Digit Recognition: Nearest Neighbors vs. Perceptron

Harnoor Singh
Department of Computer Science & Engineering
University of Washington
hsingh@cs.washington.edu

Brian Walker
Department of Computer Science & Engineering
University of Washington
bdwalker@cs.washington.edu

Abstract

Optical Character Recognition (OCR) is a complex computer science problem which has several effective machine learning solutions. We implemented the K Nearest Neighbors (KNN) and Kernelized Perceptron algorithms and compared the effectiveness of both methods in classifying a testing data set of 28,000 images. Each image is formatted as an array of 768 pixel values, with each value indicating how dark each pixel was in the image. Both of our algorithms were trained on a training data set of 42,000 images. We used cross validation to tune the K value in KNN and to choose a kernel function and dimension value for Perceptron. Despite its relative simplicity, we found that KNN was more effective than Perceptron in classifying the digits. The nearest neighbors algorithm correctly classified 96.857% of the testing samples on a weighted neighbors algorithm which looked at 4 neighbors. Our best Perceptron implementation correctly classified 95.67% of the testing samples using an exponential kernel with a $\sigma = 5$. Despite its higher accuracy, the runtime of KNN was 8x slower than that of the Perceptron.

1 The Problem

1.1 Image Data

Optical Character Recognition (OCR) is a computational problem where a machine attempts to classify an image of text. We tackled a subset of this problem, attempting to classify the digits 0-9 from data provided by the Kaggle Digit Recognition competition[1].

The Kaggle data provides two groups of data, 42,000 training samples with labels and 28,000 unlabeled testing samples. Each training sample contains a 48 pixel by 48 pixel image containing a handwritten number. The image is represented as an array of 768 pixel values which represent how dark each pixel of the image is. Table 1 is an example of a training sample.

[1]http://www.kaggle.com/c/digit-recognizer

Table 1: Subset of Kaggle Data

Label	Pixel 77	Pixel 78	Pixel 79
8	0	0	0
6	89	208	135

2 K Nearest Neighbors (KNN)

2.1 Overview

K Nearest Neighbors is a relatively simple algorithm which often gets surprisingly good results given its simplicity. Given a testing sample X, the algorithm looks through all the training samples it has previously seen and finds the *K* most similar samples. The algorithm then implements some form of voting mechanism among the *K* samples to classify the testing sample as a digit 0-9. The voting mechanism can either be as straightforward as selecting the majority label, to something more complicated which involves weighting the samples based on some distance metric.

2.2 Implementation Details

The first step in implementing KNN is to find the *K* nearest neighbors. This is accomplished by taking the euclidean distance between the testing sample we are classifying and every training sample in our training data set.

$$Euclidean(TestX, TrainX) = \sum_{i \in TrainX} (TestX - TrainX_i)^2$$

Because the image data is defined as an array of 768 pixels, $TestX - TrainX_i$ is actually doing a 768 dimension comparison between data points.

Once we identified the *K* nearest neighbors to our testing sample, we had to use this information to classify the testing point. We considered two approaches when deciding how to perform this classification.

2.2.1 Unweighted KNN

The first method we implemented to classify our testing sample was a simple majority voting algorithm which selects the majority label from the *K* closest neighbors. For example, if we ran the nearest neighbors algorithm with $K = 5$ and 3 of the neighbors had a label of 6, we would classify our testing sample as a 6.

2.2.2 Weighted KNN

Given the *K* nearest neighbors, the weighted version of the algorithm multiplies each label by a weight which indicates how close it is to our testing sample. These weights help to emphasize points that are closer to our testing sample while reducing the impact further away training points have on classifying our testing sample.

2.3 Cross Validation

Our implementation of KNN has a number of parameters which require tuning. In particular, we need to decide whether or not to use the weighted version of our KNN algorithm and how many neighbors to consider when implementing our algorithm.

To do this, we implemented K folds cross validation using 5,000 training samples. The reduced training set allowed us to run our algorithm using various configurations to see which configuration was most accurate.

Table 2: KNN Cross Validation Error Rates

K	Unweighted KNN Error	Weighted KNN Error
1	.077	.077
2	.092	.092
3	.075	.075
4	.078	.073
5	.078	.075
10	.085	.079
15	.095	.089
20	.102	.098

Our cross validation results suggest that running weighted KNN with $K = 4$ yields the most successful classification rate. One caveat to this is that our cross validation code ran using 5,000 training samples instead of the 42,000 samples in our full training set. It is possible that data we used to cross validate is no representative of the full data set.

