CS 4780/5780 Machine Learning Due September 11, 2013

Assignment 1: Concept Learning and kNN

Instructor: Thorsten Joachims *Total Points: 100*

Course Policy:

Read all the instructions below carefully before you start working on the assignment, and before you make a submission

- Assignments are due at the beginning of class on the due date in hard copy.

- Write your NetIDs with submission date and time on the first page of the hard copy.

- No assignment will be accepted after the solution is publicized (about 4 days after due)

- The submission time of whatever you submit last (hard copy or CMS) is counted as the official submission time and will determine the late penalty

- Upload only the code you wrote to CMS. Do not upload any additional libraries. Provide a README file with your submission that includes a list of all libraries and instructions needed to run your code.

- Attach a hard copy of your code to the submission.

- Late assignments can be submitted in class or to Ian Lenz in Upson Hall 5151. Since the fifth floor of Upson is locked on the weekends, weekend submissions should be made digitally via CMS, with a hard copy delivered to Ian as soon as possible afterwards.

- All sources of material must be cited. Assignment solutions will be made available along with the graded homework solutions. The University Academic Code of Conduct will be strictly enforced, including running cheating detection software.

- No more than one submission per group. Groups of one or two students only.

Problem 1: Version Spaces [30 points]

In this problem we will investigate the geometry of version spaces for different hypotheses. For all parts of this question, training instances are vectors in $\mathbb{Z}^2$, i.e. points on a 2D lattice, with labels $c \in \{-1, 1\}$.

(a) *Rectangular hypothesis* Consider a hypothesis of the form (p_{TL}, p_{BR}), where $p_i = \{x_i, y_i | x_i, y_i \text{ are integers and } 0 \leq x_i, y_i \leq 8\}$ define the diagonally-opposite corners of a rectangle (*Top Left, Bottom Right*, respectively). An instance (x, y) is classified positive if it falls inside, or on the boundary of this rectangle, and negative otherwise. Note, that the definition allows for degenerate cases, where two or four of the corners overlap. In addition, allow for p_{TL} and p_{BR} to take on a special value $\emptyset$. If either $p_{TL} = \emptyset$ or $p_{BR} = \emptyset$, all instances are classified as negative. This hypothesis definition applies to problems $1a$ to $1d$.

What is the size of this hypothesis space?

(b) *Rectangular hypothesis* A hypothesis h_1 is considered more *general* than hypothesis

h_2 (and h_2 more *specific* than h_1) if h_2 implies h_1 The *most general (most specific)* hypothesis h^* is a hypothesis, such that no other hypothesis is more general (more specific) than h^*.

Draw the *most general* hypothesis that satisfies the training data D_1 in Figure 1. Draw the *most specific* hypothesis.

(c) *Rectangular hypothesis* What is the size of the version space for the training data D_1?

(d) *Rectangular hypothesis* In a form of *Active Learning*, a learner can query the teacher for more data. The goal of the learner would be to pick query instances, such that the size of the version space is reduced the most (which means that the greatest number of inconsitent hypotheses is pruned after observing the query instance). Consider the following 3 candidates for a query instance: (3,7), (4,3), (4,6). Compute, for each, the expected size of the version space after observing the training set D_1 and the label for that query instance (considering the two possible labels have an equal chance of occuring). Between the three query candidates, is there a best choice?

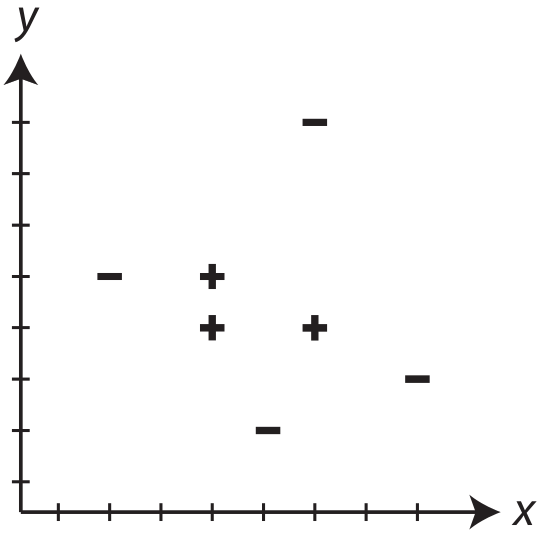

Figure 1: Training set D_1

(e) *Decision Tree hypothesis* Now consider a hypothesis space formed by all *1-level* decision trees. Since all of the attributes for our data are integer-valued, consider that splitting thresholds in our decision tree are also constrained to integers, and each node in the tree splits on a single attribute, x or y, with a splitting criterion **attribute $\geq$ threshold**. Give the size of the version space for a *1-level* decision tree and the training set D_2. Note that in general multiple trees can represent the same function, and we are interested in counting functions.

