COMS 4771-2 F18 Homework 3 (due October 29, 2018)

Instructions

Submit your write-up on Gradescope as a neatly typeset (not scanned nor handwritten) PDF document by 11:59 PM of the due date.

On Gradescope, be sure to select the pages containing your answer for each problem. More details can be found on the Gradescope Student Workflow help page:

- https://gradescope.com/help#help-center-section-student-workflow

(If you don't select the pages containing your answer to a problem, you'll receive a zero for that problem.)

Make sure **your name and your UNI** appears prominently on the first page of your write-up.

Source code

Please combine all requested source code files into a *single* ZIP file[1], along with a plain text file called `README` that contains your name and briefly describes all of the other files in the ZIP file. **Do not include the data files.** Submit this ZIP file on Courseworks.

Clarity and precision

One of the goals in this class is for you to learn to reason about machine learning problems and algorithms. To reason about these things, you must be able to make *clear* and *precise* claims and arguments about them.

A clear and precise argument is not the same as a long, excessively detailed argument. Unnecessary details and irrelevant side-remarks often make an argument less clear. And non-factual statements also detract from the clarity of an argument.

Points may be deducted for answers and arguments that lack sufficient clarity or precision. Moreover, a time-economical attempt will be made to interpret such answers/arguments, and the grade you receive will be based on this interpretation.

[1]See https://en.wikipedia.org/wiki/Zip_(file_format).

Problem 1 (40 points)

In this problem, you will implement the Online Perceptron algorithm with an online-to-batch conversion, and evaluate it on a classification task with a few different data representations.

Restaurant review data set

Download the review data set `reviews_tr.csv` (training data) and `reviews_te.csv` (test data) from Courseworks. This data set is comprised of reviews of restaurants in Pittsburgh; the label indicates whether or not the reviewer-assigned rating is at least four (on a five-point scale). The data are in CSV format (where the first line is the header); the first column is the label (`label`; 0 or 1), and the second column is the review text (`text`). The text has been processed to remove non-alphanumeric symbols and capitalization.

Data representations

In this problem, you will experiment with the following four different data representations.

1. Unigram representation (i.e., *term frequency*).

   In this representation, there is a feature for every word t, and the feature value associated with a word t in a document d is

   $$\mathrm{tf}(t;d) := \text{number of times word } t \text{ appears in document } d.$$

2. *Term frequency-inverse document frequency (tf-idf).*

   This is like the unigram representation, except the feature associated with a word t in a document d from a collection of documents D (e.g., training data) is

   $$\mathrm{tf}(t;d) \times \log_{10}(\mathrm{idf}(t;D)),$$

   where $\mathrm{tf}(t;d)$ is as defined above, and

   $$\mathrm{idf}(t;D) := \frac{|D|}{\text{number of documents in } D \text{ that contain word } t}.$$

   This representation puts more emphasis on rare words and less emphasis on common words. (There are many variants of tf-idf that are unfortunately all referred to by the same name.)

   Note: When you apply this representation to a new document (e.g., a document in the test set), you should still use the idf defined with respect to D. This, however, becomes problematic if a word t appears in a new document but did not appear in any document in D: in this case, $\mathrm{idf}(t;D) = |D|/0 = \infty$. It is not obvious what should be done in these cases. For this homework assignment, simply ignore words t that do not appear in any document in D.

3. Bigram representation.

   In addition to the unigram features, there is a feature for every *pair* of words (t_1, t_2) (called a *bigram*), and the feature value associated with a bigram (t_1, t_2) in a given document d is

   $$\mathrm{tf}((t_1,t_2);d) := \text{number of times bigram } (t_1,t_2) \text{ appears consecutively in document } d.$$

   In the sequence of words "a rose is a rose", the bigrams that appear are: $(\mathrm{a}, \mathrm{rose})$, which appears twice; $(\mathrm{rose}, \mathrm{is})$; and $(\mathrm{is}, \mathrm{a})$.

4. One more data representation of your own choosing (e.g., trigrams, skip-grams). It should be non-trivially different from the representations above (e.g., not just bigrams plus a few extra features).

In all four of these representations, include an "intercept" feature whose value is always equal to one (as in affine expansion).

