Astrology: The Study of Astro Teller

Stuart Andrews, Lijuan Cai, David Gondek, Amy Greenwald, Daniel Grollman, Arni Mar Jonsson, Keith Hall, Matthew Lease, Bryant Ng, John Raiti, Victoria Sweetser, and Jenine Turner

Department of Computer Science
Brown University, Box 1910
Providence, RI 02912

Abstract

In this paper, we describe how we address the ICML 2004 Physiological Data Modeling Contest. For the gender prediction task, we employ 5 off-the-shelf machine learning methods: decision tree, neural networks, naive bayes, logistic regression, and Support Vector Machines. We use neural networks for the context prediction tasks. Most of the methods perform reasonably well, acknowledging the success of machine learning as a field. Moreover, we point out that characteristic attributes are highly correlated to the gender in the training data. Hence we argue if using characteristics for the gender prediction will generalize well.

1 Introduction

Data used for the Physiological Data Modeling Contest was generated from a human subject study conducted over two years. In the course of this study, subjects wore an armband sensor package as they went about their daily activities, and they timestamped the beginning and ending of the activities. This armband could be worn or removed at the subject's discretion, and each wearing of the armband produced a *session* of sensor data. The armband's sensor package produced continuous-valued readings of the subject's motion (accelerometer), heat flux, galvanic skin response, skin temperature, and near-body temperature. The accelerometer produced 5 *channels* of data, and each of the other sensors produced an additional channel, for a total of 9 channels. During each session, sensors were periodically polled for updated readings, and the relative time (in relation to the start of the session) was recorded in addition to the actual sensor readings. Finally, the study also reported 2 discrete *characteristics* for each subject and included these values as part of the subjects' session data.

A data set is composed of sessions and a session in turn is composed of minutes. Sensor data are collected for every minute in a session. Moreover, each minute is annotated with a code which represents the current activity of the subject. The training data consisted of approximately 10,000 hours of data (580K data points distributed over 1410 sessions), with an additional 12,000 hours held out for evaluation. The held out portion contained data from both subjects seen in the training data and "new" individuals and was designed to be distributionally similar to the training data.

Task 1 of the contest required participants to make a binary gender prediction for each session. Tasks 2 and 3 are to identify if the subject is carrying out a particular activity at each minute, indicated by annotation codes 3004 and 5102 respectively. The paper is structured as follows. Section 2 deals with task 1. Section 3 then addresses tasks 2 and 3. In each section, we formulate the problem, describe the methods, and present experimental results on training data. Section 4 concludes our work.

2 Task 1

2.1 Problem formulation

Task 1 is to predict gender for each session. Let $S = \{(\mathbf{x}^i, y^i)\}_{i=1}^{M}$ be the set of M training examples, where $\mathbf{x}^i \in \dot{\mathcal{X}}^+$ is a session instance and $y^i \in \mathcal{Y} = \{-1, 1\}$ is the label. The session space $\dot{\mathcal{X}}^+ = \cup_{t=0}^{\infty} \dot{\mathcal{X}}^t$, where $\dot{\mathcal{X}}^t$ is the Cartesian product of t copies of $\dot{\mathcal{X}}$. The domain of minute readings is denoted by $\dot{\mathcal{X}}$. Let $\mathbf{x}^i = \langle \mathbf{x}_1^i, \ldots \mathbf{x}_{n_i}^i \rangle$, where $\mathbf{x}_j^i \in \dot{\mathcal{X}}$ is the data collected for the j-th minute in the i-th session. The length of session i is n_i. We call $\mathbf{x}_j^i$ a data point or an entry, since it corresponds to a row in the data table. A label of 1 encodes gender 1 and -1 encodes gender 0. The goal of task 1 is to learn a classification function $F : \dot{\mathcal{X}}^+ \rightarrow \mathcal{Y}$ that assigns a gender to a session. Because the data set contained an enormous gender imbalance (95.9% gender 0 by point, 97.4% gender 0 by session), the contest organizers chose to use a balanced accuracy metric rather than simple accuracy in evaluating task 1. Balanced accuracy of F over the data set is defined as

$$bacc(F) \equiv \frac{1}{2} \times \frac{\left|\{i : y^i = 1 \wedge F(\mathbf{x}^i) = 1\}\right|}{\left|\{i : y^i = 1\}\right|} + \frac{1}{2} \times \frac{\left|\{i : y^i = -1 \wedge F(\mathbf{x}^i) = -1\}\right|}{\left|\{i : y^i = -1\}\right|}. \quad (1)$$

A session can have a variable number of data points and can be very long. Hence the question of effectively representing sessions rises. To this end we use a histogram approach. The characteristics are the same across a session. So we simply keep them. The value range of every sensor in the session is divided into 50 buckets. For each sensor, we then devise 53 features: minimum of the sensor value, max value, mean value, and the number of data points falling into each of the 50 buckets. Additionally the session representation includes a feature of the number of data points in the session. We call data formed by the histogram method the compressed data.

