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

- Slides:
  - A supervised learning approach
  - Objective: categorizing some unknown items into a discrete set of categories or "classes" using labeled data
- Notebook:
  1. A simple classifier
  2. Performance measures
  3. Demonstration of how the different classifiers work

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...

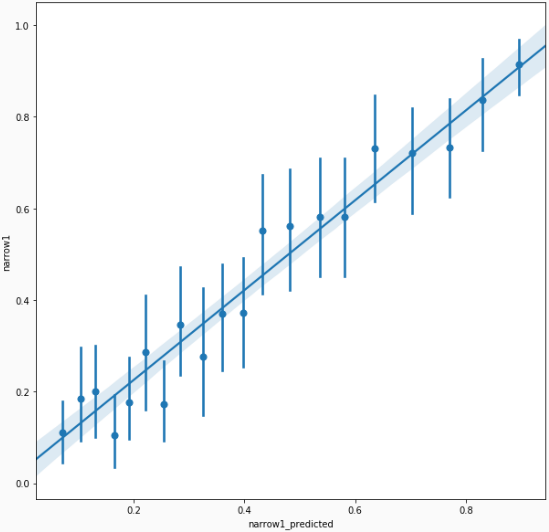
Predicting corruption based on public finance account

Ash, Galetta and Giommoni (2020) A Machine Learning Approach to Analyzing Corruption in Local Public Finances.

- Predict corruption from budget accounts in Brazilian municipalities that have been audited for corruption
  - train set: Using an innovative anticorruption program: **audit lotteries**
  - Features: local public finance data
  - Gradient boosting algorithm: Test-set accuracy of 75%, much better thanguessing (58%) and predictions from OLS (59%)
- Used to evaluate the dynamic (and spillover) effects of audits

→ inputs to policy decisions about corruption

Predicting corruption based on public finance account

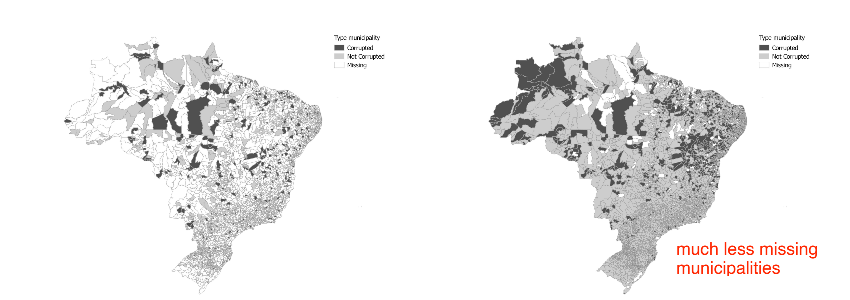


Notes. Binscatter diagram of average true corruption (vertical axis) against binned predicted corruption (horizontal axis).

Applying to Full Dataset

Take model trained on audited municipality-terms and predict probability of corruption in all municipalities and all years

Figure 1: The Geography of (Predicted) Corruption



(a) Actual Corruption

(b) Predicted Corruption

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

distance between categories
often doesn't make sense

→ 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

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

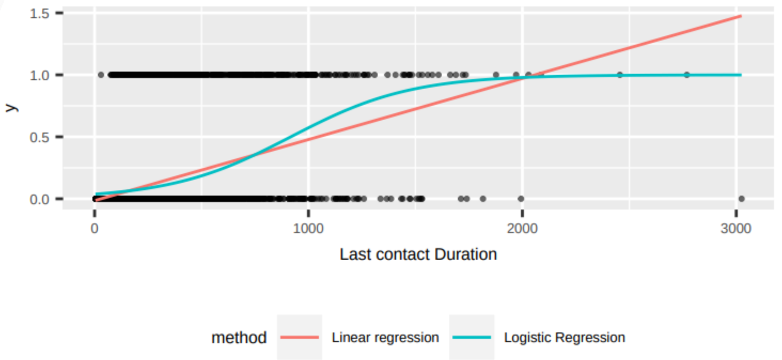
- We can rely on this probability to assign a class to the observation.
  - For example, we can assign the class yes for all observations where $P(y = 1|x) > 0$
  - But we can also select a different **threshold**.

