

Introduction to Machine Learning 2008/B

Assignment 6

Michael Orlov
Department of Computer Science
orlov@m.cs.bgu.ac.il

September 27, 2008

Abstract

Submission of Assignment 6 in Introduction to Machine Learning, 202-2-5461.

Question 1

- Show that the weights vector in the perceptron can be represented as a linear combination of the training samples. Hint: consider how each update affects \mathbf{w} .
- Use the previous observation to derive a kernelized version of the perceptron. That is, you need to provide a kernel-based algorithm for the perceptron.

Question 1(a)

The online update to the weight vector is given by

$$\mathbf{w}^{t+1} - \mathbf{w}^t = \underbrace{\eta(r^t - \text{sigmoid}(\mathbf{w}^t \cdot \mathbf{x}^t))}_{\alpha^t} \mathbf{x}^t ,$$

where \mathbf{w}^{t+1} is the weights vector after t updates, \mathbf{x}^t is a training sample vector (with $x_0^t = 1$), and η is the learning factor, possibly decreasing gradually. Of course, functions other than sigmoid can be used. Denoting the coefficient of \mathbf{x}^t above as α^t , we have

$$\mathbf{w}^{t+1} = \alpha^t \mathbf{x}^t + \mathbf{w}^t .$$

Thus, after all training samples are learned, the weights vector is given by

$$\mathbf{w} = \sum_t \alpha^t \mathbf{x}^t + \mathbf{w}^1 ,$$

where \mathbf{w}^1 is initialized randomly. Thus, \mathbf{w} can be viewed as a linear combination of the training samples.

In a simpler case, where $r^t \in \{1, -1\}$, a sign function is used instead of sigmoid (producing 1 or -1), and $w_1 = 0$, we have

$$\begin{aligned}\mathbf{w} &= \sum_t \eta(r^t - \text{sign}(\mathbf{w}^t \cdot \mathbf{x}^t)) \mathbf{x}^t \\ &= 2\eta \sum_{\text{some } t} r^t \mathbf{x}^t .\end{aligned}$$

The next question shows why η is irrelevant when the sign function is used.

Question 1(b)

The output of the simple perceptron discussed above after training on all samples, and given input vector \mathbf{x} , is

$$\begin{aligned}o(\mathbf{x}) &= \text{sign}(w \cdot \mathbf{x}) = \text{sign}\left(2\eta \sum_{\text{some } t} r^t \mathbf{x}^t \cdot \mathbf{x}\right) \\ &= \text{sign}\left(\sum_{\mathbf{x}^t \in S} r^t \mathbf{x}^t \cdot \mathbf{x}\right) ,\end{aligned}$$

where S is the set of training support vectors—inputs that were used during training to update \mathbf{w} . If (\mathbf{x}, r) is also a training sample, \mathbf{x} is added to S if $o \neq r$.

Thus, if we want to use a kernel function K , perceptron's output given input vector \mathbf{x} is

$$o(\mathbf{x}) = \text{sign}\left(\sum_{\mathbf{x}^t \in S} r^t K(\mathbf{x}^t, \mathbf{x})\right) ,$$

after which \mathbf{x} is possibly added to the multiset S if it is a training sample, and the desired output is different from $o(\mathbf{x})$.

The kernelized perceptron is not limited to the simple sign version. Using the original definition in the previous question,

$$\begin{aligned}o(\mathbf{x}) &= \text{sigmoid}(\mathbf{w} \cdot \mathbf{x}) = \text{sigmoid}\left(\sum_t \alpha^t \mathbf{x}^t \cdot \mathbf{x} + \mathbf{w}^1 \cdot \mathbf{x}\right) \\ &= \text{sigmoid}\left(\sum_t \alpha^t K(\mathbf{x}^t, \mathbf{x}) + K(\mathbf{w}^1, \mathbf{x})\right) ,\end{aligned}$$

after which, if (\mathbf{x}, r) is also a training sample, α is computed,

$$\alpha = \eta(r - o(\mathbf{x})),$$

and (\mathbf{x}, α) pair is saved for future perceptron activations.

Question 2

Implement the perceptron algorithm for classifying the following data set from the UCI repository: <http://archive.ics.uci.edu/ml/datasets/Adult>. The classification task is to predict whether the person makes more than \$50K/year.

See the appendix for implementation based on the perceptron training algorithm [1, p. 239]. The sigmoid function has been used for perceptron output.

Upon reading the training and the test datasets, samples with unknown (“?”) attributes are ignored. There are 30162 good samples in the training set, and 15060 in the test set. The attributes are normalized as follows. Some discrete attributes are logically reordered, e.g., high school education comes before academic education in the *education* attribute. Discrete attributes with n possible values are converted to integers in $\{0, \dots, n - 1\}$ range. After that, both converted discrete values and numeric values are linearly scaled to $[0, 1]$ floating-point range.

The resulting accuracy is 82.9% for 10 passes over the training dataset, and learning rate $\eta = 0.25$, if false positives and false negatives are given the same weight. Varying the parameters such as η and number of passes over the training set consistently results in accuracies in the range of 82.5%–83.5%.

