University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Krzysztof Jerzy Geras
Student no. 248264

Prediction Markets for Machine Learning

Master thesis
in COMPUTER SCIENCE

Supervisor:
Prof. Andrzej Szałas
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

External supervisor:
Dr. Amos Storkey
School of Informatics
University of Edinburgh

September 2011

Supervisor's statement

Hereby I confirm that the present thesis was prepared under my supervision and that it fulfils the requirements for the degree of Master of Computer Science.

Date Supervisor's signature

Author's statement

Hereby I declare that the present thesis was prepared by me and none of its contents was obtained by means that are against the law.

The thesis has never before been a subject of any procedure of obtaining an academic degree.

Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Author's signature

Abstract

The main topic of the thesis is prediction markets as a machine learning tool. The thesis gives an overview of current state of the art in this research area, relates artificial prediction markets to existing well-known model combination techniques and shows how they extend them. It also develops techniques introduced in one of previously known frameworks, contains a description of a first practical implementation of this framework and evaluates its performance on both synthetic and real data sets from UCI machine learning repository. Finally, results of this evaluation are utilised to understand strengths and weaknesses of this approach and to suggest future directions in this research area.

Keywords

machine learning, artificial prediction markets, classification algorithm, prediction markets, model combination

Thesis domain (Socrates-Erasmus subject area codes)

11.4 Artificial Intelligence

Subject classification

I. Computing Methodologies
I.2. Artificial Intelligence
I.2.6. Learning

Contents

Acknowledgements

I would like to thank a few people who helped me. My supervisors, ***Dr. Amos Storkey*** *and* ***Prof. Andrzej Szałas*** *for their support and advice.* ***Jonathan Millin*** *for invaluable collaboration.* ***Catriona Marwick*** *for checking spelling and grammar in draft versions of this document.* ***Dr. Marcin Mucha*** *for giving me an idea how to generate a bistochastic matrix.* ***Dr. Charles Sutton*** *and* ***Athina Spiliopoulou*** *for advice on convex optimisation. Finally, I would like to thank the School of Informatics of the University of Edinburgh for letting me work on this project using its research facilities.*

Introduction

0.1. Project aims and contributions

This thesis has three main aims. The first one is to give an overview of current state of the art in using prediction markets as a machine learning tool, relate them to existing well-known model combination techniques and show how they extend them. Secondly, to further develop techniques in the framework from [Sto11], create its first practical implementation and evaluate its performance. Finally, using results of this evaluation, develop better understanding of strengths and weaknesses of this approach and suggest future directions in this research area.

The method of looking for the market equilibrium and the complete implementation of market mechanisms are the most important contributions of this thesis.

0.2. Thesis outline

The latter part of this thesis has the following structure:

- Chapter 1 contains a very brief general introduction to the basic concepts of machine learning and current methods for combining classifiers as well as describes various motivations for combining models.
- Chapter 2 introduces basic prediction market concepts necessary in the latter parts, describes how real prediction markets are used in practice, outlines their advantages as a machine learning tool and introduces notation for the rest of the thesis.
- Chapter 3 gives a summary and critical evaluation of one specific approach to utilizing prediction markets as a stool for machine learning.
- Chapter 4 introduces formalism alternative to the one from Chapter 3 and highlights its advantages. It also contains an explanation of how to find an equilibrium in this setup and relates this approach to other machine learning techniques.
- Chapter 5 gives a description of a practical implementation of machine learning markets in the *Edinburgh formulation*. It also contains experimental results obtained on a number of data sets and market parametrisations.
- Chapter 6 summarizes the thesis and suggests future research directions.

Chapter 1

Background

Machine learning studies computer algorithms for learning to do stuff.

Professor Robert Schapire

1.1. Supervised and unsupervised learning

If an algorithm is given a set of inputs (also called features vectors) $\{x_1, x_2, \ldots, x_n\}$ and a set of corresponding outputs (also called labels) $\{y_1, y_2, \ldots, x_n\}$ and the goal of the algorithm is to learn to produce the correct output given a new input, this is a *supervised learning* task. A supervised learning task is called *classification* if the outputs are discrete or *regression* if the outputs are continuous.

If an algorithm is only given a set of inputs $\{x_1, x_2, \ldots, x_n\}$ and no outputs, this is an *unsupervised learning* task. Unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise. Two most common examples of unsupervised learning are clustering and dimensionality reduction.

Inputs can be vectors of different types of objects, integer numbers, real numbers, strings or more complex objects. Outputs take values each representing a unique state. For example, an algorithm may be given a number of vectors representing numerically external features of a person, such as height, weight, shoe size etc., and corresponding outputs that take one value from the set $\{male, female\}$.

1.2. Classification

Classification is a particular supervised learning task. The objective of a classification algorithm is, given a set of inputs X, that come from a space F^L, and corresponding outputs Y, that belong to one of K classes, to learn a function $h : F^L \to \{1, \ldots, K\}$ that can predict an output for unseen input.

Usually a good measure of performance of a classifier is *misclassification rate*[1]:

[1]

$$[a] = \begin{cases} 1 & \text{if } a = true, \\ 0 & \text{if } a \neq true. \end{cases}$$

$$MisclassificationRate = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq y_i]. \tag{1.1}$$

Sometimes it's more convenient to use *accuracy*, which is equal to $1 - MisclassificationRate$.

If an algorithm outputs a probability distribution over classes and not just predicted class, then it learns a function $\hat{h} : F^L \times \{1, \ldots, K\} \to [0, 1]$ ($h(x_i) = \arg\max_k \hat{h}(x_i, k)$). In that case, a good complementary measure can be *log-likelihood*:

$$log\text{–}likelihood = \sum_{i=1}^{N} \log \hat{h}(x_i, y_i). \tag{1.2}$$

This may be a good measure in some circumstances as it does not asses how often the classifier was right, instead assessing how much probability mass it put on the correct class.

Practical classification algorithms include: support vector machines, logistic regression, neural networks, k nearest neighbours, decision trees and Naive Bayes. Ensemble methods (section 1.4) build more complex models using these. Please see [HasTibFri09] for a comprehensive overview.

It is important to understand that function h not only has to describe the training data (X, Y) but also has to be able to generalize to unseen instances. If a classifier has a very low misclassification rate on training data but high misclassification rate on test data, it is said to *overfit* to the training data. That means that the classifier models the noise in the training data and not the true underlying pattern that is of our interest. Overfitting occurs primarily when the model has too many parameters given the amount of training data available. Therefore, it is very often reasonable to favour simpler models to more complex ones. Figure 1.1 shows an example of this phenomenon.

Predicting classifier's performance on unseen data, should never be done on the training data, overfitting is one the reasons. It is typically the best approach, to partition the data into three sets: *training set* - to train the classifier, *validation set* - to tune classifier's parameters and *test set* - to predict performance on new instances.

1.3. Clustering

Clustering is an unsupervised learning task. The objective is to divide a set of objects, represented by inputs $\{x_1, x_2, \ldots, x_n\}$, into a set of disjoint clusters $\{\{x_{1,1}, x_{1,2}, \ldots, x_{1,n_1}\}, \{x_{2,1}, x_{2,2}, \ldots, x_{2,n_2}\}, \ldots, \{x_{3,1}, x_{3,2}, \ldots, x_{3,n_3}\}\}$, that contain objects similar to each other in some sense. Typically, similarity between two objects is defined by Euclidean distance[2] or Manhattan distance[3].

For a comprehensive overview on clustering algorithms please refer to [Fun01].

1.4. Ensemble methods

Ensemble methods use multiple models to combine them into one stronger model. Experience shows that it is common for individual algorithms to be outperformed by combinations of

[2] $d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$

[3] $d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n}|p_i - q_i|$

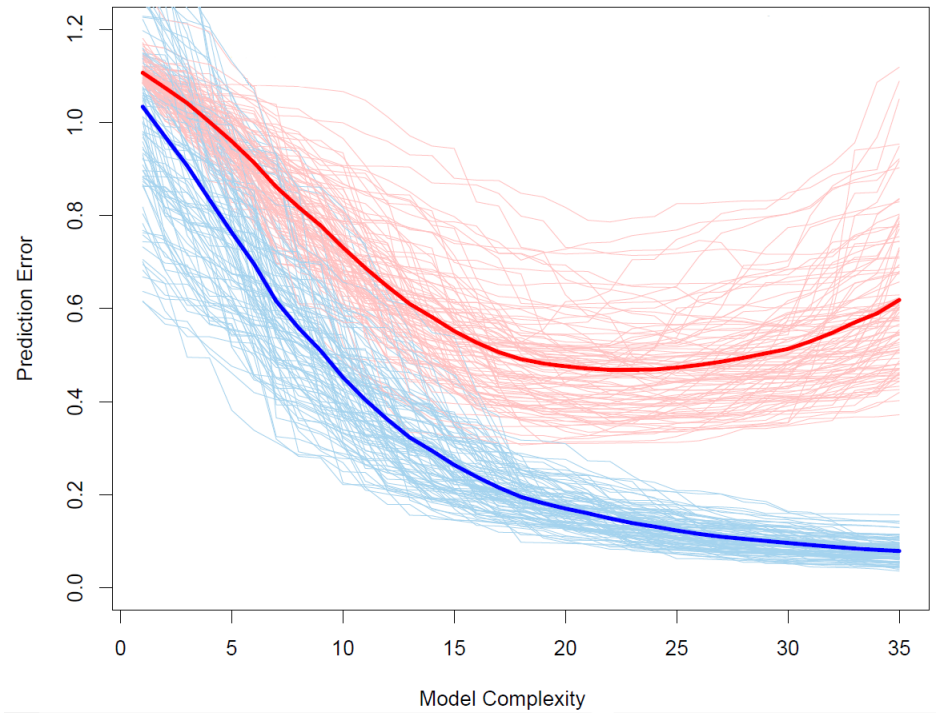
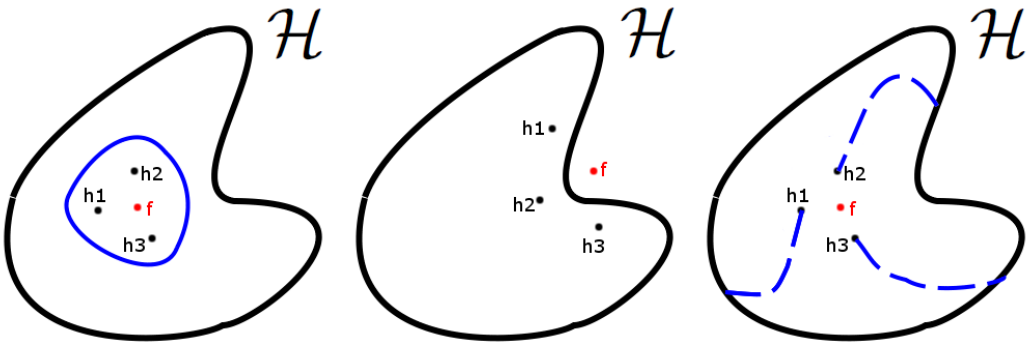


Figure 1.1: Behaviour of test sample and training sample error as the model complexity (understood in an informal manner as some measure of how complex the model is) is varied. The light blue curves show the training error, while the light red curves show the test error for 100 training sets of size 50 each, as the model complexity is increased. The solid curves show the expected test error and the expected training error. Figure from [HasTibFri09]

.



(a) Statistical (b) Representational (c) Computational

Figure 1.2: Reasons why an ensemble of classifiers may work better than a single one.

models and heterogeneous combinations are of special interest ([BelKor07]). As mentioned by [Die00] there are essentially three reasons why ensembles of models perform better than individual models: **statistical**, **computational** and **representational** (figure 1.2).

Statistical. One way to look at a learning algorithm is to view it as searching a hypothesis space $\mathcal{H}$ to find the best one. Without enough data the algorithm may find a few different hypotheses in $\mathcal{H}$ that give the same accuracy on the training (or validation) data. By constructing an ensemble, the algorithm can find a point that is, in a certain sense, an average of the members of the ensemble and thus reduces a risk of choosing the wrong classifier.

Representational. In many cases the true function f cannot be represented within hypothesis space $\mathcal{H}$. By combining classifiers into an ensemble it may be possible to expand a set of representable functions. Even though some algorithms, like neural networks (that actually are universal approximators as shown in [Hor91]), can, in theory, express a lot of functions, it is important to bear in mind that due to the finite amount of training data, they will effectively explore a finite number of hypotheses and will stop when the model fits the training data well enough.

Computational. Even when there is enough data, an algorithm performing local search for the best hypothesis may get stuck in local optima. That is the case for neural networks for example. Therefore, starting from different points and combining obtained models in an ensemble may lead to a model that is closer to the true hypothesis.

The most widely used ensemble methods are Random Forest ([Bre01]) and AdaBoost ([Sch03]).

1.4.1. Random Forest

Entropy and information gain

Definition 1.4.1 Entropy*, is a measure of the uncertainty associated with a random variable. For a discrete random variable X with support $\mathcal{X}$, entropy is defined as*

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{1.3}$$

Definition 1.4.2 *Given discrete random variable X with support $\mathcal{X}$ and Y with support $\mathcal{Y}$, the* conditional entropy *of Y given X is defined as*

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) = -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x) \tag{1.4}$$

Analogous definitions can be formulated for continuous random variables.

Definition 1.4.3 Information gain *is the change in entropy of random variable X when a value of random variable Y is observed, that is*

$$InformationGain(X; Y) = H(X) - H(X|Y). \tag{1.5}$$

Decision trees

A decision tree partitions the feature space into rectangular regions and assigns a class to every region. The process of partitioning the space is performed in such a way that the choice of the variable that will be responsible for current split is done by greedily picking a variable that gives the highest information gain. Decision trees are conceptually very simple yet accurate. Algorithm 1 gives a precise description on how to build a decision tree.

Algorithm 1 Building a decision tree

```
BuildTree(D, attributes)
begin
T ← empty tree
if all instances in D have the same class c then
  label(T) ← c
  return T
else if attributes = ∅ or no attribute has positive information gain then
  label(T) ← most common class in D
  return T
else
  A ← attribute with highest information gain
  label(T) ← A
  for each value a of A do
    D_a ← instances in D with A = a
    if D_a = ∅ then
      T_a ← empty tree
      label(T_a) ← most common class in D
    else
      T_a ← BuildTree(D_a, attributes\{A})
    end if
    add a branch from T to T_a labelled by a
  end for
end if
end
```

Forest of random decision trees

A Random Forest is a set of random trees $\{h_m\}_{m=1}^{M}$ each of which is trained using a slight modification of algorithm 1. Classification is done by majority voting, that is picking a class that was predicted most frequently by the trees in the ensemble. A Random Forest is

Algorithm 2 Building a Random Forest

Input: $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
for $b = 1 \ldots B$ **do**
  $Z \leftarrow$ set of size N drawn from D by sampling with repetitions
  Grow a tree T_b by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
    i. Select m variables at random from the p variables.
    ii. Pick the best variable among the m.
    iii. Split the node into two children nodes.
  $T_B = T_B \cup T_b$
end for
Output: ensemble of trees T_B

conceptually simple, is computationally inexpensive to train, is resistant to overfitting and is a surprisingly accurate classification algorithm.

