Big Data for Public Policy

5. Machine Learning Essentials - Classification

Elliott Ash & Malka Guillot

Where we are

- Past weeks:
  - w1: Overview and motivation
  - w2: Finding datastests using webcrawling and API
  - w3: Intro to supervised Machine Learning (ML) - regressions
  - w4: Text analysis fundamentals
- This week (w5):
  - Supervised ML - classification
  - Corresponding references: Geron chap 3, chap 4 (pages 142 to 151)
- Next:
  - w6: Unsupervised ML

Today: supervised ML - classification

- Mainly:
  - A supervised learning approach
  - Objective: categorizing some unknown items into a discrete set of categories or “classes” using labeled data
- But also:
  - Application:

Outline

Classification Framework

- Response/target variable y is **qualitative** (or **categorical**):
    - 2 categories $\rightarrow$ binary classification
    - More than 2 categories $\rightarrow$ multi-class classification
- Features X:
    - can be high-dimensional
- We want to assign a class to a **quantitative response**
  $\rightarrow$ probability to belong to the class
- **Classifier**: An algorithm that maps the input data to a specific category.
- Performance measures specific to classification

Application examples

- In business:
    - Loan default prediction
    - Type of costumer
- In public economics:
    - Tax evasion prediction
- In political sciences:
    - political affiliation of author of texts
- In medical sciences:
    - Diagnostic diseases, drug choice
- Other:
    - email filtering, speech recognition...

Why not fitting a linear regression?

- **Technically possible** to fit a linear model using a categorical response variable but it implies
  - an **ordering** on the outcome
  - a **scale** in the class difference

$\rightarrow$ If the response variable was coded differently, the results could be completely different

- Less problematic if the response variable is **binary**
  - The result of the model would be stable
  - But prediction may lie outside of $[0, 1]$: hard to interpret them in terms of probabilities

Linear Regression vs Binary Classifier

- We model the probability of belonging to a category

$$P(y = 1 \mid X)$$

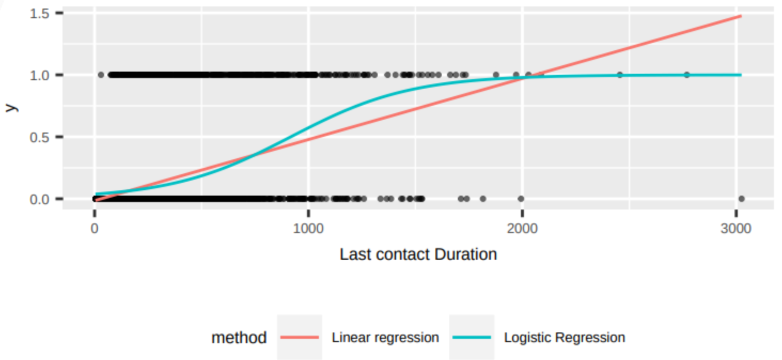
- We can rely on this probability to assign a class to the observation.
    - For example, we can assign the class yes for all observations where $P(y = 1|x) > 0$
    - But we can also select a different **threshold**.

Example

- We predict y, the **occupation of individuals**:

$$y = \begin{cases} 0 \text{ if blue-collar} \\ 1 \text{ if white-collar} \end{cases}$$

- based on their characteristics X (gender, wage, contract duration, experience, age...)



Outline

Confusion Matrix

- For comparing the predictions of the fitted model to the actual classes.
- After applying a classifier to a data set with known labels *Yes* and *No*:

		Predicted class	
		no	yes
True class	no	TN	FP
	yes	FN	TP

Precision and Recall

- Precision

  $= $ accuracy of positive predictions.

  $= \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$

  - decreases with false positives.

- Recall

  $= $ true positive rate.

  $= \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$

  - decreases with false negatives.

F1 Score

- The F_1 score provides a single combined metric it is the **harmonic mean** of precision and recall

$$
\begin{aligned}
F_1 &= \frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = 2 \times \frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}} && (1)\\
&= \frac{\text{Total Positives}}{\text{Total Positives}+\frac{1}{2}(\text{False Negatives}+\text{False Positives})} && (2)
\end{aligned}
$$

- The harmonic mean gives **more weight to low values**.
- The F1 score values precision and recall **symmetrically**.