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*:

Confusion matrix maps true to predicted classes

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

Precision

Precision and Recall

- Precision

  $$\begin{aligned} &= \text{accuracy of positive predictions.} \\ &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \end{aligned}$$

  - decreases with false positives.

- Recall

  $$\begin{aligned} &= \text{true positive rate.} \\ &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \end{aligned}$$

  - 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

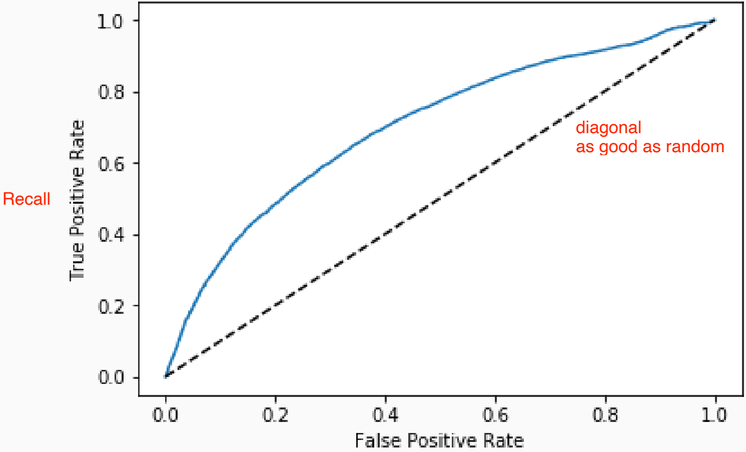
- F_1 favors classifiers with similar precision and recall, but sometimes you want **asymmetry**:
- low recall + high precisions is better:
  - e.g. **deciding "guilty" in court**, you might prefer a model that
  - lets many actual-guilty go free (high false negatives $\leftrightarrow$ low recall)...
  - ... 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 → ROC curve toward the top-left corner
- Good measure for model comparison

Outline

Outline

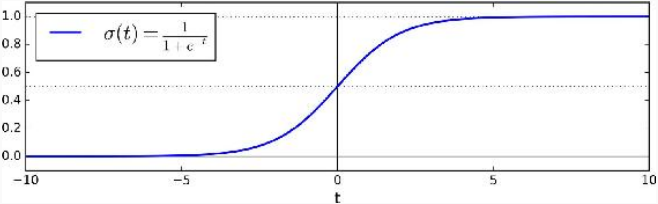
Logistic Regression

- 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

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



- 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

if p ~ 0 -> log(p) ~ inf.
if p ~ 1 -> log(p) ~ 0

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

  - 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.:

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

scikit learn hyperparameters
associated with J(theta), not alpha_2

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. BUT you can do it in a local neighbourhood. It locally approximates the Bayes decision rule.

$\rightarrow$ 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.

→ computing the Bayes classifier is not possible.

- The K-nearest neighbors (KNN) classifier estimates the conditional distribution of Y given X.
- **Non-parametric method** often successful in classification situations where the **decision boundary is very irregular**

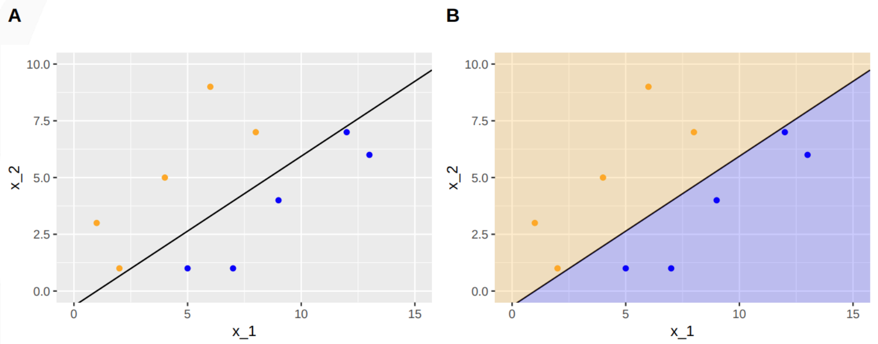
Outline

Support Vector Machine: context and concepts

- Context: developed in the mid-1990s
- A generalization of the early logistic regression (1930s)
- One of the best "out of the box" classifiers
- Core idea: hyperplane that separates the data as well as possible, while allowing some violations to this separation
- Pieces of the puzzle:
  1. A **maximal margin classifier**: requires that classes be separable by a linear boundary.
  2. A **support vector classifier**: extension of the maximal margin classifier.
  3. **Support vector machine**: further extension to accommodate non-linear class boundaries.
- For binary classification, can be extended to multiple classes