Online Perceptron with online-to-batch conversion

Implement the Online Perceptron algorithm with the following online-to-batch conversion process (similar to one suggested in lecture):

1. Run Online Perceptron to make *two* passes through the training data. Before each pass, randomly shuffle the order of the training examples. Note that a total of $2n + 1$ linear classifiers $\hat{w}_1, \ldots, \hat{w}_{2n+1}$ are created during the run of the algorithm, where n is the number of training examples.
2. Return the linear classifier $\hat{w}_{\text{final}}$ given by the simple average of the final $n + 1$ linear classifiers:

$$\hat{w}_{\text{final}} := \frac{1}{n+1} \sum_{i=n+1}^{2n+1} \hat{w}_i .$$

Of course, you should not use (or even look at the source code for) any existing implementation of Perceptron or Online Perceptron; that would defeat the purpose of this assignment. You can, however, use existing library functions for basic operations needed in Online Perceptron (e.g., vector inner products), and also for loading the CSV files and creating the desired data representations. (You are responsible for determining whether or not an existing library function correctly implements the desired data representation.) As always, provide proper citations for any software packages you use.

Recall that as Online Perceptron is making its pass through the training examples, it only updates its weight vector in rounds in which it makes a prediction error. So, for example, if $\hat{w}_1$ does not make a mistake in classifying the first training example, then $\hat{w}_2 = \hat{w}_1$. Use this fact to keep the memory usage of your code relatively modest.

You will be using this learning algorithm with the restaurant review data represented in four ways described above. The total number of features with each representation can be very large. However, because each review has only a relatively small numbers of words, it should still be possible to compactly represent the data. Use this fact to keep the running time of your code close to linear in the size of the training data.

Do the following with each of the four data representations above. Run your code on the training data to learn a linear classifier $\hat{w}_{\text{final}}$. Compute the training error rate of $\hat{w}_{\text{final}}$ on this training data, and compute the test error rate of $\hat{w}_{\text{final}}$ on the test data.

Post-hoc analysis

Now consider the classifier based on the unigram representation. To get a sense as to its behavior, determine the 10 words that have the highest (i.e., most positive) weights; also determine the 10 words that have the lowest (i.e., most negative) weights.

Pick two training examples that this classifier (based on unigram representation) incorrectly classifies. For each of these examples $x = (x_1, \ldots, x_d)$, identify the 10 words with the highest (i.e., most positive) $w_i x_i$ value, and also the 10 words with the lowest (i.e., most negative) $w_i x_i$ value. Attempt to explain why the classifier incorrectly classifies these examples.

What to submit in your write-up:

(a) A precise specification of your fourth chosen data representation. Briefly explain why you chose this representation. Provide proper citations of any sources you used to come up with this representation.

(b) Training error rates of the linear classifiers with all four data representations.

(c) Test error rates of the linear classifiers with all four data representations.

(d) The lists of words with highest/lowest weights, described above. (Sort each list *alphabetically*.)

(e) The text of two misclassified examples, with associated lists of words, and explanations for why the examples were misclassified.

Please submit your source code on Courseworks.

Problem 2 (30 points)

In this problem, you'll analyze an algorithm that finds a large margin linear separator whenever one exists.

Recall that the Perceptron algorithm quickly finds a linear separator for a data set S whenever there exists a large margin linear separator for S. But the linear separator returned by Perceptron is not necessarily one that achieves a large margin on S.

An alternative is the following algorithm, which we call Margin Perceptron. This algorithm, like Perceptron, takes as input a collection S of labeled examples from $\mathbb{R}^d \times \{-1, +1\}$).

- Begin with $\hat{w}_1 := 0 \in \mathbb{R}^d$.
- For $t = 1, 2, \ldots$:
  - If there is a labeled example in S (call it (x_t, y_t)) such that $y_t \hat{w}_t^\top x_t < 1$, then set $\hat{w}_{t+1} := \hat{w}_t + \eta y_t x_t$.
  - Else, return $\hat{w}_t$.

Above, there are two differences relative to the original Perceptron algorithm. The first is the criteria under which an example in S is used to perform an update: $y_t \hat{w}_t^\top x_t < 1$ (rather than ≤ 0). The second is the update itself: $\hat{w}_{t+1} := \hat{w}_t + \eta y_t x_t$. Here, $\eta > 0$ is a parameter of the algorithm, which we'll assume is set as

$$\eta := \frac{1}{L^2}$$

where $L := \max_{(x,y) \in S} \|x\|_2$.

If Margin Perceptron terminates, it returns $\hat{w}_t$ satisfying

$$y \hat{w}_t^\top x \geq 1 \quad \text{for all } (x, y) \in S.$$

But this fact alone does not mean that $\hat{w}_t$ achieves a large margin on S. After all, if w is *any* linear separator satisfying $y w^\top x > 0$ for all $(x, y) \in S$, we can construct another $\tilde{w}$ satisfying $y \tilde{w}^\top x \geq 1$ for all $(x, y) \in S$, simply by letting $\tilde{w}$ be a particular scaling of w.

