

- There are different ways to code categorical variables.
- There are two levels of gender: $Gender_i = \begin{cases} 1 \text{ if female} \\ -1 \text{ if male} \end{cases}$
- The regression model (without interaction) is:

$$Balance_i \approx \beta_0 + \beta_1 Income_i + \beta_2 Gender_i$$
$$= \begin{cases} \beta_0 + \beta_1 Income_i + \beta_2 & \text{if female} \\ \beta_0 + \beta_1 Income_i - \beta_2 & \text{if male} \end{cases}$$

- β_2 is the average amount that females are above the average, for any given income level; note that males are *baseline* again.

Extensions of the Linear Model

- Allow for *interaction effects*. Note that if interaction is included in the model, all of the *main effects* should be include as well (even if not statistically significant).

$$\begin{aligned} \texttt{sales} &= \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times (\texttt{radio} \times \texttt{TV}) + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 \times \texttt{radio}) \times \texttt{TV} + \beta_2 \times \texttt{radio} + \epsilon. \end{aligned}$$

- Accommodate non-linear relationships using *polynomial regression*. For example, you can include transformed versions of the predictors in the model.

$$\texttt{mpg} = \beta_0 + \beta_1 \times \texttt{horsepower} + \beta_2 \times \texttt{horsepower}^2 + \epsilon$$

Potential Problems

- There are several potential problems that may occur when fitting a linear regression model.
  1. Non-linearity of the response-predictor relationships
  2. Correlation of residuals
  3. Non-normality and non-constant variance of the residuals
  4. Outliers
  5. Collinearity

Non-linearity of the Data

- The linear regression model assumes that there is a straight-line relationship between predictors and the response.
- If the true relationship is non-linear then conclusions are suspect.
- Examine the *residual plots*, as strong patterns (U-shape) in the residuals indicate non-linearity in the data.
- If there are non-linear associations in the data, then use non-linear transformations of the predictors (e.g. log X).

Correlation of Residuals

- An important assumption of the linear regression model is that the residuals are uncorrelated.
- If there is correlation among the residuals (Durbin-Watson test), then the estimated standard errors will tend to *underestimate* the true standard errors – this makes the CIs and PIs narrower than they should be.
- These correlations frequently occur in the context of *time series* data, so consider employing *time series analysis* methods (such ARIMA, etc.).

Non-normality and Non-constant Variance of Residuals

- Another important assumption of the linear regression model is that the residuals are normally distributed and have constant variance across all levels of X
- If the residuals are not normally distributed, you can perform a Box-Cox transformation on the response Y
- If there is heteroscedasticity (Breusch-Pagan, Modified Levene, or Special White's tests), then you can consider transforming the response Y. If this doesn't fix the problem, consider computing *robust standard errors* or conduct *weighted least squares regression*

Collinearity

- In case of high collinearity between explanatory variables, OLS regression is not a good solution
- You can use PLS regression
  - PLS regression is based on orthogonal components used to fit a regression model

```
from sklearn.cross_decomposition import PLSRegression
pls = PLSRegression(n_components=2)
pls.fit(X, Y)
Y_pred = pls.predict(X)
```

Run the PLS regression model.

Application

- Split your sample using train / test using 80 / 20
- Fit the model on the training sample
- Test the model on the test sample
- Obtain RMSE

Shrinkage

- Intuition: continuous version of subset selection
- Goal: imposing penalty on complexity of model to get lower variance
- Two examples:
  - Ridge regression
  - Lasso

Ridge Regression

- Penalize by sum-of-squares of parameters
- Or

$$\hat{\beta}^{ridge} = \arg\min_{\beta}\{\sum_{i}(y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{\pi}\beta_j^2\}$$

$$\hat{\beta}^{ridge} = \arg\min_{\beta}\{\sum_{i}(y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2\}$$

$$subject\ \text{to}\ \sum_{j=1}^{p}\beta_j^2 \le s$$

Understanding of Ridge Regression

- Find the orthogonal principal components (basis vectors), and then apply greater amount of shrinkage to basis vectors with small variance.
  - Assumption: y vary most in the directions of high variance
  - Intuitive example: stop words in text classification if assuming no covariance between words
- Relates to MAP Estimation

If: $\beta \sim N(0, \tau I)$, $y \sim N(X\beta, \sigma^2 I)$

Then:

$$\hat{\beta}^{ridge} = \arg\max_{\beta}(P(Data \mid \beta)P(\beta))$$

Lasso

- Penalize by absolute value of parameter

$$\hat{\beta}^{ridge} = \arg\min_{\beta}\{\sum_{i}(y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2\}$$

$$subject\ to \sum_{j=1}^{p} |\beta| \leq s$$

Regularized methods

- Model:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$$

- OLS estimation: $\min SSE = \sum \left(Y - \hat{Y} \right)^2$
- LASSO estimation: $\min SSE = \sum_{i=1}^{n} \left(Y - \hat{Y} \right)^2 + \sum_{j=1}^{p} \left| \beta_j \right|$
- Ridge regression estimation: $\min SSE = \sum_{i=1}^{n} \left(Y - \hat{Y} \right)^2 + \sum_{j=1}^{p} \left| \beta_j^{\ 2} \right|$