1.4.2. AdaBoost

Boosting, and also more specifically the AdaBoost algorithm, is based on the observation that finding a single, very accurate prediction rule is much harder than finding many rough rules. AdaBoost builds multiple weak classifiers, feeding a classification algorithm with a different distribution of training examples each round. At the end of that process, the algorithm combines those weak learners, that have to be only slightly better than random, into one strong classifier.

The two important questions in this approach are how to change the distribution of training examples and how to combine classifiers. AdaBoost answers the first question by putting more weight on those examples that are harder to learn and combines classifiers by plain weighted majority voting, where the weights are chosen according to equations (1.6) and (1.7), i.e.:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)[y_i \neq h_t(x_i)] \tag{1.6}$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{1.7}$$

1.5. Bayesian model averaging

Bayesian model averaging is an alternative approach. It answers the following question: "if all the data was generated by exactly one of the hypotheses, what is the probability of observing the new pair (x, y)?". The weights represent uncertainty about what the true hypothesis is. That is[4]:

[4] $\propto$ means "is proportional to".

Algorithm 3 AdaBoost algorithm.

Input: $(x_1, y_1), \ldots, (x_m, y_m)$, where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
for $i = 1 \ldots m$ **do**
  $D_1(i) \leftarrow 1/m$
end for
for $t = 1 \ldots T$ **do**
  train base learner using distribution D_t
  get base classifier $h_t : X \rightarrow \mathbb{R}$
  choose $\alpha_t \in \mathbb{R}$
  update $D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$, where Z_t is a normalization factor such that D_{t+1} will be a distribution
end for
Output: the final classifier $h(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

$$p((x, y)|D) \propto \sum_h p((x, y), D|h)p(h) \tag{1.8}$$

However, as the algorithm is presented with more data, the weight for the most probable hypothesis is going to converge to 1 and all the remaining to 0.

Chapter 2

Basic concepts

2.1. Definitions

Definition 2.1.1 *A* market *is a mechanism for the exchange of goods. The market itself is neutral with respect to the goods or the trades. The market itself cannot acquire or owe goods. Unless explicitly stated otherwise, perfect liquidity*[1] *is assumed and there is no transaction fee. All participants in the market use the same currency for the purposes of trade.*

Definition 2.1.2 Prediction market *(also* predictive market, information market*) is a speculative market created for the purpose of making predictions. Market prices in the equilibrium can be interpreted as predictions of the probability of an event ([Man06], [WolZit04]). In this type of market one type of good is a bet, and the other is a currency. A bet pays a fixed amount dependent on a particular outcome of a future occurrence, and pays nothing otherwise.*

Definition 2.1.3 Artificial prediction market *(also* machine learning market*) is a special type of prediction market. Market participants in the artificial prediction market are classifiers that bet for classes. Equilibrium prices estimate probabilities over classes. The classifiers do not modify their beliefs once they join the market.*

Definition 2.1.4 *The* efficient-market hypothesis *asserts that financial markets are* informationally efficient*. That is, one cannot consistently achieve returns in excess of average market returns on a risk-adjusted basis, given the information publicly available at the time the investment is made.*

Definition 2.1.5 *The* market equilibrium *is a price point for which all agents acting in the market are satisfied with their trades and do not want to trade any further.*

Definition 2.1.6 *A* homogeneous market *is a prediction market such that all the participants are of the same type, that is, they have the same buying function.*

Definition 2.1.7 *An* inhomogeneous market *is a prediction market that aggregates participants who are of different types, that is, they do not all have the same buying function.*

Definition 2.1.8 *A* utility function *is a function $U : X \to \mathbb{R}$ that measures a market participant's satisfaction with its current wealth.*

[1] That is, every good can be bought or sold at every point of time.

Contract type	Example	Reveals market expectation of ...
Winner-takes-all	Event e: Scottish National Party wins elections. Contract costs $\$p$, market's estimation of probability of outcome e. Pays \$1 if and only if event e occurs.	Probability that event e occurs.
Index	Contract pays \$1 for every percentage point won by Scottish National Party. Contract costs amount of money that is equal to the market's expectation of the outcome.	Mean value of outcome e.
Spread	Contract costs \$1. Contract pays \$2 if Scottish National Party wins more than e*% of votes and pays \$0 otherwise.	Median value of outcome e.

Table 2.1: Types of contracts traded in real prediction markets. Based on Table 1 from [WolZit04].

Definition 2.1.9 *A* buying function *(also* betting function*) represents how much a market participant is willing to pay for a good depending on its price. It can be either defined arbitrarily ([Lay09], [LayBar10], [BarLay11]) or derived from a specific utility function ([Sto11], [StoMilGer11]).*

2.2. Real prediction markets

When information is widely dispersed among economic actors, a prediction market offers a mechanism to collect and aggregate that information. Much of the enthusiasm for prediction markets comes from the efficient-market hypothesis (definition 2.1.4). In a truly efficient prediction market, the equilibrium price is the best predictor of how likely an event is, and no other use of available information may lead to improved market-generated forecasts.

Prediction markets were used in a number of contexts, including sporting events, political elections and foreign affairs ([WolZit04]). Evidence from Iowa Electronic Markets (www.biz.uiowa.edu/iem) suggests that prediction markets outperform large-scale polling organisations in political domain ([BerForNelRie08]).

As shown in the table 2.1, there are multiple contract types in real prediction markets. Interestingly, by combining a number of contracts, it is possible to learn more about underlying distribution. For example, consider two index contracts, one that just pays in linearly and a second one that pays according to the square to the index, this is enough to infer the market's belief about the variance of e as $Var[e] = E[e^2] - E[e]^2$. For the purposes of this thesis, however, the only market type considered will be a *winner-takes-all* market type (cf. table 2.1).

One important difference between real prediction markets and artificial prediction markets considered in this thesis is in real prediction markets prices of contracts change as more information becomes available. There is no such a phenomenon here as all the information, that is, the feature vector, is available instantly for all market participants. It is, however, perfectly correct, to consider artificial prediction markets exhibiting different behaviour, even though the benefits of that from a standpoint of classification are doubtful.

2.3. Advantages of artificial prediction markets

Artificial prediction markets have strong foundations in existing knowledge on prediction markets, that have proven to be successful in various scenarios, and have a number of desirable properties that motivate study on them as a machine learning tool. These include:

Flexibility. No assumptions about the type of underlying classifier have to be made as long as it outputs its beliefs as probability distribution over classes. This allows for greater inhomogeneity. Agents can come to the market and leave the market, which means that not all the agents have to participate in the market for every data item.

Probabilistic interpretability. A number of standard machine learning techniques can be shown to be equivalent to this approach.

Parallelism. Agents in the market are naturally independent, the only centralised mechanism in this architecture is the process of finding market equilibrium based on agents' beliefs that can be generated by the agents completely in parallel.

Expressiveness. When constructing standard machine learning procedures, prior beliefs have to be expressed in an explicit form. In this formalism however, the class of possible priors is greater as they can be implicitly defined by the interaction of a number of market participants. Mixing agents of different types capture some of the benefits of both factorial and mixture models.

2.4. Notation

For the purposes of this thesis I am going to follow notation from [Sto11] and [StoMilGer11], that is:

* Market goods are enumerated by $k = 1, 2, \ldots, N_G$, each good corresponds to a value of discrete random variable K (events corresponding to market goods are mutually exclusive and jointly certain).

* The price of k-th market good is denoted by c_k. All prices form a vector $\mathbf{c} = (c_1, c_2, \ldots, c_{N_G})^T$.

* Agents acting in the market are enumerated by $i = 1, 2, \ldots, N_A$.

* The wealth of agent i is denoted by W_i.

* Stockholding (also called position, the amount of good an agent wants to buy) of agent i in good k is denoted by s_{ik}. Stockholding of agents i in all goods is denoted by $\mathbf{s}_i = (s_{i1}, s_{s2}, \ldots, s_{iN_G})^T$. If short positions are allowed, then short position in s_{ik} is indicated by a negative value.

* Each agent in the *Edinburgh formulation* of artificial prediction market has a utility function U_i that implies its buying function $s_{ik}(W_i, \mathbf{c})$ or is given its buying function explicitly (in *Florida formulation*).

* Each agent has a belief P_i, where $P_i(k)$ is the belief of that particular agent i that the value of random variable K will be k, as k enumerates all possible values, $\sum_k P_i(k) = 1$. In the context of classification, this represents agent's belief that for given feature vector $\mathbf{x}$ the correct class is k, more formally $P_i(k)$ is equal to $P(k|\mathbf{x}, i)$.

* A sequence of first t data items in data set D is denoted by D_T and the tth data item is denoted by D_t.

Chapter 3

Florida formulation

This formulation of prediction markets as a tool for machine learning has been introduced first in [Lay09], then subsequently developed in [LayBar10] and [BarLay11]. The key concept in this approach is a *betting function*, which is, essentially, a different name for *buying function*. It determines what fraction of agent's wealth (also called budget) it is going to bet for each class, therefore it is a function of a type $F^L \times [0,1]^{N_G} \to [0,1]^{N_G}$. However, in contrary to *Edinburgh formulation*, these betting functions are arbitrary, meaning that they are not justified by any underlying utility function. Considered functions are (shown in the Figure 3.1 in a binary setting):

- constant,
$$s_{ik} = P_i(k), \tag{3.1}$$
- linear,
$$s_{ik} = (1 - c_k)P_i(k) \tag{3.2}$$
- aggressive,
$$s_{ik} = \begin{cases} 1 & \text{if } c_k \leq P_i(k) - \epsilon, \\ 0 & \text{if } c_k > P_i(k), \\ \frac{P_i(k) - c_k}{\epsilon} & \text{otherwise.} \end{cases} \tag{3.3}$$

Another important concept exploited by the authors of this approach is *specialization*, which, in this context means, that not all agents participate in the trades for all data items. Figure 3.2 gives an example how this can leveraged in a very simple setting. A generic type of base classifier that can easily be adopted to that is a leaf of a tree from a Random Forest.

The authors show that, on a number of data sets, their approach gives better accuracy than Random Forrest or AdaBoost. However, it is important to keep in mind, that the results presented in the papers mentioned above were obtained with parameters chosen experimentally using the validation set. Therefore, it is not necessarily a fair comparison.

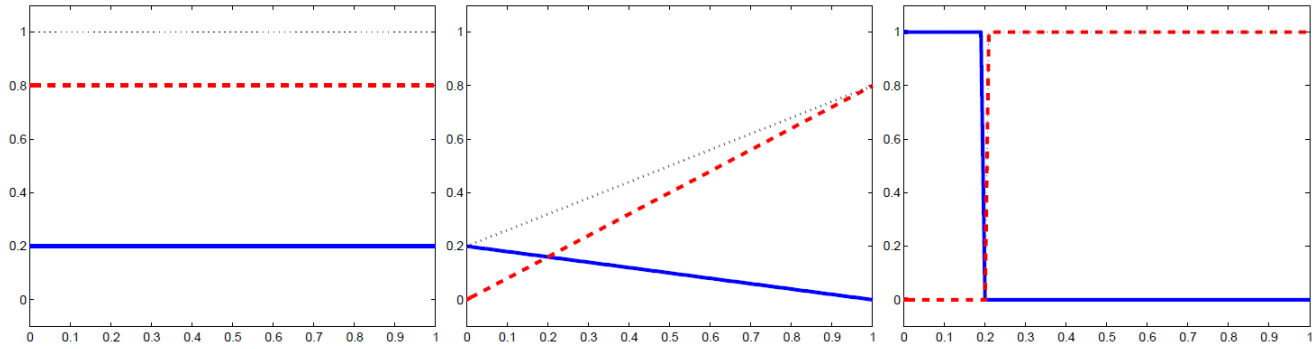


Figure 3.1: Betting functions for two-class problem. Left: constant betting, middle: linear betting, right: aggressive betting. Shown are s_{i2} (red), s_{i1} (blue), and the total amount bet $s_{i1} + s_{i2}$ (black dotted). In this example, the classifier probability is $P_i(1) = 0.2$. Figure from [LayBar10].

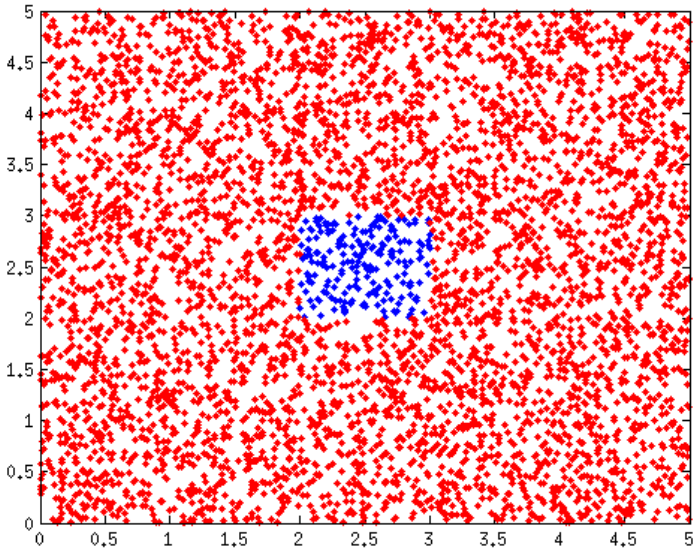


Figure 3.2: An example of a dataset for which using artificial prediction markets may lead to perfect accuracy. There are eight very simple classifiers: the first always betting that the class is *red* if $x < 2$, the second always betting that the class is *blue* if $x \geq 2$, the third always betting that the class is *red* if $x \geq 3$, the fourth always betting that the class is *blue* if $x < 3$, the fifth always betting that the class is *red* if $y < 2$, the sixth always betting that the class is *blue* if $y \geq 2$, the seventh always betting that the class is *red* if $y \geq 3$, the eighth always betting that the class is *blue* if $y < 3$. Training this market makes the classifiers betting that the class is *blue* lose their money, it is transferred to the classifiers betting that the class is *red*. That means, that when a new blue point comes, the classifiers that bet anything are all betting that the class is *blue* and when a new red point comes, the classifiers that bet that the class is *red* have big influence on the market and the ones betting that the class is *blue* have small influence on the market. Therefore, in both cases, the point is classified correctly.