An alternative way to look at the problem is to classify each data point and then do a majority vote within a session. Therefore the problem becomes to learn $f : \dot{\mathcal{X}} \rightarrow \mathcal{Y}$, given a training set of $\{(\mathbf{x}_j^i, y^i) | 1 \leq i \leq M, 1 \leq j \leq n_i\}$. We call this an entry-based perspective. Then given a session $\mathbf{x}$, the final classifier is $F(\mathbf{x}) \in \text{argmax}_y \left|\{j : f(\mathbf{x}_j) = y\}\right|$.

Both of the two above perspectives can be unified into the standard binary classification problem. That is, given a training set of $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, to learn a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$. The instance $\mathbf{x}_i$ can represent either a session or an entry, depending on the underlying framework of choice.

We applied five well-established machine learning algorithms to task 1: *decision tree*, *neural networks*, *naive bayes*, *logistic regression*, and *Support Vector Machines (SVM)*. The algorithms and instance representations are two independent issues. In our experiments, we use entry-based representation for the first three algorithms, and session-based compressed representation for logistic regression and SVM. To compare the performances of the various approaches, 10-fold cross-validation is used.

2.2 Data analysis

It is well known that machine learning efficacy on a given task is affected by feature selection as well as by the choice of the learning algorithm. As such, we performed some rudimentary analysis of the data to guide our approach. One challenge in performing this analysis was that no distinguishing labels were provided for the 9 sensor channels or the 2 characteristics (i.e. there was no indication which channel corresponded to which sensor, and the characteristics were not defined). However, we were still able to glean several insights from the data in spite of this. For notational convenience, we will henceforth arbitrarily assume Gender=0 denotes male and Gender=1 denotes female.

By counting the number of unique subject IDs in the training set, we determined that the set was derived from 18 individuals, consisting of 12 males and 6 females. Given that the imbalance of male to female data was much higher than 2:1, this demonstrated that males provided significantly more data on average than did females, as is shown in Figure 1. Moreover, the figure demonstrates a large variance in data supplied per individual: for example, subject 19 accounted for almost 75% of the female data.

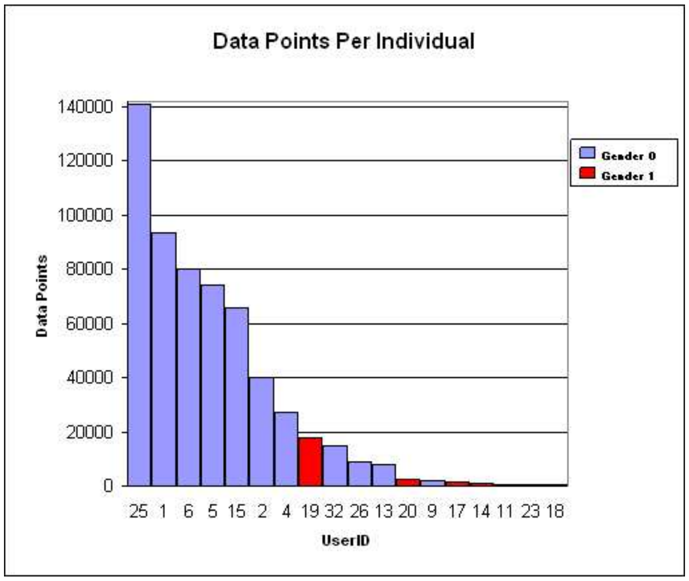

Figure 1: The number of data points each participant contributes.

Characteristic 1 was discrete and ranged from 24 to 58, with most of the subjects being between 30-45. We also noticed that 10 of the subjects had two sequential values for this characteristic. Given this information, we hypothesize characteristic 1 corresponds to age, with the change in value corresponding to a birthday. If correct, this hypothesis suggests characteristic 1, at least by itself, should not correlate with gender (assuming the study subjects represented a random sampling of the population). A quick calculation shows the average value for each gender is nearly identical (36 male, 35.8 female, where the younger age of each subject is used). To test if this is a significant difference of means, we performed a two-tailed t-test of means-independent samples (we use a two-tailed test because we are interested in deviance from the mean in either direction) [1, 2]. Our null hypothesis H_0 was that there is no significant difference between population means ($\mu_m = \mu_f$):

$$\begin{aligned} S_P^2 &= \frac{\sigma_m(n_m - 1) + \sigma_f(n_f - 1)}{\upsilon} \\ t &= \frac{(\overline{x}_m - \overline{x}_f) - (\overline{\mu}_m - \overline{\mu}_f)}{\sqrt{S_P^2\left(\frac{1}{n_m} + \frac{1}{n_f}\right)}}, \end{aligned}$$

where σ denotes variance, the number of samples is n, the degrees of freedom $\upsilon = n_m + n_f - 2$, S_P^2 denotes pooled estimate of the population variance, and $\overline{x}$ denotes sample mean. For $\upsilon = 16$, $t_{0.05} \approx 2.11$, and so our calculated $t = 0.035 < 2.11$ means that at the 0.05 level we cannot reject the null hypothesis.