PCR vs. PLS vs. Ridge Regression

- PCR discards the smallest eigenvalue components (low-variance direction). The mth component v_m solves:

$$\max_{|\alpha|<1, v_l^T S\alpha=0, l=1,\ldots,m-1} Var(X\alpha)$$

- PLS shrink the low-variance direction, while inflate high variance direction. The mth component v_m solves:

$$\max_{|\alpha|=1, v_l^T S\alpha=0, l=1,\ldots,m-1} Corr^2(y, X\alpha)Var(X\alpha)$$

- Ridge Regression: Shrinks coefficients of the principle components. Low-variance direction is shrinked more

Application of ridge and lasso regression

- Try different values of the regularization parameter using the regression model
- linear_model.Ridge(alpha =…

```
>>> from sklearn import linear_model
>>> clf = linear_model.Lasso(alpha=0.1)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
>>> print(clf.coef_)
[ 0.85  0.  ]
>>> print(clf.intercept_)
0.15
```

Logistic regression

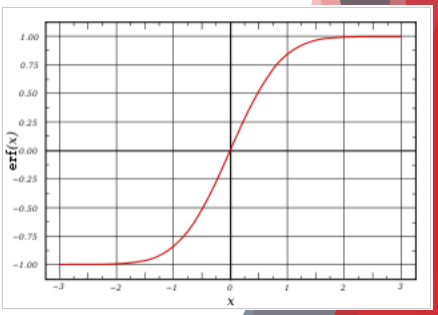
Logistic regression

- Name is somewhat misleading. Really a technique for classification, not regression. technique for classification, not regression.
- "Regression" comes from fact that we fit a linear model to the feature space.
- Involves a more probabilistic view of classification.

Logistic Regression

- Let X be the data instance, and Y be the class label: Learn P(Y|X) directly
  - Let W = (W1, W2, … Wn), X=(X1, X2, …, Xn), **WX** is the dot product
  - Sigmoid function:

$$P(Y=1|\mathbf{X}) = \frac{1}{1+e^{-\mathbf{wx}}}$$



Logistic Regression

- In logistic regression, we learn the conditional distribution P(y|x)
- Let $p_y(x;w)$ be our estimate of P(y|x), where w is a vector of adjustable parameters.
- Assume there are two classes, y = 0 and y = 1 and $p_0(\mathbf{x};\mathbf{w}) = 1 - \frac{1}{1+e^{-\mathbf{wx}}}$

$$p_1(\mathbf{x};\mathbf{w}) = \frac{1}{1+e^{-\mathbf{wx}}}$$

- This is equivalent to

- That is, the log odds of class 1 is a linear function of x
- Q: How to find **W**?

$$\log \frac{p_1(\mathbf{x};\mathbf{w})}{p_0(\mathbf{x};\mathbf{w})} = \mathbf{wx}$$

44

Constructing a Learning Algorithm

- The conditional data likelihood is the probability of the observed Y values in the training data, conditioned on their corresponding X values. We choose parameters w that satisfy

$$\mathbf{w} = \arg\max_{\mathbf{w}} \prod_{l} P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

- where $\mathbf{w} = \langle w_0, w_1, \ldots, w_n \rangle$ is the vector of parameters to be estimated, y^l denotes the observed value of Y in the l th training example, and $\mathbf{x}^l$ denotes the observed value of $\mathbf{X}$ in the l th training example

45

Summary of Logistic Regression

- Learns the Conditional Probability Distribution P(y|x)
- Local Search.
  - Begins with initial weight vector.
  - Modifies it iteratively to maximize an objective function.
  - The objective function is the conditional log likelihood of the data – so the algorithm seeks the probability distribution P(y|x) that is most likely given the data.

46

What you should know LR

- Advantages:
  - Makes no assumptions about distributions of classes in feature space
  - Easily extended to multiple classes (multinomial regression)
  - Natural probabilistic view of class predictions
  - Quick to train
  - Very fast at classifying unknown records
  - Good accuracy for many simple data sets
  - Resistant to overfitting
  - Can interpret model coefficients as indicators of feature importance
- Disadvantages:
  - Linear decision boundary

47

Logistic regression application with python

- We apply regression on the digit database
- We use linear_model.LogisticRegression()

http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits