Model Selection, Evaluation Metrics, and Learning from Imbalanced Data

Piyush Rai

Introduction to Machine Learning (CS771A)

November 1, 2018
Yet to come..

- Today: Model Selection, Evaluation, Learning from Imbalanced Data
- Reinforcement Learning
- Ensemble methods (e.g., boosting)
- Learning with time series data
- Learning with limited supervision, other practical aspects (e.g., debugging ML algorithms)
Model Selection
What is Model Selection?

Given a set of models $\mathcal{M} = \{M_1, M_2, \ldots, M_R\}$, choose the model that is expected to do the best on the test data. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - $K$-Nearest Neighbors: Different choices of $K$
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees
  - Kernel Methods: Different choices of kernels
  - Regularized Models: Different choices of the regularization hyperparameter
  - Architecture of a deep neural network (\# of layers, nodes in each layer, activation function, etc)

- Different types of learning models (e.g., SVM, KNN, DT, etc.)

**Note:** Usually considered in supervised learning contexts but unsupervised learning too faces this issue (e.g., “how many clusters” when doing clustering)
Held-out Data

- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.

- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training.
- Train each model using the remaining training data.
- Evaluate error on the held-out data (*cross-validation*)
- Choose the model with the smallest held-out error.

- **Problems:**
  - Wastes training data. Typically used when we have plenty of training data.
  - What if there was an unfortunate train/held-out split?
**K-fold Cross-Validation**

- Create $K$ (e.g., 5 or 10) equal sized partitions of the training data
- Each partition has $N/K$ examples
- Train using $K-1$ partitions, validate on the remaining partition
- Repeat this $K$ times, each with a different validation partition

- Average the $K$ validation errors
- Choose the model that gives the smallest average validation error
Special case of $K$-fold CV when $K = N$. Each partition is now a single example.

Train using $N - 1$ examples, validate on the remaining example.

Repeat the same $N$ times, each with a different validation example.

Average the $N$ validation errors. Choose the model with smallest error.

Can be expensive in general, especially for large $N$.

Very efficient when used for selecting $K$ in nearest neighbor methods (NN requires no training).
Random Subsampling based Cross-Validation

- Subsample a fixed fraction $\alpha N$ ($0 < \alpha < 1$) as examples as validation set
- Train using the rest of the examples, calculate the validation error
- Repeat $K$ times, each with a different, randomly chosen validation set

- Average the $K$ validation errors. Choose the model with smallest error
Finding the best hyperparameters

- Each setting of the hyperparameter values is a different model
- Picking the best model = Finding the best hyperparameter setting (which gives best heldout error)
- Picking the best hyperparameter(s) means cross-validation on lots of different models
- Typically done using grid search. But expensive if there are lots of hyperparameters
- The search can be “automated” using hyperparameter search techniques

**Idea:** Instead of grid-search, sequentially decide which hyperparam, config. should be tried next
  - Random Search (see “Random Search for Hyper-Parameter Optimization”)
  - Bayesian Optimization (see “Practical Bayesian Optimization of Machine Learning Algorithms”)
Metrics for Evaluating ML Algorithms
Binary Classification Evaluation Metrics

- Easy to visualize via a $2 \times 2$ matrix (Confusion Matrix)
- Sum of diagonals = # of correct predictions
- Sum of off-diagonals = # of mistakes
- Standard evaluation measure is classification accuracy
  \[
  \text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}
  \]

Various other metrics are also used to evaluate classification performance

- **Precision** = Of all positive predictions, what fraction is actually positives
- **Recall** = Of all actual positives, what fraction is predicted as positives

\[
\begin{align*}
\text{Precision (P)} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\
\text{Recall (R)} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\
\text{F1-score} &= \frac{2\text{PR}}{\text{P} + \text{R}} \quad \text{(harmonic mean)}
\end{align*}
\]
True Positive Rate (TPR) and False Positive Rate (FPR) are also commonly used metrics.