We found the opposite to be true with regard to characteristic 2: men and women differed significantly. Characteristic 2 was binary, and whereas it was true for none of the males, it was true for 4 of the 6 females. To test the significance of this, we used the Dunning statistic [3] to evaluate whether the two binomial distributions were equivalently parameterized ($H_0 : p_m = p_f$).

$$\begin{aligned}
L(p, n, k) &= p^k (1-p)^{(n-k)} \\
p_m &= \frac{k_m}{n_m},\ p = \frac{k_m + k_f}{n_m + n_f} \\
\lambda &= -(\frac{L(p_m, k_m, n_m) L(p_f, k_f, n_f)}{L(p, k_m, n_m) L(p, k_f, n_f)}\bigg)
\end{aligned}$$

Dunning showed that $-2 \log \lambda$ is distributed asymptotically according to χ^2 with 1 degree of freedom. For the training data, $-2 \log \lambda = 4.965$, indicating that a difference in parameterization is nearly 12 times more likely according to χ^2 ($p = 0.026$). Therefore, we concluded that characteristic 2 was correlated with gender.

In analyzing the characteristic data, we saw that it was possible to achieve 100% accuracy on the training data using only the characteristics with decision rule

$$\text{gender} = 1 \ \Leftrightarrow\ (char2 = 1) \vee (char2 = 0 \wedge (char1 = 29 \vee char1 = 42)) . \tag{2}$$

Based on this as well as simple analysis of the sensor data, we saw the characteristic data to be significantly more discriminative than the sensor data, and so we expected our knowledge-blind learning algorithms would likely give these characteristics much more weight than the sensor data. This was a concern since, given the analysis above, we doubted characteristic 1 would be valuable in determining gender. Consequently, we considered manually discounting characteristic 1 or excluding it altogether in making gender predictions. We were similarly concerned that our automated techniques might rely too heavily on characteristic 2; while characteristic 2 was false for all 12 males in the training set (a significant correlation), this did not guarantee characteristic 2 would also be false for all males in the test set. Therefore all our experiments are run with three feature sets comprising only the 9 sensor features, both the sensor features and characteristic 2, and the 9 sensor features together with the 2 characteristic features respectively.

2.3 Decision trees

2.3.1 Overview

A decision tree is a set of simple classifiers arranged in a tree. Each internal node is a classifier, and each leaf represents a single class. A classification is done by performing a test at a node, and based on the result of that test, traveling down to one of its children, and so on until a leaf is reached. The class of the leaf is then the classification result.

Each classifier is very simple, and classifies using only a single attribute. It can use any kind of test including equality, inequality, or numerical comparisons. A decision tree can be translated into a logical formula for each class. If you take a single path down the tree, it can be translated into a conjunction of conditions, and the conjunctions for paths to leafs of the same class can be combined disjunctively. This means that decision trees can represent any logical formula. This also means that it is relatively easy to understand the reasons for the decisions in small trees.

The number of decision trees is potentially infinite, For this reason most growing algorithms are heuristic based, they successively select a best candidate leaf and split that, until some terminating criteria are reached. One of the first growing algorithm is called ID3 [4]. ID3 builds a decision tree for datasets which have discrete attributes. The pseudocode is shown in Algorithm 1

The information gain, $Gain(A, S)$ measures how well a given attribute A separates the training instances S according to their classes. Ideally, we want to split a leaf such that each of its children contain only a single class, or as much of one class as possible. $Gain(A, S)$ is defined in terms of $Entropy(S)$, the entropy

Algorithm 1 ID3(R, C, S)

```
Input: $R$: attributes; $C$: non-empty set of classes; $S$: non-empty training set.
if all instances of $S$ are of the same class then
  return leaf of that class
else if $R$ is empty then
  return a leaf of the most common class in $S$
else
  Let $A$ be the attribute with largest $Gain(A, S)$ of attributes in $R$.
  Let $S_j$ $(1 \leq j \leq n)$ be the disjoint subsets having separate values of $A$.
  return a node classifying on $A$, having children ID3($R - \{A\}$, $C$, $S_1$), ..., ID3($R - \{A\}$, $C$, $S_n$)
end if
```

Algorithm 2 BUILD(S)

```
Input: $S$: training samples
repeat
  for all leaf $l$ containing samples $S_l$ do
    for all attribute $A$ and all possible splits along the attribute do
      evaluate the information gain. let the maximum be $Gain(A, S_l)$
    end for
  end for
  let $(\hat{l}, \hat{A}) = \text{argmax}_{l,A}\, Gain(A, S_l)$
  split leaf $\hat{l}$ with the corresponding split over attribute $\hat{A}$.
until $Gain(\hat{A}, S_{\hat{l}}) < \alpha$
```

of S relative to the binary classes. The definitions are in Eq. (3) and (4).

$$Entropy(S) = \sum_y P_s(y) \log(P_s(y)) \text{ with } P_s(y) = \frac{|\{i : i \in S \wedge y_i = y\}|}{|S|} \tag{3}$$

$$Gain(A, S) = Entropy(S) - \sum_{v \in Values(A)} \left(\frac{|S_v|}{|S|} * Entropy(S_v) \right) \tag{4}$$

$P_s(y)$ in Eq. (3) is the fraction of instances in S of class y. For a binary classification problem, $Entropy(S)$ is maximized when $P_s(0) = P_s(1)$, and minimized when $|P_s(0) - P_s(1)| = 1$. $Gain(A, S)$ is the difference between the current node entropy and the entropy of the resulting nodes, weighted by their relative size.