Chapter 4

Edinburgh formulation

In contrary to *Florida formulation*, buying functions in this approach are not arbitrary and have their justification in underlying utility functions. That allows for relating this approach to known methods. However, by introducing isoelastic agents, the standard methods can be extended.

4.1. General setup

This formulation was first introduced in [Sto11]. There are N_G goods in the market and there are N_A agents, each endowed with some utility function U, some wealth W and some beliefs P about the domain from which the data items come from. Only one of the goods pay 1, the remaining goods pay 0. Agents are presented data items from data set D_T in certain order. After seeing every data item D_t, agents express their beliefs about the class of that data item. Given these beliefs, agents' utility functions and wealths, the market equilibrates, the prices express aggregated market belief about the class current data item belongs to. The key missing pieces of information in this description are: how the market equilibrates, how the wealth of participants is decided and from where agents' beliefs come. The latter part of this chapter is an attempt to fill those missing pieces.

An important assumption is that there are no arbitrage opportunities in the market. This means that it is not possible to make a profitable jointly risk free trade. No-arbitrage assumption implies that:

$$\sum_{k=1}^{N_G} c_k = 1. \tag{4.1}$$

This is because one class always pays 1, therefore, if $\sum_{k=1}^{N_G} c_k < 1$ then an agent could buy equal amounts of goods of all classes and always earn the difference. Similarly, if $\sum_{k=1}^{N_G} c_k > 1$, then an agent could sell equal amounts of goods of all classes and again earn the difference.

To be in the equilibrium, the market has to satisfy the following constraint:

$$\sum_{i=1}^{N_A} \mathbf{s}_i(W_i, \mathbf{c}) = \mathbf{0}. \tag{4.2}$$

This means that the total number of goods bought matches the total number of goods sold.

4.2. Utility functions

One can think of various utility functions an agent may be endowed with. The class of useful functions for the our purposes, however, can be greatly limited. From a computational point of view it is desirable that the function is continuous. Among continuous functions we are only interested in those that have a positive and strictly decreasing derivative (to be strictly increasing and grow slower for large values than for small values). The function of that form makes the agent act rationally, that is, always willing to have more wealth and willing to have more wealth less when it has a lot of wealth already. Some well known utility functions ([Sto11], [StoMilGer11]) that satisfy these constraints are:

- **Exponential**

$$U_{exp}(x) = -e^{-x}. \tag{4.3}$$

  One important property of this utility function is that $-e^{W-x} = -e^{W}e^{-x}$, that implies that a relative change of utility is independent of the current wealth of the agent. This implies that decisions made by such an agent do not depend on its wealth.

- **Logarithmic**

$$U_{log}(x) = \begin{cases} \log(x) & \text{if } x > 0, \\ -\infty & \text{otherwise.} \end{cases} \tag{4.4}$$

- **Isoelastic** (for $\eta > 0$)

$$U_{iso}(x) = \frac{x^{1-\eta} - 1}{1 - \eta}. \tag{4.5}$$

  η parameter is controlling how risk averse the agent is. If $\eta = 0$ then the agent is risk-neutral as his utility grows linearly with wealth. As η gets bigger, the agent gets more and more risk averse. When $\eta \to \infty$, the agent is infinitely risk averse. For $0 < \eta < 1, \lim_{x\to\infty} U_{iso}(x) = \infty$ and for $\eta > 1, \lim_{x\to\infty} U_{iso}(x) = \frac{1}{\eta-1}$.

  Strictly speaking[1], $\lim_{\eta\to 1} U_{iso}(x) = U_{log}(x)$. However, for computational reasons (equilibrium prices for market with isoelastic agents can't be computed analytically[2]), these two functions will be treated distinctively.

All these utility functions are shown in Figure 4.1.

4.2.1. Where the utility functions come from

One, however, may ask where all these functions come from. They may be derived fairly straightforwardly using the following chain of reasoning ([Pet05]). Given an agent with some utility function U, and continuous random variable X, representing a lottery that pays off with probability density function f, the problem is to find a lottery ticket price such that expected utility of the payoff is equal to the utility of initial wealth of the agent. That will be the maximum price an agent is willing to pay to participate in this lottery. This is exactly what Equation (4.6) says:

$$U(W) = \int_{-\infty}^{\infty} U(W - p + x) f(x) dx \tag{4.6}$$

[1] See section D.3 for derivation.

[2] see section 4.4 and C.3 for details

An approximate simplification of this equation is[3]:

$$p = \int_{-\infty}^{\infty} x f(x) dx + \frac{U''(W)}{U'(W)} \int_{-\infty}^{\infty} \frac{(x-p)^2}{2} f(x) dx \tag{4.7}$$

This expression shows that the maximum amount that an agent is willing to pay for a lottery ticket is equal to the mean of X minus a term consisting of two parts. One part is the integral (always of positive value) related to the variability of the outcome. The other part depends only on the agent's utility function. A few categories of utility functions can be defined depending on what function the ratio $\frac{U''(W)}{U'(W)}$ is.

Hyperbolic absolute risk aversion (HARA) is the most general class of utility functions used in practice. A utility function exhibits HARA if its absolute risk aversion A is a hyperbolic function, namely:

$$A(c) = -\frac{U''(c)}{U'(c)} = \frac{1}{ac+b} \tag{4.8}$$

Special cases of this occur when:

- $a = 0$, then $A(c) = -\frac{U''(c)}{U'(c)} = \frac{1}{b} = const$ and utility is said to exhibit constant absolute risk aversion. Then the solution of this differential equation is, up to multiplicative and additive constant, the exponential utility function.
- $b = 0$, then $A(c) = -\frac{cU''(c)}{U'(c)} = \frac{1}{a} = const$ and utility is said to exhibit constant relative risk aversion. Then the solution of this differential equation is, up to multiplicative and additive constant, the isoelastic utility function.

Therefore, since isoelastic utility generalizes logarithmic utility, I have shown that functions exhibiting hyperbolic absolute risk aversion generalize all utility functions used in this thesis.

4.3. Buying functions

The next step is to find out what amounts of different stock agents want to buy. As the outcome is uncertain, a rational agent wants to maximise its expected utility

$$\mathbb{E}[U] = \sum_k P_i(k) U(W_i - \mathbf{s}_i^{\mathrm{T}} \mathbf{c} + s_{ik}), \tag{4.9}$$

which leads to equation (4.10).

$$\mathbf{s}_i^* = \arg\max_{\mathbf{s}_i} \sum_k P_i(k) U(W_i - \mathbf{s}_i^{\mathrm{T}} \mathbf{c} + s_{ik}). \tag{4.10}$$

However, there is more than just one set of parameters maximising expected utility, hence there is no unique function satisfying this optimisation problem as well. In fact, there is a continuum of functions that maximise expected utility. The reason for this is that every agent can make financially-neutral risk-free bets just by buying equal amounts of all goods, utilities of $\mathbf{s} + \alpha\mathbf{1}$ are equal. The important thing is what the relative amounts of goods bought and sold are, therefore, to deal with this issue, a neutral agent, willing to make only financially-neutral risk-free trades, is introduced. Then it is necessary to specify agent's position only

[3]See section D.4 for derivation.

up to an equivalence class. To represent each class by one position only, it is necessary to use some standardization constraint. Some sensible constraints to impose on each agent are $\mathbf{s}_i^T \mathbf{c} = 0$ (agent purchases and sells equal amount of goods) and $s_{iN_G} = 0$ (agent does not purchase or sell the last good). It is important to understand that those constraints do not change agents' behaviour in the market, they are introduced only for convenience of representing classes of equivalent positions. It is also correct to switch between constraints in the calculations.

$\mathbf{s}_i(W_i, \mathbf{c})$ is a smooth function of $\mathbf{s}_i$ if the utility function is a smooth function. Therefore, its maximum with respect to $\mathbf{s}_i$ can be found by simple differentiation. Buying functions corresponding[4] to the utility functions introduced earlier are:

- **Exponential**

$$s_{ik}^* = \log P_i(k) - \log c_k \tag{4.11}$$

- **Logarithmic**

$$s_{ik}^* = \frac{W_i \left(P_i(k) - c_k\right)}{c_k} \tag{4.12}$$

- **Isoelastic** (for $\eta > 0$)

$$s_{ik}^* = W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{\sum_j c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} - 1 \right] \tag{4.13}$$

  Again, when $\eta \to 1$, this becomes equivalent to the buying function derived for logarithmic utility.

All these buying functions are shown in Figure 4.2. The unexpected change of shape of the function in the Figure 4.2(c) and in the Figure 4.3 as η gets closer to 0 is a consequence of the fact that, in a binary setting, when $\eta \to 0$, then[5]

$$s_{ik}^* = \begin{cases} \frac{W_i}{c_k} & \text{if } P_i(k) > c_k, \\ 0 & \text{if } P_i(k) = c_k, \\ -\frac{W_i}{1-c_k} & \text{if } P_i(k) < c_k. \end{cases} \tag{4.14}$$

Also, when $\eta \to \infty$, then

$$s_{ik}^* = 0 \tag{4.15}$$

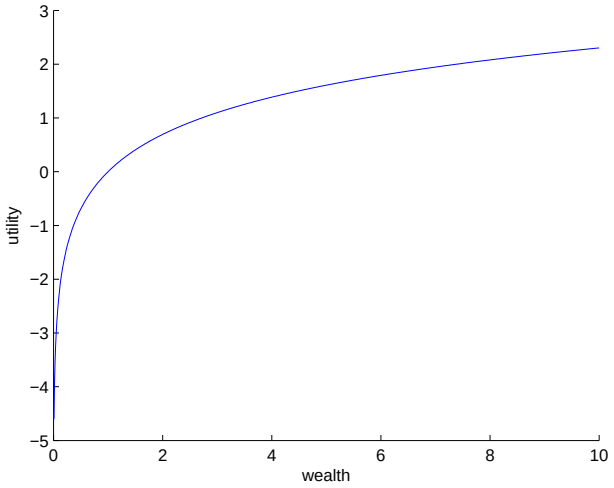
4.4. Finding an equilibrium

The market has to satisfy the market constraint (4.2). All agents simultaneously try to maximise their utility functions. According to [ArrDeb54], if all individual agents have concave utility functions, there is a unique market equilibrium. To find this equilibrium in this formalism, it is necessary to solve equation (4.2). When the market is homogeneous, attempting to solve market equation analytically gives the following[6]:

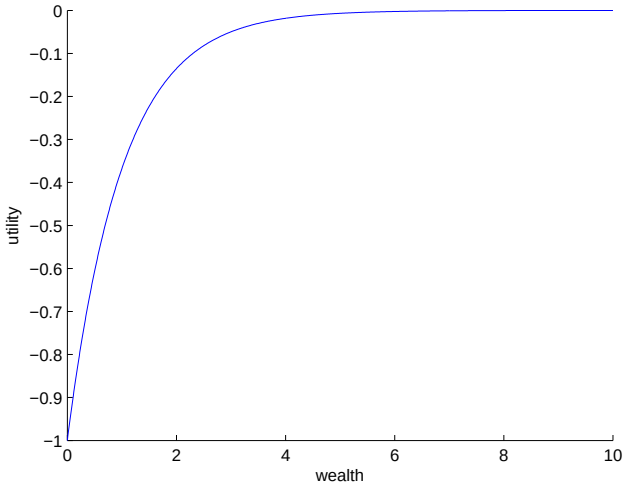
[4] Please look at appendix A for derivations of buying functions from utility functions.

[5] See section D.5 for details.

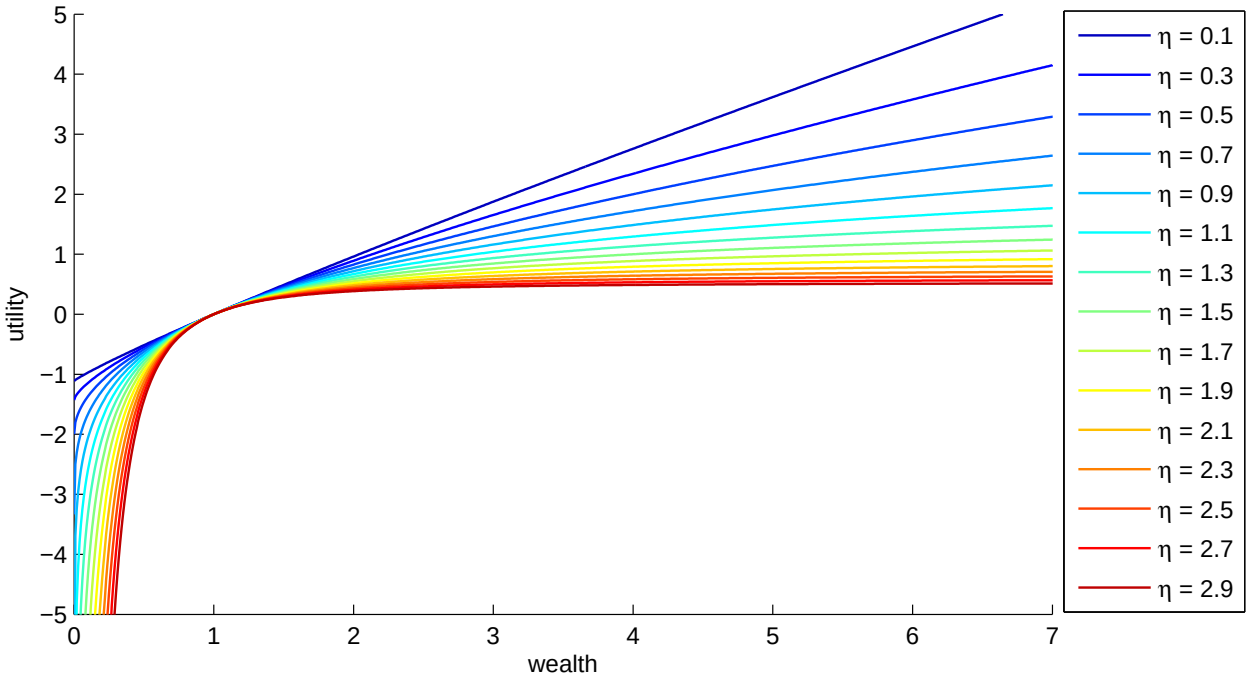
[6] See appendix C for details.



