Lab #6: Tweaking Classifiers

Everything Data
CompSci 216 Spring 2015
Announcements (Mon. Feb. 23)

• **Project team formation** due tonight!
  – Submit `team.txt` to `proj-team`
  – Don’t confuse it with submitting `team.txt` for this lab

• **Same team assignment** as last lab
  – Seating by project team assignment will begin next week

• **Sample solution to Homework #6** posted

• **Jun is moving office hours** to Fridays 3:30-4:45pm in LSRC D327
Winners from Lab #5

Team 9:
• Anthony Hagouel
• Dianwen Li
• Janvi Shah
• Alexander Shih
Format of this lab

• Introduction
• Two challenges
• Discussion
Introducing Lab #7

So you are not happy with your classifier (or any prediction algorithm in general); what can you do?

• *Try a different algorithm?*
• *Try different parameters of the algorithm?*
• *Get more training examples?*
• *Try fewer features?*
• *Try more features?*
Team challenge 1

- In Homework #6, we used hundreds of votes as features to predict party affiliation.
- It turned out that 10 arbitrary votes were enough!
- But would any 10 work? Can you find 10 bad features to screw up Naïve Bayes?

5% extra credit if you get <70% accuracy
First to achieve the lowest accuracy wins!
Feature selection

Why?
• Faster, less prone to overfitting, easier to interpret model

How?
• One simple approach: rank all features by some utility measure, and use only the top $k$
  – A popular utility measure is $\chi^2$
    • A high $\chi^2$ means it’s unlikely that the feature value and the class label are independent

• When does this fail?
Tweaking classifiers: Scenario 1

If you increase # training examples and see
• Test error continues to decrease
• Gap between test and training errors remains big

More training examples will help
Possible overfitting, consider fewer features

From Andrew Ng: http://see.stanford.edu/materials/aimlcs229/ML-advice.pdf
Tweaking classifiers: Scenario 2

But if

- Even training error is unacceptably high
- Gap between test and training error is narrow

Try additional and/or alternative features

From Andrew Ng: [http://see.stanford.edu/materials/aimlcs229/ML-advice.pdf](http://see.stanford.edu/materials/aimlcs229/ML-advice.pdf)
Tuning algorithm parameters

• Systematic search of parameter space
  – You don’t get to see test data, yet
  – Use cross-validation on training data

• Understanding of how algorithms (and parameters) work will help
  – E.g., if you observe overfitting, try increasing $k$ in the $k$NN classifier
Intuition behind linear SVM

• Points labeled with two classes
• Find a hyperplane separating the two classes

But which one would you pick?
Max-margin classifier

☞ Pick the hyperplane with the widest margin
– Turns out this problem can be solved efficiently

✓ Better!
Not linearly separable?

• Transform data to make it separable, e.g.:

\[(x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2)\]

Instead of really transforming data, pick a distance metric (kernel), and the “kernel trick” will keep SVM efficient!

• In other cases, you can make the SVM “soft”
  – Allow misclassified points but pay a penalty

Team challenge 2

• Classify articles into 6 newsgroups
• Naïve Bayes vs. kNN vs. SVM
• Various tweaking can be done by modifying lab.py and supplying additional command-line arguments

5% extra credit if you get >0.79 F-measure
First to achieve the highest accuracy wins!
Lessons learned

• Getting started should be easy; getting really good results is hard
• So many tools and knobs, so little time!
  – Automatic searches through feature and parameter spaces can help
  – Better understanding of the tools/knobs helps
• “Careful design” vs. “build-and-fix”
  – Andrew Ng
Finally

• Remember to submit lab team.txt under lab06 by midnight
• And also project team.txt under proj-team
• Slides on and sample solutions to Lab #6 will be posted by tonight