Extending this algorithm to handle continuous attributes is simple. The number of possible splits for a continuous attribute is infinite, but to simplify things only splits halfway between attribute values of training samples are considered. Also, a different terminating criterion is used. Instead of stopping when we run out of attributes, we stop when the maximum entropy reduction falls below a given threshold α. Algorithm 2 describes the procedure.

2.4 Experiments

For the experiment, we consider various α values in the range of $[10^{-15}, 0.02]$. The rough trend is that the smaller α is, the better the performance is. When α is 0.02 and only sensor features are used, 9 out of 10 times there are no candidate splits that can achieve an entropy reduction that high. So the generated decision tree simply return the dominant class, i.e. gender 0.

Experimental results are summarized in Table 1. When both characteristics are present in the feature set, they appear to be the features that best classify the training examples. Actually using only the two characteristics generate the same decision tree as using additional sensor data.

α	nochar	char2	chars
0.0001	65.1	91.2	93.1
0.001	62.8	87.9	88.4
0.15	61.2	81.2	83.4
0.02	50.0	81.3	83.4

Table 1: Balanced accuracy of decision trees with various parameter settings. 10-fold cross validation is performed. 'nochar' refers to using only the 9 sensor features, 'char2' using sensor features plus characteristic 2, and 'chars' using features plus the two characteristics.

2.5 Neural Networks

2.5.1 Overview

Inspired by biology, neural networks are a way to perform complicated computations using many simple units. Each node, or neuron, in the network does a simple arithmetic operation, and gives the result to all of its successor nodes. When arranged in the appropriate network topology, arbitrary functions from the input to the output can be learned.

The simplest neural network consists of some input nodes and one output node, with all the input nodes connected directly to the output node [5]. These are known as perceptrons, and have limited representational power, due to their simplicity. In a perceptron, the output is usually some function of the weighted sum of the inputs. That is, the output node computes $g(\sum_j w_j * x_j)$ over j inputs and weights. g is the activation function, e.g. a step or sign function.

For representing complex relations, it is common to see networks with one or more hidden layers, that are neither inputs nor outputs, but intermediaries between them. It has been shown that with one layer of a large enough number of hidden units, any continuous function of the inputs can be represented, and with two hidden layers, discrete functions can be realized as well. Each node in a hidden layer can be seen as a perceptron, and computes the above equation over its own inputs, with its own weights.

For the gender prediction task, we chose to use a 3-layered feed-forward neural net. A feed-forward network topology is one where information flows only in the forward direction, and the input to a node never depends on its output. Our three layers are the input layer comprising 9 to 11 nodes depending on the feature set, a 1-node output layer, and a 5-node hidden layer, to bridge the gap.

We chose sign function as the activation function. Hence each non-input node i calculates $\text{sign}(b^i + \sum_j w_j^i * x_j)$. This is the sign of a weighted sum of its inputs, and a node-specific offset. In the case of the output node, a sign of 1 is mapped to gender 1, and a sign of -1 is mapped to gender 0. Each data point in a session is tagged with its own gender prediction, and the gender for the session is computed from these individual genders by majority vote, with ties going to gender 1.

2.5.2 Genetic algorithms

Genetic algorithms depend on randomness to take decent solutions and make better ones [5]. This is done by combining solutions in such a way that the resulting solution may be better than the initial ones. Once a new set of solutions are created, the best are taken, and the process is repeated. This is analogous to Darwinian evolution, or survival of the fittest. In each generation, the best solutions survive, and reproduce, to make the new generation, which is once again tested for survival. After a pre-set number of generations, the algorithm terminates, and the best solution is returned.

In our setup, since the network topology is set, a solution is just the weights on each link. To combine solutions, they are 'bred' in a pseudo-genetic way as described in algorithm 3.

We start our algorithm with 10 randomly generated seed solutions, which are bred pairwise to create an initial 55-solution genetic pool. The fitness of each solution is measured by the balanced accuracy, and the top 10 are selected for breeding. This generates 45 new children solutions, which, combined with the 10

Algorithm 3 Breed(A, B)

```
Input: $A$ and $B$ are two parent neural networks with identical topologies.
Output: $C$: a new child solution.
for all link $j$ in the topology do
    with probability MUTATION, assign a random value to child C's weight $C.w_j$
    with probability (1-MUTATION)/2, assign $A.w_j$ to $C.w_j$
    with probability (1-MUTATION)/2, assign $B.w_j$ to $C.w_j$
end for
return C
```

Algorithm 4 The entire genetic training algorithm

```
Randomly initialize 10 solutions in $top10$ buffer
for each generation do
    Breed the $top10$ solutions to create 45 new solutions
    Test the fitness of all 55 solutions
    Select the 10 best fittest solutions and store in $top10$
end for
Return fittest solution in $top10$ buffer
```

parents, provides the pool for the next generation. The 10 parents are carried over to guarantee that the best fitness in the pool is monotonically increasing. After a pre-set number of generations, the best solution is returned. Algorithm 4 describes the entire genetic training .