(a) Logarithmic

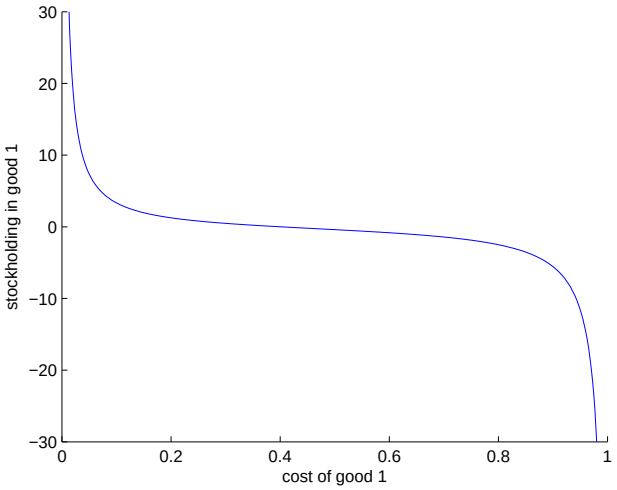


(b) Exponential

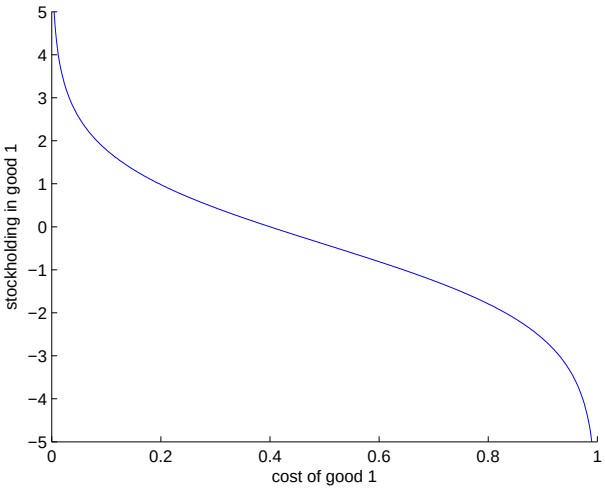


(c) Isoelastic

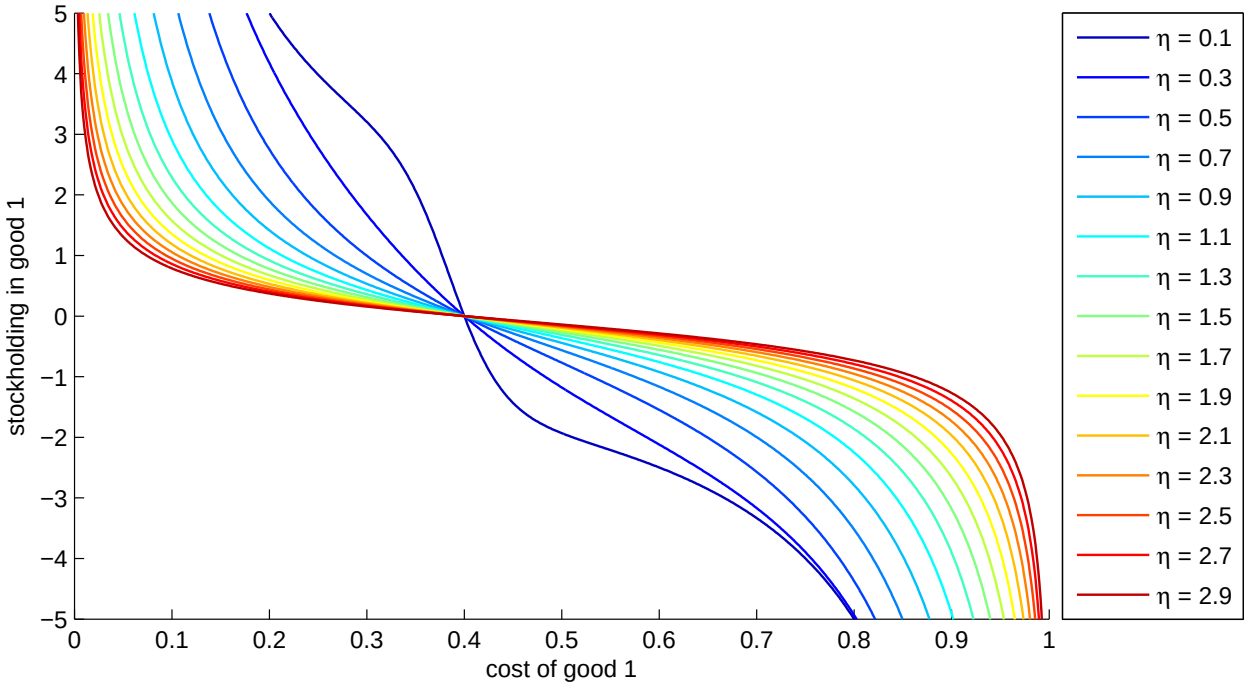
Figure 4.1: Utility functions in a binary setting.



(a) Logarithmic



(b) Exponential



(c) Isoelastic

Figure 4.2: Buying functions in a binary setting, for an agent that has a belief $P_i(1) = 0.4$, after applying a constraint $s_{i2} = 0$.

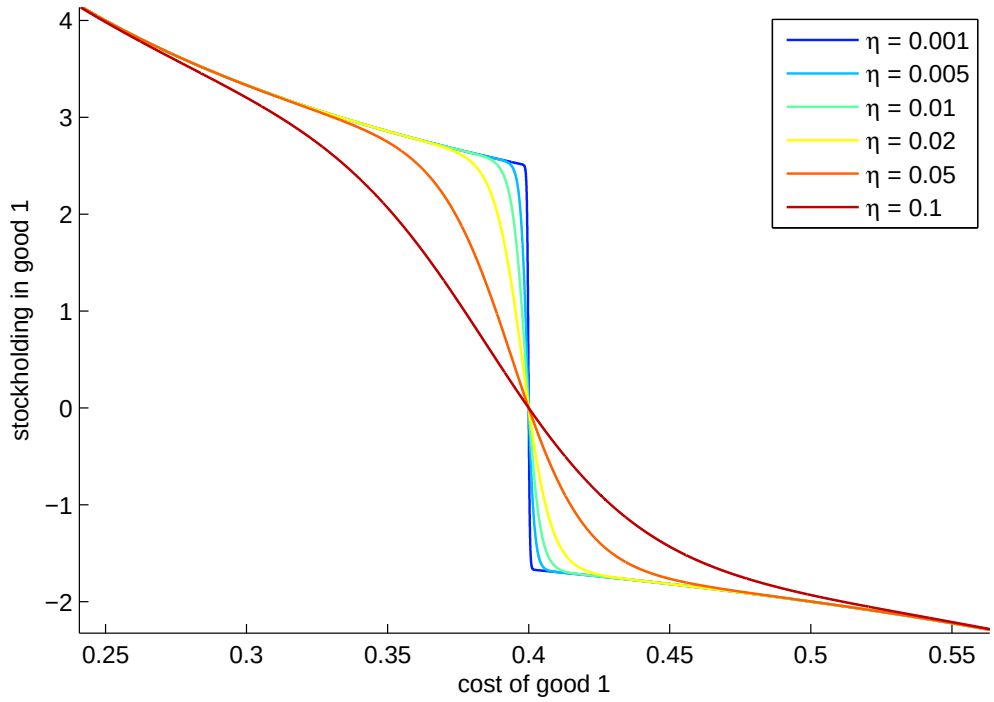


Figure 4.3: Buying function for isoelastic utility when $\eta \to 0$.

- **Logarithmic**

$$c_k = \frac{\sum_i W_i P_i(k)}{\sum_i W_i} \tag{4.16}$$

  Interestingly, such a market aggregates participants' beliefs the same way as a model known as the mixture of experts.

- **Exponential**

$$c_k \propto \prod_{i=1}^{N_A} P_i(k)^{1/N_A} \tag{4.17}$$

  This market aggregates participants' beliefs in such a way as a model known as product of experts ([Hin02]) does.

- **Isoelastic** (for $\eta > 0$)

$$c_k \propto \sum_{i=1}^{N_A} W_i \left[\frac{c_k \left(\frac{P_i(k)}{c_k} \right)^{1/\eta}}{\sum_{j=1}^{N_G} c_j \left(\frac{P_i(j)}{c_j} \right)^{1/\eta}} \right] \tag{4.18}$$

  This is not a closed form solution, hence, it has to be solved numerically, except for the case when $\eta \to 1$, then the equilibrium price vector is the same as for the homogeneous logarithmic market.

There is no general analytical solution known to the author when the market is inhomogeneous. However, the solution, can also be found numerically by minimizing the following

function (which will be referred to as *market equilibrium function*):

$$E(\mathbf{c}) = \sum_k \left(\sum_i s_{ik}(W_i, \mathbf{c}) \right)^2 \tag{4.19}$$

If $E > 0$ there is a mismatch between those who want to sell and those who want to buy goods at a given price. When $E = 0$, constraint (4.2) is satisfied. Minimizing function (4.19) can, in practice, be done using some variant of Newton's method[7]. One good initial value of $\mathbf{c}$ that approximates the solution is just a vector of average belief about classes, that is $\left[\frac{1}{N_A}\sum_{i=1}^{N_A} P_i(1), \ldots, \frac{1}{N_A}\sum_{i=1}^{N_A} P_i(N_G)\right]$.

4.5. Minimizing market equilibrium function

When plugging function (4.19) into an unconstrained optimisation procedure, the solution found will not necessarily be a probability distribution (it will not necessarily be a vector of positive numbers summing up to 1). In order to mitigate against these issues without formulating this as constrained optimisation, equation (4.19) can be rewritten as:

$$E(\mathbf{z}) = \sum_{k=1}^{N_G} \left(\sum_{i=1}^{N_A} s_{ik}\left(W_i, c(\mathbf{z})\right) \right)^2 \tag{4.20}$$

where $\mathbf{c}$ is the softmax function of $\mathbf{z}$:

$$c_k(\mathbf{z}) = \frac{e^{z_k}}{\sum_{j=1}^{N_G} e^{z_j}}$$

Then derivative of E with respect to $\mathbf{z}$ can be found using the chain rule:

$$\frac{\partial E}{\partial \mathbf{z}} = \frac{\partial E}{\partial \mathbf{c}} . \frac{\partial \mathbf{c}}{\partial \mathbf{z}}$$

Starting with

$$\frac{\partial \mathbf{c}}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial c_1}{\partial z_1} & \cdots & \frac{\partial c_1}{\partial z_j} & \cdots & \frac{\partial c_1}{\partial z_{N_G}} \\ \vdots & \ddots & & & \vdots \\ \vdots & & \frac{\partial c_i}{\partial z_j} & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial c_{N_G}}{\partial z_1} & \cdots & \frac{\partial c_{N_G}}{\partial z_j} & \cdots & \frac{\partial c_{N_G}}{\partial z_{N_G}} \end{bmatrix}$$

where[8] [9]

$$\frac{\partial c_i}{\partial z_j} = c_i \left(I_{ij} - c_j\right)$$

[7] Experimental work in thesis was done using subspace trust-region method based on the interior-reflective Newton method implemented by `fminunc` function from MATLAB Optimization Toolbox.

[8] See section D.1 for the derivation.

[9]

$$I_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

and

$$\frac{\partial E}{\partial \mathbf{c}} = \left[\frac{\partial E}{\partial c_1} \cdots \frac{\partial E}{\partial c_k} \cdots \frac{\partial E}{\partial c_{N_G}}\right]$$

where

$$\frac{\partial E}{\partial c_q} = \sum_{k=1}^{N_G} 2 \left(\sum_{i=1}^{N_A} s_{ik}(W_i, \mathbf{c})\right) \left(\sum_{i=1}^{N_A} \frac{\partial s_{ik}(W_i, \mathbf{c})}{\partial c_q}\right)$$

the derivative is equal to:

$$\frac{\partial E}{\partial \mathbf{z}} = \left[\frac{\partial E}{\partial c_1} \cdots \frac{\partial E}{\partial c_k} \cdots \frac{\partial E}{\partial c_{N_G}}\right] \begin{bmatrix} \frac{\partial c_1}{\partial z_1} & \cdots & \frac{\partial c_1}{\partial z_j} & \cdots & \frac{\partial c_1}{\partial z_{N_G}} \\ \vdots & \ddots & & & \vdots \\ \vdots & & \frac{\partial c_i}{\partial z_j} & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial c_{N_G}}{\partial z_1} & \cdots & \frac{\partial c_{N_G}}{\partial z_j} & \cdots & \frac{\partial c_{N_G}}{\partial z_{N_G}} \end{bmatrix} \tag{4.21}$$

Thus, it is necessary to differentiate the buying functions with respect to the cost vector in order to use gradient based optimisation methods. Derivatives of the functions introduced earlier can be derived analytically[10]. If they could not, numerical derivatives can be used instead, however, they slow the computations significantly.

4.6. Training the market

4.6.1. Wealth update

Agent i has wealth W_i. Wealth affects agent's behaviour and, therefore, affects its influence on the market (except for exponential agents, whose buying function (4.11) does not depend on wealth). For every market training data item agent i gets $\mathbf{s}_i^T\mathbf{c}$ for goods that he bought and sold and gets return of s_{ik^*} for the good that represented the correct class, hence, in total $\mathbf{s}_i^T\mathbf{c} + s_{ik^*}$. However, because of the constraint $\mathbf{s}_i^T\mathbf{c} = 0$, $\mathbf{s}_i^T\mathbf{c} + s_{ik^*} = s_{ik^*}$.

A number of ways of updating the budgets could be considered. The two most obvious are: *online update* and *batch update*. Online update means that budget of every participant of the market changes after every training sample. In that context, W_i^t denotes wealth of agent i after the market saw t data items. Batch update means that agent's wealth is divided into equal pieces, one piece for each training point. In that context W_i' denotes the wealth of agent i after seeing all T data items.

One interesting property of the market that has exclusively isoelastic agents is that the wealths can all be multiplied by the same constant and the prices are going to be exactly the same as before.

4.6.2. Update for logarithmic agents

The wealths are initialized in such a way that $\sum_{i=1}^{N_A} W_i = 1$.

[10] See appendix B for derivation of $\frac{\partial s_{ik}}{\partial c_q}$.