The Precision/Recall Tradeoff

- F_1 favors classifiers with similar precision and recall, but sometimes you want **asymmetry**:

The Precision/Recall Tradeoff

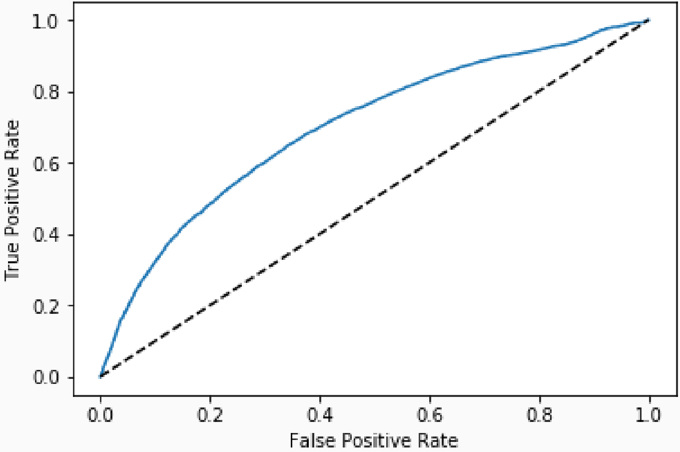
- F_1 favors classifiers with similar precision and recall, but sometimes you want **asymmetry**:
- low recall + high precisions is better:
  - e.g. **deciding "guilty" in court**, you might prefer a model that
  - lets many actual-guilty go free (high false negatives $\leftrightarrow$ low recall)...
  - ... but has very few actual-innocent put in jail (low false positives $\leftrightarrow$ high precision

The Precision/Recall Tradeoff

- F_1 favors classifiers with similar precision and recall, but sometimes you want **asymmetry**:
- low recall + high precisions is better:
    - e.g. **deciding "guilty" in court**, you might prefer a model that
    - lets many actual-guilty go free (high false negatives $\leftrightarrow$ low recall)...
    - ... but has very few actual-innocent put in jail (low false positives $\leftrightarrow$ high precision
- high recall + low precisions is better:
    - e.g classifier to **detect bombs during flight screening**, you might prefer a model that:
    - has many false alarms (low precision)...
    - ... to minimize the number of misses (high recall).

ROC Curve and AUC

- Plots *true positive rate* (recall) against the *false positive rate* ($\frac{FP}{FP+TN}$):



ROC Curve and AUC

- The area under the ROC curve (AUC) is a popular metric ranging between:
  - 0.5
    - **random classification**
    - ROC curve = first diagonal
  - and 1
    - **perfect classification**
    - = area of the square
  - better classifier $\rightarrow$ ROC curve toward the top-left corner
- Good measure for model comparison

Outline

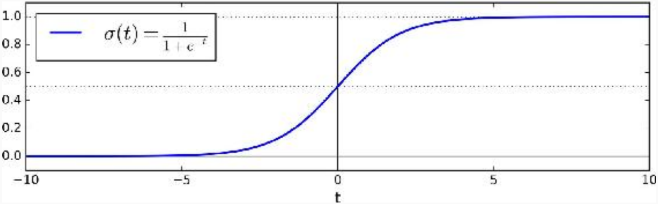
Outline

- Like OLS, logistic "regression" computes a weighted sum of the input features to predict the output.
  - But it transforms the sum using the **logistic function**.

$$\hat{p} = \Pr(Y_i = 1) = \sigma(\theta' \boldsymbol{x})$$

where $\sigma(\cdot)$ is the sigmoid function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



- Prediction: $\hat{y} = \begin{cases} 0 \text{ if } \hat{p} \geq .5 \\ 1 \text{ if } \hat{p} < .5 \end{cases}$

Logistic Regression Cost Function

- The cost function to minimize is

$$J(\theta) = \underbrace{-\frac{1}{m}}_{\text{negative}} \sum_{i=1}^{m} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{p}_i)}_{\text{log prob} y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{p}_i)}_{\text{log prob} y_i=0}]$$

  - this does not have a closed form solution
  - but it is convex, so gradient descent will find the global minimum.

Logistic Regression Cost Function

- The cost function to minimize is

$$J(\theta) = \underbrace{-\frac{1}{m}}_{\text{negative}} \sum_{i=1}^{m} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{p}_i)}_{\text{log prob} y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{p}_i)}_{\text{log prob} y_i=0}]$$

  - this does not have a closed form solution
  - but it is convex, so gradient descent will find the global minimum.

- Just like linear models, logistic can be regulared with L1 or L2 penalties, e.g.:

$$J_2(\theta) = \mathrm{J}(\theta) + \alpha_2 \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Feature selection for classification tasks

- χ^2 is a very fast feature selection routine for classification tasks
    - features must be non-negative
    - works on sparse matrices
- With negative predictors, use `f_classif`.

Outline

Naive Bayes Classifier

- Relies on the observed conditional probabilities (and the Bayes theorem)
- For a 2-class problem for a given observation $X = x_0$:
  - Predict class 1 if $P(Y = 1|X = x_0) \geq 0.5$
  - Predict class 0 if $P(Y = 1|X = x_0) < 0.5$
- Relies on the independance assumption

K-Nearest Neighbors

- With real data, we do not know the conditional distribution of Y given X.

