Ensemble methods

(Combining predictions)

M. Stefanucci

Fall 2021

University of Trieste

Ensemble methods

Learning with imbalanced data

Ensemble methods

Combining multiple predictions: Model averaging

- Classification trees can be simple, but often produce noisy (bushy) and weak classifiers
  - **Bagging** *(Breiman, 1996)*: Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote
  - **Boosting** *(Freund & Shapire, 1996)*: Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote
  - **Random Forests** *(Breiman 1999)*: Fancier version of bagging.
- Note however that the idea of combining multiple predictions or classifications can be used for any technique (i.e., logistic classification, NN, etc.) and it is not limited to trees
- This idea is closely related with model averaging: a strategy for model selection and evaluation of uncertainty in Bayesian analyses

Combining predictions (classifications)

- The idea is to combine the output of different learners for each data point $(y, \boldsymbol{x})$. This help when learners have complementary strengths.
- Suppose training data are available in the form of the p covariates $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ and the response is (target) is y . Let $\hat{y}_1 = f_1(\boldsymbol{x}), \ldots, \hat{y}_M = f_1(\boldsymbol{x})$ be M different predictions (estimates, "experts evaluations") for the same data point.
- A simple combined vote takes their average

$$f_{comb}(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(\boldsymbol{x})$$

In the classification setting for each class k we have a prediction $f_m^k(x, t)$ equal to 0 or 1. Then

$$f_{comb}(x) = \frac{1}{M} \sum_{m=1}^{M} f_m^k(\boldsymbol{x})$$

for each class k and $f_{comb}^k(x, t)$ is the proportion of votes for class k. We predict the class with the most number of votes **(majority vote)**.

Bagging

- Bagging or bootstrap aggregation averages a given procedure over many samples, to reduce its variance
- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.
- Instead, we can
  - bootstrap, by taking repeated samples from the same training data set
  - use the b-th bootstrapped training set to get the prediction $\hat{f}^b(x)$ and
  - average all the predictions, to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(\boldsymbol{x})$$

- This is called bagging. In classification problems it uses majority votes.
- Bagging can dramatically reduce the variance of unstable procedure (like trees), leading to improved prediction.
- Bagging averages many trees, produces smoother decision boundaries, reduces the variance, but can slightly increase bias

Out-of-bag

- Using random samples of observations allows the use of the out-of-bag tool, for easy estimation of prediction errors.
- In each bootstrap sample, some of the data of the original training set are excluded.
- On average, each bagged sample makes use of around two-thirds of the observations
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations
- For each classifier $\hat{f}^b(\boldsymbol{x})$ the data of training set that are not in sample can be used as a test set. This will give $B/3$ predictions for the i-th observation.
- Estimate the misclassification error on these data outside the sample used for the fit (out-of-bag), so avoiding cross-validation for large data sets

Random Forest

- A quite popular refinement of bagging. Particularly when bagging trees for which was originally developed. We will describe this version.
- At each tree split, a random sample of m features is drawn, and only those m features are considered for splitting.
- Typically $m = \sqrt{p}$ or $\log_2 p$, where p is the number of features (covariates)
- For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored (out-of-bag)
- Random forests tries to improve on bagging by "de-correlating" the trees, and reduce the variance.
- Each tree has the same expectation, so increasing the number of trees does not alter the bias of bagging or random forests.

Variable importance

Random forests can be used to rank the importance of variables in a regression or classification problem.

- For each tree grown in a random forest, calculate number of votes for the correct class in out-of-bag data.
- Now perform random permutation (shuffling) of a predictor's values (let's say variable-k) in the OOB data and then check the number of votes for correct class.
- Subtract the number of votes for the correct class in the variable-k-permuted data from the number of votes for the correct class in the original OOB data.
- The average of this number over all trees in the forest is the raw importance score for variable k. The score is normalized by taking the standard deviation.
- Variables having large values for this score are ranked as more important. It is because if building a current model without original values of a variable gives worse prediction, it means the variable is important.

Boosting

- Designed, initially, exclusively for classification problems
- Idea: Like bagging, but take unequal probability bootstrap samples. Put more weight on observations that are misclassified, to make the classifier work harder on those points. Invention of Freund e Schapire (1997)
- Details
  - Start with equal observation weights $p_i = 1/n$
  - At iteration t, draw a bootstrap sample with the current probabilities $p_1, p_2, \ldots, p_n$, compute the classifier and e_t , the error rate of the classifier on the original sample. Let $\beta_t = e_t/(1 - e_t)$
  - For those points that are classified correctly, decrease their probabilities $p_i = p_i \beta_t$ and normalize them
  - Do this for many (say 1000) iterations.