2.5.3 Experiments

With a fixed network, the two main parameters to adjust in the genetic algorithm are the number of generations, and the probability of mutation. We present results for networks trained without characteristics, using only characteristic 2, and with both characteristics in Fig. 2. Each result is the average fitness over 10-fold cross-validation.

The best network appears to be the one that was trained for 60 generations, with a mutation rate of 10%. More generations seems to lead to overfitting, that is, the network learns the training data too well, and does poorly on the testing data.

2.6 Naive Bayes

2.6.1 Overview

Naive Bayes combines Bayes' Rule with a naive assumption of conditional independence to estimate the posterior probabilities of each gender. This creates a simple but elegant method of classification.

Generation	Mutation	nochar	char2	chars
30	0.1	0.557695	0.676181	0.657635
	0.2	0.600293	0.559535	0.670882
60	0.1	0.630027	0.648306	0.725815
	0.2	0.595819	0.690261	0.666031
100	0.1	0.585729	0.617024	0.663296
	0.2	0.579335	0.633216	0.598101

Table 2: Balanced accuracy of neural networks with various parameter settings.

sensors only	48.7%
characteristics only	85.3%
sensors and characteristic 2	48.7%
sensors and both characteristics	48.7%

Table 3: Experimental results of naive bayes

Bayes theorem states that

$$P(Y = y|\psi_1(X) = a1, \ldots, \psi_k(X) = a_k) = \frac{P(\psi_1(X) = a1, \ldots, \psi_k(X) = a_k|Y = y)P(Y = y)}{P(\psi_1(X) = a1, \ldots, \psi_k(X) = a_k)}, \quad (5)$$

where $\psi_1(X), \ldots, \psi_k(X)$ are the features of independent variable X and Y is the class variable. Choosing the gender y which maximizes Eq. (5) yields the maximum a posteriori classification. We can drop $P(\psi_1(X) = a1, \ldots, \psi_k(X) = a_k)$ because it remains constant over all possible genders.

In order to combine evidence from all of the sensor and characteristic data, the features are assumed to be conditionally independent of each other. This is clearly not true. For example, a person is more likely to have a high skin temperature when he or she also has a high near-body temperature. However, this simplifying assumption allows us to easily combine a large number of features, and is often quite effective. The modified decision rule is

$$P(Y = y|\psi_1(X) = a1, \ldots, \psi_k(X) = a_k) \propto P(Y = y) \prod_{i=1}^{k} P(\psi_i(X) = a_i|Y = y) \quad (6)$$

To calculate $P(\psi_i(X) = a_i|Y = y)$, a Maximum Likelihood estimation from the labeled training corpus is most often used. For this application, however, we wanted to capture the continuous nature of the sensor data. The distribution of each sensor, conditioned on each gender, was fitted with a Gaussian curve.

2.6.2 Experiments

Using the sensor data to predict gender never helped at all. We had a much better result just using the characteristic data, as is shown in table 3. The results for the sensors were so because the naive bayes method nearly always guessed gender 0.

2.7 Logistic regression and Support Vector Machines

2.7.1 Hyperplane Learning

In this section, we describe three learning algorithms under a common framework that is motivated by regularization theory. We applied two of these algorithms to the compressed data described in Section 2.1; i.e. we assume the data we have are a labelled data set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ where each $\mathbf{x}_i \in \mathbb{R}^d$ is the compressed representation for a session, and $y_i \in \{-1, +1\}$ denotes the gender. We further assume that the data are drawn i.i.d. from a fixed, but unknown joint distribution $P(\mathbf{x}, y)$.

In each case, the *learning* will consist of selecting a simple linear function $f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$ parameterized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Notice that the equation $f(\mathbf{x}; \mathbf{w}) = 0$ describes a hyperplane; we use the hyperplane to classify patterns $\mathbf{x}$ via $\hat{y} = F(\mathbf{x}) = \operatorname{sign} f(\mathbf{x})$. For simplicity of presentation, we will drop the bias term b.