$\rightarrow$ computing the Bayes classifier is not possible.

- The K-nearest neighbors (KNN) classifier estimates the conditional distribution of Y given X.
- Approximate Bayes decision rule in a subset of data around the testing point

K-Nearest Neighbors

- With real data, we do not know the conditional distribution of Y given X.

$\rightarrow$ computing the Bayes classifier is not possible.

- The K-nearest neighbors (KNN) classifier estimates the conditional distribution of Y given X.

Outline

Outline

Multi-Class Models

- Many interesting machine learning problems involve multiple un-ordered categories:
  - categorizing a case by area of law
  - predicting the political party of a speaker in a proportional representation system

Multi-Class Confusion Matrix

		Predicted Class		
		Class A	Class B	Class C
True Class	Class A	Correct A	A, classed as B	A, classed as C
	Class B	B, classed as A	Correct B	B, classed as C
	Class C	C, classed as A	C, classed as B	Correct C

- More generally, can have a confusion matrix M with items M_{ij} (row i, column j).

Multi-Class Performance Metrics

Confusion matrix M with items M_{ij} (row i, column j).

$$\text{Precision for } k = \frac{\text{True Positives for } k}{\text{True Positives for k} + \text{False Positives for } k} = \frac{M_{kk}}{\sum_j M_{kj}}$$

$$\text{Recall for } k = \frac{\text{True Positives for } k}{\text{True Positives for } k \text{ + False Negatives for } k} = \frac{M_{kk}}{\sum_i M_{ik}}$$

$$F_1(k) = 2 \times \frac{\text{precision}(k) \times \text{recall}(k)}{\text{precision}(k) + \text{recall}(k)}$$

Metrics for whole model

- Macro-averaging:
  - average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n}\sum_{k=1}^{n} F_1(k)$$

  - treats all classes equally

Metrics for whole model

- Macro-averaging:
  - average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n}\sum_{k=1}^{n} F_1(k)$$

  - treats all classes equally
- Micro-averaging:
  - Compute model-level sums for true positives, false positives, and false negatives; compute precision/recall from model sums.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}, \text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - favors bigger classes

Metrics for whole model

- Macro-averaging:
  - average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n}\sum_{k=1}^{n} F_1(k)$$

  - treats all classes equally
- Micro-averaging:
  - Compute model-level sums for true positives, false positives, and false negatives; compute precision/recall from model sums.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}, \text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - favors bigger classes
- "Weighted": same as macro averaging, but classes are weighted by number of true instances in data.

Multinomial Logistic Regression

- Logistic can be generalized to multiple classes.
  - When given an instance $\boldsymbol{x}_i$, multinomial logistic computes a score $s_k(\boldsymbol{x}_i)$ for each class k,

$$s_k(\boldsymbol{x}_i) = \theta_k' \boldsymbol{x}_i$$

  - If there are n features and K output classes, there is a $K \times n$ parameter matrix Θ, where the parameters for each class are stored as rows.

Multinomial Logistic Regression

- Logistic can be generalized to multiple classes.
  - When given an instance $\boldsymbol{x}_i$, multinomial logistic computes a score $s_k(\boldsymbol{x}_i)$ for each class k,

$$s_k(\boldsymbol{x}_i) = \theta_k' \boldsymbol{x}_i$$

  - If there are n features and K output classes, there is a $K \times n$ parameter matrix Θ, where the parameters for each class are stored as rows.
- Using the scores, probabilities for each class are computed using the softmax function

$$\hat{p}_k(\boldsymbol{x}_i) = \Pr(Y_i = k) = \frac{\exp(s_k(\boldsymbol{x}_i))}{\sum_{j=1}^{K} \exp(s_j(\boldsymbol{x}_i))} = \frac{e^{\theta_k \boldsymbol{x}_i}}{\sum_{j=1}^{K} e^{\theta_j \boldsymbol{x}_i}}$$

  - And the prediction $Y_i \in \{1, ..., K\}$ is determined by the highest-probability category.

Multinomial Logistic Cost Function

- The binary cost function generalizes to the cross entropy

$$J(\theta) = \underbrace{-\frac{1}{m}}_{\text{negative}} \sum_{i=1}^{m} \sum_{k=1}^{K} \underbrace{\mathbf{1}[y_i = k]}_{y_i = k} \underbrace{\log(\hat{p}_k(\boldsymbol{x}_i))}_{\text{log prob} y_i = k}$$

  - again, this is convex, so gradient descent will find the global minimum.

Outline

Types of Classification Algorithms

- Linear Classifiers
  - Logistic regression
  - Naive Bayes classifier
- Support vector machines
- Kernel estimation
  - k-nearest neighbor
- Decision trees
  - Random forests
-