2.4 Runtime

2.5 Results

3 Kernelized Perceptron

3.0.1 Overview

3.0.2 Implementation Details

3.0.3 Results

4 Citations, figures, tables, references

These instructions apply to everyone, regardless of the formatter being used.

4.1 Citations within the text

Citations within the text should be numbered consecutively. The corresponding number is to appear enclosed in square brackets, such as [1] or [2]-[5]. The corresponding references are to be listed in the same order at the end of the paper, in the **References** section. (Note: the standard BIBTEX style `unsrt` produces this.) As to the format of the references themselves, any style is acceptable as long as it is used consistently.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]", not "In our previous work [4]". If you cite your other papers that are not widely available (e.g. a journal paper under review), use anonymous author names in the citation, e.g. an author of the form "A. Anonymous".

4.2 Footnotes

Indicate footnotes with a number[2] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).[3]

4.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction; art work should not be hand-drawn. The figure number and caption always appear after the figure. Place one line space before the figure caption, and one line space after the figure. The figure caption is lower case (except for first word and proper nouns); figures are numbered consecutively.

Make sure the figure caption does not get separated from the figure. Leave sufficient space to avoid splitting the figure and figure caption.

You may use color figures. However, it is best for the figure captions and the paper body to make sense if the paper is printed either in black/white or in color.

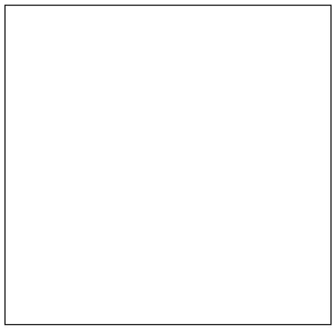

Figure 1: Sample figure caption.

4.4 Tables

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. The table number and title always appear before the table. See Table 3.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

5 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

[2]Sample of the first footnote
[3]Sample of the second footnote

Table 3: Sample table title

PART	**DESCRIPTION**
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

6 Preparing PostScript or PDF files

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- LaTeX users:
  - Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.
  - Otherwise, please generate your PostScript and PDF files with the following commands:

    ```
    dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
    ps2pdf mypaper.ps mypaper.pdf
    ```

    Check that the PDF files only contains Type 1 fonts.
  - xfig "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
  - The `\bbold` package almost always uses bitmap fonts. You can try the equivalent AMS Fonts with command

    ```
    \usepackage[psamsfonts]{amssymb}
    ```

    or use the following workaround for reals, natural and complex:

    ```
    \newcommand{\RR}{I\!\!R} %real numbers
    \newcommand{\Nat}{I\!\!N} %natural numbers
    \newcommand{\CC}{I\!\!\!\!C} %complex numbers
    ```
  - Sometimes the problematic fonts are used in figures included in LaTeX files. The ghostscript program `eps2eps` is the simplest way to clean such figures. For black and white figures, slightly better results can be achieved with program `potrace`.
- MSWord and Windows users (via PDF file):
  - Install the Microsoft Save as PDF Office 2007 Add-in from `http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=4d951911-3e7e-4ae6-b059-a2e79ed87041`

    - Select "Save or Publish to PDF" from the Office or File menu

- MSWord and Mac OS X users (via PDF file):
    - From the print menu, click the PDF drop-down box, and select "Save as PDF..."
- MSWord and Windows users (via PS file):
    - To create a new printer on your computer, install the AdobePS printer driver and the Adobe Distiller PPD file from `http://www.adobe.com/support/downloads/detail.jsp?ftpID=204` *Note:* You must reboot your PC after installing the AdobePS driver for it to take effect.
    - To produce the ps file, select "Print" from the MS app, choose the installed AdobePS printer, click on "Properties", click on "Advanced."
    - Set "TrueType Font" to be "Download as Softfont"
    - Open the "PostScript Options" folder
    - Select "PostScript Output Option" to be "Optimize for Portability"
    - Select "TrueType Font Download Option" to be "Outline"
    - Select "Send PostScript Error Handler" to be "No"
    - Click "OK" three times, print your file.
    - Now, use Adobe Acrobat Distiller or ps2pdf to create a PDF file from the PS file. In Acrobat, check the option "Embed all fonts" if applicable.

If your file contains Type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

6.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the graphicx package. Always specify the figure width as a multiple of the line width as in the example below using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

References follow the acknowledgments. Use unnumbered third level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to 'small' (9-point) when listing the references. **Remember that this year you can use a ninth page as long as it contains *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer-Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.