CS109a Midterm Review

ofanning

October 2024

Contents

1 Lecture 1: Intro to Data Science

Contents

1.1 The Potential of Data Science

Positive effects

- Disease diagnosis
- Generative AI
- Drug discovery
- Transportation

Negative things

- Gender bias
- Racial bias

1.2 What is Data Science?

The DS process

1. Ask an interesting questino
2. Get the data
3. Explore the data
4. Model the data
5. communicate/Visualize the REsults

2 Lecture 2: Data and Visualization

Contents

2.1 What are Data?

Ways to gather online data

- API (Application Programming Interface): using a prebuilt set of functions developed by a company to access their services. Often pay to use. For example: Google Map API
- RSS (Rich Site Summary): summarizes frequently updated online content in standard format. Free if site has one. For example: news-related sites, blogs
- Webs Scraping: using software, scripts or by-hand extracting data from what is displayed on a page or what is contained in the HTML file (often in tables)

Types of data representation and storage

- Tabular Data. Each type of measurement is a column (usually) and is called a variable, attribute, or feature. Number of attributes is the dimension.
- Structure Data: each data record is presented in a form of a [possibly complex and multi=-tiered] dictionary (json, xml, etc.)
- Semistructured Data: not all records are represented by the same set of keys or some data recrods are not represented using hte key-value pair structure

Classes of variables/attributes

- Quantitative variable: numerical, either
  - discrete: a finite number of values are possible in a bounded interval. For ex: "number of siblings"
  - continuous: an infinite number of a values are possible in any bounded interval. for ex: "height"
- categorical variable: no inherent order among the values. for ex: "what kind of pet do you have?" gets a categorical variable in response

2.2 Exploratory Data Analysis (EDA)

Sampling basics AKA descriptive statistics

- Population: is the entire set of objects or events under study. Can be hypothetical or literal
- Sample: "representative" subset of the objects or events under study. It's impossible to obtain/compute using population data
- Selection bias: some subjects or records are more likely to be selected
- Volunteer/nonresponse bias: subjects or records who are not easily available are not represented
- Mean: of a set of n observations of a variable is denoted as

$$\overline{x} = \frac{x_1 + x_2 + ... + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  - the mean is sensitive to outliers, which results in skewness
  - Algorithmic complexity is at most $O(n)$
- Median: of a set of n number of observations in a sample, ordered by value, of a variable is defined by

$$\text{Median} = \begin{cases} x_{(n+1)/2} & \text{if } n \text{ is odd} \\ \frac{x_{n/2} + x_{(n+1)/2}}{2} & \text{if } n \text{ is even} \end{cases}$$

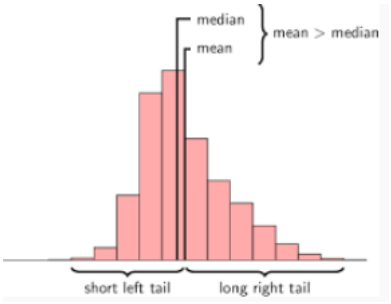

Figure 1: right-skewed distribution (mean greater than median)

  - Algorithmic complexity is at most $O(n \log n)$

- Measures of spread: the spread of a sample of observatoins measures how well the mean or median describes the sample
  - Range = Maximum value - Minimum Value
  - Variance: measures how much on the average the sample value deviates from the mean, defined as

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} |x_i - \overline{x}|^2$$

    note: s^2 is sensitive to extreme outliers and doesn't have the same units as x_i
  - Standard Deviation: is the square root of the variance

$$s = \sqrt{s^2} = \sqrt{s^2 = \frac{1}{n-1} \sum_{i=1}^{n} |x_i - \overline{x}|^2}$$

    note: s now has the same units as x_i

Basic visualizations during EDA. Types of things visualizations show

- Distribution: how a variable or variables in the dataset distribute over a range of possible values
- Relationship: how the values of multiple variable in the dataset relate
- Composition: how the dataset breaks down into subgroups
- Comparison: how trends in multiple variable or datasets compare

Common visualization types include

- Histogram: a way to visualize how 1-demensional data is distributed across certain values
- Bar Plot: a way to visualize the composiiton (aka, distribution) of a categorical variable
- Pie Chart: way to visualize the static compositoin(aka, distribution) of a variable. Pie charts are frowned upon (VIDEO CHECK???)
- Scatter Plot: way to visualize the relationship between two different attributes of multi-dimensional data
- Stacked Area Graph: way to visualize the composition of a group as it changes over time
- Multiple Histograms: on the same axes are a way to visualize how different variables compare (or how a variable differs over specific groups)
- Boxplot: simplified visualization to compare a quantitative variable across groups. Hightlights the range, quartiles, median, and any outliers present in a data set

2.3 Effective Visualizations

Edward Tufte's Principles of Graphical Excellence. Graphical Excellence...

- is the well-designed presentation of interesting data— a matter of substance, of statistics, and of design
- consists of complex ideas communicated with clarify, precision and effeiciency
- is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space
- is nearly always multivariate
- requires telling the truth

3 Lecture 3: Pandas (lab-like)

Contents

4 Lecture 4: Regression

Contents

4.1 Statistical Modeling and k-Nearest Neighbors (kNN)

We generally classify variables in a dataset into two main categories

- Predictor Variables: variables used as inputs to inform or prediction. AKA features, covariates, independent variables
- Response Variable(s): variables whose values we aim to predict. AKA outcome, response variable, dependent variable

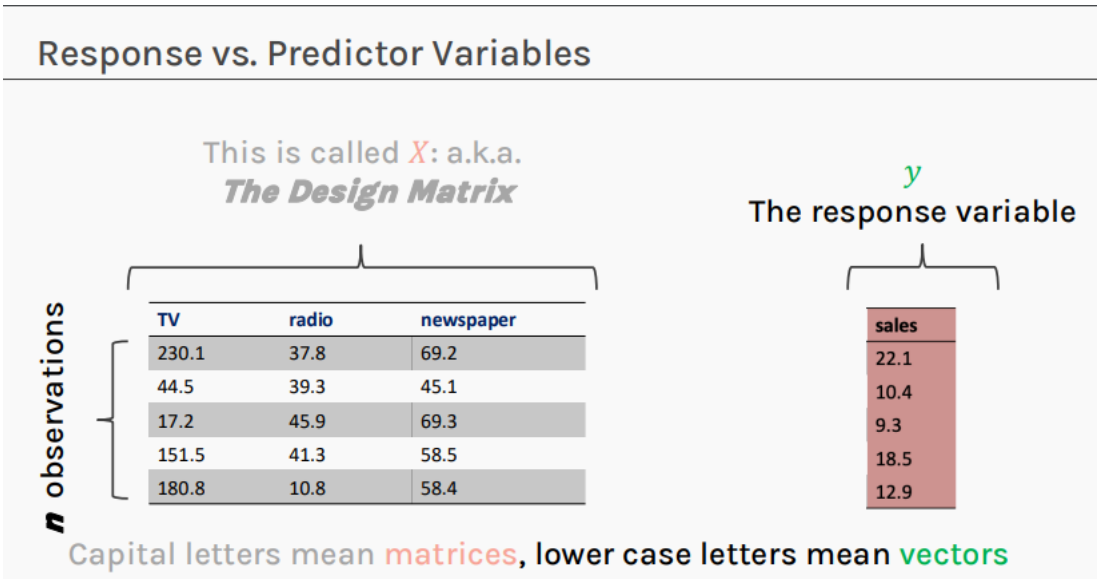

Figure 2:

We use statistical models instead of trying to capture the exactly true model, which is intractable. We assume the relationship

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

where f represents the unknown underlying rule for relating Y and X. ϵ is the random amount this relation varies from the rule. A statistical model is any algorithm used to estimate f, where the estimated function is $\hat{f}$.

There are two main types in statistical modeling

- Inference Problems: objective is to understand the form and characteristics of $\hat{f}$
- Prediction Problems: objective is to minimize the difference between $\hat{y}$ and y. An example of this is K-Nearest Neighbor algo

kNN is a very human way of decision making by similar examples. Non-parametric learning algorithm. Given a dataset $D = (x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})$, for every new X where you want to predict a new Y:

1. Find the k-number of observations in D most similar to X (typically by distance along the predictor variable you are considering):
$$(x^{(n_1)}, y^{(n_2)}), ..., (x^{(n_k)}, y^{(n_k)})$$
2. Average the output of those k-nearest neighbors of x
$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} y^{n_k}$$

4.2 Error Evaluation and Model Comparison

To determine which model is "best," we need a measure. To do this, we first perform the train-test split. The test set data is used to train our model to estimate $\hat{4}$, and the test set data is used to evaluate the model's performance. We use the points in the test set to calculate the residual (or error) at each observation by taking $r_i = |y_i - \hat{y}_i|$. Our loss function is then defined by aggregating those errors to get the Mean Square Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

note: cost usually refers to the total loss, whereas loss refers to a single training point.

