Ensemble Learning

Danna Gurari
University of Texas at Austin
Spring 2020

https://www.ischool.utexas.edu/~dannag/Courses/IntroToMachineLearning/CourseContent.html
Review

• Last week:
  • Evaluating Machine Learning Models Using Cross-Validation
  • Naïve Bayes
  • Support Vector Machines

• Assignments (Canvas):
  • Problem set 4 due yesterday
  • Lab assignment 2 due next week

• Next week: class will be taught by Samreen Anjum

• Questions?
Today’s Topics

• One-vs-all multiclass classification

• Classifier confidence

• Evaluation: ROC and PR-curves

• Ensemble learning

• Lab
Today’s Topics

• One-vs-all multiclass classification

• Classifier confidence

• Evaluation: ROC and PR-curves

• Ensemble learning

• Lab
Recall: Binary vs Multiclass Classification

**Binary**: distinguish 2 classes

**Multiclass**: distinguish 3+ classes

Figure Source: http://mlwiki.org/index.php/One-vs-All_Classification
Recall: Binary vs Multiclass Classification

**Binary**: distinguish 2 classes

- Perceptron
- Adaline
- Support Vector Machine

**Multiclass**: distinguish 3+ classes

- Nearest Neighbor
- Decision Tree
- Naïve Bayes
One-vs-All (aka, One-vs-Rest): Applying Binary Classification Methods for Multiclass Classification

• Given ‘N’ classes, train ‘N’ different classifiers: a single classifier trained per class, with the samples of that class as positive samples and all other samples as negatives; e.g.,

Figure Source: http://mlwiki.org/index.php/One-vs-All_Classification
One-vs-All (aka, One-vs-Rest): Limitation

• Often leads to unbalanced distributions during learning; i.e., when the set of negatives is much larger than the set of positives

Figure Source: http://mlwiki.org/index.php/One-vs-All_Classification
One-vs-All (aka, One-vs-Rest): Class Assignment

• (Imperfect) Approach: use majority vote from N classifiers; since multiple classes can be predicted for a sample, this requires the classifiers to produce a real-valued confidence score for its decision.

Figure Source: http://mlwiki.org/index.php/One-vs-All_Classification
Today’s Topics

• One-vs-all multiclass classification

• Classifier confidence

• Evaluation: ROC and PR-curves

• Ensemble learning

• Lab
Classifier Confidence: Beyond Classification

• Indicate both the predicted class and uncertainty about the choice

• When and why might you want to know about the uncertainty?
  • e.g., weather forecast: 25% chance it will rain today
  • e.g., medical treatment: when unconfident, start a patient on a drug at a lower dose and decide later whether to change the medication or dose
Classifier Confidence: How to Measure for K-Nearest Neighbors?

- Proportion of neighbors with label y; e.g.,

When K=3:

Classifier Confidence: How to Measure for Decision Trees?

- Proportion of training samples with label $y$ in the leaf where for the test sample; e.g.,

```
14 yes; 6 no
100 no
120 yes
30 yes; 70 no
```
Classifier Confidence: How to Measure for Naïve Bayes?

• Conditional probability $P(Y|X)$ for the most probable class
Classifier Confidence: How to Measure for Support Vector Machines?

- Distance to the hyperplane: e.g.,

http://chem-eng.utoronto.ca/~datamining/dmc/support_vector_machine.htm
Classifier Confidence vs Probability

• Classifiers can make mistakes in estimating their confidence level

• External calibration procedures can address this issue (e.g., using calibration curves/reliability diagrams)
Today’s Topics

• One-vs-all multiclass classification

• Classifier confidence

• Evaluation: ROC and PR-curves

• Ensemble learning

• Lab
Classification from a Classifier’s Confidence

• Observation: A threshold must be chosen to define the point at which the example belongs to a class or not

• Motivation: how to choose the threshold?
  • Default is 0.5
  • Yet, it can tuned to avoid different types of errors
Review: Confusion Matrix for Binary Classification

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P$</td>
</tr>
<tr>
<td>$P$</td>
<td>True Positives (TP)</td>
</tr>
<tr>
<td></td>
<td>False Negatives (FN)</td>
</tr>
<tr>
<td>$N$</td>
<td>False Positives (FP)</td>
</tr>
<tr>
<td></td>
<td>True Negatives (TN)</td>
</tr>
</tbody>
</table>
Receiver Operating Characteristic (ROC) curve

Summarizes performance based on the positive class
- A positive prediction is either correct (TP) or not (FP)

\[
FPR = \frac{FP}{N} = \frac{FP}{FP + TN}
\]

\[
REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}
\]
Receiver Operating Characteristic (ROC) curve

To create, vary prediction threshold and compute TPR and FPR for each threshold

Summarizes performance based on the positive class
- A positive prediction is either correct (TP) or not (FP)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$
Receiver Operating Characteristic (ROC) curve

What is the coordinate for a perfect predictor?