Boosting

- At the end, take a weighted vote of the classifications, with weights $\alpha_t = log(1/\beta_t)$ (more weight on classifiers with lower error).
- Boosting can improve bagging in many instances

Weighting decorrelates the trees, and focuses on regions missed by past trees.

In the classification setting for each class k we have a prediction $f_m^k(x,t)$ equal to 0 or 1. Then

$$f_{comb}(x) = \frac{1}{M}\sum_{m=1}^{M} f_m^k(\boldsymbol{x})$$

for each class k and $f_{comb}^k(x,t)$ is the proportion of votes for class k. We predict the class with the most number of votes **(majority vote)**.

Learning with imbalanced data

Classification with imbalanced datasets

- The problem of data imbalance emerges in supervised classification problems and here we will only mention the (most relevant) case of binary classification
- It is an issue that occurs when one of the two classes which represents the target variable (usually also the class of main interest) is **rare** and then it is much less represented then the other class in the dataset
- It is a situation encountered in many real word applications:
  - in many fraud detection problems the number of observed frauds is (luckily) a rare event
  - when predicting the insolvency of a firm the event of failing in a given year is not very frequent
  - in medicine, many specific diseases in the population have usually a very low frequency
  - there are many examples where customers are very loyal and observing customer churn is rare

Degree of imbalance

- The degree of imbalance for the response variable can be measured by the imbalance ratio IR which is defined as the ratio between the cases of the prevalent class divided by the number of data points in the rare class. Saying that IR is 100 mean that for 1 data point in the rare (positive) class there are 100 cases of the prevalent (negative) class.
- Actually any dataset presents a certain degree of imbalance, but the severity of the problem for a two-class classification problem emerges usually when the IR is at least larger than 10
  - IR larger than 100 defines a strong imbalance
  - IR larger than 1000 is an extremely skewed dataset (very strong imbalance)
- In the two last cases dealing with the imbalance before bulding any classification rule or machine learning algorithm cannot be avoided
- The size of the dataset also matters when defining the severity of the problem

Why standard ML algorithms can fail with imbalanced datasets?

- Standard classifiers such as logistic regression, Support Vector Machine (SVM), classification tree or other ML algorithms are suitable for balanced training sets.
- When facing imbalanced scenarios, these models often provide suboptimal classification results, i.e., a good coverage of the majority examples, whereas the minority examples are distorted
  - The learning process often guided by global performance metrics such as prediction accuracy induces a bias towards the majority class
  - Rare minority examples may possibly be treated as noise by the learning model.
- Many machine learning and statistical approaches have been developed in the past two decades to cope with imbalanced data classification, most of which have been based on three strategies: (i) sample techniques, (ii) cost sensitive learning and (iii) possible modification of the learning algorithm

Performance metrics in a two-class problems

- The quality of a learning algorithm is evaluated by looking at its performance on test data.
- The simplest performance measures are based on comparison between the predictions of the classifier and the true values (confusion matrix).

		predicted		
		positive	negative	total
	positive	TP	FN	POS
Actual	negative	FP	TN	NEG
	total	PredPOS	PredNEG	N

- The simplest measure, accuracy (ACC) is defined as

$$ACC = \frac{TP + TN}{N}$$

- note that ACC **cannot** be used as a performance measure in imbalanced dataset: a trivial classifier which always classify new cases into the majority class will have an accuracy equal to the proportion of the majority class in the sample
- If we have 0.1% of the sample of cases with a rare type of cancer, a (dull) strategy which always predicts that you are free from the disease will anyway obtain ACC equal to 99.9%

Other metrics and their use for imbalanced datasets

- True positive rate (recall or sensitivity) $= \frac{TP}{POS}$
- True negative rate (specificity) $= \frac{TN}{NEG}$
- Positive predictive value (precision) $= \frac{TP}{predPOS}$
- $F1 = (1+\beta^2)\frac{\text{precision}\cdot\text{recall}}{(\beta^2\text{precision})+\text{recall}}$

β is often taken to be 1 (that is precision and recall have the same weight)

- Classification is obtained by setting a threshold for those methods (logistic regression, trees) which estimate a score (probability) and this threshold can be somewhat arbitrary and should be changed appropriately when using some of the remedies for imbalance (such as oversampling the rare class).