Other cost functions are sometimes used, such as

- Max Absolute Error
- Mean Absolute Error
- Huber Loss
- Root Mean Square Errors (RMSE)

We notice that if we change the units of the y axis of of our model, we can dramatically alter the value of the MSE (think of Sales vs TV Budget, where Sales can be in units of single sales or units of 1000s). This implies we need a more meaningful benchmark to compare between models that isn't affected by the scale of the data. The better benchmark is R-squared. We use two reference models for comparison:

1. The simplest model, often considered the worst possible, where the predicted value $\hat{y}$ is the mean of all observations:
$$\hat{y} = \overline{y} = \frac{1}{n}\sum_i y_i \tag{1}$$
2. The ideal or best possible model, where the predicted value $\hat{y}$ is identical to the actual value y

With these we define the R^2 as

$$R^2 = 1 - \frac{\sum_i(\hat{y}_i - y_i)^2}{\sum_i(\overline{y}_i - y_i)^2}$$

- If our model is only as good as the mean value, $\overline{y}$, then $R^2 = 0$
- If our model is perfect, then $R^2 = 1$
- R^2 can be negative if the model is worse than the average, this can happen when we evaluate the model in the testset

5 Lecture 5: Simple and Multiple Linear Regression

Contents

5.1 Simple Linear Regression

We can build a model by first assuming a simple form of f as

$$f(x) = \beta_0 + \beta_1 X$$

It follows that our estimate is

$$\hat{Y} = \hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are estimates that we compute using observations. We perform this estimate by again using the MSE as our loss function s.t.

$$L(\beta_0, \beta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

and we choose β_1 and β_0 so they minimize the predictive errors made by our model, i.e., minimize our loss function, s.t.

$$\hat{\beta}_0, \hat{\beta}_1 = \arg\min_{\beta_0,\beta_1} L(\beta_0, \beta_1)$$

The global minima or maxima of $L(\beta_0, \beta_1)$ must occur at a point where the gradient is:

$$\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1}\right] = 0$$

In the case of linear regression, this has an analytical solution, which is rare. The solution is

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sum_i (x_i - \overline{x})^2}$$
$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

where $\overline{y}$ and $\overline{x}$ are sample means. The regression line is given by

$$\hat{Y} = \hat{\beta}_1 X + \hat{\beta}_0$$

5.2 Multi-linear Regression

In multiple linear regression we assume a multi-linear form for f

$$f(x_1, ...x_p) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

Essentially, we are assuming that the response variable, $y = y_1, ..., y_n$ is now dependent on many predictors, so the design matrix is now $X = x_1, ..., x_p$. In linear algebra form

$$Y = X\beta + \epsilon$$

We again choose the MSE as our loss function, which in vector notation is

$$MSE(\beta) = \frac{1}{n}\|Y - X\beta\|_2^2$$

For a two predictor problem, this would be

$$MSE(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$

Doing a similar MSE minimization as for linear, we get to the closed form solution

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

5.3 Interpreting Model Parameters

When we have a large number of predictors, there will be a large number of model parameters. Looking at the values of the β's is impractical, so we visualize the with a feature importance graph, which shows which predictors have the most impact on the model prediction.

5.4 Scaling

Standardization (Z-Score): transforms data to have mean = 0 and standard deviation = 1.

$$\frac{X - mean}{std}$$

- more helpful if data also follows a Gaussian (AKA Normal, which is confusing) distribution
- less affected by outliers, as there is no predefined range of features

Normalization (Min-Max Scaling): rescales data to range between 0 and 1.

$$\frac{X - mean}{max - min}$$

- recommended for data that already has fixed range
- or data that do not already follow Gaussian distribution
- more vulnerable to outliers

MAYBE ADD THE QUESTIONS FROM LECTURE 5 NOTES:SCALING

5.5 Collinearity

Collinearity refers to a situation where two or more predictors in a regression model are highly correlated. It does not violate the assumptions of linear regression, but can make it difficult to determine the individual effect of each predictor on the response. It also affects our confidence in the estimated coefficients.

5.6 Qualitative Predictors

To deal with qualitative predictors, we one hot encode by creating dummy variables. For example:

$$x_{i,1} = \begin{cases} 1 & \text{if } i\text{th person is Asian} \\ 0 & \text{if } i\text{th person is not Asian} \end{cases}$$

$$x_{i,2} = \begin{cases} 1 & \text{if } i\text{th person is Caucasian} \\ 0 & \text{if } i\text{th person is not Caucasian} \end{cases}$$

Using these new variables as predictors, the regression then becomes

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} = \begin{cases} \beta_0 + \beta_1 & \text{if } i\text{th person is Asian} \\ \beta_0 + \beta_2 & \text{if } i\text{th person is Caucasian} \\ \beta_0 & \text{if } i\text{th person is African American} \end{cases}$$

Linear Regression Assumptions (so far):

- linear relationship between X and Y
- the residuals $r_i = y_i - \hat{y}_i$ are uncorrelated

We should check these assumptions according to Figure 3

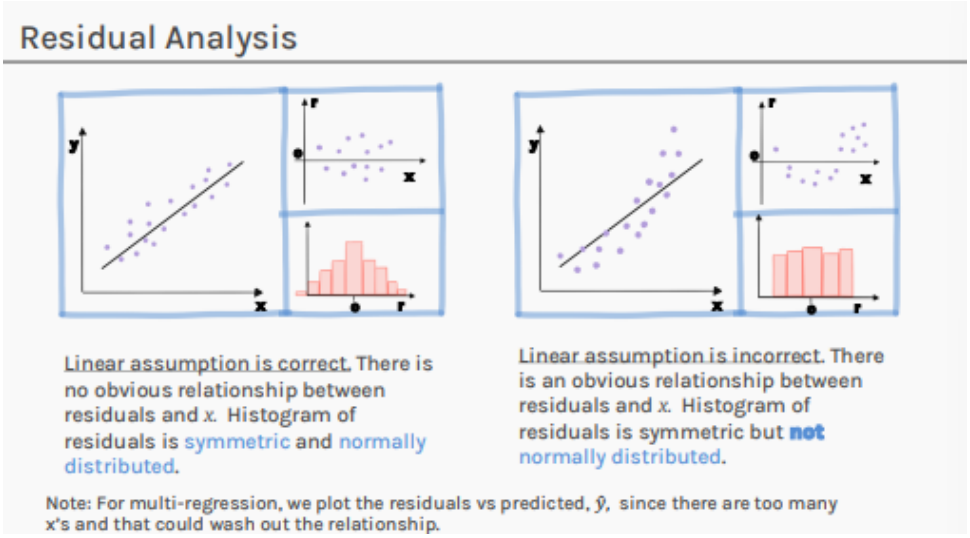
Residual Analysis
y
x
r
x
r
Linear assumption is correct. There is no obvious relationship between residuals and x. Histogram of residuals is symmetric and normally distributed.
y
x
r
x
r
Linear assumption is incorrect. There is an obvious relationship between residuals and x. Histogram of residuals is symmetric but not normally distributed.
Note: For multi-regression, we plot the residuals vs predicted, ŷ, since there are too many x's and that could wash out the relationship.

Figure 3:

6 Lecture 6: Polynomial Regression, Model Selection, and Cross Validation

Contents

6.1 Interaction Effects in Regression Models

Assumptions of Linear Regression:

- Linearity: Relationship between variables is linear

$$f(x) = \beta_0 + \beta_1 x$$

- Independence: No correlation between errors and predictors
- Homoscedasticity: Constant variance of residuals
- Normality of Residuals: Residuals are normally distributed

$$y = f(x) + \epsilon$$
$$L(\beta_0, \beta_1) = MSE$$

- Other things to consider:
  - Fixed X: Independent variables are error-free
  - No Multicollinearity: Low correlation between predictors

Interaction or Synergy Effect: when an increase on the radio budget affects the effectiveness of the TV spending on sales. We model this by adding an extra term

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

Too many predictors, collinearity, and too many interaction terms leads to OVERFITTING!

6.2 Polynomial Regression: Extending Linear Models

The simplest non-linear model we can consider, for a response Y and a predictor x, is a polynomial model of degree M

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_M x^m$$

This is just a multi-linear regression where the predictors in the design matrix are powers of x, so we can still solve via

$$\hat{\beta} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

NOTE: sklearn will include the interaction terms and the new design matrix will include the 1 column automatically so no need to fit for intercept.

Linear regression is invariant under scaling. If X multiplied by some number λ, then β will be scaled by $\frac{1}{\lambda}$ and the MSE will remain the same.

However, small or large numbers to high degrees can cause issues, so its a good idea to scale X when considering polynomial regression. CONFIRM THAT SHOULD STANDARDIZE RATHER THAN NORMALIZE.

Too many predictors, collinearity, too many interaction terms and high degree of polynomial leads to OVERFITTING!.

6.3 Model Selection Techniques: Focus on Cross-Validation

Model selection is the application of a principled method to determine the complexity of the model, e.g., choosing a subset of predictors, choosing the degree of the polynomial model, etc. The goal of model selection is to choose the model that generalizes the best. There are four ways to do model selection

- Exhaustive search: with J predictors and only linear terms, we have 2^J models, which is intractable.
- Greedy algorithms: stepwise variable selection, forward method. $O(J^2)$, which is faster than greed 2^J for large J
    1. Start with no predictors, make null model M_0
    2. Using the best model so far, pick one new predictor that improves the model the most according to validation set MSE
    3. Make that new model. Repeat this for all predictors
    4. Select the model from the set of models created through the above process with the lowest validation MSE
- Fine tuning hyper-parameters: we recast model selection as choosing hyper-parameters. For example, polynomial regression requires choosing a degree - so we select the model by "tuning" (choosing) the degree hyper-parameter
- Regularization: focus of Lecture 7

Using a single validation set creates possibility of overfitting to the validation set. To combat this, we do some form of cross validation

- K-Fold Cross Validation: split the data into K uniformly sized chunks. Gives K train/val splits, each using one chunk for val and the rest for train. We fit the model on each training set, giving us $\hat{f}_{C_{-i}}$, and eval it on the corresponding val set, $\hat{f}_{C_{-i}}(C_i)$. The cross val score is then the average performance of the model across all val sets, where L is a loss function

$$CV(Model) = \frac{1}{K}\sum_{i=1}^{n}(L(\hat{f}_{C_{-i}}(C_i))$$

- Leave-One-Out Cross Validation: K now equals the number of data points, N, in the dataset. Each data point serves as its own test set, while the model is trained on all other $N-1$ points.

$$CV(Model) = \frac{1}{K}\sum_{i=1}^{n}(L(\hat{f}_{X_{-i}}(X_i))$$

7 Lecture 7: Regularization: Ridge and LASSO

Contents

7.1 Generalization Error, Bias Variance Tradeoff

When models do well on new data (typically test data), it is called generalization. There are at least three ways a model can have high-test error.