- **TPR** is the same as recall: Fraction of actual positives predicted as positives
  
  \[
  TPR = \frac{TP}{TP + FN}
  \]

- **FPR** is the fraction of actual negatives wrongly predicted as positive
  
  \[
  FPR = \frac{FP}{TN + FP}
  \]
Most classifiers predict a score (a real-valued number or a probability)

Can set a threshold to decide what to call positive and what to call negative

Can adjust the threshold to control the TPR and FPR

Plot of TPR vs FPR for all possible values of the threshold is called Area Under the Receiving Operating Curve (AUCROC or AUC)

The max AUC score is 1. AUC = 0.5 means close to random.
For $K$ classes, we will have a $K \times K$ confusion matrix.

Can define precision and recall w.r.t. each class.
Regression Evaluation Metrics

- Assume true responses $y = [y_1, \ldots, y_N]$, predicted responses $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_N]$
- Traditional metric: Residual sum of squares: $SS_{res} = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$
- Suppose the mean of true responses is $\mu = \frac{\sum_{n=1}^{N} y_n}{N}$
- Define total sum of squares: $SS_{tot} = \sum_{n=1}^{N} (y_n - \mu)^2$. Prop. to original variance
- Regression sum of squares: $SS_{reg} = \sum_{n=1}^{N} (\hat{y}_n - \mu)^2$. Prop. to variance “explained” by the model
- The coefficient of determination metric is defined as
  
  $R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{n=1}^{N} (y_n - \hat{y}_n)^2}{\sum_{n=1}^{N} (y_n - \mu)^2}$
- A close to perfect model will have $R^2$ close to 1
- For linear regression, $SS_{tot} = SS_{reg} + SS_{res} \Rightarrow R^2 = \frac{SS_{reg}}{SS_{tot}}$ (fraction of explained variance)
Evaluation Metrics for Unsupervised Learning

- Some clustering metrics exist if the ground truth clusters are known (rarely the case)
  - Accuracy, normalized mutual information (NMI), rand index, purity, etc.
  - But need to account for cluster label permutations

- Metrics if no ground truth is known
  - Distortion: Sum of squared errors from the closest clusters (need to penalize the number of clusters)
  - Distortion on a “held-out data” (not used to learn the clusters)
  - For probabilistic models, can look at the negative log-likelihood (penalized by number of clusters)

- Distortion/reconstruction error can also be used for evaluating dimensionality reduction methods

- External evaluation is often preferred when evaluating unsupervised learning models
  - Use the new representation to train a supervised learning model
Learning from Imbalanced Data
Learning from Imbalanced Data

- Consider binary classification
- Often the classes are highly imbalanced

Should I feel happy if my classifier gets 99.997% classification accuracy on test data?
Learning from Imbalanced Data

- Other problems can also exhibit imbalance (e.g., binary matrix completion)

- Should I feel happy if my matrix completion model gets 99.999% matrix completion accuracy on the test entries?
True Definition of Imbalance Data?

- Debatable..
- Scenario 1: 100,000 negative and 1000 positive examples
- Scenario 2: 10,000 negative and 10 negative examples
- Scenario 3: 1000 negative and 1 negative example
- Usually, imbalance is characterized by absolute rather than relative rarity
  - Finding needles in a haystack..
Minimizing Loss

- Any model to minimize the loss, e.g.,

\[
\hat{w} = \text{arg min}_w \sum_{n=1}^{N} \ell(y_n, w^T x_n)
\]

Matrix Completion: \((\hat{U}, \hat{V}) = \text{arg min}_{U,V} \|X - UV^T\|^2\)

.. will usually get a high accuracy

- However, it will be highly biased towards predicting the majority class
  - Thus accuracy alone can’t be trusted as the evaluation measure if we care more about predicting minority class (say positive) correctly
  - Need to use metrics such as precision, recall, F1 score, AUC, etc (that specifically care about positives)
Dealing with Class Imbalance

- Modifying the training data (the class distribution)
  - Undersampling the majority class
  - Oversampling the minority class
  - Reweighting the examples

- Modifying the learning model
  - Use loss functions customized to handle class imbalance

- Reweighting can be also seen as a way to modify the loss function
Modifying the Training Data
Undersampling

