Choosing a Machine Learning Classifier

How do you know what machine learning algorithm to choose for your classification problem? Of course, if you really care about accuracy, your best bet is to test out a couple different ones (making sure to try different parameters within each algorithm as well), and select the best one by cross-validation. But if you're simply looking for a "good enough" algorithm for your problem, or a place to start, here are some general guidelines I've found to work well over the years.

How large is your training set?

If your training set is small, high bias/low variance classifiers (e.g., Naive Bayes) have an advantage over low bias/high variance classifiers (e.g., kNN), since the latter will overfit. But low bias/high variance classifiers start to win out as your training set grows (they have lower asymptotic error), since high bias classifiers aren't powerful enough to provide accurate models.

You can also think of this as a generative model vs. discriminative model distinction.

Advantages of some particular algorithms

Advantages of Naive Bayes: Super simple, you're just doing a bunch of counts. If the NB conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. And even if the NB assumption doesn't hold, a NB classifier still often does a great job in practice. A good bet if want something fast and easy that performs pretty well. Its main disadvantage is that it can't learn interactions between features (e.g., it can't learn that although you love movies with Brad Pitt and Tom Cruise, you hate movies where they're together).

Advantages of Logistic Regression: Lots of ways to regularize your model, and you don't have to worry as much about your features being correlated, like you do in Naive Bayes. You also have a nice probabilistic interpretation, unlike decision trees or SVMs, and you can easily update your model to take in new data (using an online gradient descent method), again unlike decision trees or SVMs. Use it if you want a probabilistic framework (e.g., to easily adjust classification thresholds, to say when you're unsure, or to get confidence intervals) or if you expect to receive more training data in the future that you want to be able to quickly incorporate into your model.

Advantages of Decision Trees: Easy to interpret and explain (for some people – I'm not sure I fall into this camp). They easily handle feature interactions and they're non-parametric, so you don't have to worry about outliers or whether the data is linearly separable (e.g., decision trees easily take care of cases where you have class A at the low end of some feature x, class B in the mid-range of feature x, and A again at the high end). One disadvantage is that they don't support online learning, so you have to rebuild your tree when new examples come on. Another disadvantage is

that they easily overfit, but that's where ensemble methods like random forests (or boosted trees) come in. Plus, random forests are often the winner for lots of problems in classification (usually slightly ahead of SVMs, I believe), they're fast and scalable, and you don't have to worry about tuning a bunch of parameters like you do with SVMs, so they seem to be quite popular these days.

Advantages of SVMs: High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if you're data isn't linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm. Memory-intensive, hard to interpret, and kind of annoying to run and tune, though, so I think random forests are starting to steal the crown.

But…

Recall, though, that better data often beats better algorithms, and designing good features goes a long way. And if you have a huge dataset, then whichever classification algorithm you use might not matter so much in terms of classification performance (so choose your algorithm based on speed or ease of use instead).

And to reiterate what I said above, if you really care about accuracy, you should definitely try a bunch of different classifiers and select the best one by cross-validation. Or, to take a lesson from the Netflix Prize (and Middle Earth), just use an ensemble method to choose them all.

LinkedIn
Email

Recent Posts

Surge: Data Labeling You Can Trust

Exploring LSTMs

Moving Beyond CTR: Better Recommendations Through Human Evaluation

Propensity Modeling, Causal Inference, and Discovering Drivers of Growth

Product Insights for Airbnb

Improving Twitter Search with Real-Time Human Computation

Edge Prediction in a Social Graph: My Solution to Facebook's User Recommendation Contest on Kaggle

Soda vs. Pop with Twitter

Infinite Mixture Models with Nonparametric Bayes and the Dirichlet Process

Instant Interactive Visualization with d3 + ggplot2

Movie Recommendations and More

via MapReduce and Scalding

Quick Introduction to ggplot2

Introduction to Conditional Random Fields

Winning the Netflix Prize: A Summary

Stuff Harvard People Like