1. Noise: irreducible or aleatoric error
2. Underfitting: reducible or epistemic error. Fixed through model selection
3. Overfitting: reducible or epistemic error. Fixed through model selection

There is a tradeoff between increasing complexity/overfitting and decreasing complexity/underfitting, referred to as the bias variance tradeoff. This is explained well visually by Figures4 and 5k

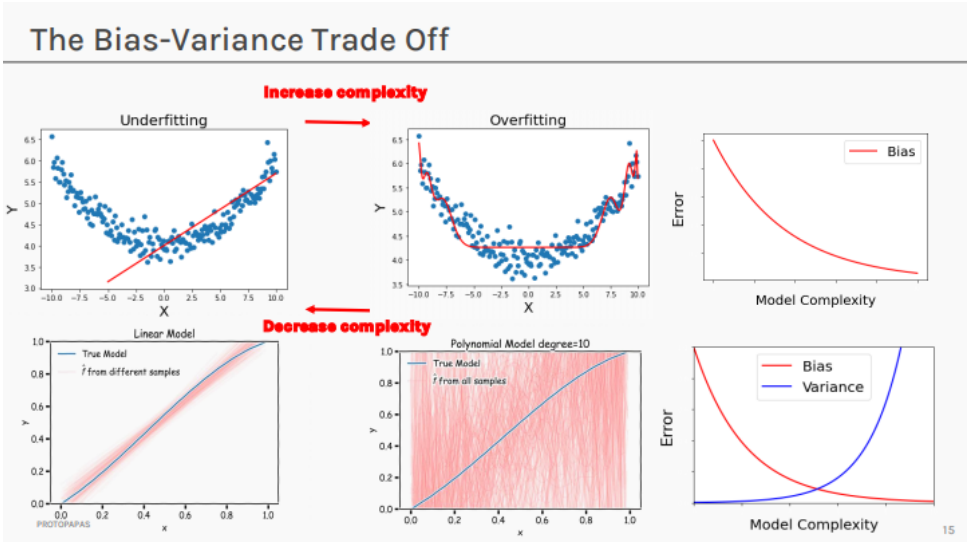

Figure 4:

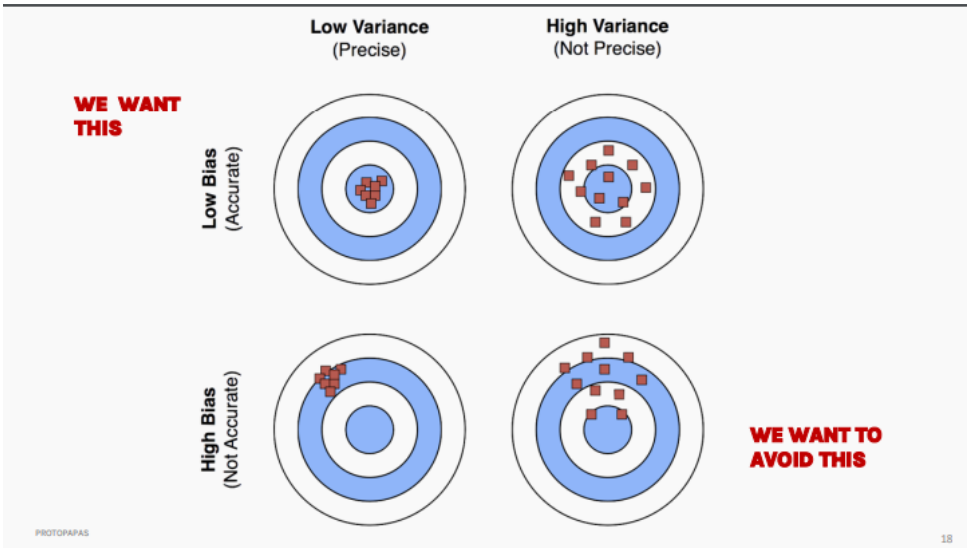

Figure 5:

7.2 Regularization Techniques: Lasso, Ridge

Regularization is a way of combating coefficient values that are too extreme and would cause overfitting. The two methods we learn are LASSO and Ridge.

Lasso regression: minimize $\mathcal{L}_{LASSO}$ w.r.t β's

$$\mathcal{L}_{LASSO} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \beta^T x_i\right|^2 + \lambda\sum_{j=1}^{J}|\beta_j|$$

- No analytical solution, has to be solved with solver
- Algo is essentially same as ridge
- Totally zeros out some coefficients, resulting in easier to interpret models. "Automatic feature selection"

Ridge regression: minimize $\mathcal{L}_{RIDGE}$ W.R.T β's

$$\mathcal{L}_{LASSO} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \beta^T x_i\right|^2 + \lambda\sum_{j=1}^{J}\beta_j^2$$

- Analytical solution for the coefficients

$$\hat{\beta}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y$$

- Basic algorithm is to try a set of λ's and select the one that minimizes the MSE loss on the validation set, then retrain with train and val and report those scores

8 Lecture 8: Probability in Regression/MLE

Contents

9 Lecture 9: Inference in Regression: Bootstrap and CI, Hypothesis Testing

Contents

10 Lecture 10: Principal Component Analysis

Contents

11 Lecture 11: Missingness

Contents

12 Lecture 13: Classification and Logistic Regression Basics

Contents

12.1 What is Classification?

Regression performs well on tasks that require the prediction of a quantitative response variable, i.e. What is the temperature going to be tomorrow? Classification performs well on tasks that require the prediction of a categorical or qualitative response variable. It classifies an observation into a category or class labeled by Y, i.e. Is it going to be hot or cold tomorrow?

12.2 Why not Linear Regression?

If you encode a bunch of categories as numbers (1,2,3, etc.) than the model implies a specific ordering of the outcomes, so changing the order of the encoding would alter the predictions, which doesn't make sense. If the categorical response variable is ordinal, then a linear regression could maybe make sense, but not ideal.

12.3 Estimating the Simple Logistic Model

For a simple two category case encoded with 0 and 1, we could use linear regression to predict $P(Y = 1)$, and if its above some boundary, we classify. However, linear regression can return above 0 and 1 values, which doesn't make sense for probabilities, so we switch to Logistic Regression, where

$$P(Y = 1) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} = \frac{1}{1 + e^{-(\beta_0+\beta_1 X)}}$$

This can be rewritten as follows

$$\ln\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \beta_0 + \beta_1 X$$

Where the part inside the log function are the odds. Thus, a one unit change in X causes a β_1 change in the log-odds of $P(Y = 1)$.

NOTE ON Probability vs. Likelihood

- **Probability:** If we assume we know the underlying parameters of a model, probability gives us the likelihood of seeing a certain outcome. For example, if we know the probability of heads for a coin is 0.5, then the probability of seeing 5 heads in 10 flips can be computed directly.
- **Likelihood:** If we have observed data (e.g., we saw 5 heads in 10 flips) and we want to find which parameter value (e.g., probability of heads) makes this observed data most plausible, we consider the likelihood. Here, the data is known, and the parameter is the unknown we are trying to understand.
- **Mathematical Formulation:** Suppose we have data points $x_1, x_2, \ldots, x_n$ and a model that depends on some parameter vector θ. The probability of observing this specific data, assuming a particular θ, is given by the probability density or mass function:

  $$p(x_1, x_2, \ldots, x_n \mid \theta).$$

  When viewed as a function of θ for fixed data, this is called the **likelihood function**:

  $$L(\theta) = p(x_1, x_2, \ldots, x_n \mid \theta).$$

  The goal in many statistical inference problems is to choose the parameter θ that maximizes this likelihood.

In simple logistic model, likelihood of a single observation for p given x and true label y is

$$L(p_i|Y_i) = P(Y_i = y_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

Assuming observed data points are independent, the total likelihood is then

$$L(p_i|Y_i) = \prod_i P(Y_i = y_i) = \prod_i p_i^{y_i}(1-p_i)^{1-y_i}$$

The best model is model that maximizes likelihood that the parameters explain the data, thus

$$\arg\max_\beta(L(p|y)) = \arg\max_\beta(\log L(p|y))$$

Expanding the argument and using log rules...

$$\begin{aligned}\log L(p|Y) = \log \prod_i p_i^{y_i}(1-p_i)^{1-y_i} &= \sum_i \log\{p_i^{y_i}(1-p_i)^{1-y_i}\} \\ &= \sum_i \log p_i^{y_i} + \log(1-p_i)^{1-y_i} \\ &= \sum_i y_i \log p_i + (1-y_i)\log(1-p_i)\end{aligned}$$

Since maximizing a function is equivalent to minimizing the negative...

$$\beta^* = \arg\min_\beta(-\log L(p|Y)) = -\sum_i y_i \log p_i + (1-y_i)\log(1-p_i)$$

So the final loss function, called the Binary Cross Entropy, is

$$L_{BCE} = -\sum_i y_i \log p_i + (1-y_i)\log(1-p_i)$$

12.4 Inference in Logistic Regression

The uncertainty of the estimates $\hat{\beta}_0$ and $\hat{\beta}_0$ can be quantified and used to calculate confidence intervals and hypothesis tests. Bootstrapping and permutation testing (hypothesis testing) are methods to perform these inferences

12.5 Multiple Logistic Regression

Just like with linear regression, logistic regression extends to multiple predictors (case above was just one). Interactions can still be done, multicollinearity is a concern, and so is overfitting, we still solve using regularization and cross-validation.

$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p$$

The coefficient $e^{\hat{\beta}_j}$ is the multiplicative change in odds between the j^{th} predictor and the response, holding constant the influence of the other predictors in the model. Multicollinearity matters for this just like in linear regression.

12.6 Classification Decision Boundaries

In the classic case, we classify $\hat{P}(Y=1) \geq 0.5$ as the positive case and vice versa. This means that the decision boundary is defined when $X\beta = 0$. However, the decision boundary doesn't have to look like a straight line, it can also be almost any shape depending on what interaction terms and polynomial features are added. For example: we could choose the following set of predictors if we wanted a circular decision boundary

$$X = \{X_1, X_1^2, X_2, X_2^2, X_1 X_2\}$$

13 Lecture 14: Logistic Regression 2

Contents

13.1 Interpreting Interactions in Logistic Regression

Just be careful in making sure to realize which interaction terms go away when certain predictors are 1 or 0. The interaction term is basically the correction in the intercept and the slope to the original curve. Think of the heart disease example:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1(\text{age}) + \beta_2(\text{female}) + \beta_3(\text{females * age})$$

Assuming females are 1 and males are 0, the respective models are

- for males:
$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1(\text{age})$$
- for females:
$$\ln\left(\frac{p}{1-p}\right) = (\beta_0 + \beta_2) + (\beta_1 + \beta_2)(\text{age})$$

13.2 Regularization in Logistic Regression

Recall that maximizing the log-likelihood is equivalent to minimizing the binary cross-entropy loss function. We can add a regularization term to penalize large values of parameters. The parameter is chosen through CV

$$\arg\min_{\beta}\left[-\sum\left(y_i \log p_i + (1-y_i)\log(1-p_i)\right) + \lambda\sum_{j=1}^{p}\beta_j^2\right]$$

NOTE: SKLEARN USES $C = 1/\lambda$

13.3 Multiclass Logistic Regression

The two most common extensions to logistic regression when the response variable Y has more than 2 categories are:

1. Nominal: categories have no inherent order
2. Ordinal logistic Regression: categories have a specific hierarchy/order

We still maximize the log-likelihood, which gives us the "multinomial logistic loss"

$$\ell = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\left[\mathbb{1}(y_i = k)\ln(\hat{P}(y_i = k)) + \mathbb{1}(y_i \neq k)\ln(1 - \hat{P}(y_i = k))\right]$$

13.3.1 Multinomial Logistic Regression

Multinomial Logistic Regression is for nominal categorical varibles. For K classes, you basically choose a reference category and then do a binary logistic regression between each of the other k-1 categories and the reference category.