(f) *Decision Tree hypothesis* Now consider a hypothesis space formed by all decision trees with 3 leaf nodes, with the same splitting criteria as above. What is the size of the version-space now? Draw all hypotheses that belong to the version space in the (x, y) space of Figure 2.

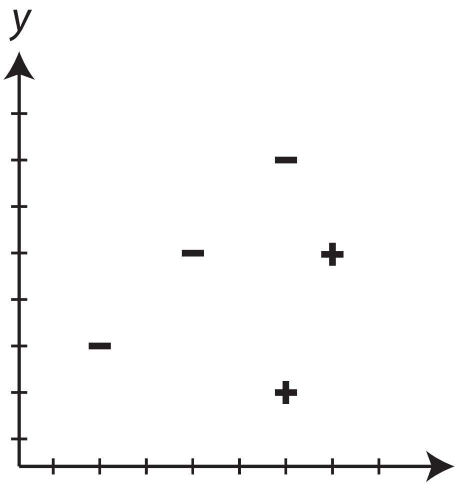

Figure 2: Training dataset D_2

Problem 2: Regression with kNN [30 points]

In this problem, we will investigate the application of *kNN* to regression in a setting of partial image completion. You have been hired by the NSA to help them reconstruct corrupted surveillance photos of possible criminals. Due to a glitch in their cameras, some images are partially blank. Your goal will be to provide the “best” completion of the image using *kNN*.

Included with the assignment is a subset of the *Olivetti* face dataset. The dataset is a collection of 64×64 grayscale images of peoples faces. This dataset was processed into the *SVMLight* format for this assignment. This file format will be used for all programming assignments in this course. In this file, each line corresponds to a training instance, providing the label, and the feature-value pairs for that instance:

```
<label>:  <feature idx>:<feature value> ....
```

For this problem, each image has been vectorized into this format using a row-major encoding, i.e. features with indices $0 - 63$ correspond to the grayscale intensities of the first row, features with indices $64 - 127$ correspond to the grayscale intensities of the second row, etc. Grayscale intensities are in the range 0.0 to 1.0, where 0.0 is black, and 1.0 is white.

You are provided with a set of training images in the file $faces.train$, and a set of test images in the file $faces.test$. We will treat the vectorized form of the top half of the image (32×64 pixels) as a feature vector $\mathbf{x}$, and the vectorized form of the bottom half of the image, $\mathbf{y}$ (also 32×64 pixels), as a target (label). Note, that in contrast to the first problem, the target variable is both multidimensional, and continuous. Our goal is: given an input $\mathbf{x}$ of someone's top part of the face, predict what the lower part $\mathbf{y}$ might look like.

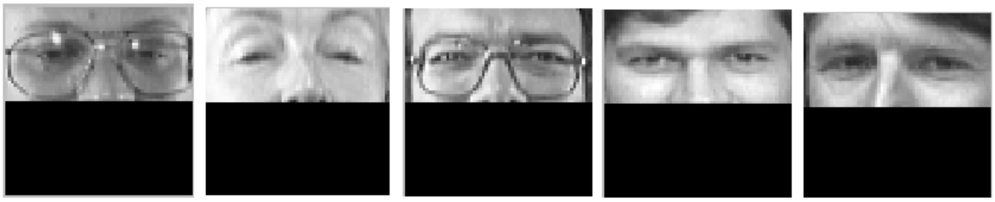

Figure 3: Faces – test set

Your instructions for implementation: Complete all five faces in the $faces.test$ file by performing regression with *kNN*. Use *inverse euclidean distance* as a similarity metric (and take the similarity between identical instances to be 1). As an example: if for k=2, your two nearest neighbors to a test instance $\mathbf{x}_u$ have labels $\mathbf{y}_v$ and $\mathbf{y}_w$ with corresponding distances to the test instance d_{uv} and d_{uw}, then the predicted output would be $\mathbf{y}_\mathbf{u} = (1/d_{uv})\mathbf{y}_v + (1/d_{uw})\mathbf{y}_w$. Experiment with $k \in \{1, 5, 10, 50\}$. What do you observe about the sharpness of the completed face when k is increased, and how do you account for this observation? Present "completed" faces corresponding to $k = 5$ and $k = 50$ in your submission.