The quality of a proposed classification rule $F(\mathbf{x})$ is measured by its ability to *generalize*, or predict the correct genders y, for new compressed session patterns $\mathbf{x}$. Thus, we want to minimize the error of the classifier, averaged over the distribution of instances $(\mathbf{x}, y)$. Mathematically, we write the expected error

$$\operatorname{err}(F) = \int \operatorname{loss}_0(y, f(\mathbf{x}))dP, \quad (7)$$

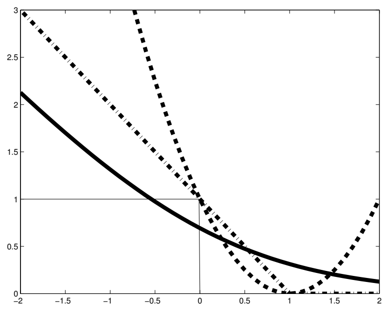

Figure 2: Zero-one, residual sum of squares[dot], Hinge[dash dot], and NLL-binomial[solid] loss functions. Each loss$(y,\ f(\mathbf{x}))$ is plotted versus the quantity $yf(\mathbf{x})$, where $y \in \{-1,\ 1\}$, and $f(\mathbf{x}) \in \mathbb{R}$.

where the integral is evaluated over the unknown joint distribution $P(\mathbf{x}, y)$. The *zero-one* loss function $\mathrm{loss}_0(y,\ f(\mathbf{x}))$ is 1 if and only if the predicted classification $\hat{y} = F(\mathbf{x})$ is incorrect (see Figure 2). Since the distribution $P(\mathbf{x}, y)$ is unknown, the best we can do is to approximate the right-hand side of Equation (7) using a finite sample $\mathcal{S}$.

$$\frac{1}{N}\sum_{i=1}^{N} \mathrm{loss}_0(y_i,\ f(\mathbf{x}_i))\,.$$

Notice that the accuracy of the classifier is $\mathrm{acc}(F) = 1 - \mathrm{err}(F) = \frac{tp+tn}{tp+fn+tn+fp}$, and the balanced-accuracy score is $\mathrm{bacc}(F) = \frac{1}{2}\left(\frac{tp}{tp+fn} + \frac{tn}{tn+fp}\right) \geq \frac{1}{2}\mathrm{acc}(F)$. Minimizing the expected error is equivalent to maximizing the accuracy, which in turn is a lower bound on the balanced-accuracy score.

2.7.2 Regularization Framework

As we have seen, we can hope to find a suitable classifier $F(\mathbf{x})$, by selecting $\mathbf{w} \in \mathbb{R}^d$ such that $\sum_i \mathrm{loss}_0(y_i,\ f(\mathbf{x}_i))$ is minimized. This turns out to be a difficult optimization problem (c.f. [6]) and approximations are needed. To avoid over-fitting, we simultaneously minimize a term $J(F)$ intended to penalize overly-complex classifiers, a technique known as regularization. Thus, the general formulation in this framework is

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \mathrm{loss}(y_i,\ f(\mathbf{x}_i)) + \frac{\lambda}{2} J(F)$$

for a given loss function, and a regularization term. The techniques considered in this section differ according to how they approximate $\mathrm{loss}_0(y,\ f(\mathbf{x}))$, and the methods used to optimize their respective objective functions.

1. Ridge Regression

   For the first algorithm, we use the *residual sum of squares* (RSS) loss function in place of the zero-one loss

   $$\mathrm{loss}_{RSS}(y,\ f(\mathbf{x})) = (y - f(\mathbf{x}))^2$$

   and use the norm $\|\mathbf{w}\|^2$ of the weight vector as the regularizer. This results in the optimization problem

   $$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

   which is identical to a least-squares approach apart from the regularization term.

While the RSS loss (see Figure 2) is indeed an upper-bound on the zero-one loss, it has the undesirable property of increasing from zero when $yf(\mathbf{x}) \geq 1$, despite the fact that the classifier is correct in this region. Moreover, since the RSS loss increases quadratically as the prediction error increases, it places most emphasis on a small number of outlier points, and practically ignores those with small prediction errors. The solution is found by solving the normal form equations for this problem.

2. Logistic Regression

   For logistic regression, we use as the loss function the negative log-likelihood of a binomial probability density function (c.f. [7])

$$\text{loss}_{NLLB}(y,\, f(\mathbf{x})) = \log(1 + \exp(-yf(\mathbf{x})))$$

   and again use the norm $\|\mathbf{w}\|^2$ of the weight vector as the regularizer. This results in the optimization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \log(1 + \exp(-y_i f(\mathbf{x}_i))) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \ .$$

   Except for a narrow range just below 0, the NLLB loss is an upper-bound on the zero-one loss (see Figure 2). This loss is an improvement over the RSS loss for two reasons: for positive $yf(\mathbf{x})$, the NLLB loss is close to zero; and, as $yf(\mathbf{x})$ approaches negative infinity, the NLLB loss increases only linearly. One advantage of logistic regression is that it yields an estimate for the conditional probability of the label $p(y|\mathbf{x}) = (1 + \exp(-yf(\mathbf{x})))^{-1}$, whereas other methods only estimate $\text{sign}(p(y|\mathbf{x}) - \frac{1}{2})$. To solve this problem, we use the Newton-Raphson method to find a zero of the objective's derivative w.r.t. $\mathbf{w}$. The result is an iterative re-weighted least squares algorithm (IRLS).