NOTE: the traditional implementation implies $K-1$ sets of parameters for K classes, with the K^{th} just being inferred from the others, but sklearn directly computes k sets of parameters for k classes, which could result in different sets of coefficients.

13.3.2 One-vs-Rest Logistic Regression

OVR turns K class problem into K binary logistic regressions, each class vs all others.
MAYBE ADD PICTURES?
NOTE: sklearn normalizes the output of each of the k models when predicting probabilities

13.4 Bayes Theorem and Misclassification Rates

Definition of conditional probability gives us

$$P(B|A) = \frac{P(A \text{ and } B)}{P(A)}$$

And since $P(A \text{ and } B) = P(A|B)P(B)$, we get Bayes' Theorem

$$\begin{aligned} P(B|A) &= \frac{P(A|B)P(B)}{P(A)} \\ &= \frac{P(A|B)P(B)}{P(A|B)P(B) + P(A|B^C)P(B^C)} \end{aligned}$$

Best example of the effect of this is Diagnostic Testing, see Figures 6

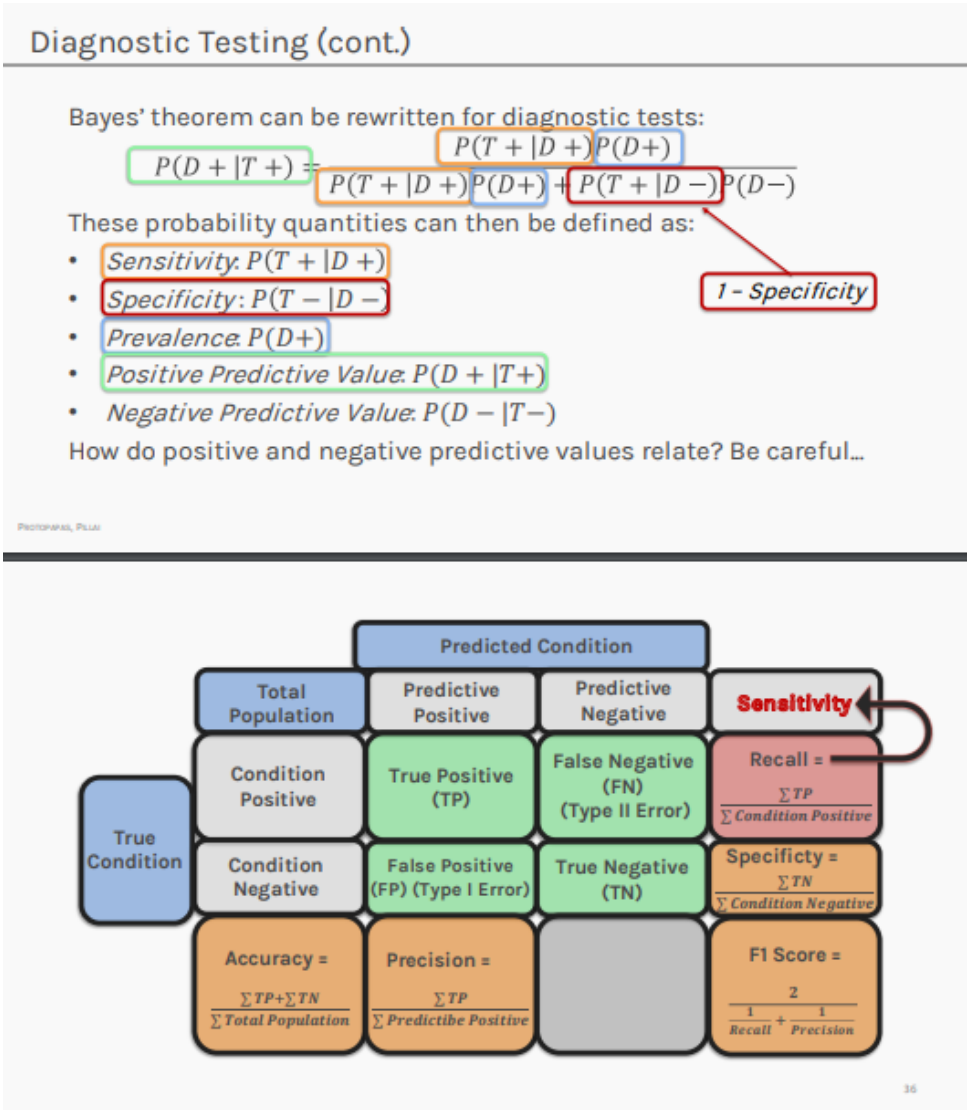

Figure 6:

The results of a classification algorithm are either summarized in

1. Confusion table ($k \times k$). Example in Figure 7. We can tune the results of the table for accuracy in a specific category by altering the decision boundary
2. ROC Curve. Next subsection.

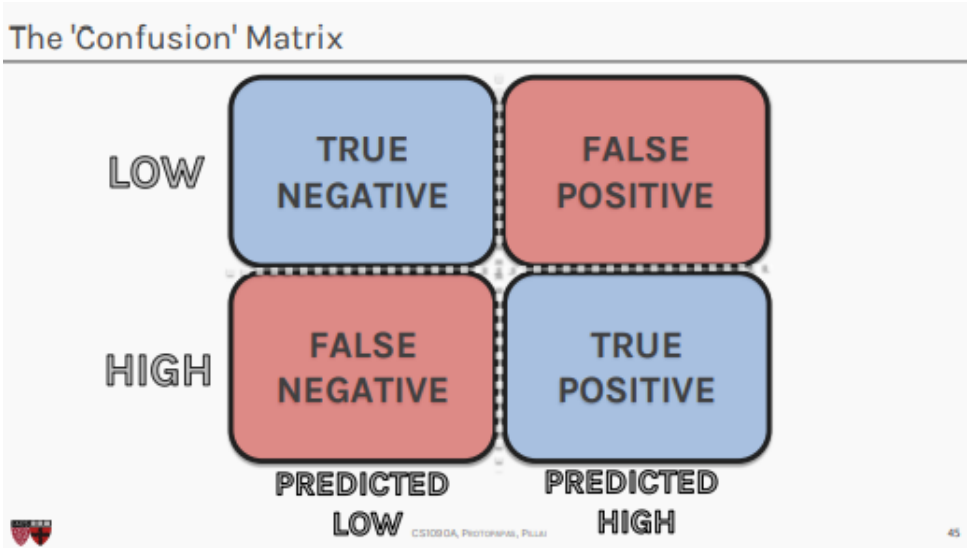

Figure 7: Caption

13.5 ROC Curves

Radio Operator Characteristics curve shows the trade-off between true positive rate and false positive rate across all thresholds, as shown in Figure 8. The overall performance of a classifier, calculated across all thresholds, is given by the area under the ROC curve, or AUC.

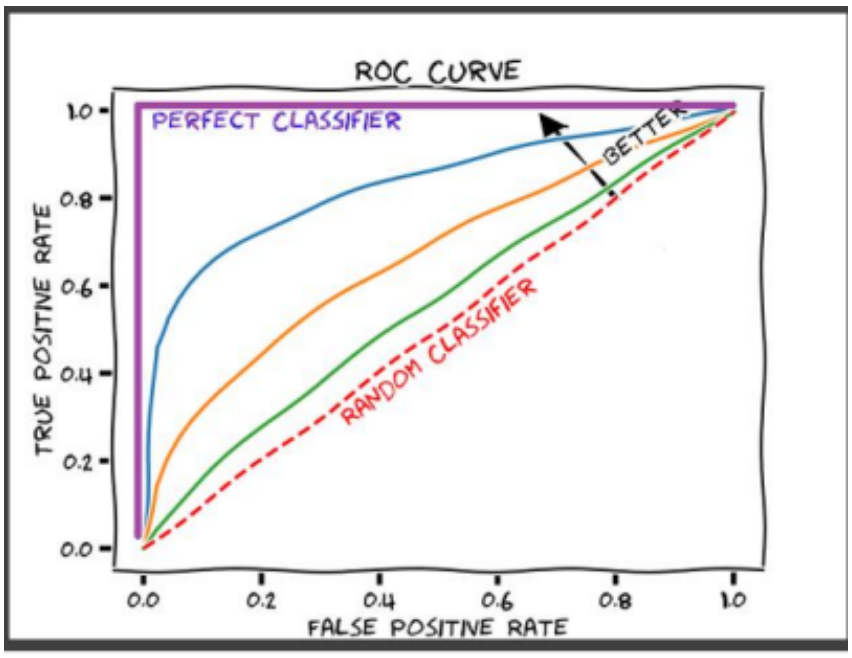

Figure 8:

14 Lecture 15: Causal Inference

Contents

If a β coefficient is significantly different from 0, then evidence than the response is associated with the predictor. But what about causal association? There is potential for confounding factors which actually cause the association. The fixes for this are either

1. Model all possible confounder by putting them all in the model. Or use some other sophisticated method to account for them
   - Advantage: cheap
   - Disadvantage: probably didn't measure every possible confounder
2. Randomized experiment where scientist manipulates the levels of the predictor (treatment) to see how this leads to changes in the response
   - Advantage: confounders will be balanced on average across treatment groups
   - Disadvantage: expensive, can be too artificial of an environment

14.1 Simpson's Paradox

Simpson's Paradox is when an observed trend between two variables is different at a group level than just looked at the aggregate data. It happens because the initial groups you're comparing have different sizes and are influenced by a third, confounding variable.

14.2 Causal Structure

Think about the causal structure diagrams. Does treatment affect condition? Or vice versa? And which affect the outcome on the target?

14.3 Correlation and causation

Example: Sleeping fully dressed correlates with waking up with headaches. Common cause: drinking the night before.
Association vs Correlation: association is a statistical dependence and correlation a linear statistical dependence (aka, direction), but in practice we use them interchangeably.

14.4 Causal Effects

The notation for causal inference is shown in Figure 9. The fundamental problem is that you can only observe either the factual or counterfactual, not both, thus this becomes a missing data problem.