Problem 3: kNN Classification [40 points]

In this problem, we will be performing text categorization with the *kNN* algorithm. We will be classifying books from *amazon.com* based on their previews available online (usually one or two chapters). The most common way to classify text is to first model a document as a *bag-of-words*. That means that the order of words in the document is ignored, allowing us to treat each unique English word as an independent feature. The number of times a particular word occurs in the document is then the value of the respective feature. That way, each English word is mapped to a unique *id*, allowing us to represent a single document in the *SVMlight* file as a line in this format:

```
<document label> <word_id>:count <word_id>:count ....
```

Supplied to you are 2 such files: *books.train, books.test*, corresponding to the training and test sets respectively. Additionally, you are provided with the *id-word* mappings in the *books.vocab* file, and the titles of books in the **.titles* files with a line to line correspondence with the *.train* and *.test* files. There is a total of (almost) 10,000 books, divided equally between the training and test sets. Each book belongs to exactly one of the 5 categories (genres) — *0: Action-and-adventure, 1: Horror, 2: Mystery-thrillers, 3: Romance, 4: Science-fiction*, where indices are the class labels corresponding to the genres, and are given as document labels in the *SVMlight* files. Our goal in this problem will be to classify books according to their genre.

(a) *Content-based book recommendation* As a warm up exercise, consider a problem of recommending books to read based on the books you already like. For each of the 2 books listed below, provide the top 10 recommendations by finding its 10 closest neighbors. Use *cosine similarity* as a measure of "closeness" between instances. Cosine similarity is given by:

$$sim(\mathbf{x}_u, \mathbf{x}_v) = \frac{\mathbf{x}_u \cdot \mathbf{x}_v}{||\mathbf{x}_u|| ||\mathbf{x}_v||}$$

where $\mathbf{x}_u, \mathbf{x}_v$ are feature vectors corresponding to two document instances, and $|| \cdot ||$ indicates the euclidean length (L_2 norm). Suppose your two most favorite books are:

```
Fifty Shades of Grey:  Book One of the Fifty Shades Trilogy
                  Brains:  A Zombie Memoir
```

Report the top 10 recommendations corresponding to *each* book, using the instances in the *books.train* file only. Qualitatively, do these recommendations make sense? *Note that some books have an amazon product ID number, instead of the title. You can look up the book corresponding to the number by searching for the number on amazon.com.*

(b) *Baseline.* Before trying *kNN*, let's consider a simple baseline for the task of genre classification. For each class (genre[1]), compute a "centroid" feature vector by first normalizing the feature vector for each instance (using the L_2 norm), and then summing the normalized feature vectors across all training instances corresponding to that class. To classify a test instance, compute *cosine similarity* between the test instance and the 5 "centroids". Label the test instance with the class of the nearest centoid.

Run this baseline on the entire test set *books.test*. Report the following metrics for this baseline: *Accuracy* (1 value), *Precision* for each class (5 values), *Recall* for each class (5 values). These metrics are defined as follows:

[1] the two terms are interchangeable for this problem

$$Accuracy = n_{y_{predict}=y_{true}}/n$$

$$Precision(\text{class } c) = n_{y_{predict}=y_{true}=c}/n_{y_{predict}=c}$$

$$Recall(\text{class } c) = n_{y_{predict}=y_{true}=c}/n_{y_{true}=c}$$

where n is the size of the test set (number of instances) and the expression $n_{y_{predict}=y_{true}=c}$ reads: the number of test instances for which our algorithm *correctly* predicts class c. Note that for the case that the denominators in the $Precision$ and $Recall$ expressions are zero, report zero for the corresponding metric. $Precision$ and $Recall$ are valuable for investigating the performance of our classifier on individual classes, whereas $Accuracy$ provides only an aggregate metric.

(c) *kNN Implementation* Implement the *unweighted kNN* algorithm that takes k, the training and testing sets as inputs, and outputs a class label c for every test instance. Use *cosine similarity*, and run the algorithm for values of k in the range {1, 2, 5, 10, 100, 200, 300, 500, 1,000, 2,000, 3,000, 4,000, 5,000}. Plot *Accuracy* as a function of $\log(k)$. In your *kNN* implementation, break ties by choosing a class with the smallest class index.

(d) What value of k yields the highest accuracy on the test set? Report $Precision$ and $Recall$ for that value of k for every class (10 values in total).

(e) What do you observe with respect to the $Precision$ and $Recall$ when $k = 5,000$?. Does this match your expecations?

(f) Consider a hypothetical instance of an unweighted *kNN* algorithm with $k = n_{train}$. Consider an unbalanced distribution of class labels in the training set: $\{n_0 = 1010, n_1 = 999, n_2 = 998, n_3 = 997, n_4 = 996\}$, and $n_0 + n_1 + n_2 + n_3 + n_4 = n_{train}$. What $Precision$ and $Recall$ do we expect on a test set of size $n_{test} = 5000$ where each of the five classes has an equal number of instances?

(g) Compare the performance of *kNN* to the baseline in part *b*. It turns out that this baseline is quite strong. What does this say about the geometry of our instance space (book data)? Show a qualitative example of an instance space in 2D where we would expect the baseline to perform significantly worse compared to *kNN*.