3. Support Vector Machines

   Support Vector Machines (SVM) are a learning algorithm that efficiently seek the maximum-margin separating hyperplane in feature space. This approach is motivated from within the regularization framework by considering the hinge loss

$$\text{loss}_{hinge}(y,\, f(\mathbf{x})) = (1 - yf(\mathbf{x}))_+$$

   where we define the truncation $(\mathbf{x})_+ = x$ if $\mathbf{x}$ is positive, and $(\mathbf{x})_+ = 0$ otherwise. Notice that the hinge loss is a true upper bound on the zero-one loss. The norm $\|\mathbf{w}\|^2$ of the weight vector is again used as the regularizer which leads to the following optimization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (1 - y_i f(\mathbf{x}_i))_+ + \frac{\lambda}{2} \|\mathbf{w}\|^2 \ .$$

   Note the connection, via Lagrangian optimization, to the traditional soft margin linear SVM formulation, which is derived with the goal of margin maximization

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{s.t.}\ \ y_i f(\mathbf{x}_i) \geq 1 - \xi_i,\ \xi_i \geq 0\ (\forall i) \end{aligned}$$

   Due to the truncation in the hinge loss, the contribution $\text{loss}(y_i,\, f(\mathbf{x}_i))$ is often zero for many data points. Thus, the solution w depends only on points that are on the correct side and close to the hyperplane, or those that are misclassified. These are the *support vectors* because they "hold up" the hyperplane.

Kernel Function	Parameters	Formula
Linear	-	$\langle \mathbf{x}_i, \mathbf{x}_j \rangle$
Polynomial	degree d	$\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d$
RBF	sigma σ	$exp^{\frac{-\lvert \mathbf{x}_i - \mathbf{x}_j \rvert^2}{(2\sigma)^2}}$

Table 4: Kernel functions

	λ	T	balanced accuracy
nochar	100	5	0.8664
char2	10	5	0.9527
chars	1	1	0.9519

Table 5: Results of logistic regression.

2.7.3 Kernelization

One can often improve the performance of a classification algorithm by augmenting the feature representation of the patterns. For example, one might consider adding features that are the products of existing feature pairs. Intuitively, by increasing the dimension of the patterns, we are creating more "space", making it easier for the classifier to separate the patterns. More generally, we consider functions ϕ mapping patterns $\mathbf{x} \in \mathbb{R}^d$ to a higher dimensional representation $\phi(\mathbf{x}) \in \mathbb{R}^D$ where $D \gg d$. While we are still learning a hyperplane, the resulting discriminant, when evaluated in the original feature space $\mathbb{R}^d$, will often be non-linear.

A naive implementation would simply perform this mapping as a data preprocessing step. The drawback of the naive approach, however, is that the complexity of the algorithm will scale with D. A better approach is to perform the feature mapping ϕ implicitly, as is briefly described below.

Each of the three algorithms outlined above can be reformulated using only inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ (c.f. [8]). Thus, under the naive approach, we must evaluate inner products between high dimensional vectors $\phi(\mathbf{x})$. Instead, we restrict attention to mappings ϕ for which the *kernel* $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ may be computed efficiently, without first mapping the patterns to $\mathbb{R}^D$. The formulae for several common kernels are shown in Table 4. We used this property in our experiments with the SVM.

2.7.4 Experiments

1. Logistic Regression

   This approach is a specific instance of a larger class of techniques referred to as *generalized linear models* (c.f. [9]). As such, we used the `glmfit` routine from Matlab's Statistics Toolbox, specifying the `binomial` link function. Some minor modifications were required to incorporate the $\frac{\lambda}{2} \|w\|^2$ regularization term, which is not currently supported by `glmfit`.

   One feature of the `glmfit` routine is that it accepts weights on the instances. We took advantage of this capability to offset the uneven class sizes. Specifically, the weights for instances from the smaller class were set to $T \times R$. The first factor T was a user-supplied constant, and the second factor $R = N_L/N_S \geq 1$, where N_L, N_S were the sizes of the large, and small classes respectively. The weights for instances from the larger class were set to one.

   We ran full cross-validation experiments to select the best weight scaling factor $T \in \{0.5, 1, 2, 5, 10\}$, and regularization parameter $\lambda \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3\}$. The optimal parameters for different feature sets are reported in Table 5.

2. SVM

   To find the best SVM for gender classification we perform some parameter tuning. For each set of parameters, we trained and tested the SVM over a 10-fold cross validation of data and took the balanced

Kernel	Kernel Parameter	C	nochar	char2	chars
Linear	-	1000	0.77476	0.88293	0.91428
Polynomial	2	1000	0.74559	0.78761	0.78418
	3	1000	0.73792	0.79719	0.81604
RBF	40	1000	0.73131	0.84001	0.88277
	60	1000	0.76048	0.85141	0.88167
	80	1000	0.76520	0.87824	0.86610

Table 6: SVM Results

accuracy average. We chose the best parameters based on this average. For penalty factor parameter C, we tried values of ($C = 10^1, 10^3, 10^6, \infty$). For the polynomial kernel, we tested degrees 1 through 6. For the RBF kernel, we tested σ values from 20 to 80.

We used a matlab library called *Spider*[1] that computed our SVM and the separating hyperplane. Spider is an object-oriented machine learning library that created our SVM model. We put the data into a matrix and gave it to Spider to solve the quadratic optimization problem.