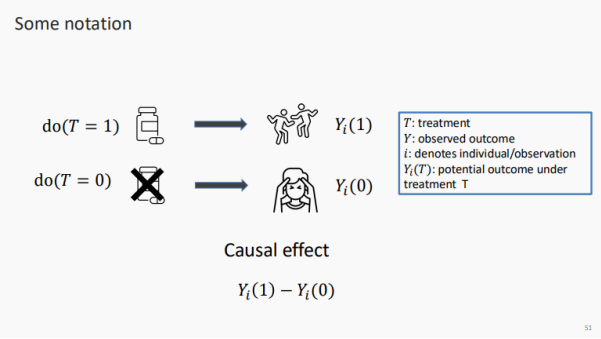

Figure 9: Caption

If you somehow new the individual treatment effect (ITE)

$$Y_i(1) - Y_i(0)$$

Then it would make sense to know the average treatment effect (ATE) as

$$\mathbb{E}[Y_i(1) - Y_i(0)] = \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)]$$

But it is a misconception that this is always equivalent to the following. This is Simpson's Paradox!

$$\mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]$$

But that can be done when you have a randomized experiment!

14.5 Randomized Control

Randomized experiment is the gold standard! By randomizing those who receive the treatment and those who do not, the two groups should on average have the same background characteristics, thus eliminating the effect of confounders and making the following true

$$\mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] = \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]$$

15 Lecture 16: Causal Inference II

Contents

15.1 AB Testing (Randomized Control Trials)

Is an example of an RCT. Think of google color example or Obama button example.

15.2 Adjusting for Confounders

If you know the confounders you can use conditional probability to account. This is the most simple way. Example calculation shown in Figure 10. The final numbers are the confounder adjusted treatment effects

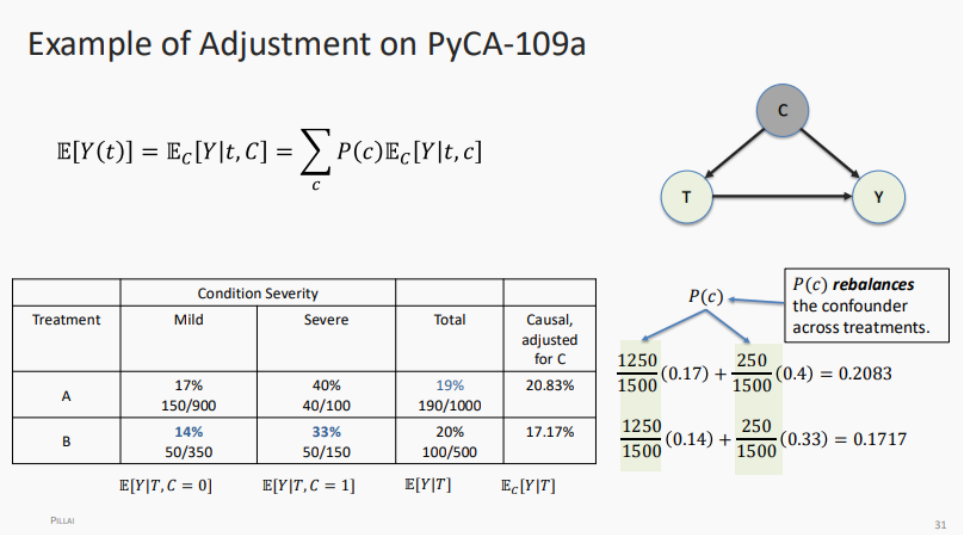

Figure 10:

15.3 Propensity Scores

What is no RCT is possible, probably for ethical reasons (not give some kids education)? This means we can't use our ATE calculation from above. We are now doing "Observational Studies" based off of data that we weren't part of the collection of.

Propensity score addresses this by mapping all background features of a person to a single score. Then people from T group can be matched with someone in C group, and a new group of matches is made where the ATE calculation is now valid again.
Propensity score: the probability that a subject/observation (ex: user, person, classroom, etc.) is assigned to a particular treatment, T_i, given their set of observed predictors, X_i.

$$e(X_i) = P(T_i = 1|X_i)$$

Standard approach is to fit a logistic regression with the treatment being the outcome, and all the background features you have being the data. This is still not as good as RCT because its impossible for us to have been given data for every possible background feature.

16 Lecture 17: Decision Trees - Classification

Contents

16.1 Motivation

Logistic Regression works well when the classes are well-separated in the feature space and the classification boundary has a nice geometry. We can use nonlinear decision boundaries, but they are hard to interpret. We want models

- Complex decision boundaries
- Easily interpretable
- Efficient to compute

Which leads us to the Decision Tree!

16.2 Intuition

Every flow chart node corresponds to a partition of the feature space by lines parallel to the axis or (hyper) planes, as seen in Figure 11

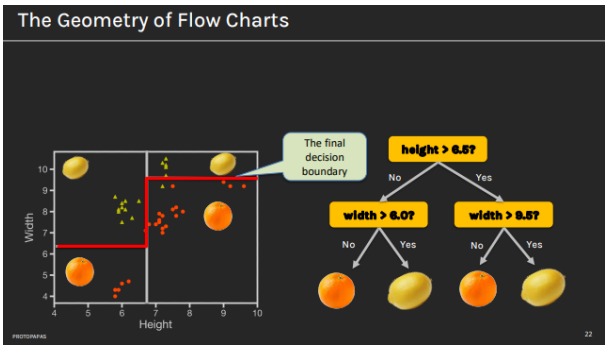

Figure 11:

16.3 Predictions

How to predict with (or traverse) a tree, see Figure 12

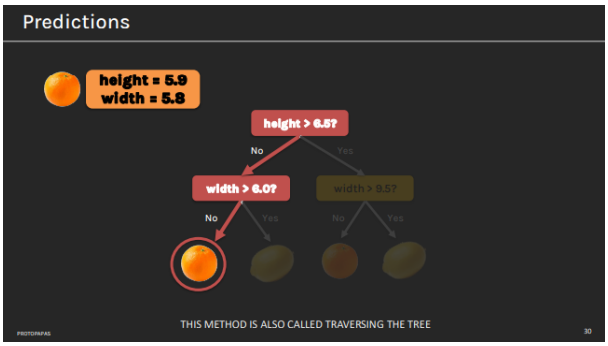

Figure 12:

16.4 Splitting Criteria

There is no correct way to define an optimal split, but they should follow these rules

1. Regions should get progressively purer with more splits
2. No empty regions, every region must have training points

16.4.1 Classification Error

Assume P predictors and K classes. Suppose we select the p^{th} predictor and split a region along the threshold t_p. We can assess the quality of this specific split by considering the class error in each created region, where $\max_k(\Psi(k|R_r))$ is the proportion of training points in R_r that are labeled class k

$$Error(R_r|p,t_p) = 1 - \max_k(\Psi(k|R_r))$$

However, we need to take the weighted average over both regions created from the split t_p so that the number of points in each region is taken into account. Take N_r as the number of training points inside region R_r

$$\min_{p,t_p}\left[\frac{N_1}{N}Error(R_1|p,t_p) + \frac{N_2}{N}Error(R_2|p,t_p\right]$$

16.4.2 Gini Index

Gini index is the same as class error, but now taking the sum of squares of the proportions of points from the k^{th} class in the region r (instead of the max of non squared). This accentuates the contrast betwen pure regions and mixed ones

$$Gini(R_r|p,t_p) = 1 - \sum_K \Psi(k|R_r)^2$$

We can now try to find the predictor p and the threshold t_p that minimizes the weighted average Gini Index over the two regions, just like above

$$\min_{p,t_p}\left[\frac{N_1}{N}Gini(R_1|p,t_p) + \frac{N_2}{N}Gini(R_2|p,t_p)\right]$$

16.4.3 Entropy

A third way is entropy, which quantifies the strength of a signal in a particular region. Entropy quantifies the uncertainty or randomness in a region's class distribution, by minimizing entropy we create splits that maximize information gain (lower uncertainty of labels). For the same assumed situation as above, we can compute the conditional entropy of the distribution of training points amongst the K classes given that a point is in region r as

$$Entropy(R_r|p,t_p) = -\sum_k \Psi(k|R_r)\log_2\Psi(k|R_r)$$

And the weighted analog is

$$\min_{p,t_p}\left[\frac{N_1}{N}Entropy(R_1|p,t_p) + \frac{N_2}{N}Entropy(R_2|p,t_p)\right]$$

Entropy penalizes impurity the most of these metrics!

16.5 Stopping Conditions

Learning the optimal decision tree for a given data set is intractable, so we use a greedy algorithm as follows

1. start with empty tree
2. choose the 'optimal' predictor on which to split and optimal split value
3. recurse on each new node until the stopping condition is met
4. for the case of classification, predict each region to have a class label based on the largest class of training points in that region (majority class)

1em
If we don't limit the growth of the tree, each leaf node will just end up with 1 data point and 100% train accuracy. Some potential stopping conditions are

- `max_depth`: limit maximum depth
- don't split if all leaf nodes are pure
- `min_samples_leaf`: define the minimum number of samples for any node
- `max_leaf_nodes`: define the max total number of leaves in the tree
- `min_impurity_decrease`: compute the gain in purity of splitting a region R into R_1 and R_2, don't split if the gain is less than a threshold. m is the error in use

$$Gain(R) = \Delta(R) = m(R) - \frac{N_1}{N}m(R_1) - \frac{N_2}{N}m(R_2)$$

Normally sklearn grows trees in "level-order" fashion, for example, by using max-depth as a criteria. However, max leaf nodes value will cause it to grow in a "best-first" fashion. You can sometimes get the same trees, just grown in different ways. See Figure reffig:level order vs best first

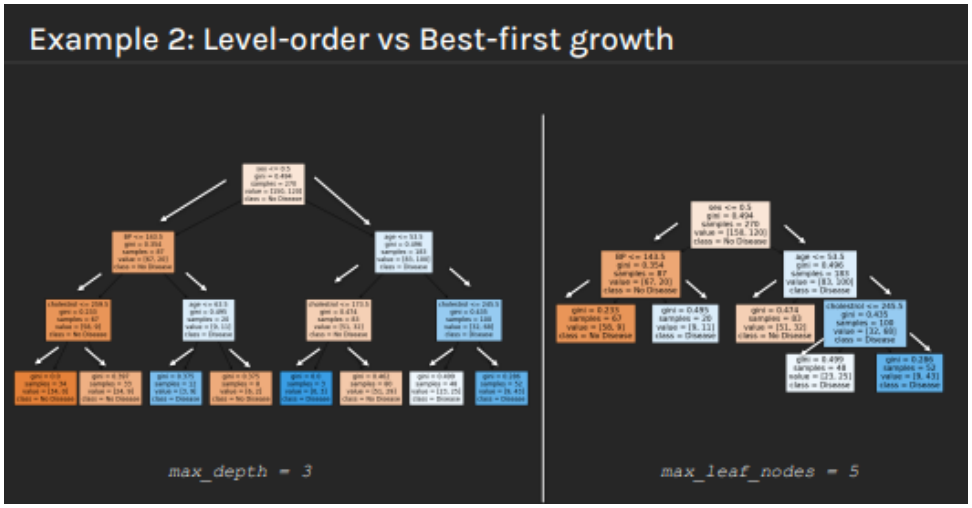

Figure 13:

How do we determine the best stopping criterion? Cross-validation! Consider the tradeoff between Variance vs Bias of trees, as explained in Figure 14

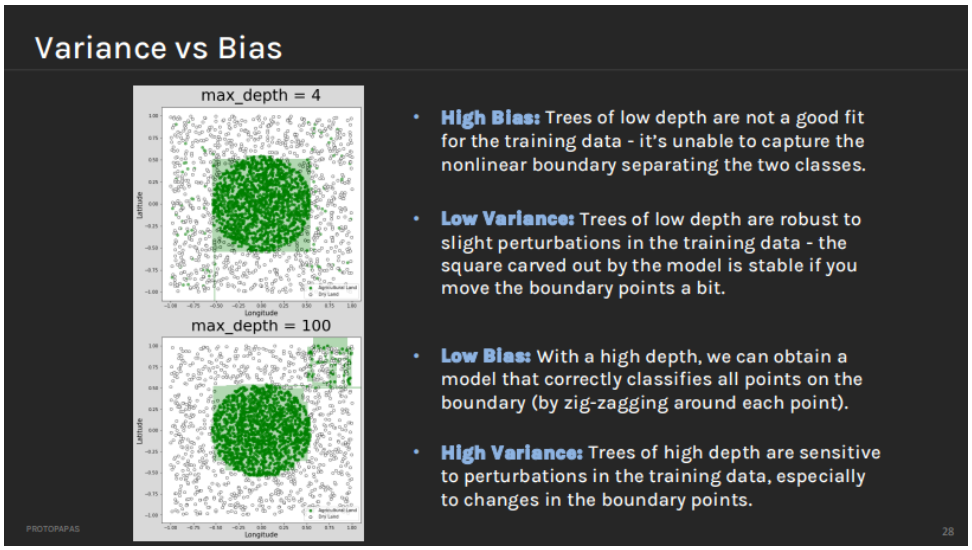

Figure 14: Caption

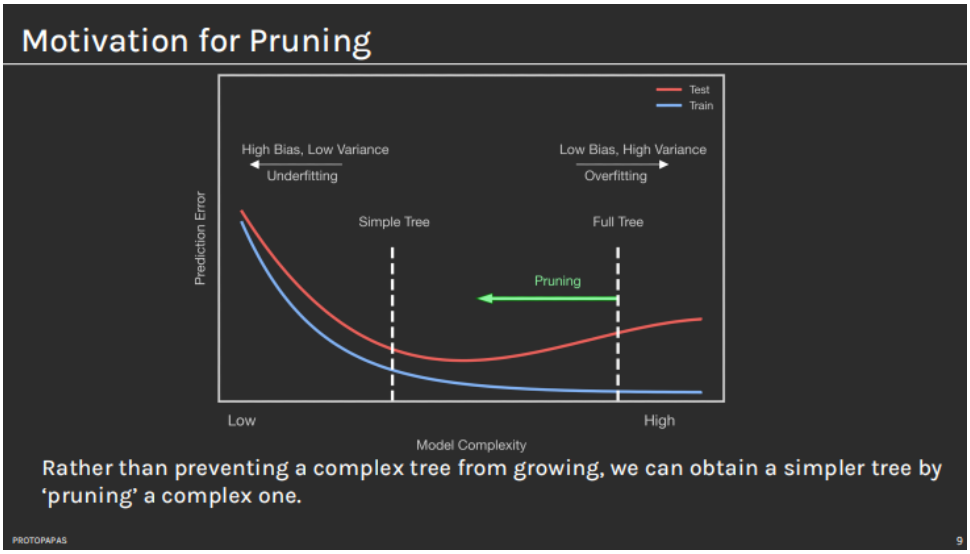

Figure 15: Caption

17 Lecture 18: Decision Trees - Regression

Contents

We make decision trees work for regression by reintroducing MSE as the error metric (analog to Gini, class error, or entropy). Within each region we split, we have our prediction $\bar{y}_{R_r}$ be the mean of the region, then assess the quality via

$$MSE(R_r) = \frac{1}{n_r} \sum_{i \in R_r} (y_i - \bar{y}_{R_r})^2$$

The region weighted average then is

$$\min_{p,t_p} \left[\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2) \right]$$

Most previous stopping conditions can be applied the exact same way. The analog to purity gain is accuracy gain (MSE reduction)

$$Gain(R) = \Delta(R) = MSE(R) - \left[\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2) \right]$$

17.1 Numerical vs Categorical Attributes

We deal with categorical features by encoding. We can't use ordinal coding (unless the feature is ordinal), otherwise different encodings will produce different trees. So we use one-hot-encoding.
NOTE: sklearn doesn't handle this by itself as of now, you have to give it the one-hot-encoded feature columns

17.2 Pruning

Pre-specifying a stopping condition is dangerous. Instead, we can either

- Choose several stopping criteria (e.g. set minimal Gain(R) at various levels) and cross-val to decide which one is best
- Prune! See Figure 15 for the motivation

Cost Complexity pruning is a common method, basically regularization. T is the tree, $Error(T)$ is the classification error, α is the complexity parameter, $|T|$ is the number of leaves in the tree, and $\alpha|T|$ as a

whole is the regularization term. A higher α favors smaller trees with potentially higher error, and vice versa

$$C(T) = Error(T) + \alpha|T|$$

To find the optimal tree, we follow these steps:

1. we identify all possible pruning locations
2. then find the maximal $C(T) - C(T*)$, where $T*$ is the pruned tree

$$\begin{aligned}
&\arg\max_{T^*} [C(T) - C(T^*)] \\
&\arg\max_{T^*} [E(T) - E(T^*) + \alpha|T| - \alpha|T^*|] \\
&\arg\max_{T^*} \left[\frac{E(T) - E(T^*)}{\alpha|T^*| - \alpha|T|} - 1 \right] \quad \text{(divide by } \alpha|T^*| - \alpha|T|) \\
&\arg\max_{T^*} \left[\frac{E(T) - E(T^*)}{\alpha|T^*| - \alpha|T|} \right] \\
&\arg\min_{T^*} \left[\frac{E(T) - E(T^*)}{\alpha|T^*| - \alpha|T|} \right]
\end{aligned}$$

3. We make that prune, then continue this iteratively, saving each one as $T^{(i)}$, until we hit the root tree, $T^{(L0}$
4. Then we cross validate all the saved trees to choose the optimal α

18 Lecture 19: Bagging

Contents

18.1 Motivation

Decision trees often underperform when compared to other classifier methods in situations with complex decision boundaries, because they require a deep tree to capture the complexity, and deep trees have high variance and are prone to overfitting.

Ensemble learning corrects for this by combining several models to produce one single final prediction. There are many methods, but we will starting with Bagging.

18.2 Bagging

Bagging is short for Bootstrap Aggregating

- Bootstrap: we generate multiple samples of training data by sampling with replacement from the original sample. We train a model on each sample
- Aggregating: for a given input, we output the average outputs of all the models for that input
- Advantages of Bagging:
  - High Expressiveness: by using deeper trees each model is able to capture complex functions and decision boundaries
  - Low Variance: averaging the prediction of all the models reduces the variance in the final prediction, assuming that we choose a sufficiently large number of trees
- Drawbacks of Bagging:
  - Interpretability: the averaged model is no longer easily interpretable, you can't trace the logic of an output through a concrete series of decisions. Addressed in Random Forest Lecture
  - Underfitting and Overfitting: shallow enough trees, even with tons of bootstraps, might not be able to capture the real pattern. The more depth we add to the underlying tree, the faster the model overfits
  - Expensive: cross validation is cool but computationally expensive and requires a larger dataset

18.3 Out-of-bag Error

Out-of-bag error is a new metric for ensemble methods for assessing the predictive performance of the model. Given a training set and an ensemble of models, we compute this by

1. for each point x_i in the training set, we average the predicted outputs $\hat{y}_i'$s. We only predict with the B trees whose bootstrap training set excludes this point.
2. We compute the error of this averaged prediction. We call this the point-wise out-of-bag error.
3. We average the point-wise out-of-bag error over the full training set N

See Figure 16 for the math
Why OOB instead of cross-val?

