6. Classification for Imbalanced Data

Statistical Modelling & Machine Learning

Jaejik Kim

Department of Statistics
Sungkyunkwan University

STA3036

Imbalanced Data

- Imbalanced data: One or more classes have very low proportions in the training data.

- Examples of imbalanced data:
  - Insurance claims: The number of customers with claims is very low relatively to the whole customers.
  - Manufacturing plants: The number of defected products is very low relatively to products from the plant.
  - Credit card fraud: Proportion of transactions due to fraud is very low.
  - Medical diagnosis: In clinical database, a disease group is fairly smaller than a normal group.
  - etc.

- Problem of class imbalance:
  - Poor prediction performance of classifiers.
  - Most classifiers tend to be biased towards the majority class.

Class Imbalance Problem

- The reasons for poor prediction:
  - The goal of most classifiers is to minimize the overall error to which the minority class contributes very little.
  - They usually assume that classes have the equal distribution.
  - They also assume that the errors coming from different classes have the same cost.
- Nature of imbalance problem:
  - Imbalanced class distribution: Relatively balanced distribution attains better results.
  - Sample size: As the training set size increases, the error rate caused by class imbalance decreases.
  - Class separability: Overlapped classes are more serious problems than imbalance. Linear classifiers are not sensitive to any amount of imbalance.

Difficulties of Standard Classifiers for Imbalanced Data

- Tree model:
  - The split may be terminated before small classes are detected.
  - Branches for small classes may be pruned as being susceptible to overfitting.
  - Correct prediction of small classes may not much contribute to reduce the error rate.
- Support vector machine (SVM):
  - SVM is believed to be less prone to the class imbalance problem because it depends on only few support vectors.
  - However, as training data gets more imbalanced, the support vector ratio between the large and small classes also becomes more imbalanced.
  - The small amount of error on the small class count for very little in estimation of decision boundary.
  - SVM simply learn to classify everything as the large class to make the margin the maximum and the error the minimum.

Model Assessment Criteria for Imbalanced Data

- For the classification with the class imbalance problem, accuracy (e.g., misclassification rate) is no longer a proper criterion.
- E.g., small class: 1% of training data $\Rightarrow$ Classifying all obs. into the large class $\Rightarrow$ Misclassification error rate: 1% (99% accuracy).
- Objective of classifiers for imbalanced data:
  - Balance the identification abilities between the small and large classes.
  - Improve the recognition success on the small class.

Model Assessment Criteria for Imbalanced Data

- Binary classification: Confusion matrix

	Predicted as Positive	Predicted as Negative
Actually Positive	True Positive (TP)	False Negative (FN)
Actually Negative	False Positive (FP)	True Negative (TN)

  - Positive class: Small class size but high identification importance.

- Measures from the confusion matrix:
  - True positive rate (Sensitivity): $TPR = \frac{TP}{TP+FN}$.
  - True negative rage (Specificity): $TNR = \frac{TN}{TN+FP}$.
  - Positive predictive value (Precision): $PPV = \frac{TP}{TP+FP}$.
  - False positive rate: $FPR = \frac{FP}{FP+TN}$.
  - False discovery rate: $FDR = \frac{FP}{FP+TP} = 1 - PPV$.

Model Assessment Criteria for Imbalanced Data

- If we focus on the prediction of the positive class, TPR and PPV are important.
- F-measure (F_1 score):

$$F = \frac{2\,TPR \cdot PPV}{TPR + PPV}.$$

  - The harmonic mean of TPR and PPV.
  - High F-measure $\Rightarrow$ High TPR and PPV
- G-mean: If we concern the performance of both classes, TPR and TNR are expected to be high simultaneously.

$$G = \sqrt{TPR \cdot TNR}.$$

Methods to Solve Class Imbalance Problem

Methods to Solve Class Imbalance Problem:

- Data-level approaches: Sampling methods.
  - Up-sampling (oversampling).
  - Down-sampling (undersampling).
  - Hybrid of up and down-sampling.
- Algorithmic-level approaches:
  - Alternate cut-off: Instead of usual cut-off prob. 0.5, use alternative cut-off values.
  - Adjusting prior probability: Use balanced prior prob. (LDA).
  - One-class learning (anomaly detection): Estimate the area of the large class using only obs. of the large class. Obs. outside of the area can be classified into the small class.
  - Cost-sensitive learning: Varying costs of different misclassification types.
- Ensemble-based approaches: Boosting with cost-sensitive learning.