Summarizes performance based on the positive class
- A positive prediction is either correct (TP) or not (FP)

\[ FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \]

\[ REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \]
Which of the first three methods performs best overall?

Summarizes performance based on the positive class
- A positive prediction is either correct (TP) or not (FP)

\[
FPR = \frac{FP}{N} = \frac{FP}{FP + TN}
\]

\[
REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}
\]
ROC Curve: Multiclass Classification

• Plot curve per class:

Precision-Recall (PR) Curve

Summarizes performance based only on the positive class (ignores true negatives):

\[ PRE = \frac{TP}{TP + FP} \]

\[ REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \]
Precision-Recall (PR) Curve

To create, vary prediction threshold and compute precision and recall for each threshold

Summarizes performance based only on the positive class (ignores true negatives):

\[ PRE = \frac{TP}{TP + FP} \]

\[ REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \]
Precision-Recall (PR) Curve

What is the coordinate for a perfect predictor?

Summarizes performance based only on the positive class (ignores true negatives):

\[
PRE = \frac{TP}{TP + FP}
\]

\[
REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}
\]
PR Curve: Area Under Curve (AUC)

- Which classifier is the best?
PR Curve: Multiclass Classification

- Plot curve per class:

1. Assume you are building a classifier for these applications:
   • Detecting offensive content online
   • Medical diagnoses
   • Detecting shoplifters
   • Deciding whether a person is guilty of a crime

   What classifier threshold would you choose for each application and why?

2. When would you choose to evaluate with a PR curve versus a ROC curve?

   • Each student should submit a response in a Google Form (tracks attendance)
     • What is the area under the ROC and PR curves for a perfect classifier?
Today’s Topics

• One-vs-all multiclass classification

• Classifier confidence

• Evaluation: ROC and PR-curves

• Ensemble learning

• Lab
Idea: How Many Predictors to Use?

More than 1: Ensemble
Why Choose Ensemble Instead of an Algorithm?

• Reduces probability for making a wrong prediction, assuming:
  • Classifiers are independent (not true in practice!)

• Suppose:
  • n classifiers for binary classification task
  • Each classifier has same error rate $\epsilon$
  • Probability mass function indicates the probability of error from an ensemble:

$$P(y \geq k) = \sum_{k} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} = \epsilon_{\text{ensemble}}$$

  - Number of classifiers
  - Classifier error rate
  - Error probability
  - # ways to choose $k$ subsets from set of size $n$
  - e.g., $n = 11$, $\epsilon = 0.25$; $k = 6$: probability of error is $\sim 0.034$ which is much lower than probability of error from a single algorithm (0.25)
Why Choose Ensemble Instead of an Algorithm?

• Reduces probability for making a wrong prediction, assuming:
  • Classifiers are independent (not true in practice!)

• Suppose:
  • n classifiers for binary classification task
  • Each classifier has same error rate \( \epsilon \)
  • Probability of error from individual classifiers is \( \epsilon \). For an ensemble:

\[
P(y \geq k) = \sum_{k} \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} = \epsilon_{\text{ensemble}}
\]

• e.g., n = 11, \( \epsilon = 0.25 \); k = 6: probability of error is \(~0.034\) which is much lower than probability of error from a single algorithm (0.25)

How to Get Diverse Classifiers?
Why Choose Ensemble Instead of an Algorithm?

- Reduces probability for making a wrong prediction, assuming:
  - Classifiers are independent (not true in practice!)

Suppose:
- $n$ classifiers for binary classification task
- Each classifier has same error rate
- Probability mass function indicates the probability of error from an ensemble:

$$P(y \geq k) = \sum_{k} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} = \epsilon_{ensemble}$$

- e.g., $n=11$, $\epsilon=0.25$, $k=6$: probability of error is ~0.034 which is much lower than probability of error from a single algorithm (0.25)

1. Use different algorithms
2. Use different features
2. Use different training data
How to Predict with an Ensemble?

• Majority Voting
  • Return most popular prediction from multiple prediction algorithms

• Bootstrap Aggregation, aka Bagging
  • Resample data to train algorithm on different random subsets

• Boosting
  • Reweight data to train algorithms to specialize on different “hard” examples

• Stacking
  • Train a model that learns how to aggregate classifiers’ predictions
How to Predict with an Ensemble of Algorithms?