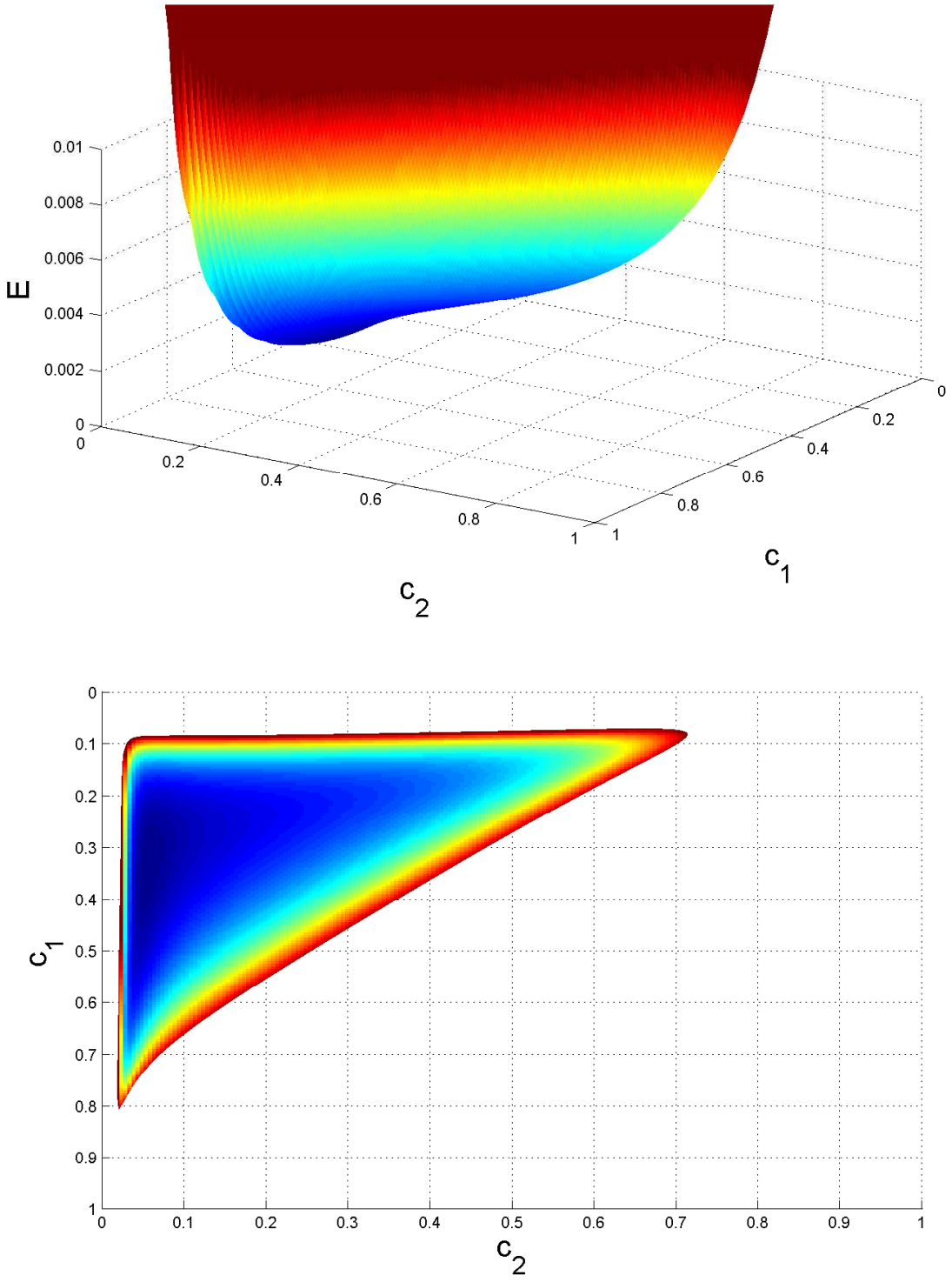


Figure 4.4: Typical shape of function E when there are three classes. Effectively, this is a function of just two variables, c_1 and c_2 as $c_3 = 1 - (c_1 + c_2)$. The curve is convex, as it should be.

Online update

$$W_i^{t+1} = W_i^t + s_{ik_t} = W_i^t + \frac{W_i^t(P_i(k_t) - c_{k_t})}{c_{k_t}} = W_i^t \frac{P_i(k_t)}{c_{k_t}} \tag{4.22}$$

Assuming that the distribution of vectors of features $\mathbf{x}$ is independent of i, the following chain of equalities holds $\frac{P(D_t|i)}{P(D_t)} = \frac{P(\mathbf{x},k|i)}{P(\mathbf{x},k)} = \frac{P(k|\mathbf{x},i)P(\mathbf{x}|i)}{P(k|\mathbf{x})P(\mathbf{x})} = \frac{P(k|\mathbf{x},i)}{P(k|\mathbf{x})} = \frac{P_i(k_t)}{c_{k_t}}$. Therefore, by equating W_i^t to $P(i|D_T)$, budget update takes the form of

$$W_i^{t+1} = P(i|D_{T+1}) = \frac{P(D_{t+1}|i)P(i|D_T)}{P(D_{t+1})} \tag{4.23}$$

This is Bayesian update rule, initial wealths can be interpreted as a prior distribution. Then using the assumption that $\sum_{i=1}^{N_A} W_i^t = 1$, it follows that equation (4.16) can be interpreted in the following way:

$$c_k = \frac{\sum_{i=1}^{N_A} W_i^t P_i(k)}{\sum_{i=1}^{N_A} W_i^t} = \sum_{i=1}^{N_A} P(i|D_T)P_i(k) \tag{4.24}$$

This is Bayesian averaging procedure mentioned in section 1.5. This is a good method if we believe that there are many competing hypotheses and only one is correct. However, this assumption may not always, and, in fact, in most cases is inappropriate ([Min02]). However, it is still an interesting example of how prediction market mechanism is equivalent to standard machine learning algorithm.

One way of preventing a scenario where all the wealth available on the market goes to the best player would be to introduce a *salary* (a fixed income at the beginning of each turn or training epoch) and a *tax* on wealth. This way no agents would lose all their money and be completely ignored and the agents who have great wealth would not dominate the market totally. That idea is in the spirit of a technique known as *regularisation*.

Batch update

Similarly, an interpretation can be given to batch update.

$$W_i' = \frac{1}{T}\sum_{t=1}^{T}(W_i + s_{ik_t}) = W_i^t + \frac{1}{T}\sum_{t=1}^{T}\frac{W_i^t(P_i(k_t) - c_{k_t})}{c_{k_t}} = \frac{W_i}{T}\sum_{t=1}^{T}\frac{P_i(k_t)}{c_{k_t}} \tag{4.25}$$

$$W_i' = P(i|D_T) = \frac{P(i)}{T}\sum_{t=1}^{T}\frac{P(D_t|i)}{P(D_t)} = \frac{1}{T}\sum_{t=1}^{T}P(i|D_t) \tag{4.26}$$

This is known as maximum posterior mixture model.

Chapter 5

Experimental results

5.1. Experimental setup

5.1.1. Data sets

A number of data sets of different characteristics were used in the process of experimental validation of the algorithms introduced in this thesis.

Data set	**#Data points**	**#Features**	**#Classes**	**Type of attributes**
Waveform	5000	21	3	real
Image	2310	19	7	real
Vehicle	946	18	4	real
Sonar	208	60	2	real
Breast cancer	699	10	2	integer
Ionosphere	351	34	2	integer, real
Magic	19020	11	2	real
Yeast	1484	8	10	real

5.1.2. Splitting the data

Every data set was split into three parts (figure 5.2):

- data to train base classifiers and the market,
- data to tune the parameters,
- data to test the market.

5.1.3. Classifiers

Agents need beliefs to participate in the market. Those beliefs can be just arbitrarily set prior, however, it makes more sense to derive them from data. Therefore, every agent used in the experiments had some classifier that was supplying it with probabilistic beliefs about classes of incoming data items. The base classifiers used were:

- **Decision trees.** Implemented by `classregtree` class from MATLAB Statistics toolbox. Unless specified otherwise, the trees were pruned in such a way that the leaves had at least 10 data items.

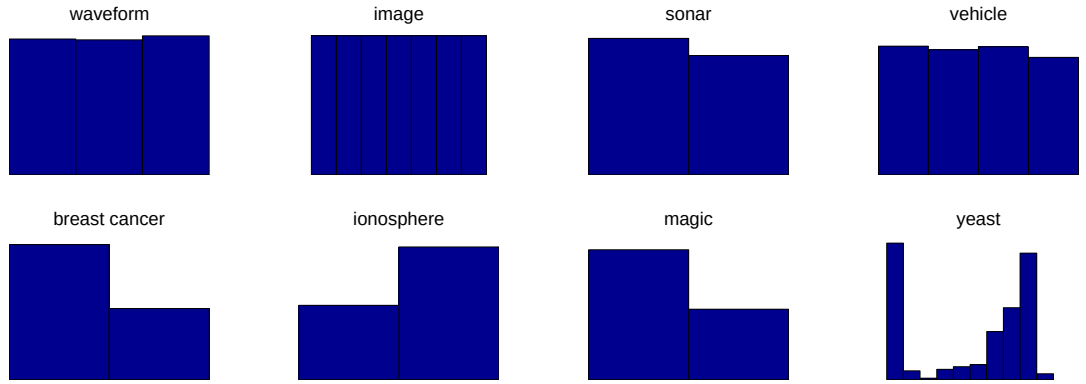


Figure 5.1: Class distributions in the data sets. The data sets were chosen in such a way that there is a variety in the number of features and classes. Waveform is an artificially generated data set and the others contain real data.

Figure 5.2: Data split. Base classifiers and the market was trained on the same data set.

- **Neural networks.** Implemented by `mlp` class in Netlab ([Netlab]) library. Unless specified otherwise, networks had 10 hidden units and were trained in 300 iterations.
- **Logistic regression.** Implemented by `mnrfit` and `mnrval` functions from MATLAB Statistics toolbox.
- **Naive Bayes.** Implemented by `NaiveBayes` class from MATLAB Statistics toolbox.
- **Support Vector Machines.** Implemented by `svmtrain` and `svnpredict` functions from LIBSVM ([LIBSVM]) library.

To avoid beliefs equal to zero, which may result in prices equal to zero, which may cause division by zero, beliefs have been smoothed in the following way, for all N_G classes, $P_i(k) := \frac{0.1+P_i(k)T}{0.1N_G+T}$, where T denotes the size of the data set.

5.1.4. Sampling methods[1]

To generate multiple data sets of size n out of one data set of size N, one of the following sampling methods was applied.

- **No sampling**. Just permuting the whole data set.
- **Bootstrapping**. Sampling the original data set n times with replacement.

[1]Sampling in this context means picking data items from the original data set to create a new data set and should not be confused with sampling as in the context of MCMC methods.

- **Sample selection bias**. This procedure is described by algorithm 4[2].

Algorithm 4 Sample selection bias

Input: $D = ((x_1, y_1), \ldots, (x_m, y_m))$,
B - number of biased sets to generate,
S - number of training samples in every generated data set,
K - number of classes.
Generate B probability distributions, such that $\forall_{k \in \{1,\ldots,K\}} \sum_{b=1}^{B} P_b(k) = \frac{B}{K}$
for $b = 1 \ldots B$ **do**
  $D_b \leftarrow \emptyset$
  while $|D_b| < S$ **do**
    $(x, y) \leftarrow$ random data point from D
    if $rand(0, 1) \geq P_b(y)$ **then**
      $D_b \leftarrow D_b \cup (x, y)$
    end if
  end while
end for
Output: $(D_1, \ldots, D_B)$

- **Subsampling**. Sampling the original data set n times without replacement.

5.1.5. Other parameters

Other parameters were:

- number of agents in the market,
- number of iterations of the whole procedure - to have more reliable estimates,
- maximum number of data items - to truncate very large data sets,
- percentages of the data assigned to parts as defined in section 5.1.2.

5.2. Results

Unless specified otherwise, the parameters were set to the following values:

- 10 agents were participating in the market,
- sample selection bias was a sampling method with sample size of 75% of the original sample,
- experiments were repeated 25 times,
- 60% of the data was used for training, 20% for validation, remaining 20% was left as test data,
- data sets larger than 1000 data items were truncated to 1000 data items,
- there was one training epoch with batch update.

[2]See section D.2 to see how to generate such probability distributions as mentioned in the algorithm.

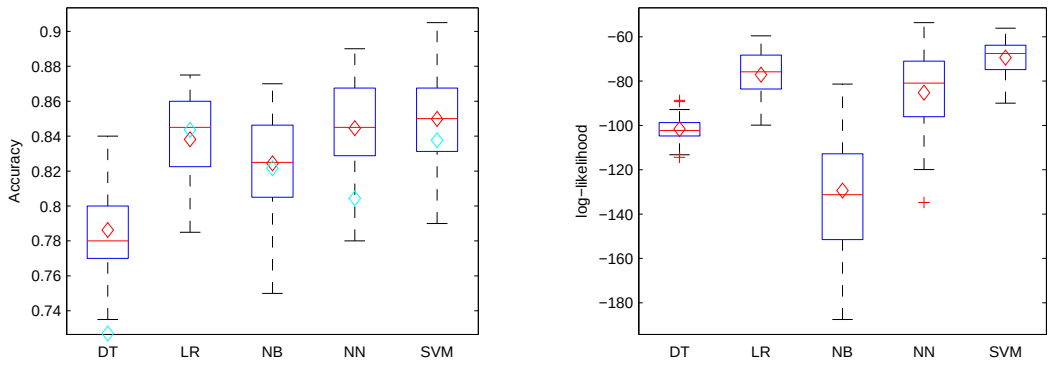
A number of experiments was executed using this setup:

1. Comparison of individual classifiers with markets consisting of agents endowed with these base classifiers (Figure 5.3).
2. Comparison of sampling methods (Figure 5.4).
3. Comparison of markets with different number of agents (Figure 5.5).
4. Comparison of markets trained using diffent number of data items (Figure 5.6). For this experiment only the number of training items was a variable and the number of validation data items was set to 200.
5. Comparison of homogeneous markets of isoelastic agents with different η (Figure 5.7).
6. Comparison of market training techniques (Figure 5.8). The number of repetitions was set to 50 for this experiment.
7. Comparison of learning rates for online training (Figure 5.9).

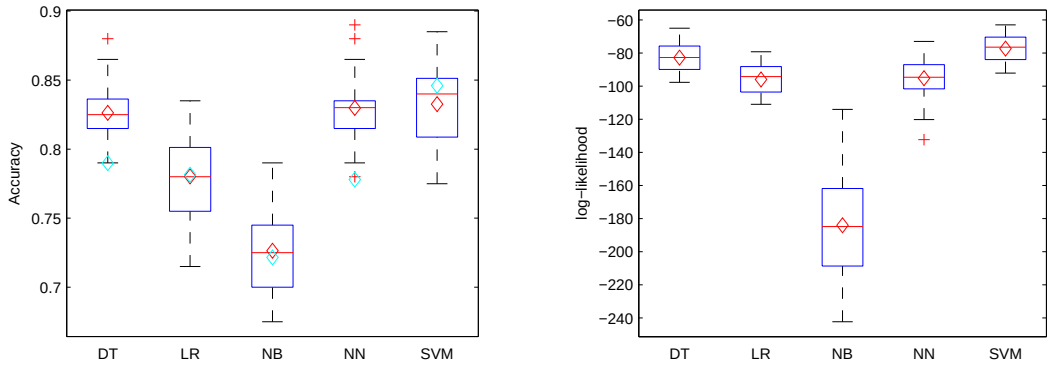
Some additional figures that did not fit in the main part of the thesis, extending those mentioned above, are contained in appendix E.