Data-level approaches

- Sampling methods: Similar idea to unequal weights for individual obs.

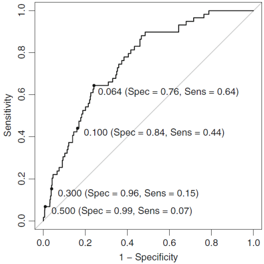
- Up-sampling: Add obs. to the small class to make the dataset balanced.
  - The small class is sampled with replacement until each class has the same size (This works like large weights for obs. in the small class).
  - SMOTE (synthetic minority over-sampling technique):
    1. Randomly select an obs. $\boldsymbol{x}_i$ from the small class.
    2. Find k obs. in the small class closest to $\boldsymbol{x}_i$ in the input space.
    3. Randomly select any/all of the k-nearest neighbors (say $\boldsymbol{x}_k$).
    4. Generate a uniform(0,1) random number u.
    5. New synthetic obs. for the small class: $\boldsymbol{x}^* = \boldsymbol{x}_i + u(\boldsymbol{x}_k - \boldsymbol{x}_i)$.

Data-level approaches

- Down-sampling: Remove obs. from the large class. to make the dataset balanced.
  - Random down-sampling: Randomly remove obs. from the large class or generate bootstrap samples with balanced class by stratification (random forests for the bootstrap sample with balanced classes).
  - Informative down-sampling: It selects only obs. in the large sample based on pre-specified selection criteria.
- Hybrid of up and down-sampling: Add obs. to the small class using the up-sampling and remove obs. from the large sample to have balanced training set.

Algorithmic-level approaches

- Alternate cut-off:
  - ROC (receiver operating characteristic) curve: Plot of (1-specificity) vs. sensitivity (i.e., *FPR* vs. *TPR*) by different cut-off values in binary classification.
  - AUC (area under ROC): Large AUC ⇒ Better classification model.

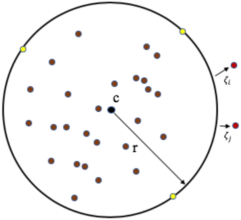


  - New cut-off: Find the point on ROC curve closest to the perfect model (sensitivity 1 & specificity 1).
  - Trade-off of sensitivity and specificity.

Algorithmic-level approaches

- Adjusting prior prob.:
  - In LDA and QDA, increase the prior prob. for the small class.

- One-class learning:
  - Support vector data description (SVDD): It considers the hypersphere with the center $\boldsymbol{c}$ and the radius r containing most training data by allowing small fraction of obs. to be outside of the hypersphere.



- SVDD:

$$\min_{r, \boldsymbol{c}, \xi_1, \ldots, \xi_n} r^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i,$$
$$\text{subject to } \|\phi(\boldsymbol{x}_i) - \boldsymbol{c}\|^2 \leq r^2 + \xi_i, \; i = 1, \ldots, n.$$

  - ν: Positive parameter that specifies the trade-off between the sphere volume and the number of outliers.
  - $\phi(\boldsymbol{x})$: Basis function of $\boldsymbol{x}$.
  - ξ_i: Slack variables.
  - By the optimality condition, $\boldsymbol{c} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i)$.
  - Another representation of the optimization above: Using the kernel function $\kappa(\cdot, \cdot)$,

$$\max_{\alpha_1, \ldots, \alpha_n} \sum_{i=1}^{n} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i=1}^{n} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
$$\text{subject to } \sum_{i=1}^{n} \alpha_i = 1, \; 0 \leq \alpha_i \leq \frac{1}{\nu n}, \; i = 1, \ldots, n.$$