• Majority Voting
  • Return most popular prediction from multiple prediction algorithms

• Bootstrap Aggregation, aka Bagging
  • Train algorithm repeatedly on different random subsets of the training set

• Boosting
  • Train algorithms that each specialize on different “hard” training examples

• Stacking
  • Train a model that learns how to aggregate classifiers’ predictions
Majority Voting

Figure Credit: Raschka & Mirjalili, Python Machine Learning.
Majority Voting

Prediction Model → Prediction

Prediction Model → Prediction

Prediction Model → Prediction

Majority Vote
Majority Voting: Binary Task

e.g., “Is it sunny today?”

- “Yes”
- “No”
- “Yes”
- “Yes”

“Yes”
Majority Voting: “Soft” (not “Hard”)

Prediction Model

Probability

Prediction Model

Probability

Prediction Model

Probability

Majority Vote
Majority Voting: Soft Voting on Binary Task

e.g., “Is it sunny today?”

90% “Yes”
20% Yes
55% “Yes”
45% “Yes”

“Yes” (210/4 = 52.5% Yes)
Plurality Voting: Non-Binary Task

e.g., “What object is in the image?”

“Cat”

“Dog”

“Pig”

“Cat”

“Cat”
Majority Voting: Regression

e.g., “Is it sunny today?”

90% “Yes”
20% Yes
55% “Yes”
45% “Yes”

52.5% (average prediction)
Majority Voting: Example of Decision Boundary

Figure Credit: Raschka & Mirjalili, Python Machine Learning.
How to Predict with an Ensemble of Algorithms?

• Majority Voting
  • Return most popular prediction from multiple prediction algorithms

• Bootstrap Aggregation, aka Bagging
  • Train algorithm repeatedly on different random subsets of the training set

• Boosting
  • Train algorithms that each specialize on different “hard” training examples

• Stacking
  • Train a model that learns how to aggregate classifiers’ predictions
Bagging

Figure Credit: Raschka & Mirjalili, Python Machine Learning.
Bagging: Training

• Build ensemble from “bootstrap samples” drawn with replacement
• e.g.,

<table>
<thead>
<tr>
<th>Sample indices</th>
<th>Bagging round 1</th>
<th>Bagging round 2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>7</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>7</td>
<td>...</td>
</tr>
</tbody>
</table>

Duplicate data can occur for training
Some examples missing from training data; e.g., round 1
Each classifier trained on different subset of data

Figure Credit: Raschka & Mirjalili, Python Machine Learning.
Bagging: Training

- Build ensemble from “bootstrap samples” drawn with replacement
- e.g.,

<table>
<thead>
<tr>
<th>Sample indices</th>
<th>Bagging round 1</th>
<th>Bagging round 2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Class Demo:
- Pick a number from the bag

Figure Credit: Raschka & Mirjalili, Python Machine Learning.
Bagging: Predicting

- Predict as done for “majority voting”
  - e.g., “hard” voting
  - e.g., “soft” voting
  - e.g., averaging values for regression
Bagging: Random Forest

• Build ensemble from “bootstrap samples” drawn with replacement
• e.g.,

<table>
<thead>
<tr>
<th>Sample indices</th>
<th>Bagging round 1</th>
<th>Bagging round 2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>7</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>7</td>
<td>...</td>
</tr>
</tbody>
</table>

Fit decision trees by also selecting random feature subsets

Figure Credit: Raschka & Mirjalili, Python Machine Learning.
How to Predict with an Ensemble of Algorithms?

• Majority Voting
  • Return most popular prediction from multiple prediction algorithms

• Bootstrap Aggregation, aka Bagging
  • Train algorithm repeatedly on different random subsets of the training set

• Boosting
  • Train algorithms that each specialize on different “hard” training examples

• Stacking
  • Train a model that learns how to aggregate classifiers’ predictions
Boosting

• Key idea: sequentially train predictors that each try to correctly predict examples that were hard for previous predictors

• Original Algorithm:
  • Train classifier 1: use random subset of examples without replacement
  • Train classifier 2: use a second random subset of examples without replacement and add 50% of examples misclassified by classifier 1
  • Train classifier 3: use examples that classifiers 1 and 2 disagree on
  • Predict using majority vote from 3 classifiers
Boosting – Adaboost (Adaptive Boosting)

1. Assign equal weights to all examples

2. • Assign larger weights to previous misclassifications
   • Assign smaller weights to previous correct classifications

3. • Assign larger weights to training samples $C_1$ and $C_2$ disagree on
   • Assign smaller weights to previous correct classifications