The seed for random numbers generator was set in such a way that data was split the same way for every set of parameters in a given iteration.

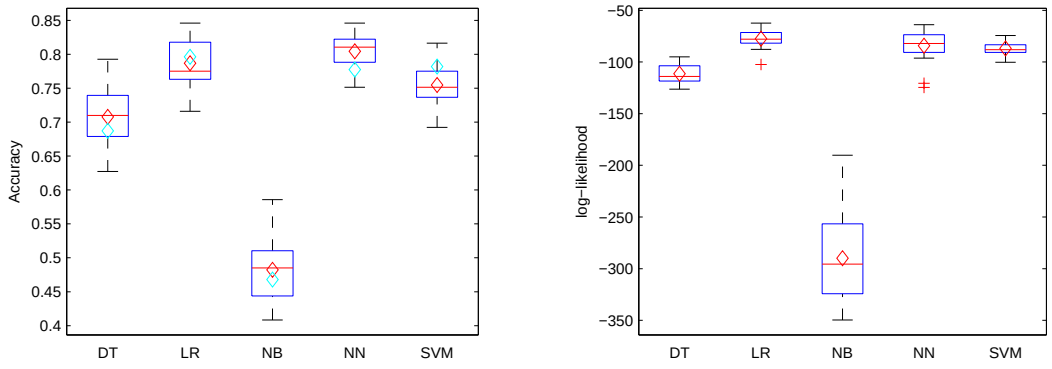
Two ways of visualising the data was used. Box plot and plotting just the mean. In the box plot, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.



(a) waveform

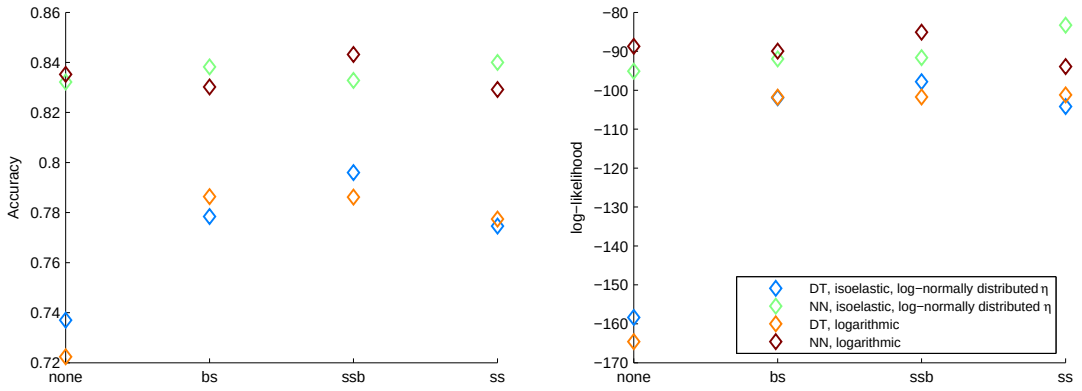


(b) magic

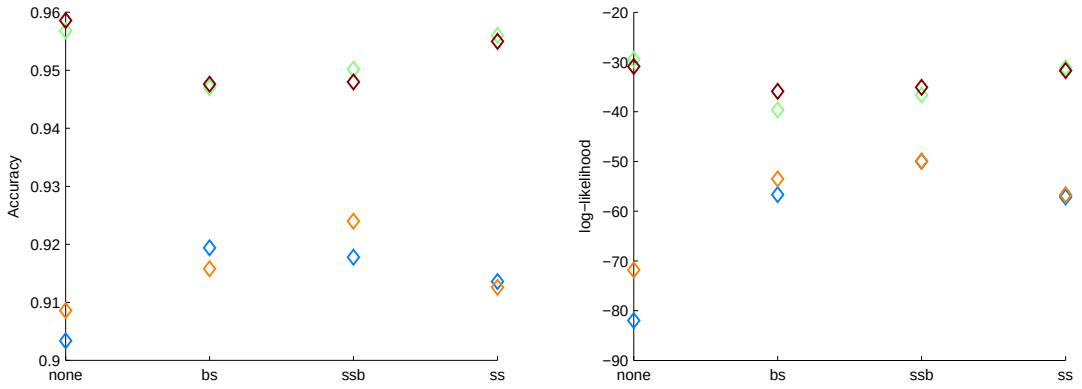


(c) vehicle

Figure 5.3: Accuracy and log-likelihood for markets consisting of agents endowed with different classifiers. The red diamonds plotted over box plots are the means of results of the markets and cyan diamonds are means of the results of standalone classifiers trained on whole data sets. Individual decision trees and neural networks vary much more for data sampled using sample selection bias, which results in greater improvement in accuracy.

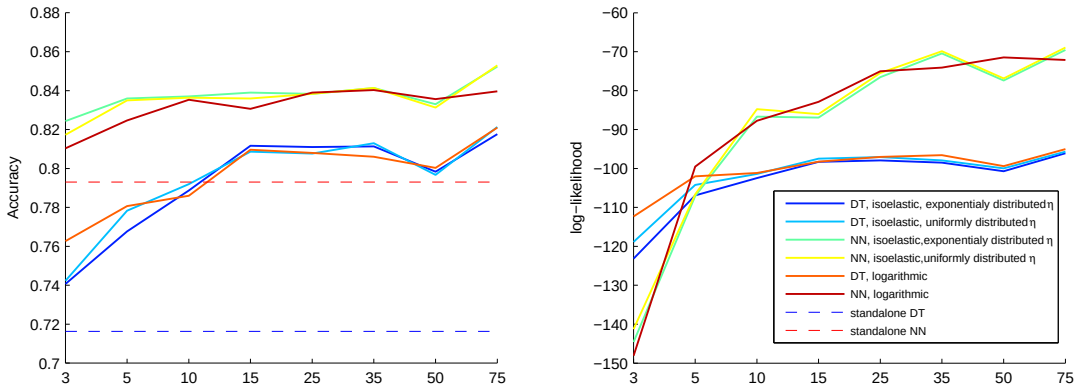


(a) waveform

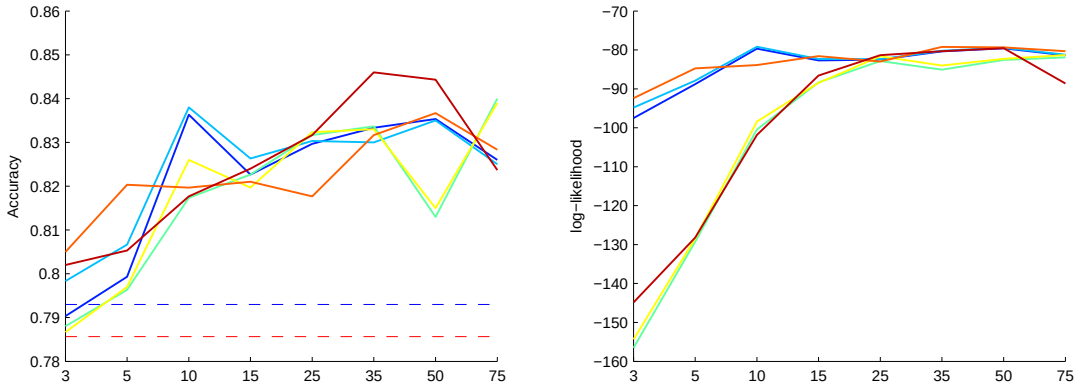


(b) image

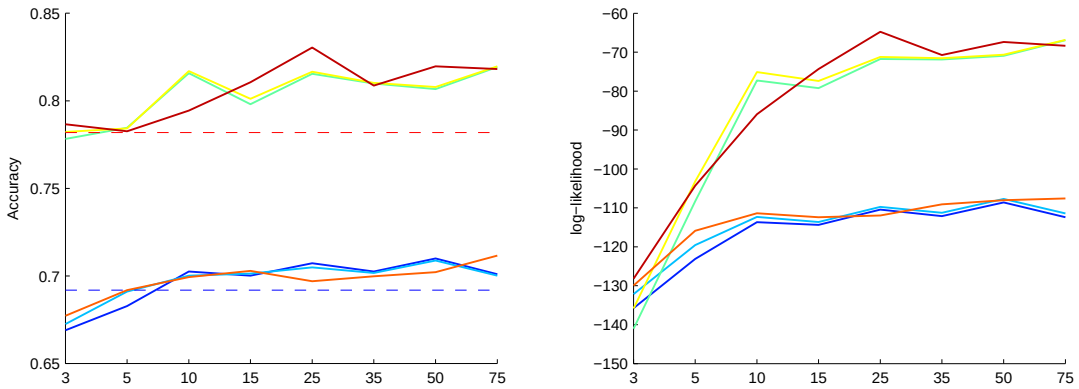
Figure 5.4: Accuracy and log-likelihood for different sampling methods. The diamonds are the means. Sample selection bias and subsampling outperform other methods for both datasets. Sample selection bias is of particular interest as this is a kind of situation when machine learning markets could be used.



(a) waveform

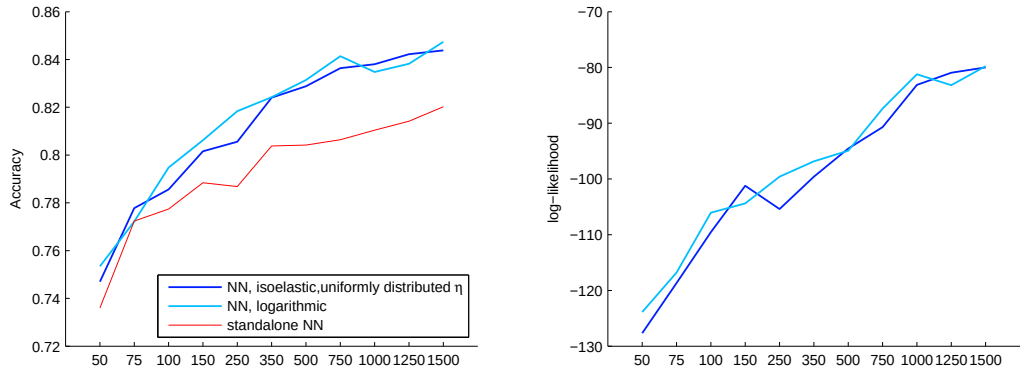


(b) magic

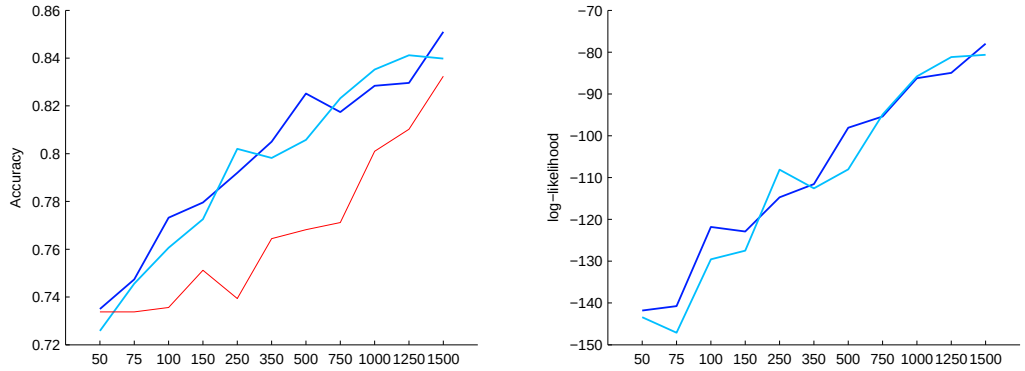


(c) vehicle

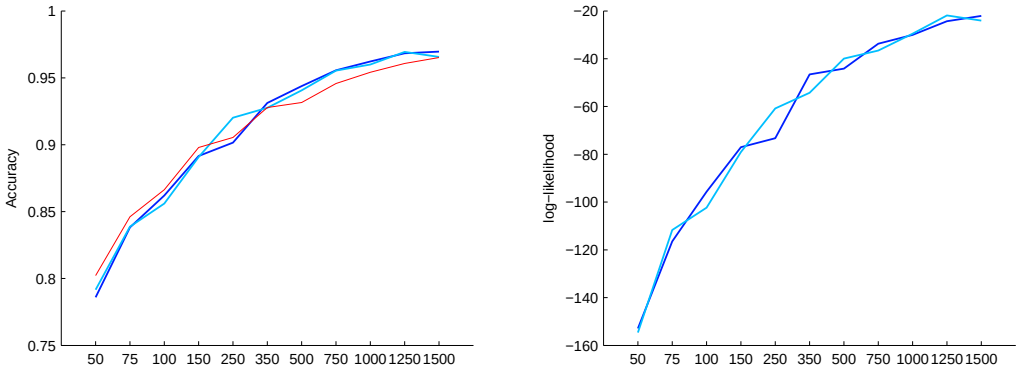
Figure 5.5: Accuracy and log-likelihood as a function of the number of agents in the market. The values plotted are the means. Even for small number of agents accuracy of the market is better than accuracy of a single classifier trained with the whole training set. However, after reaching some point, increasing the number of agents does not increase accuracy or log-likelihood. It is also clearly visible that neural networks can gain much more in log-likelihood while increasing the number of agents in the market that decision trees.