Algorithmic-level approaches

- SVDD (Continued.)
  - $\alpha_i = 0$: Obs. inside of the sphere.
  - $0 < \alpha_i < \frac{1}{\nu n}$: Obs. that lies on the sphere boundary.
  - $\alpha_i = \frac{1}{\nu n}$: Obs. outside of the sphere.
  - Support vectors: Obs. corresponding to $\alpha_i > 0$.
  - Decision rule: If $\|\phi(\boldsymbol{x}) - \boldsymbol{c}\| > r \Rightarrow$ Outlier.

$$\|\phi(\boldsymbol{x}) - \boldsymbol{c}\|^2 = \sum_{i,j \in A} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{i \in A} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) + \kappa(\boldsymbol{x}, \boldsymbol{x}),$$

where A is a set of indices of support vectors.

Algorithmic-level approaches

- Cost-sensitive learning: Learning methods to optimize a cost or loss function that differentially weights specific types of errors.
  - $C(+,-)$: The cost of misclassifying a positive (small class) obs. as a negative (large class) obs. $\Rightarrow$ Cost of False Negative.
  - $C(-,+)$: The cost of False Positive.
  - $C(-,-) = C(+,+) = 0$: Correct classification $\Rightarrow$ No cost.
  - Class imbalance: Set $C(+,-) > C(-,+)$.
  - In general, cost-sensitive learning seeks to minimize the total misclassification cost.
  - Expected cost of classifying $\boldsymbol{x}$ into class $+$ or $-$: When $C(-,-) = C(+,+) = 0$,

$$
\begin{aligned}
E[C(+|\boldsymbol{x})] &= P(-|\boldsymbol{x})C(+,-) + P(+|\boldsymbol{x})C(+,+) \\
&= P(-|\boldsymbol{x})C(+,-), \\
E[C(-|\boldsymbol{x})] &= P(+|\boldsymbol{x})C(-,+).
\end{aligned}
$$

    where $P(-|\boldsymbol{x})$ & $P(+|\boldsymbol{x})$ are the prob. of classifying $\boldsymbol{x}$ into class $-$ & $+$, respectively.

Cost-sensitive Learning

- Decision rule: Minimization of the expected cost.
  Classify $\boldsymbol{x}$ into class $+$ if and only if

$$P(-|\boldsymbol{x})C(+,-) \leq P(+|\boldsymbol{x})C(-,+). \tag{1}$$

- Since $P(-|\boldsymbol{x}) = 1 - P(+|\boldsymbol{x})$, by arranging (1) in terms of $P(+|\boldsymbol{x})$, we get the optimal threshold

$$P(+|\boldsymbol{x}) \geq \frac{C(+,-)}{C(+,-)+C(-,+)}.$$

Ensemble-based Approaches

Balanced classification with two class ($Y = \{-1, 1\}$).

Algorithm: AdaBoost.M1

1. Initialize weights $w_i = 1/n,\ i = 1, \ldots, n$.
2. For $m = 1, \ldots, M$,
   - Fit a classifier $G_m(x)$ to the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ using the weights w_i.
   - Compute the weighted training error rate.

$$err_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq G_m(x))}{\sum_{i=1}^{n} w_i}.$$

   - Compute $\alpha_m = \log\left[\frac{1-err_m}{err_m}\right]$.
   - Update $w_i = w_i \exp\{\alpha_m I(y_i \neq G_m(x_i))\}$ (obs. that are difficult to classify have larger weights).
3. $G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$.

Ensemble-based Approaches

Imbalanced classification with two class ($Y = \{-1, 1\}$).
Algorithm: AdaC2

1. Initialize weights $w_i = 1/n$ & set costs c_i, $i = 1, \ldots, n$.
2. For $m = 1, \ldots, M$,
   - Fit a classifier $G_m(x)$ to the training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ using the weights w_i & costs c_i.
   - Compute the cost weighted training error rate.

$$err_m = \frac{\sum_{i=1}^{n} c_i w_i I(y_i \neq G_m(x))}{\sum_{i=1}^{n} c_i w_i}.$$

   - Compute $\alpha_m = \log\left[\frac{1-err_m}{err_m}\right]$.
   - Update $w_i = c_i w_i \exp\{\alpha_m I(y_i \neq G_m(x_i))\}$.
3. $G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$.