4. Predict with weighted majority vote

Freund and Schapire, Experiments with a New Boosting Algorithm, 1996.
Boosting – Adaboost (Adaptive Boosting)

e.g., 1d dataset

<table>
<thead>
<tr>
<th>Sample indices</th>
<th>$x$</th>
<th>$y$</th>
<th>Weights</th>
<th>$\hat{y}(x \leq 3.0)$?</th>
<th>Correct?</th>
<th>Updated weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
<tr>
<td>4</td>
<td>4.0</td>
<td>-1</td>
<td>0.1</td>
<td>-1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>-1</td>
<td>0.1</td>
<td>-1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
<tr>
<td>6</td>
<td>6.0</td>
<td>-1</td>
<td>0.1</td>
<td>-1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
<tr>
<td>7</td>
<td>7.0</td>
<td>1</td>
<td>0.1</td>
<td>-1</td>
<td>No</td>
<td>0.167</td>
</tr>
<tr>
<td>8</td>
<td>8.0</td>
<td>1</td>
<td>0.1</td>
<td>-1</td>
<td>No</td>
<td>0.167</td>
</tr>
<tr>
<td>9</td>
<td>9.0</td>
<td>1</td>
<td>0.1</td>
<td>-1</td>
<td>No</td>
<td>0.167</td>
</tr>
<tr>
<td>10</td>
<td>10.0</td>
<td>-1</td>
<td>0.1</td>
<td>-1</td>
<td>Yes</td>
<td>0.072</td>
</tr>
</tbody>
</table>

Round 1: training data, weights, predictions

Round 2: update weights

Raschka and Mirjalili; Python Machine Learning
Boosting – Adaboost (Adaptive Boosting)

e.g., 1d dataset

1. Compute error rate (sum misclassified examples’ weights):
   \[ \varepsilon = 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 0 + 0.1 \times 1 + 0.1 \times 1 + 0.1 \times 1 \\ + 0.1 \times 1 + 0.1 \times 0 = \frac{3}{10} = 0.3 \]

2. Compute coefficient used to update weights and make majority vote prediction:
   \[ \alpha_j = 0.5 \log \left( \frac{1 - \varepsilon}{\varepsilon} \right) \approx 0.424 \]

3. Update weight vector:
   \[ w := w \times \exp \left( -\alpha_j \times \hat{y} \times y \right) \]
   • Correct predictions will decrease weight and vice versa
   
   \[ 0.1 \times \exp(-0.424 \times 1 \times 1) \approx 0.065 \quad 0.1 \times \exp(-0.424 \times (-1) \times 1) \approx 0.153 \]

4. Normalize weights to sum to 1:
   \[ \sum_i w_i = 7 \times 0.065 + 3 \times 0.153 = 0.914 \]
   \[ w := \frac{w}{\sum_i w_i} \]

Raschka and Mirjalili; Python Machine Learning
Boosting – Adaboost (Adaptive Boosting)

To predict, use $\alpha$ calculated for each classifier as its weight when voting with all trained classifiers.

Idea: value the prediction of each classifier based on the accuracies they had on the training dataset.

Raschka and Mirjalili; Python Machine Learning
How to Predict with an Ensemble of Algorithms?

• Majority Voting
  • Return most popular prediction from multiple prediction algorithms

• Bootstrap Aggregation, aka Bagging
  • Train algorithm repeatedly on different random subsets of the training set

• Boosting
  • Train algorithms that each specialize on different “hard” training examples

• Stacking
  • Train a model that learns how to aggregate classifiers’ predictions
Stacked Generalization, aka Stacking

• Train meta-learner to learn the optimal weighting of each classifiers’ predictions for making the final prediction

• Algorithm:
  1. Split dataset into three disjoint sets.
  2. Train several base learners on the first partition.
  3. Test the base learners on the second partition and third partition.
  4. Train meta-learner on second partition using classifiers’ predictions as features
  5. Evaluate meta-learner on third prediction using classifiers’ predictions as features

Ensemble Learner Won Netflix Prize “Challenge”

• In 2009 challenge, winning team won $1 million using ensemble approach:
  • [https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf](https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf)
  • Dataset: 5-star ratings on 17770 movies from 480189 “anonymous” users collected by Netflix over ~7 years. In total, the number of ratings is 100,480,507.

![Diagram of data sets](attachment:image.png)

• Netflix did not use ensemble recommendation system. Why?
  • “We evaluated some of the new methods offline but the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment” - [https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429](https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429)
  • Computationally slow and complex from using “sequential” training of learners
Today’s Topics

• One-vs-all multiclass classification

• Classifier confidence

• Evaluation: ROC and PR-curves

• Ensemble learning

• Lab