(a) waveform

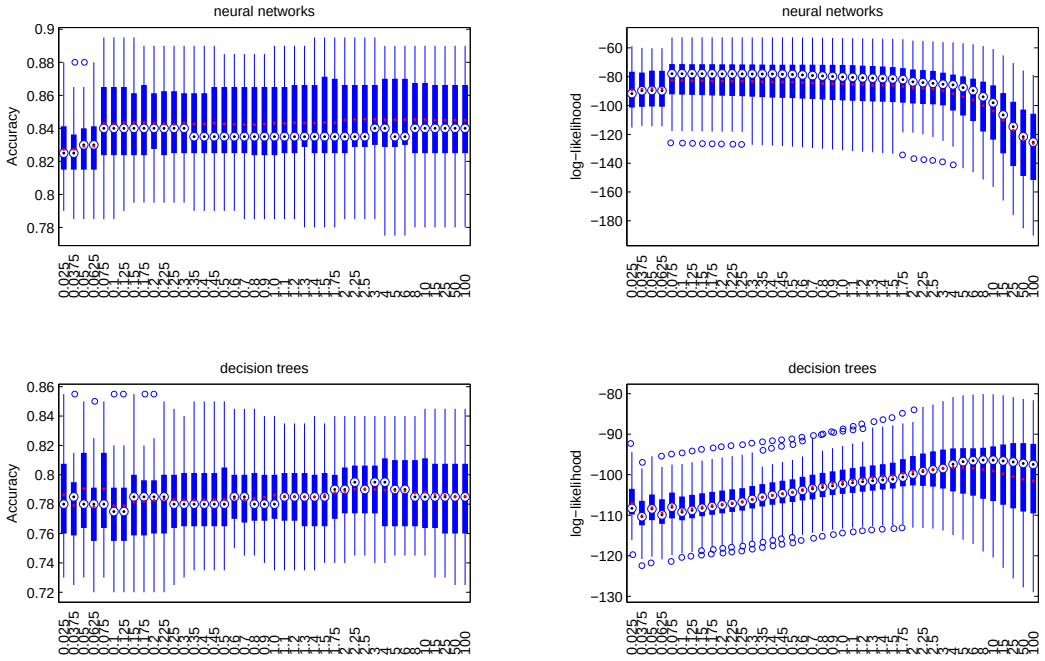


(b) magic

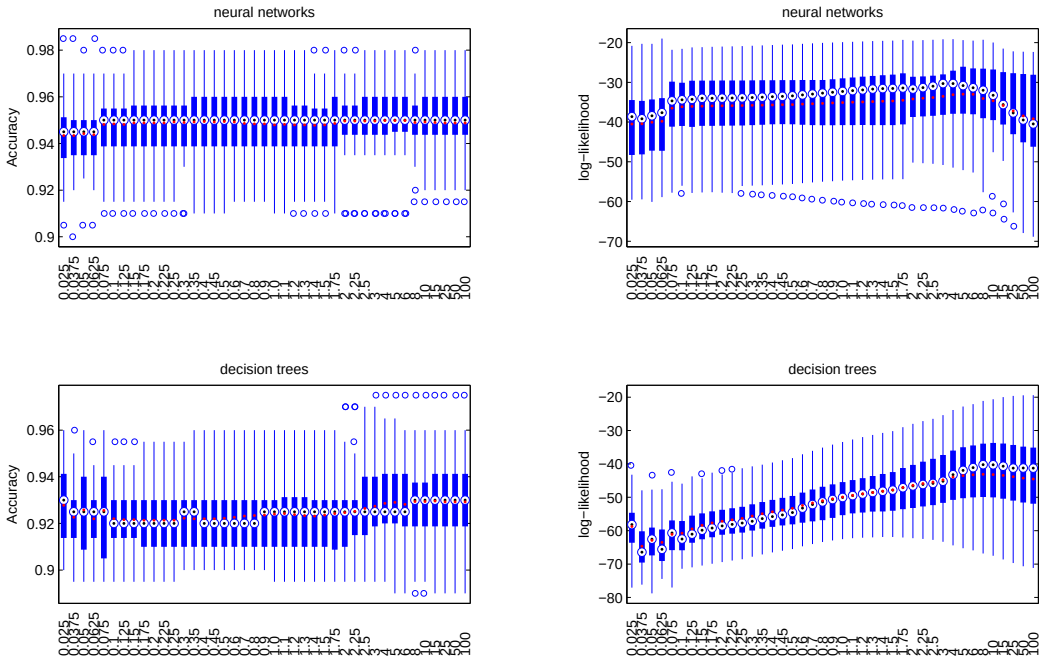


(c) image

Figure 5.6: Accuracy and log-likelihood as a function of the number of points in the training set. The values plotted are the means. Both metrics for the market improve while increasing the number of training data items. However, the difference between the accuracy of a standalone classifier trained on the whole training data set is a different function in each case. The market with agents endowed with all types of utility functions outperforms a single neural network regardless of the number of training data items.

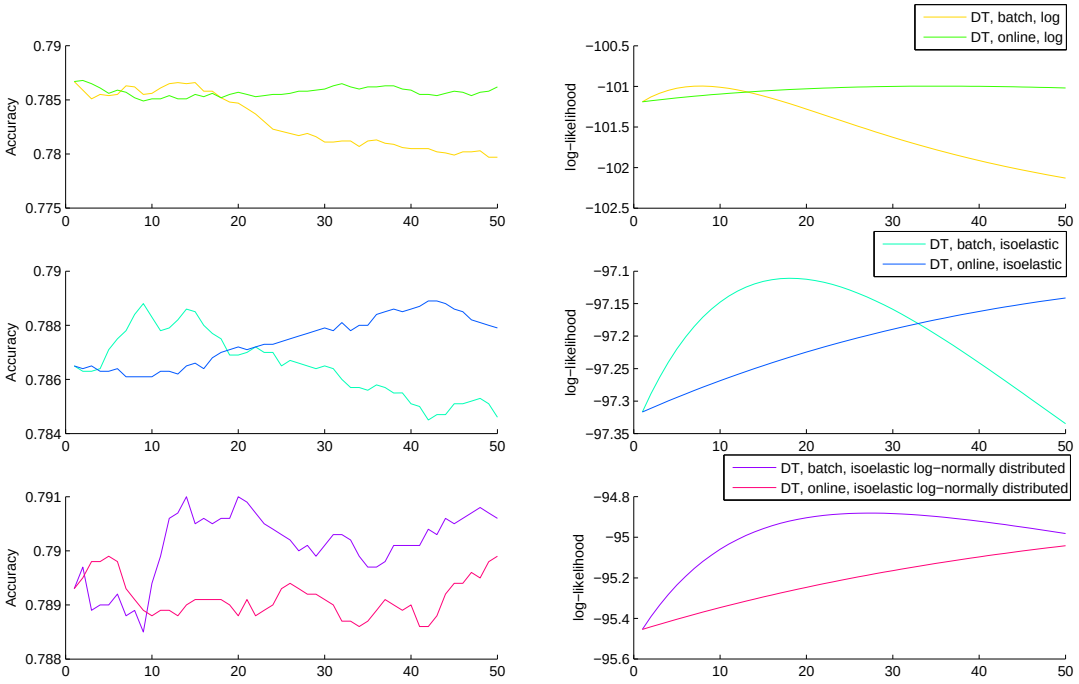


(a) waveform

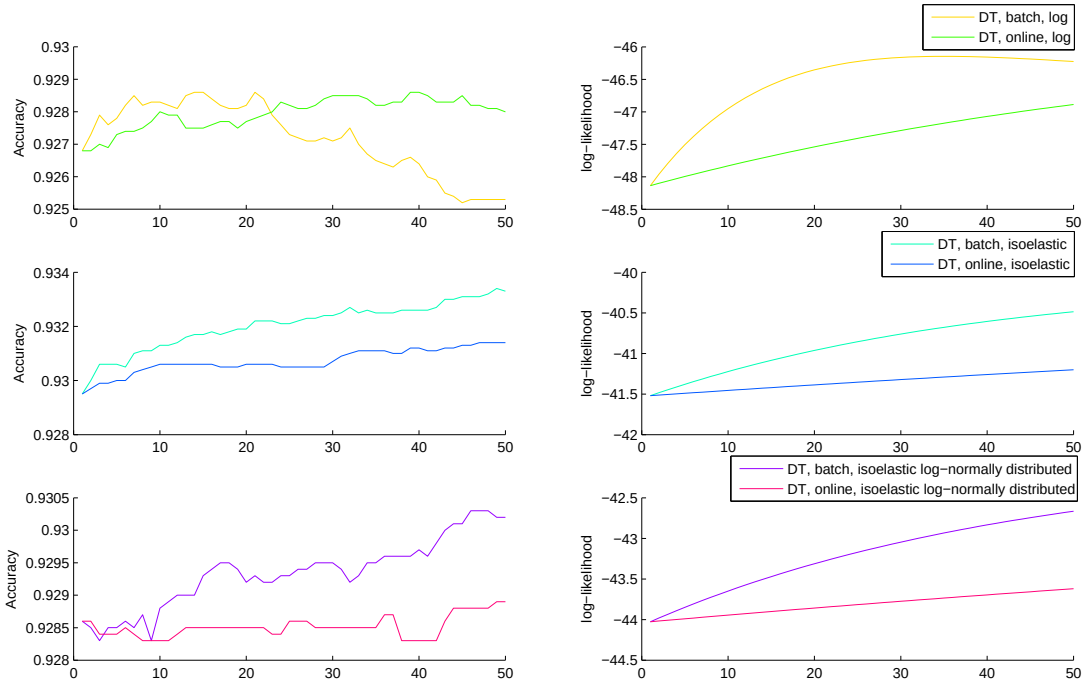


(b) image

Figure 5.7: Accuracy and log-likelihood as a function of η. The red dots plotted over the box plots are the means. Even though the change of η does not result in big changes in accuracy, a significant effect is achieved for log-likelihood. Interestingly, the change in log-likelihood does not necessarily have a direct impact on accuracy. Also, the shapes of these curves are very different for decision trees and neural networks.

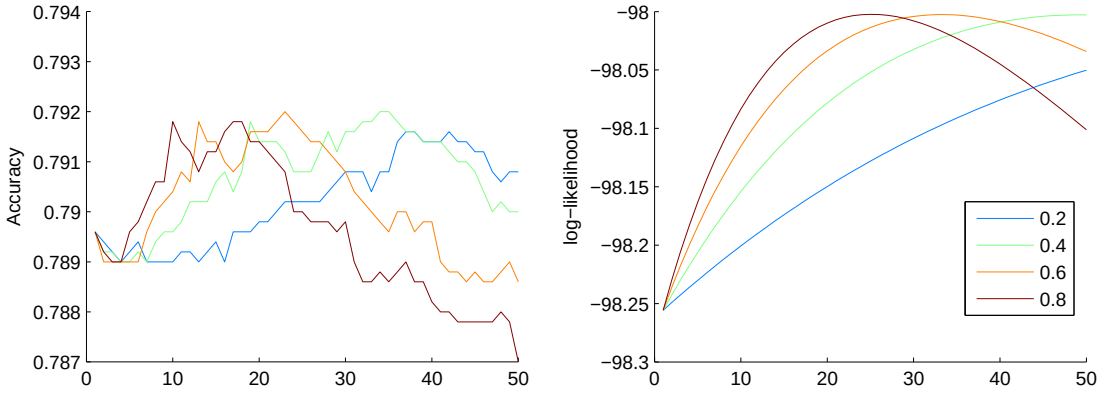


(a) waveform



(b) image

Figure 5.8: Accuracy and log-likelihood as a function of the number of training epochs for batch and online training. The values plotted are the means. The effects of training are small, however, it is worth noting that, in both cases, isoelastic agents with log-normally distributed η have gained more in log-likelihood than isoelastic agents all with the same η.



(a) waveform

Figure 5.9: Accuracy and log-likelihood as a function of the number training epochs for online training for different learning rates. The values plotted are the means. Learning rate changes the number of epochs necessary to reach to the best combination, however, the maximum value is the same for both accuracy and log-likelihood.

Chapter 6

Summary

A market of agents endowed with classifiers trained on data sampled with sample selection bias outperforms a single classifier presented with equivalent data. A market consisting of logarithmic agents is equivalent to a mixture model and a market consisting of exponential agents is equivalent to a product model. Logarithmic agents can be extended by isoelastic agents. Even though the effects of introducing isoelastic agents are not as dramatic as I initially anticipated, there are a couple of important conclusions to make:

- Homogeneous markets of isoelastic agents with η picked using validation data outperform, especially with respect to log-likelihood, homogeneous markets of logarithmic agents.
- The structure of the market, that is, utility functions and base classifiers agents are endowed with, is much more important parameter of the system influencing accuracy and log-likelihood than the market training procedure. However, when the structure is already decided, the training procedure may lead to significant improvements in both metrics.

Some future research directions in this area may be:

- Using specialized agents that participate in the market only on subspaces of the feature space like done in [LayBar10]. Such agents could be created for example by clustering the feature space and training each agent on the data from one cluster.
- Introducing new agents to the market in the process of training. New agents coming to the market could be trained only on the instances that are harder or harder instances could get a higher weight. That would be in the spirit of boosting.
- Adopting this approach to regression. This is especially promising as the markets exhibit much better behaviour with respect to log-likelihood, which is a continuous quantity, than with respect to accuracy which is discretized.
- Better, more computationally efficient, ways of looking for market equilibrium. Current method does not scale well for problems that have a large number of possible class labels. For batch update the training procedure can also be speeded up by being executed in parallel for individual data items.
- Studying effect of additional inhomogeneity by introducing agents that do not all have the same base classifier.

- Introducing *tax* on winnings and *salary* so that no agent goes bankrupt. That would result in a similar effect to a technique known as *regularisation*.
- Developing better understanding of how updates for isoelastic agents can be interpreted and how they can be related to other methods.

Appendix A

Derivations of buying functions from utility functions

In general, the function to be optimised is (4.9) and the constraint that is to be satisfied is $\mathbf{s}_i^{\mathrm{T}}\mathbf{c} = 0$. Therefore, Lagrange multiplier λ is introduced and the function to be optimised is now

$$\hat{U}_i^E(W_i, \mathbf{c}, \mathbf{s}_i) = \sum_{k=1}^{N_G} P_i(k)U(W_i - \mathbf{s}_i^{\mathrm{T}}\mathbf{c} + s_{ik}) + \lambda \mathbf{s}_i^{\mathrm{T}}\mathbf{c} \tag{A.1}$$

Using $\mathbf{s}_i^{\mathrm{T}}\mathbf{c} = 0$ constraint we get:

$$\hat{U}_i^E(W_i, \mathbf{c}, \mathbf{s}_i) = \sum_{k=1}^{N_G} P_i(k)U(W_i + s_{ik}) + \lambda \mathbf{s}_i^{\mathrm{T}}\mathbf{c} \tag{A.2}$$

Then differentiating with respect to s_{iq} (treating all other elements of the vector $\mathbf{s}_i$ as constants) and equating the derivative to 0 gives:

$$0 = P_i(q)U'(W_i + s_{ik}) + \lambda c_q \tag{A.3}$$

And finally:

$$s_{iq} = -W_i + (U')^{-1}\left(\frac{-\lambda c_q}{P_i(q)}\right) \tag{A.4}$$

Usually, however, it is easier to start from equation (A.3). It is also very handy to manipulate λ as it remains a constant anyway.