Table 6 summarizes our results. The table only shows the top results for each kernel and parameters. It turned out that a C value of 1000 gave the best results for all kernels. It shows that the linear kernel gave the best results.

3 Tasks 2 and 3

3.1 Problem description

Tasks 2 is to determine if a data point is in context 1, identified by annotation code 3004. Task 3 is to decide if a data point is in context 2, identified by annotation code 5102. The metric provided for the tasks are

$$\text{score} = 0.3 \times \frac{\text{TruePositives}}{\text{TruePositives+FalseNegatives}} + 0.7 \times \frac{\text{TrueNegatives}}{\text{TrueNegatives+FalsePositives}} \tag{8}$$

Also 10-fold cross-validation is conducted in the experiments.

3.2 Neural Networks

3.2.1 Feed-forward network

For tasks 2 and 3, we adapted the gender classification code to operate on the annotations. That is, for the same inputs, the nine sensor readings and characteristics, the network now output a binary value symbolizing if the data point was in the given context or not. The fitness function was changed to the score in Eq. (8). In addition, since in each context there are several vague annotations that can be validly classified either way, all outputs for data points bearing these annotations were ignored during fitness calculations.

Results for the two contexts were very different, with the neural net performing much better on context 2 than on context 1. With the basic feed-forward neural network, as described in Section 2.5, the best results for context 1 were 70.2%, and for context 2 86.5%. These results were obtained with 60 generations of evolution, and a mutation rate of 10%.

[1] http://www.kyb.tuebingen.mpg.de/bs/people/spider

3.2.2 Recurrent network

In an attempt to improve the results, especially for context 1, the neural network was changed to allow recurrence. That is, the results from one time step, or data point, were fed back into the network to influence the results at the next data point. This enables the network to learn relationships not only between the sensor readings at a data point and its annotation, but also between the previous annotation and the current annotation. This is done by taking the predicted annotation of the previous data point, and using it as another input into the neural network. In addition, the annotations from more than one previous data point can be used, to increase the amount of information available to the network.

However, since the neural network only outputs a binary value representing whether or not the data point is in the context, and not the actual annotation number, much information is already lost. In actuality, the network learns a relationship between whether or not it thinks the previous data points were in the context, and whether or not it thinks the current data point is in the context.

Our initial experiments with 5 previous classification results being fed back into the network appeared unsatisfactory. Little to no improvement over the feed-forward network was seen. For instance, the best results for context 1 were 70.5%, a difference that could be entirely attributed to the randomness inherent in the neural network. Closer examination of the weightings discovered by the network showed that the previous annotations were weighted very lightly, suggesting that they were not good indicators of the current context classification.

4 Conclusions

Task 1 of the contest is to predict gender for sessions. We formulate this as a binary classification task, with each instance either being a session or being a data point. In the former case, a histogram-based compression is performed to efficiently represent the session data. In the latter case, a majority vote is conducted to reach consensus classification within a session. We applied 5 well-known classification algorithms: decision trees, neural networks, naive bayes, logistic regression, and SVM. Most of them perform reasonably well, with decision trees, logistic regression and SVM perform the best.

In the training set, we observe the two characteristics are very strongly correlated with the gender. Throughout the various methods, we observe using additional characteristic features lead to significantly better performance than using sensor features only. Actually we hand-crafted a decision tree using only the characteristics that are able to classify the training data perfectly. But since there are only 18 subjects and hence 18 pairs of characteristic data, we predict that the simple classifier will suffer from the sparse data problem and hence not generalize well.

The only metric in task 1 is the balanced accuracy, the average of positive instance accuracy and negative instance accuracy. The methods we employ, however, aim at optimizing other metrics such as accuracy or separation margin. This then creates a gap between the learning purpose and the application purpose. Although we did not explore this direction, it is reasonable to project methods that directly optimize the balanced accuracy on the training set will perform better in terms of the balanced accuracy. The measure causes an additional problem for our entry-based approaches, since the training procedure optimizes an entry-wise objective while the testing is session-wise.

Tasks 2 and 3 are to predict if a data point is in a particular context. We use neural networks trained with genetic algorithm for these two tasks. Both feed-forward networks and recurrent networks are employed. In the experiments, the network does no better if recurrence is allowed. In addition, context 2 appears to be much more amenable to classification, with results a full 15% better than those of context 1 from a basic feed-forward network.

References

[1] Pinkmonkey.com statistics study guide. URL, 2004.
http://www.pinkmonkey.com/studyguides/subjects/stats/contents.asp.

[2] Anthony Woods. *Statistics in Language Studies.* Cambridge University Press, 1986.

[3] Ted Dunning. Accurate methods for statistical surprise and coincidence. *Computational Linguistics*, pages 61–74, 1993.

[4] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[5] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley Interscience, XXXX, 2000.

[7] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems (NIPS*13)*, pages 1081–1088, 2001.

[8] B. Scholkopf and A. Smola. *Learning with kernels.* MIT Press, Cambridge, 2002.

[9] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall, London, 1983.