TTIC 31020: Introduction to Statistical Machine Learning
Autumn 2015

Problem Set #2

Out: November 4, 2016

Due: Friday November 18, 11:59pm

Instructions

Please read these instructions carefully and follow them precisely. Feel free to ask the instructor if anything is unclear!

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want: typeset the solution in LaTeX (recommended), type it in Word or a similar program and convert/export to PDF, or even hand write the solution (legibly!) and scan it to PDF. Please name this document ⟨`firstname-lastname`⟩`-sol1.pdf`.

2. The empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file.

In addition, you will need to submit your predictions on the handwritten digit recognition tasks to Kaggle, as described below, according to the competition rules. Please contact the TAs with any questions of difficulties.

Late submissions: there will be a penalty of 25 points for any solution submitted within 24 hours past the deadlne. No submissions will be accepted past then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked

to plot something, please include the figure as well as the code used to plot it (and clearly explain in the `README` what the relevant files are). If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important.

When submitting code, please make sure it's reasonably documented, and describe succinctly what is done in the relevant bits of code in the notebook.

1 Decision trees

In the problems below, we will only consider *existence* of decision trees with certain properties; do not worry about how these trees (if they exit) could be found by a particular algorithm, if at all.

Problem 1 **[10 points]**
Consider a set of N two-dimensional data points $\mathbf{x}_i, \ldots, \mathbf{x}_N \in \mathbb{R}^2$, with associated labels $y_1, \ldots, y_N \in \{\pm 1\}$, that is linearly separable. That is, there exists $\mathbf{w} \in \mathbb{R}^2$ and $w_0 \in \mathbb{R}^2$ such that for every i, $y_i = \text{sign}\left(\mathbf{w}^T\mathbf{x}_i + w_0\right)$.

Can a decision tree correctly classify these N points? If yes, decribe (as specifically as you can) what such a tree would look like; in particular, what is its depth? If not, explain why not. **End of problem 1**

Problem 2 **[10 points]**
Now assume that these N points are *not* linearly separable, i.e., there is no linear boundary that can separate +1 from -1. Can these points be correctly (zero mistakes) classified by a decision tree? If yes, decribe (as specifically as you can) what such a tree would look like; in particular, what is its depth? If not, explain why not. **End of problem 2**

2 Boosting

In stepwise fit-forward (least squares) regression, in each iteration a simple regressor is fit to the residuals obtained by the ensemble model up to that iteration. As a result, it is easy to see that *after* this regressor is added, the

new residuals are uncorrelated with its predictions, due to a general property of least squares regression.

We will now investigate a similar phenomenon that occurs with weak classifiers in AdaBoost.

Problem 3 **[10 points]**
Consider an ensemble classifier $H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ constructed by T rounds of AdaBoost on N training examples. Now we add next classifier h_{T+1} to the ensemble, by minimizing the training error weighted by $W_1^{(T)}, \ldots, W_N^{(T)}$, compute α_{T+1}, and update the weights. Show that the training error of the just added h_{T+1} (note: not the error of H_{T+1}) weighted by the *updated* weights $W_1^{(T+1)}, \ldots, W_N^{(T+1)}$, is exactly 1/2. With this fact in mind, could we select the same classifier again in the immediately following round, i.e., can we have $h_{T+2} = h_{T+1}$? **End of problem 3**

Advice: For this and the next problem, you can assume that we normalize the weights so that $\sum_i W_i^{(t)} = 1$ *after each iteration* t*, to make math/notation a bit less cluttered.*

Problem 4 **[10 points]**
Recal the expression of the vote strength α_t for a weak classifier h_t in AdaBoost,

$$\alpha_t = \frac{1}{2} \frac{1 - \epsilon_t}{\epsilon_t}, \tag{1}$$

where ϵ_t is the weighted training error of that weak classifier under the current weights at the beginning of iteration t in which it is chosen.

Show that (1) minimizes the empirical exponential loss (i.e., exponential loss on the training data) given the selection of h_t, **End of problem 4**

3 Support Vector Machines

Now we will consider some details of the dual formulation of SVM, the one in which we optimize the Lagrange multipliers α_i. In class we saw how to derive a constrained quadratic program, which can then be "fed" to an off-the-shelf quadratic program solver. These solvers are usually constructed to handle certain standard formulations of the objective and constraints.