A Listings

Question 2

```
import static java.lang.Math.*;
import java.util.Random;

public class Perceptron {
    static final double eta = 0.25;
    Random rand = new MersenneTwisterFast(123);
    int d;
    double[] w;

    Perceptron(int d) {
        this.d = d;
        w = new double[d+1];
        for (int i = 0; i <= d; ++i)
            w[i] = rand.nextDouble() * 0.02 - 0.01;
    }

    double sigmoid(double v) {
        return 1 / (1 + exp(-v));
    }

    double getOutput(double[] x) {
        double y = w[0];
        for (int i = 0; i < d; ++i)
            y += w[i+1] * x[i];
        return sigmoid(y);
    }

    void update(double[] x, double r) {
        double alpha = eta * (r - getOutput(x));
        w[0] += alpha;
    }
}
```

```

        for (int i = 0; i < d; ++i)
            w[i+1] += alpha * x[i];
    }

    int getPrediction(double[] x) {
        return getOutput(x) >= 0.5 ? 1 : 0;
    }
}

public class Adult {

    final static String[] WC = {
        "Never-worked", "Without-pay", "Private",
        "Local-gov", "State-gov", "Federal-gov",
        "Self-emp-not-inc", "Self-emp-inc" };
    final static String[] ED = {
        "Preschool", "1st-4th", "5th-6th", "7th-8th",
        "9th", "10th", "11th", "12th", "HS-grad",
        "Prof-school", "Some-college", "Assoc-voc",
        "Assoc-acdm", "Bachelors", "Masters", "Doctorate" };
    final static String[] MS = {
        "Never-married", "Widowed", "Divorced",
        "Separated", "Married-spouse-absent",
        "Married-AF-spouse", "Married-civ-spouse" };
    final static String[] OC = {
        "Tech-support", "Craft-repair", "Other-service", "Sales",
        "Exec-managerial", "Prof-specialty", "Handlers-cleaners",
        "Machine-op-inspct", "Adm-clerical", "Farming-fishing",
        "Transport-moving", "Priv-house-serv", "Protective-serv",
        "Armed-Forces" };
    final static String[] RE = {
        "Unmarried", "Not-in-family", "Own-child",
        "Other-relative", "Wife", "Husband" };
    final static String[] RA = {
        "Other", "Black", "Amer-Indian-Eskimo",
        "Asian-Pac-Islander", "White" };
    final static String[] SX = { "Female", "Male" };
    final static String[] NC = {
        "United-States", "Cambodia", "England", "Puerto-Rico",
        "Canada", "Germany", "Outlying-US(Guam-USVI-etc)",
        "India", "Japan", "Greece", "South", "China", "Cuba",
        "Iran", "Honduras", "Philippines", "Italy", "Poland",
        "Jamaica", "Vietnam", "Mexico", "Portugal", "Ireland",
        "France", "Dominican-Republic", "Laos", "Ecuador",
        "Taiwan", "Haiti", "Columbia", "Hungary", "Guatemala",
        "Nicaragua", "Scotland", "Thailand", "Yugoslavia",
        "El-Salvador", "Trinidad&Tobago", "Peru", "Hong",
        "Holand-Netherlands" };
    final static String[] SUM = { ">50K", "<=50K" };
}

```

```

double[] parse(String line, int[] rbox) {
    if (line.contains("?"))
        return null;
    if (line.endsWith("."))
        line = line.substring(0, line.length()-1);

    String[] feat = line.split(", ");
    if (feat.length != 15)
        return null;

    double[] x = new double[feat.length-1];
    rbox[0] = (int) convert(feat[14], SUM);

    x[0] = convert(feat[0], 17, 90); // age
    x[1] = convert(feat[1], WC);
    x[2] = convert(feat[2], 12285, 1490400); // fmlwgt
    x[3] = convert(feat[3], ED);
    x[4] = convert(feat[4], 1, 16); // education-num
    x[5] = convert(feat[5], MS);
    x[6] = convert(feat[6], OC);
    x[7] = convert(feat[7], RE);
    x[8] = convert(feat[8], RA);
    x[9] = convert(feat[9], SX);
    x[10] = convert(feat[10], 0, 99999); // capital-gain
    x[11] = convert(feat[11], 0, 4356); // capital-loss
    x[12] = convert(feat[12], 1, 99); // hours-per-week
    x[13] = convert(feat[13], NC);

    return x;
}

double convert(String value, int min, int max) {
    int x = Integer.parseInt(value);
    if (x < min || x > max)
        throw new Error("Value out of range: " + value);

    return (x - min) / (max - min + 0.0);
}

double convert(String value, String[] vals) {
    for (int v = 0; v < vals.length; ++v)
        if (value.equals(vals[v]))
            return v / (vals.length-1.0);

    throw new Error("Unknown value " + value);
}
}

import java.io.BufferedReader;

```

```
import java.io.FileReader;
import java.io.IOException;

public class Run {
    public static void main(String[] args) throws IOException {
        Perceptron perc = new Perceptron(14);
        Adult parser = new Adult();

        BufferedReader in;
        String line;

        for (int i = 1; i <= 10; ++i) {
            int dataLines = 0;

            in = new BufferedReader(new FileReader(
                "/tmp/.orlovm/perceptron/adult.data"));
            while ((line = in.readLine()) != null) {
                int[] rbox = new int[1];
                double[] x = parser.parse(line, rbox);
                if (x != null) {
                    ++dataLines;
                    perc.update(x, rbox[0]);
                }
            }

            System.out.println("Lines in: " + dataLines);
        }

        int testLines = 0, errors = 0;

        in = new BufferedReader(new FileReader(
            "/tmp/.orlovm/perceptron/adult.test"));
        while ((line = in.readLine()) != null) {
            int[] rbox = new int[1];
            double[] x = parser.parse(line, rbox);
            if (x != null) {
                ++testLines;
                if (perc.getPrediction(x) != rbox[0])
                    ++errors;
            }
        }

        System.out.println("Tests in: " + testLines
            + ", errors: " + errors + ", accuracy: "
            + (testLines - errors + 0.0) / testLines);
    }
}
```

References

- [1] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, October 2004. ISBN 0-262-01211-1.