AUC (area under the ROC curve)

- AUC which does not depend on a given threshold is a most appropriate measure for comparing performances with imbalanced dataset.
- ROC (Receiver Operating Characteristics) measures the accuracy of a classification prediction when prediction comes in form of a numerical scoring (a probability).
- ROC Curve is obtained by plotting sensitivity versus specificity for different thresholds. AUC measures the area under this curve and the larger the better is the performance (for a perfect classifier AUC is 1).
- Note that ROC curve, and as a consequience AUC, can be very unstable when the test dataset is small and imbalanced

Approaches to imbalanced data classification

- Preprocessing techniques (resampling and synthetic data generation)

Preprocessing is often performed before building learning model in order to obtain balanced input data in building the classifier

- Cost-sensitive learning
- Specific modifications of classification algorithms for imbalanced learning

(re)Sampling techniques: undersampling

- This strategy belongs to the first category of remedies: preprocessing techniques
- The aim is to obtain a balanced sample (let's say one of the classes has no less than 30% of the cases, or even better the two classes are made equivalent) for both the training and the test set.
- Undersampling:

this method reduces majority class. It consists in randomly selecting a subset of observations (without replacement) from majority class to make the data set balanced. This method can be very successfully used when the data set is really huge. It can be improved on by adding strategy for selecting the data most relevant for classification.

(re)Sampling techniques: oversampling

- It eliminates the harms of skewed distribution of the target by multiplying new minority class samples. Data from the rare class are re-sampled (with replacement) in order to balance the sample (note that it can induce overfitting)
- Oversampling can also be achieved by generating new instances from existing one.
- There are many methods specifically designed for generating new synthetic data (data cloning) for the minority class. We will briefly describe two of them:
  - SMOTE (Chawla et al, 2002)
  - ROSE (Menardi & Torelli, 2014)
- Hybrid methods: are a combination of the over-sampling method and the under-sampling method.

ROSE: Random OverSampling Examples

- ROSE is aimed at oversampling the rare class by creating new (syntetic) data points that are as similar as possible to the existing (real) ones
- ROSE also suggests to under-sample the majority class (possibly by cloning data with the same strategy)
- ROSE chooses a random point from the rare set and then a new point is generated in the neighborhood according to a kernel function (imagine to put a multivariate Gaussian distribution centered on the point and select a new point from that Gaussian)
- The cloning method is formally based on a kernel density estimation of the distribution of the predictor variables within the rare (or sometimes also the prevalent) class. This turns out to be equivalent to obtaining a **smoothed bootstrap** resampling scheme

ROSE: Random OverSampling Examples

- A package exists in R named *ROSE*. It can be also called directly within the *caret* package. Recently a version of Rose has been made also available within *Scikit-learn* in Python
- *ROSE* can also be used to undersample the prevalent class.
- A combination of undersampling and oversampling (possibly cloning also data from the prevalent class) is sometimes useful
- Usually the test set is left unchanged. But the authors of ROSE suggest that for a more stable estimate of some of the metrics (such as AUC) also for the test set some new data can be cloned

SMOTE

- SMOTE is a popular method for data cloning.
- It generates new data in the rare class in order to obtain as many cases as in the prevalent class.
- The generation of new cases happens by first selecting randomly a case from the rare class and considering the K points which are closer to this point.
- New points are generated selecting a random position along the line connecting the selected point to one of the K neighbouring points
- There are many variants of SMOTE
- In R SMOTE is a function available within the R package *DMwR*. It can be also called directly within the *caret* package. SMOTE and its variants are availabel in Python Scikit-learn

Some practical issues

In real applications one has to choose:

1. when imbalance is a problem. Sometimes one can consider balancing the training set even when IR is not larger than 4-5. With small data sets it could be beneficial anyway. While in case of large (or huge datasets) simple/naive strategies such as undersampling the prevalent class may give good results
2. which method is most appropriate. This is a a matter of tastes. Usually, considering alternative strategies and comparing the performances is suggested. Any method has its merits and which is the more appropriate depends on specific characteristics of the data
3. the nature of the predictors matters.

- SMOTE and ROSE work better when most of the predictors are quantitative. With categorical predictors simple undersampling or oversampling can be more appropriate.
- when applying ROSE some extra care is needed in the the case of mixed variables (for instances zero inflated variables) or for limited variables.

Other remedies

1. Cost sensitive learning

In many cases the two errors have not the same importance. A different cost can be associated to the two errors FP and FN and a higher value is assigned to the most relevant error for the specific problem. The loss function to be minimized for the algorithm will change accordingly and it could help concentrating on predicting accurately the minority class

2. Modification of the standard algorithms

One of the most notable example is modification of boosting/bagging procedures to account for data imbalance. Note that actually the use of ensemble methods itself can alleviate the problem of data imbalance