Classification and hyperplane

Figure 2: A perfectly separating linear hyperplan for a binary outcome

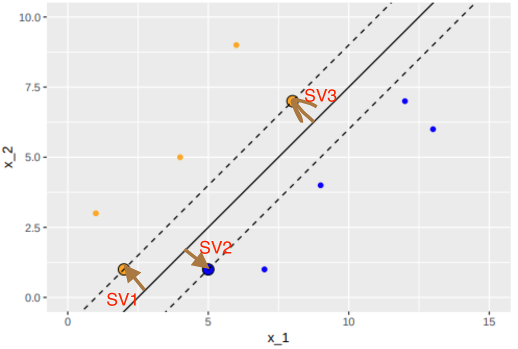


There are an infinity of such separating hyperplan

$\rightarrow$ we need to choose one

Maximum margin

Figure 3: Maximum margin classifier for a perfectly separable binary outcome variable



Criterium for optimal choice: the separating hyperplane for which the the margin is the farthest from the observations, i.e., to select the **maximal margin hyperplane**

Support vector

Support vector $=$ the 3 observations from the training set that are equidistant from the maximal margin hyperplane

$\rightarrow$ they "support" the maximal margin hyperplane (if they move, the the maximal margin hyperplane also moves)

Overcoming the perfectly separable hyperplan assumption

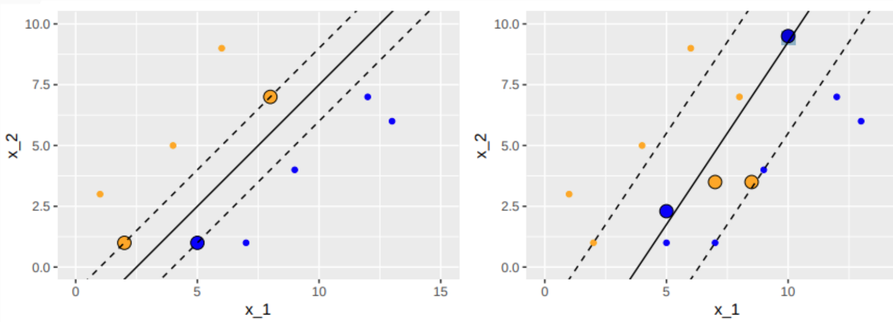
We allow some number of observations to violate the rules so that they can lie on the wrong side of the margin boundaries.

$\rightarrow$ find a hyperplane that almost separates the classes

The **support vector classifier** generalizes the maximum margin classifier to the non-separable case.

Support vector classifiers

Figure 4: Maximal margin classifier (left) and support vector classifier (right)



Overcoming the linearity assumption: support vector machines

Idea 1 (polynomial) transformation of the features + `StandardScaler` + `LinearSVC`.

Idea 2 convert a linear classifier into a classifier that produced **non-linear decision boundaries**.

→ using a Kernel such as:

- Polynomial kernel
- Gaussian RBF kernel

- **We do not open the kernel box**.
  - Just think as them as a way to construct non-linear hyperplans
  - Try out different kernel and distance specification

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 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
- Some classifier handle multi-class natively
- Others are strictly binary (SVM)

2 approaches for N classes

- One-versus-the-rest (OvR)
  - each classifier compares a classe to all the other classes
  - select the class whose classifier outputs the highest score
  - train N classifiers (1 by class) on the whole data

  $\rightarrow$ prefered solution
- One-versus-the-one (OvO)
  - each classifier compares a pair of classe
  - train $\frac{N(N-1)}{2}$ classifiers on two classes each time

  $\rightarrow$ good if training takes time on a large dataset

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.

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

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

  - favors bigger classes
- "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 [week 7]
  - Random forests