- If doing cross-val, every val set has already been seen in training by at least a few decision trees within the ensemble, there is a leakage of data
- OOB Error prevents leakage and yields a better model with lower variance or less overfitting
- there is also lesser computational cost for OOB as compared to CV for bagging

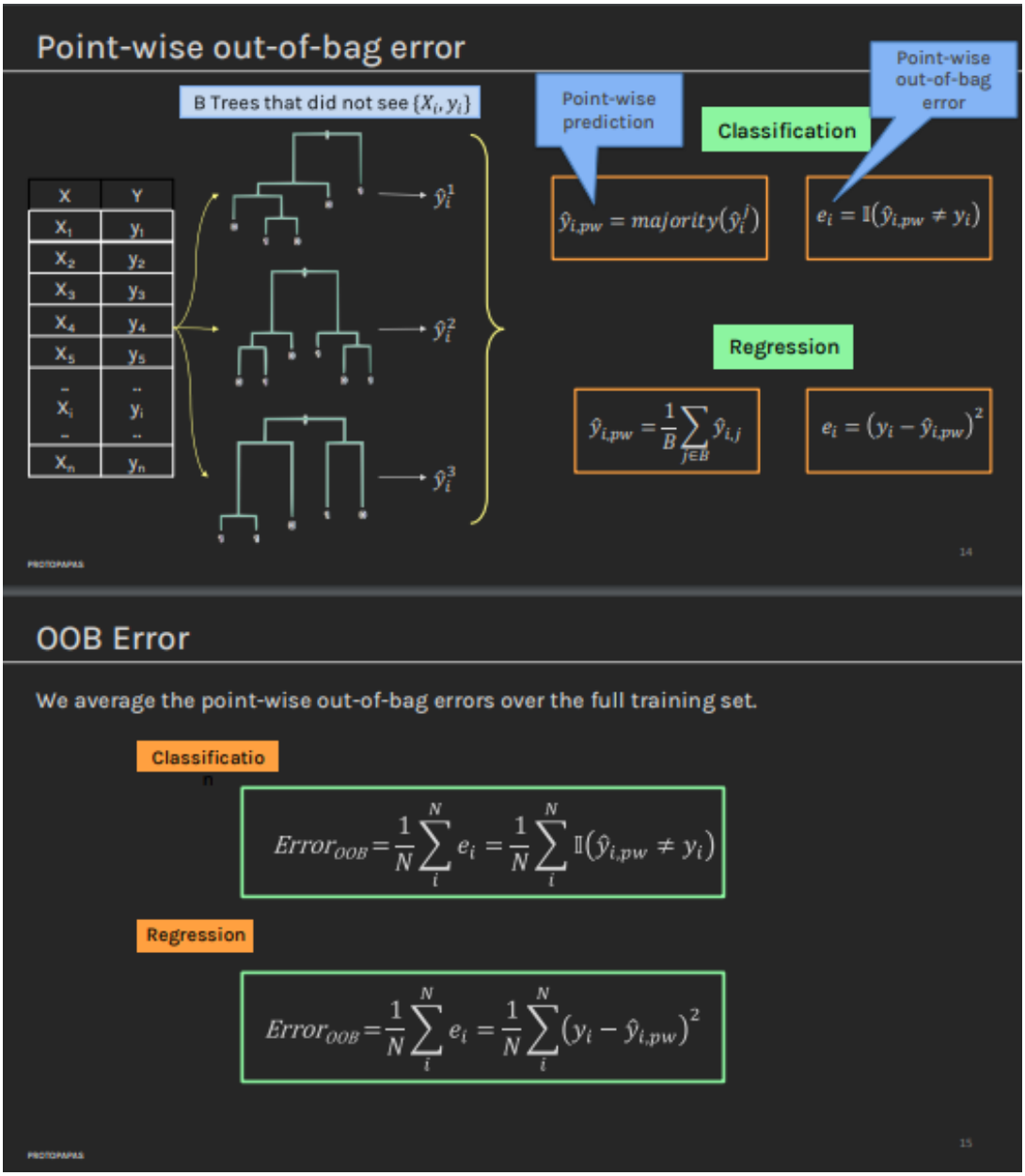
Point-wise out-of-bag error
B Trees that did not see $\{X_i, y_i\}$
Point-wise prediction
Point-wise out-of-bag error
Classification
$\hat{y}_{i,pw} = majority(\hat{y}_i^j)$
$e_i = \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$
Regression
$\hat{y}_{i,pw} = \frac{1}{B}\sum_{j \in B} \hat{y}_{i,j}$
$e_i = (y_i - \hat{y}_{i,pw})^2$
X
Y
$\hat{y}_i^1$
$\hat{y}_i^2$
$\hat{y}_i^3$
OOB Error
We average the point-wise out-of-bag errors over the full training set.
Classificatio n
$Error_{OOB} = \frac{1}{N}\sum_i^N e_i = \frac{1}{N}\sum_i^N \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$
Regression
$Error_{OOB} = \frac{1}{N}\sum_i^N e_i = \frac{1}{N}\sum_i^N (y_i - \hat{y}_{i,pw})^2$

Figure 16:

19 Lecture 20: Random Forest

Contents
Ensembles of trees tend to be correlated (in bagging), which we fix with random forest.

19.1 Random Forest

Random forest is a modified form of bagging that creates ensembles of independent decision trees. The method is

1. Create B bootstrapped datasets with all J predictors (same as bagging)
2. Initialize a random forest with B decision trees
3. For each tree, at each split, we randomly select a subset of J' predictors from the J
4. Amongst the J' predictors, we select the optimal predictor and the optimal threshold for the corresponding split

Random forests have these hyperparameters to tune. They are tuned using out-of-bag errors generally

1. Number of predictors to select at each split. Standard for classification is $\sqrt{N_j}$. Standard for regression is $\frac{N_j}{3}$
2. total number of trees in the ensemble
3. stopping criteria: max depth, minimum leaf node size, etc.
4. splitting criterium: gini, entropy

19.2 Variable Importance

19.2.1 Mean Decrease in Impurity (MDI)

Recall that trees make splits that maximize the decrease in impurity. by calculating the MDI for each feature across all trees, we get the variable importance (for bagging or rf). For each feature j in the dataset:

1. Calculate the mean decrease in impurity for each node q in the decision tree

$$\Delta I_q = \frac{n}{N}\left[Gini_n - \left(\frac{m_{left}}{n}\right)Gini_{m_{left}} - \left(\frac{m_{right}}{n}\right)Gini_{m_{right}}\right]$$

2. Calculate the importance of each feature by summing up the impurity decreases for each node where that feature is used
3. Normalize this value by dividing by the sum of all feature importance values item Average for each feature over all T trees

19.2.2 Permutation Importance

For each feature j in the dataset:

1. Record the validation/OOB (unpermuted) accuracy of RF model: s
2. Randomly permute the data for the column j. Record the validation/OOB accuracy $s_{k,j}$ of the RF model on the modified dataset
3. Repeat this permutation K times and average all the accuracies

$$s_j = \frac{1}{K}\sum_{k=1}^{K} s_{k,j}$$

4. Calculate the difference between unpermuted and average permuted accruacy to get the importance of the feature in the random forest

Final Notes on Random Forests:

- MDI and Permutation Importance probably will result in different feature importance orderings
- MDI is faster, computed during the training, but prefers high cardinality features
- Feature importance for RF is smoother than for bagging due to randomness introduced by selecting a subset of predictors to choose from
- When number of features is high, but number of relevant features is low, RFs can perform poorly because low chance of selecting relevant feature at each split
- Increasing the number of trees in the ensemble does not increase the risk of overfitting (for RF)
- RF and bagging classifiers can return probabilities. Compute the mean predicted class probabilities of the trees in the forest

19.2.3 LIME, SHAP values SEE LAB AND READING

19.3 Class Imbalance

Training a RF (or any model) on an imbalanced dataset introduces challenges. We have to vary the measure we use to evaluate our model based on the type of imbalance

- Balanced Datasets ⇒ accuracy is good measure. Leads to following prescription
- If you have balance in your train but you don't care about false positives or false negatives more than one another (think of cost/criticality), then just use accuracy
- If you have imbalance but still don't care about false positives vs false negatives, give F1 score

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision}$$

- if you have imbalance and you do care... deal with the imbalance

Ways of dealin gwith imbalance:

1. Undersampling: see Figure 17

Figure 17:

   (a) Random Sampling: select values with or without replacement from the majority class
   (b) Near Miss: select data points by using simple heuristics like finding samples from which the average distance to some data points of minority class is smallest. Points from close to decision boundary

2. Oversampling: opposite via bootstrapping

(a) Random Sampling: with replacement

(b) SMOTE: add points in between the existing points via interpolation

3. Class Weighting: provide a weight for each class which places more emphasis on the minority class. Aut

$$W_k = \frac{N}{K \times N_k}$$

where N is the total number of samples, N_k, is the number of samples in class k and K is the total number of classes

19.4 Missing Data (again)

SKIPPED?

19.5 Tree Building Algorithms (LAB)

20 Lecture 21: Gradient Boosting

Contents

20.1 Introduction to Boosting

Decision Tree Issues:

- Shallow trees (few leaves) suffer from high bias and do not train well
- Deep trees (many nodes and leaves) have low bias but high variance leading to very bad generalization

Random Forest Issues:

- Variance reduction is better in RF than bagging, but generalization error is still high
- Prediction Speed: large numbers of trees can make algorithm slow

Could we improve single decision trees by decreasing bias instead of decreasing variance (like we do with bagging/RF) - make them more expressive? $\Rightarrow$ Boosting
This is a a parallel track/alternative to bagging, not necessarily an improvement.

20.2 Gradient Boosting

Take an ensemble of simple models $(T_h)_{h \in H}$) and linearly combine them into a single, more complex model. h is a single weak learner and H is the hypothesis space of all possible weak learns

$$T = \sum_h \lambda_h T_h$$

Two main ideas in gradient boosting:

- Gradient boosting is a method for iteratively building a complex model$T$ by adding simple models
- Each new simple model added to the ensemble compensates for the weaknesses of the current ensemble

Gradient Boosting algorithm:

1. Fit a simple model $T^{(0)}$ on the training data $\{(x_1, y_1), ..., (x_N, y_N)\}$. Set $T \leftarrow T^{(0)}$
2. Compute the residuals $\{r_1, ...r_N\}$
3. For i = 1... until stopping condition is met:
   (a) Fit a simple model, $T^{(i)}$, to the current residuals.
   In other words, train $T^{(i)}$ using $\{(x_1, r_1), ..., (x_N, r_N)\}$
   (b) Set the current model $T \leftarrow T + \lambda T^{(i)}$
   (c) Compute residuals at step i, set $r_n \leftarrow r_n - \lambda T^{(i)}(x_n), n = 1, ..., N$
   where λ is a constant called the learning rate.

NOTE: we don't actually construct combined trees as we iterate, we just predict with them.

20.3 Mathematical Formulation - Gradient Boosting

Based on the above algorithm we assume that our residual is getting smaller with each iteration via

$$r_n \leftarrow r_n - \lambda T^{(i)}(x_n)$$

But we still don't know how to choose λ*correctly*; we need to formulate gradient boosting as a type of gradient descent. Leaving out review of gradient descent

Decision trees are non-parametric, so how do we do gradient descent on the (nonexistent) loss function? We move to step back and view Gradient Boosting in the larger function space.

The function space is a space where each point represents a function $f : X \to y$, so we want to optimize over this function space to find the best estimator. Parametric models like linear regression are a subspace of the function space where we already know the form of the function and just need to optimize the weights. But what do we do for non-parametric, like Boosting?

Boosting can be thought of as doing gradient descent in the prediction space.