(a) Assume that there exists a vector $w_\star \in \mathbb{R}^d$ such that

$$\min_{(x,y) \in S} y w_\star^\top x = 1.$$

Mimic the proof of the Perceptron convergence theorem to prove that Margin Perceptron halts after at most $3\|w_\star\|_2^2 L^2$ loop iterations. Clearly explain every step of the proof, especially where it differs from that of the original Perceptron convergence theorem.

(b) Under the same assumption as in Part (a), prove that Margin Perceptron returns $\hat{w}_t$ satisfying

$$\|\hat{w}_t\|_2 \leq 3\|w_\star\|_2.$$

Hint: Use the claim you proved in Part (a) (and, possibly, part of its proof).

(c) Very briefly explain why this means that the margin achieved by $\hat{w}_t$ is (almost, within a factor of three) as large as the margin achieved by $w_\star$.

(d) (Optional.) For any given $\epsilon > 0$, explain how to modify Margin Perceptron so that, under the same conditions as in the previous parts, it returns $\hat{w}_t$ satisfying

$$\|\hat{w}_t\|_2 \leq (2 + \epsilon)\|w_\star\|_2.$$

Problem 3 (30 points)

In this problem, you'll use Lagrange duality to derive the "dual" forms of ridge regression and a variant of the soft-margin SVM problem.

The ridge regression optimization problem can be written in a form that is reminiscent of the soft-margin SVM problem (for $C > 0$):

$$\begin{aligned} \min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} \quad & \frac{1}{2}\|w\|_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & \xi_i = y_i - x_i^\top w \quad \text{for all } i = 1, \dots, n. \end{aligned}$$

Above, $\xi = (\xi_1, \dots, \xi_n) \in \mathbb{R}^n$ has a similar role as the *slack variables* in soft-margin SVM, but note that they are not required to be non-negative here.

The Lagrangian function associated with this optimization problem is

$$L(w, \xi, \alpha) = \frac{1}{2}\|w\|_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 + \sum_{i=1}^n \alpha_i(y_i - x_i^\top w - \xi_i).$$

Here, $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$ are the *Lagrange multipliers*, which are unconstrained.

(a) Fix $\alpha \in \mathbb{R}^n$. Derive expressions for the w and ξ that, together, minimize $L(w, \xi, \alpha)$. The expressions should be given in terms of α (as well as the data and C). Briefly justify each step of your derivations.

(b) The dual objective function $g \colon \mathbb{R}^n \to \mathbb{R}$ is equal to the Lagrangian function evaluated at $w_\alpha, \xi_\alpha, \alpha$, where w_α and ξ_α are the values of w and ξ from Part (a) that minimize L:

$$g(\alpha) = L(w_\alpha, \xi_\alpha, \alpha) = \min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} L(w, \xi, \alpha).$$

Derive an expression for the α that maximizes $g(\alpha)$. The expression should be given in terms of the data and C. Moreover, the expression should only involve the x_i's through the matrix $K \in \mathbb{R}^{n \times n}$ whose (i, j)-th entry is $x_i^\top x_j$. Briefly justify each step of your derivation.

(c) (Optional.) A variant of the soft-margin SVM problem that is similar to ridge regression is the following optimization problem (for $C > 0$):

$$\begin{aligned} \min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} \quad & \frac{1}{2}\|w\|_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i x_i^\top w \geq 1 - \xi_i \quad \text{for all } i = 1, \dots, n. \end{aligned}$$

This is almost the same as soft-margin SVM, except the slack variables are squared in the objective. The Lagrangian is

$$L(w, \xi, \alpha) = \frac{1}{2}\|w\|_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 + \sum_{i=1}^n \alpha_i(1 - y_i x_i^\top w - \xi_i).$$

However, since the constraints are inequality constraints, the associated Lagrange multipliers $\alpha \in \mathbb{R}^n$ are constrained to be non-negative.

Derive expressions for the w and ξ that, together, minimize $L(w, \xi, \alpha)$, and then derive an expression for the dual objective $g(\alpha) = \min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} L(w, \xi, \alpha)$. Explain why the dual objective only depends on the x_i's through the matrix $K \in \mathbb{R}^{n \times n}$ whose (i, j)-th entry is $x_i^\top x_j$.