Specifically the canonical form of a quadratic program with linear constraints is, mathematically:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H}\boldsymbol{\alpha} + \mathbf{f}^T\boldsymbol{\alpha}, \tag{2}$$

$$\text{such that: } \mathbf{A}\cdot\boldsymbol{\alpha} \leq \mathbf{a}, \tag{3}$$

$$\mathbf{B}\cdot\boldsymbol{\alpha} = \mathbf{b}, \tag{4}$$

$$\tag{5}$$

The vector $\boldsymbol{\alpha} \in \mathbb{R}^N$, where N is the number of training examples, contains the unknown variables to be solved for. The matrix $\mathbf{H} \in \mathbb{R}^{N\times N}$ and vector $\mathbf{f} \in \mathbb{R}^N$ specify the quadratic objective; the matrix $\mathbf{A} \in \mathbb{R}^{k_{ineq}\times N}$ and vector $\mathbf{a} \in \mathbb{R}^{k_{ineq}}$ specify k_{ineq} inequality constraints. Similarly, $\mathbf{B} \in \mathbb{R}^{k_{eq}\times N}$ and vector $\mathbf{b} \in \mathbb{R}^{k_{eq}}$ specify k_{eq} equality constraints. Note that you can express a variety of inequality constraints by adding rows to $\mathbf{B}$ and elements to $\mathbf{b}$; think how you would do it to express, e.g., a "greater or equal" constraint.

Problem 5 **[20 points]**
Describe in detail how you would compute $\mathbf{H}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{a}$, $\mathbf{B}$, and $\mathbf{b}$, to set up the dual optimization problem for the kernel SVM

$$\underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max\left[0, 1 - y_i\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) - w_0\right)\right] \right\}$$

given a kernel function $K(\cdot,\cdot)$ corresponding to the dot product in $\boldsymbol{\phi}$ space, and N training examples $(\mathbf{x}_i, y_i)$ **End of problem 5**

4 Sentiment analysis

In this final part we will develop a *sentiment analysis* tool. We have provided a data set containing short customer reviews (or snippets of reviews) for products. Each has been labeled as a positive or negative review. For instance, below is an example of a positive review

```
i downloaded a trial version of computer associates ez firewall
and antivirus and fell in love with a computer security system all
over again .
```

and a negative one

`i dont especially like how music files are unstructured ; basically they are just dumped into one folder with no organization , like you might have in windows explorer folders and subfolders .`

from the training set. We will use Support Vector Machines to learn to classify such review sentences into positive and negative classes.

We will use will be the word occurence features: if a particular word (or more generally, a token, which may include punctuation, numbers, etc.) w occurs in an example, the corresponding feature is set to 1, otherwise to 0.

Problem 6 **[40 points]**

Fill in the missing pieces of code to fully implement the SVMs. This includes fleshing out the input to the optimization, and calculation of the model predictions.

Using the provided `dev` (development) set as a validation set, tune an SVM predictor you think is best, and use it to compute ans submit predictions on the `test` set to Kaggle.

You are free to experiment with various aspects of SVMs, but at the minimum please do the following:

1. Run linear SVM
2. Run at least one non-linear SVM (with some non-linear kernel)
3. Evaluate a range of values of C (regularization parameter) and the kernel-specific parameter(s) and discuss your findings – how do these values affect the performance of the classifier on training/validation data?

End of problem 6

Some ideas for additional (optional) exploration:

- Consider various scaling on the input features, for instance, z-scoring or unit length normalization of each example.
- Using *bigram* features. Consider pairs of consecutive words, instead of, or in addition to, the *unigram* features (individual word occurences).
- Experiment with the frequency cutoff for including a word (or bigram) in the dictionary used to extract features. The default value for this we

recommend is 5 (i.e., if a word appear less than 5 times in the data set it is ignored), but perhaps you will get better results with other values. This and the previous points are already possible with the code (see arguments in `utils.preprocess`).

- Once you identify good setting for your hyperparameters (C etc.) you could retrain the classifier on the combined `train` and `dev` sets, and then test it on `test`.

Note: you will need to install a convex optimization package `cvxopt` which is likely not included by default with your Python installation. If you are using Anaconda, you may be able to install is with the simple command

`conda install -c omnia cvxopt`

You can find other instructions on how to install it here:

`https://anaconda.org/anaconda/cvxopt`

or here:

`http://cvxopt.org/install/`

Please start working on this early, and ask course staff for help with the software issues, if you need it!