- Parametric regime: exists in parameter space

$$\hat{f} := [w_1, w_2, ..., w_n]$$

- Non-parametric regime: exists in prediction/functional space

$$\hat{f} := [\hat{f}(x_1), \hat{f}(x_2, ..., \hat{f}(x_n)]$$

Let our current estimate be $T^{(n)}(x)$. In the prediction space, this function is just a vector of the prediction values. Then the optimal direction to take a step using Gradient Descent is along the negative gradient. But since we aren't in a normal analytical space, we don't have a gradient, we use an approximation of the gradient that depends on the predictors:

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda \hat{r}_n(x_n)$$

Notice the $\hat{r}_n$, we are using a simple model to approximate the residuals (as opposed to just using the exact residuals). If we did take the derivative with respect to the predictions, we just go straight to the solution, we aren't learning to map input to output (didn't include the math for this statement).
NOTE: gradient boosting is descending in a space of models or functions mapping x_n to y_n

Learning rate:

- if λ is constant, then it should be tuned through cross validation
- for better results, use a variable λ. Around optimum, small λ because gradient is small and vice versa

$$\lambda = h(||\nabla f(x)||)$$

21 Lecture 22: AdaBoost

21.1 AdaBoost

Contents

Same two main ideas as Gradient Boosting, but now for classification we don't have a residual. The algorithm is

1. Given training data $(x_1, y_1), \ldots, (x_N, y_N)$, choose an initial distribution $w_n^{(0)} = \frac{1}{N}$.
   For $i = 0...$ until stopping condition is met:
   (a) Train a weak learner $\mathcal{S}$ using weights $w_n^{(i)}$.
   (b) Calculate the total error of the weak learner using:

$$\epsilon^{(i)} = \sum_{n=1}^{N} w_n^{(i)} \mathbf{I}(y_n \neq S^{(i)}(x_n))$$

   (c) Calculate the importance of each model $\lambda^{(i)}$:
   (d) Construct the ensemble model using:

$$T^{(i)}(x) = \begin{cases} \lambda^{(i)} S^{(i)}(x) & \text{if } i = 0 \\ T^{(i-1)}(x) + \lambda^{(i)} S^{(i)}(x) & \text{if } i = 1, 2, \ldots \end{cases}$$

   (e) Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump:

$$w_n^{(i+1)} = \frac{w_n^{(i)} e^{-\lambda^{(i)} y_n T^{(i)}(x_n)}}{Z}$$

   where

$$T^{(i)}(x) = \text{sign}\left[T^{(i-1)}(x) + \lambda^{(i)} S^{(i)}(x)\right].$$

Final model:

$$T^{(i)}(x) = \text{sign}\left[\sum_{i=1}^{T} \lambda^{(i)} S^{(i)}(x)\right].$$

Questions:

- How do I create a stump?
  In AdaBoost the stumps are created using simple decision trees with `max_depth` $= 1$.
- How to use the normalized weights to make the stump?
  - Create a new dataset of same size of the original dataset with repetition based on the newly updated sample weight
  - Or use a weighted version of the Gini Index
  - How do I calculate the scaling factor $\lambda^{(i)}$? next

In gradient boosting for regression, we minimize the MSE loss. In AdaBoost, we minimize a different function, called the exponential loss:

$$Exp\ Loss = \frac{1}{N} \sum_{i=1}^{N} e^{(-y_n \hat{y}_n)}, \text{where } y_n \in -1, 1$$

Thus, unlike Gradient Boosting for regression, we can analytically solve for the optimal learning rate for AdaBoost, by optimizing:

$$\arg\min_{\lambda} \frac{1}{N} \sum_{N=1}^{N} e^{(-y_n (T + \lambda^{(i)} S^{(i)}(x_n)))}$$

Skipping the math of taking derivative and setting to 0, this gets us...

$$\lambda^{(i)} = \frac{1}{2} \ln \left(\frac{1-\epsilon}{\epsilon} \right)$$

where

$$\epsilon = \sum_{n=1}^{N} w_n^{(i)} \mathbf{I}(y_n \neq T^{(i)}(x_n))$$

and

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} e^{-\lambda^i S^{(i)}(x_n)}}{Z}$$

21.2 Mathematical Formulation - AdaBoost

Similar to what we did for gradient boosting, we compute the gradient fo rthe loss w.r.t. the predictions. Skipping the derivation, but doing this gets us an update step of

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda w_n y_n, n = 1, ..., N$$

Just like in gradient boosting, we approximate the gradient, $w_n y_n$ with a simple model, $s^{(i)}$, that depends on x_n. See the comparison between Gradient Boosting and AdaBoost in Figure 18

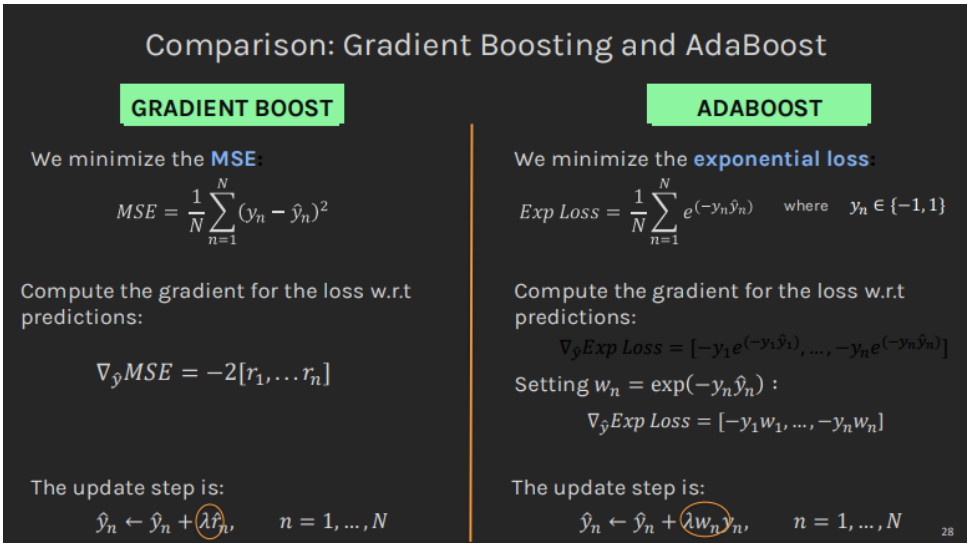

Figure 18:

Boosting is different from bagging, we will overfit as we keep adding estimators or keep increasing learning rate! So we need to tune both of these things!

21.3 Final Thoughts on Boosting

- Stopping conditions: Same as gradient boosting: max iteration, number of stumps, min improvement in loss, number of iterations without improvement in loss
- Overfitting: Unlike other ensemble methods like bagging and Random Forest, boosting methods like AdaBoost will overfit if run or many iterations. Some libraries implement regularization methods which is out of the scope in this course
- Hyper-parameters: All parameters associated with the stump and the stopping conditions mentioned above

More on XGBoost (Extreme Gradient Boosting Implementation of Boosting, better than sklearn)

- Highly optimized SoTA implementation of gradient boosting

- Regularization: L1 and L2 regularization to prevent overfitting
- Parallelization: Uses CPU cores for constructing trees in parallel
- Distributed Computing: Scales well for large datasets using frameworks like Spark, Hadoop
- Cross-Validation: Can tune hyperparameters easily using cross-validation
- Missing Values: Handles missing values automatically

22 Lecture 23: EthiCS

Contents

23 Lecture 24: Blending, Stacking, and Mixture of Experts

Contents

23.1 Ensemble Methods

Bagging Summary:

- Bagging (Bootstrap aggregating) trains model on each bootstrapped sample of the dataset and combines their predictions as the final output
- The base models are usually homogeneous strong learners, i.e.
  - They are the same type of models defined by the same hyperparameters
  - They have low bias (and high variance) and tend to overfit
- Base models are trained in parallel, and their predictions are aggregated through
  - Averaging for regression tasks
  - Majority voting for classification tasks

Bagging Summary:

- In Boosting, base models are fit sequentially on the emphasis of residuals/mistakes from the previous ensemble, and final output is aggregated with different weights on base models' predictions.
- The base models are usually homogeneous weak learners, i.e.
  - They are the same type of models defined by the same hyperparameters
  - They have high bias and tend to underfit
- The predictions from base models are aggregated through:
  - Weighted averaging for regression tasks
  - Weighted voting for classification tasks

How do we aggregate outputs from heterogeneous underlying models?

23.2 Blending

Blending trains heterogeneous learners (i.e., different models) on the dataset in parallel, and it further trains a meta-model on the base models' outputs for making predictions. The goal is to learn how to weight the outputs of the individual models based on how they do in their initial predictions

1. Split the dataset into
   - Training set (to train base models)
   - Validation set (to cross validate base models and generate predictions for training meta model)
   - Holdout set (to cross validate meta model)
   - Test set (for evaluation)
2. Fit base models on training set. Validate with validation set.
3. Generate base model predictions on validation set and holdout set.
4. Concatenate predictions, $\hat{y}$'s, and the original data, x, into a new dataset (also the original y's for the loss function)
5. Combined features from validation set will be used for training meta model
6. Fit meta model with training and validation sets built with original data and base model predictions

During inference (test set):

- Generate predictions on test set from base models
- Combine model predictions and original data
- Generate meta model predictions

Compared with Bagging and Boosting, Blending provides flexibility:

- Allows combining models of different types
- Meta model learns how to best combine diverse model outputs
- Blending models can be more interpretable if meta-model is simple (e.g. linear regressoin), where contribution of each base model can be explicitly visible
- Bad part is we are dividing the dataset into more pieces for training blending models, and the amoun to fdata for training meta model is only the size of the validation set (usually 10%-25% of the original data).

23.3 Stacking

Essentially blending with cross validation.

1. Similar to blending, we first split the dataset into
    - Training set (to train base models)
    - Holdout set (to validate meta model)
    - Test set (for evaluation)
      Note: No validation set because we will be doing cross validation
2. Assume 3 folds
    - We first take Fold 1 as our val fold
    - We train on Fold 2 and 3
    - Validate on Fold 1 and generate predictions
    - Repeat this with each of the other folds as the val fold, resulting in 3 sets of predictions
    - Combine the training folds with the base model predictions to make the training set for the meta model. Now the size of this training set is the same as the original training set (unlike blending)
    - Meanwhile, in each fold round of cross val, we want to make predictions on holdout set
3. Combined features from holdout set will be ready for validating meta model
4. The remaining steps of fitting meta model and making predictions are the same as blending

23.4 Mixture of Experts

How to combine models that don't take the same input (5 columns vs 10 columns)
Same setup as the past two, but need a Gating Network to figure out how to connect to the Meta Expert