- Throws away a lot of data/information. But efficient to train

Create a new training data set by:
- including all $k$ “positive” examples
- randomly picking $k$ “negative” examples
Oversampling

- Th repeated examples simply contribute multiple times to the loss function
- Some oversampling methods (SMOTE) based on creating synthetic examples from minority class
**Reweighting Examples**

- Similar effect as oversampling but is more efficient (because there is no multiplicity of examples)
- Also requires a model that can learn with weighted examples
Modifying the Loss Function
Loss Functions Customized for Imbalanced Data

- Traditional loss functions have the form: \( \sum_{n=1}^{N} \ell(y_n, f(x_n)) \)

- Such loss functions look at positive and negative examples individually, so the majority class tends to overwhelm the minority class

- Reweighting the loss function differently for different classes can be one way to handle class imbalance, e.g., \( \sum_{n=1}^{N} C_{y_n} \ell(y_n, f(x_n)) \)

- Alternatively, we can use loss functions that look at pairs of examples (a positive example \( x^+_n \) and a negative example \( x^-_m \)). For example:
  \[
  \ell(f(x^+_n), f(x^-_m)) = \begin{cases} 
  0, & \text{if } f(x^+_n) > f(x^-_m) \\
  1, & \text{otherwise}
  \end{cases}
  \]

- These are called “pairwise” loss functions

- Why is it a good loss function for imbalanced data?
Pairwise Loss Functions

- Using pairs with one +ve and one -ve doesn’t let one class overwhelm other

\[
\sum_{n=1}^{N_+} \sum_{m=1}^{N_-} \ell(f(x_n^+), f(x_m^-)) + \lambda R(f)
\]

- The pairwise loss function only cares about the difference between scores of a pair of positive and negative examples
  - Want the positive ex. to have higher score than the negative ex., which is similar in spirit to maximizing the AUC (Area Under the ROC Curve) score
  - AUC (intuitively): The probability that a randomly chosen pos. example will have a higher score than a randomly chosen neg. example
  - Empirical AUC of \( f \) on a training set with \( N_+ \) and \( N_- \) pos. and neg. ex.

\[
\text{AUC}(f) = \frac{1}{N_+ N_-} \sum_{n=1}^{N_+} \sum_{m=1}^{N_-} \mathbb{1}(f(x_n^+) > f(x_m^-))
\]

- Note: Commonly used pairwise loss functions maximize a proxy of the AUC score (or closely related measures such as F1 score)
Pairwise Loss Functions

- A proxy based on hinge-loss like pairwise loss function for a linear model

\[
\ell(w, x_n^+, x_m^-) = \max\{0, 1 - (w^\top x_n^+ - w^\top x_m^-)\} = \max\{0, 1 - w^\top (x_n^+ - x_m^-)\}
\]

- It basically says that the difference between scores of positive and negative examples should be at least 1 (which is like a “margin”)

- The overall objective will have the form

\[
\frac{||w||^2}{2} + \sum_{n=1}^{N_+} \sum_{m=1}^{N_-} \ell(w, x_n^+, x_m^-)
\]

- Convex objective (if using the hinge loss). Can be efficiently optimized using stochastic optimization (see “Online AUC Maximization”, Zhao et al, 2011)

- Note: Similar ideas can be used for solving binary matrix factorization and matrix completion problems as well

  - E.g., if matrix entry \(X_{nm} = 1\) and \(X_{nm'} = -1\) then \(\text{loss} = 0\) if \(u_n^\top v_m > u_n^\top v_{m'}\)
Summary

- Imbalanced data needs to be handled with care
- Classification accuracies can be very misleading for such data
  - Should look at measures such as precision, recall, or other variants that are robust to class imbalance
- Sampling heuristics work reasonably on many data sets
- More principled approaches are based on modifying the loss function
  - Instead of minimizing the classification error, optimize w.r.t. other metrics such as precision, recall, F1 score, AUC, etc.
- Another way to look at this problem could be as an anomaly detection problem (minority class is anomaly) or density estimation problem