A.1. Exponential utility

Equation (A.3) takes the form:

$$\frac{\partial \hat{U}_i^E}{\partial s_{iq}} = -P_i(q)e^{(-W_i - s_{iq})} + \lambda c_q \tag{A.5}$$

$$\begin{aligned} 0 &= -P_i(q)e^{(-W_i - s_{iq})} + \lambda c_q \\ 0 &= -P_i(q)e^{-s_{iq}} + \lambda' c_q \\ \log P_i(q) - s_{iq} &= \log \lambda' + \log c_q \\ s_{iq} &= \log P_i(q) - \log \lambda' - \log c_q \end{aligned}$$

Then using the constraint $s^*_{iN_G} = 0$ gives:

$$0 = \log P_i(N_G) - \log \lambda' - \log c_{N_G}$$
$$\log \lambda' = \log \frac{P_i(N_G)}{c_{N_G}}$$

Therefore:

$$s^*_{ik} = \log P_i(k) - \log c_q - \log P_i(N_G) + \log c_{N_G} \tag{A.6}$$

A.2. Logarithmic utility

Equation (A.3) takes the form:

$$0 = \frac{P_i(q)}{W_i + s_{iq}} + \lambda c_q \tag{A.7}$$

Then:

$$0 = P_i(q) + \lambda c_q W_i + \lambda c_q s_{iq}$$

Summing over q:

$$0 = \sum_{q=1}^{N_G} P_i(q) + \lambda W_i \sum_{q=1}^{N_G} c_q + \lambda \sum_{q=1}^{N_G} c_q s_{iq}$$

Then using the constraint $\mathbf{s}_i^{\mathrm{T}}\mathbf{c} = 0$, the fact that $\sum_{q=1}^{N_G} P_i(q) = 1$ and no-arbitrage assumption $\sum_{q=1}^{N_G} c_q = 1$:

$$0 = 1 + \lambda W_i$$
$$\lambda = -\frac{1}{W_i}$$

Therefore, combined with the constraint $s^*_{iN_G} = 0$:

$$s^*_{ik} = \frac{W_i(P_i(k) - c_k)}{c_k} - \frac{W_i(P_i(N_G) - c_{N_G})}{c_{N_G}} \tag{A.8}$$

A.3. Isoelastic utility

Equation (A.4) takes the form:

$$s_{iq} = -W_i + \lambda' \left[\frac{c_q}{P_i(q)}\right]^{-\frac{1}{\eta}} \tag{A.9}$$

Multiplying both sides by c_q, summing over q and using constraint $\mathbf{s}_i^T\mathbf{c} = 0$ to get:

$$\sum_{j=1}^{N_G} c_j s_{ij} = -W_i \sum_{j=1}^{N_G} c_j + \lambda' \sum_{j=1}^{N_G} c_j \left[\frac{c_j}{P_i(j)}\right]^{-\frac{1}{\eta}}$$
$$\lambda' = \frac{W_i}{\sum_{j=1}^{N_G} c_j \left[\frac{c_j}{P_i(j)}\right]^{-\frac{1}{\eta}}}$$

Which, combined with the constraint $s^*_{iN_G} = 0$, gives the following:

$$\begin{aligned} s^*_{ik} &= W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} - 1 \right] - W_i \left[\frac{\left[\frac{P_i(N_G)}{c_{N_G}}\right]^{\frac{1}{\eta}}}{\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} - 1 \right] \\ &= W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} - \left[\frac{P_i(N_G)}{c_{N_G}}\right]^{\frac{1}{\eta}}}{\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} \right] \end{aligned} \tag{A.10}$$

Appendix B

Derivatives of buying functions

B.1. Exponential utility

$$\frac{\partial s_{ik}^*}{\partial c_q} = \begin{cases} 0 & \text{if } k = N_G, \\ \frac{1}{c_k} & \text{if } k \neq N_G,\ q = N_G, \\ -\frac{1}{c_k} & \text{if } k \neq N_G,\ q \neq N_G,\ k = q, \\ 0 & \text{if } k \neq N_G,\ q \neq N_G,\ k \neq q. \end{cases} \tag{B.1}$$

B.2. Logarithmic utility

$$\frac{\partial s_{ik}^*}{\partial c_q} = \begin{cases} 0 & \text{if } k = N_G, \\ \frac{W_i P_i(k)}{c_k^2} & \text{if } k \neq N_G,\ q = N_G, \\ -\frac{W_i P_i(k)}{c_k^2} & \text{if } k \neq N_G,\ q \neq N_G,\ k = q, \\ 0 & \text{if } k \neq N_G,\ q \neq N_G,\ k \neq q. \end{cases} \tag{B.2}$$

B.3. Isoelastic utility

$$\frac{\partial s_{ik}^*}{\partial c_q} = \begin{cases} 0 & \text{if } k = N_G, \\ W_i \left(\frac{P_i(q)}{c_q}\right)^{\frac{1}{\eta}} \left[\frac{\left(\frac{1}{\eta c_{N_G}}\right)\left(\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}\right) - \left(1-\frac{1}{\eta}\right)\left(\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} - \left[\frac{P_i(N_G)}{c_{N_G}}\right]^{\frac{1}{\eta}}\right)}{\left(\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}\right)^2} \right] & \text{if } k \neq N_G,\ q = N_G, \\ W_i \left(\frac{P_i(q)}{c_q}\right)^{\frac{1}{\eta}} \left[\frac{\left(-\frac{1}{\eta c_q}\right)\left(\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}\right) - \left(1-\frac{1}{\eta}\right)\left(\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} - \left[\frac{P_i(N_G)}{c_q}\right]^{\frac{1}{\eta}}\right)}{\left(\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}\right)^2} \right] & \text{if } k \neq N_G,\ q \neq N_G,\ k = q, \\ -W_i \left(\frac{P_i(q)}{c_q}\right)^{\frac{1}{\eta}} \frac{\left(1-\frac{1}{\eta}\right)\left[\left(\frac{P_i(k)}{c_k}\right)^{\frac{1}{\eta}} - \left(\frac{P_i(N_G)}{c_{N_G}}\right)^{\frac{1}{\eta}}\right]}{\left(\sum_{j=1}^{N_G} c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}\right)^2} & \text{if } k \neq N_G,\ q \neq N_G,\ k \neq q. \end{cases} \tag{B.3}$$

Appendix C

Derivations of equilibrium prices

C.1. Exponential utility

By substituting s_{ik} in equation (4.2) with equation (4.11).

$$\begin{aligned} 0 &= \sum_{i=1}^{N_A} \frac{W_i(P_i(k) - c_k)}{c_k} \\ 0 &= \sum_{i=1}^{N_A} W_i(P_i(k) - c_k) = \sum_{i=1}^{N_A} W_i P_i(k) - c_k \sum_{i=1}^{N_A} W_i \\ c_k &= \frac{\sum_{i=1}^{N_A} W_i P_i(k)}{\sum_{i=1}^{N_A} W_i} \end{aligned}$$

Which is exactly the same as equation (4.17).

C.2. Logarithmic utility

By substituting s_{ik} in equation (4.2) with equation (4.12).

$$\begin{aligned} 0 &= \sum_{i=1}^{N_A} (\log P_i(k) - \log c_k - \log \lambda_i) = \sum_{i=1}^{N_A} \log P_i(k) - \sum_{i=1}^{N_A} \log c_k - \sum_{i=1}^{N_A} \log \lambda_i \\ &= \sum_{i=1}^{N_A} \log P_i(k) - N_A \log c_k - \sum_{i=1}^{N_A} \log \lambda_i \\ \log c_k &= \frac{1}{N_A} \frac{\prod_{i=1}^{N_A} P_i(k)}{\prod_{i=1}^{N_A} \lambda_i} \\ c_k = \left(\frac{\prod_{i=1}^{N_A} P_i(k)}{\prod_{i=1}^{N_A} \lambda_i} \right)^{\frac{1}{N_A}} &= \left(\prod_{i=1}^{N_A} (P_i(k))^{\frac{1}{N_A}} \right) \left(\prod_{i=1}^{N_A} \lambda_i \right)^{-1} \end{aligned}$$

From which equation (4.16) follows directly.

C.3. Isoelastic utility

By substituting s_{ik} in equation (4.2) with equation (4.13).

$$0 = \sum_{i=1}^{N_A} W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{\sum_j c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} - 1 \right] = \sum_{i=1}^{N_A} W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{\sum_j c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} \right] - \sum_{i=1}^{N_A} W_i$$

$$\sum_{i=1}^{N_A} W_i = \sum_{i=1}^{N_A} W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{\sum_j c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} \right]$$

$$c_k = \frac{\sum_{i=1}^{N_A} W_i \left[\frac{c_k \left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{\sum_j c_j \left[\frac{P_i(j)}{c_j}\right]^{\frac{1}{\eta}}} \right]}{\sum_{i=1}^{N_A} W_i}$$

From which equation (4.18) follows directly. When $\eta = 1$, it is equivalent to equation (4.16), as it should be.

Appendix D

Other derivations and algorithms

D.1. $\frac{\partial \mathbf{c}}{\partial \mathbf{z}}$

If $q = k$

$$\begin{aligned}
\frac{\partial c_q}{\partial z_q} &= \frac{e^{z_q}\sum_j e^{z_j} - e^{2z_q}}{\left(\sum_j e^{z_j}\right)^2} \\
&= \frac{e^{z_q}\left(\sum_j e^{z_j} - e^{z_q}\right)}{\left(\sum_j e^{z_j}\right)\left(\sum_j e^{z_j}\right)} \\
&= \frac{e^{z_q}}{\sum_j e^{z_j}} . \frac{\sum_j e^{z_j} - e^{z_q}}{\sum_j e^{z_j}} \\
&= \frac{e^{z_q}}{\sum_j e^{z_j}} . \left(1 - \frac{e^{z_q}}{\sum_j e^{z_j}}\right)
\end{aligned}$$

If $q \neq k$

$$\begin{aligned}
\frac{\partial c_q}{\partial z_k} &= \frac{-e^{z_q}e^{z_k}}{\left(\sum_j e^{z_j}\right)^2} \\
&= \frac{-e^{z_q}e^{z_k}}{\left(\sum_j e^{z_j}\right)\left(\sum_j e^{z_j}\right)} \\
&= \frac{e^{z_q}}{\sum_j e^{z_j}} . \frac{-e^{z_k}}{\sum_j e^{z_j}} \\
&= \frac{e^{z_q}}{\sum_j e^{z_j}} . \left(0 - \frac{e^{z_k}}{\sum_j e^{z_j}}\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial c_q}{\partial z_k} &= \frac{e^{z_q}}{\sum_j e^{z_j}} . \left(I_{qk} - \frac{e^{z_k}}{\sum_j e^{z_j}}\right) \\
&= c_q(I_{qk} - c_k)
\end{aligned}$$

D.2. Generating random bistochastic matrices

Theoretical justification of correctness of this algorithm known in the literature can be found in [CapSomBruŻyc09].

Algorithm 5 Generating random bistochastic matrices

Input: K, N
$M \leftarrow$ K-by-N matrix of random numbers between 0 and 1
$D \leftarrow$ vector of length K filled with ones
while max(D) $> \epsilon$ **do**
  for $k = 1 \dots K$ **do**
    $M(k,:) \leftarrow M(k,:)/sum(M(k,:))$
  end for
  for $n = 1 \dots N$ **do**
    $M(:,n) \leftarrow M(:,n)/(sum(M(:,n)) * (N/K))$
  end for
  for $k = 1 \dots K$ **do**
    $D(k) = |sum(M(k,:)) - 1|$
  end for
end while
Output: M

D.3. $\lim_{\eta \to 1} U_{iso}(x) = U_{log}(x)$

Using L'Hôpital's rule:

$$\lim_{\eta \to 1} \frac{x^{1-\eta} - 1}{1-\eta} =$$
$$\lim_{\eta \to 1} \frac{\frac{d}{d\eta}\left(x^{1-\eta} - 1\right)}{\frac{d}{d\eta}\left(1-\eta\right)} =$$
$$\lim_{\eta \to 1} \frac{\frac{d}{d\eta}\left(e^{\log_e x^{1-\eta}}\right)}{-1} =$$
$$\lim_{\eta \to 1} \frac{\frac{d}{d\eta}\left(e^{(1-\eta)\log_e x}\right)}{-1} =$$
$$\lim_{\eta \to 1} \frac{-e^{(1-\eta)\log_e x}\log_e x}{-1} =$$
$$\lim_{\eta \to 1} e^{(1-\eta)\log_e x}\log_e x =$$
$$\log_e x$$

D.4. Arrow-Pratt measure of absolute risk-aversion

Beginning with equating expected utility to initial utility:

$$U(W) = \int_{-\infty}^{\infty} U(W - p + x) f(x) dx.$$

Then using Taylor expansion, approximately:

$$
\begin{aligned}
U(W) &= \int_{-\infty}^{\infty} \left[U(W) + U'(W)(x-p) + U''(W)\frac{(x-p)^2}{2} \right] f(x)dx \\
U(W) &= U(W) + U'(W)\int_{-\infty}^{\infty}(x-p)f(x)dx + U''(W)\int_{-\infty}^{\infty}\frac{(x-p)^2}{2}f(x)dx \\
0 &= U'(W)\int_{-\infty}^{\infty}(x-p)f(x)dx + U''(W)\int_{-\infty}^{\infty}\frac{(x-p)^2}{2}f(x)dx \\
p &= \int_{-\infty}^{\infty} xf(x)dx + \frac{U''(W)}{U'(W)}\int_{-\infty}^{\infty}\frac{(x-p)^2}{2}f(x)dx
\end{aligned}
$$

D.5. Buying function for isoelastic utility when $\eta \to 0$

$$
\begin{aligned}
s_{ik}^* =& W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} - \left[\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}}}{c_k\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} + (1-c_k)\left[\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}}} \right] \\
=& W_i \left[\frac{\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}}}{c_k\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} + (1-c_k)\left[\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}}} - \frac{\left[\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}}}{c_k\left[\frac{P_i(k)}{c_k}\right]^{\frac{1}{\eta}} + (1-c_k)\left[\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}}} \right] \\
=& W_i \left[\frac{1}{c_k + (1-c_k)\left[\frac{c_k}{P_i(k)}\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}}} - \frac{1}{c_k\left[\frac{1}{\frac{c_k}{P_i(k)}\frac{1-P_i(k)}{1-c_k}}\right]^{\frac{1}{\eta}} + (1-c_k)} \right]
\end{aligned}
$$

Then from the fact that:

$$
\lim_{\eta\to 0}\left[\frac{c_k}{P_i(k)}\frac{1-P_i(k)}{1-c_k}\right]^{\frac{1}{\eta}} = \begin{cases} 0 & \text{if } P_i(k) > c_k, \\ \infty & \text{if } P_i(k) < c_k, \\ 1 & \text{if } P_i(k) = c_k. \end{cases}
$$

It follows that:

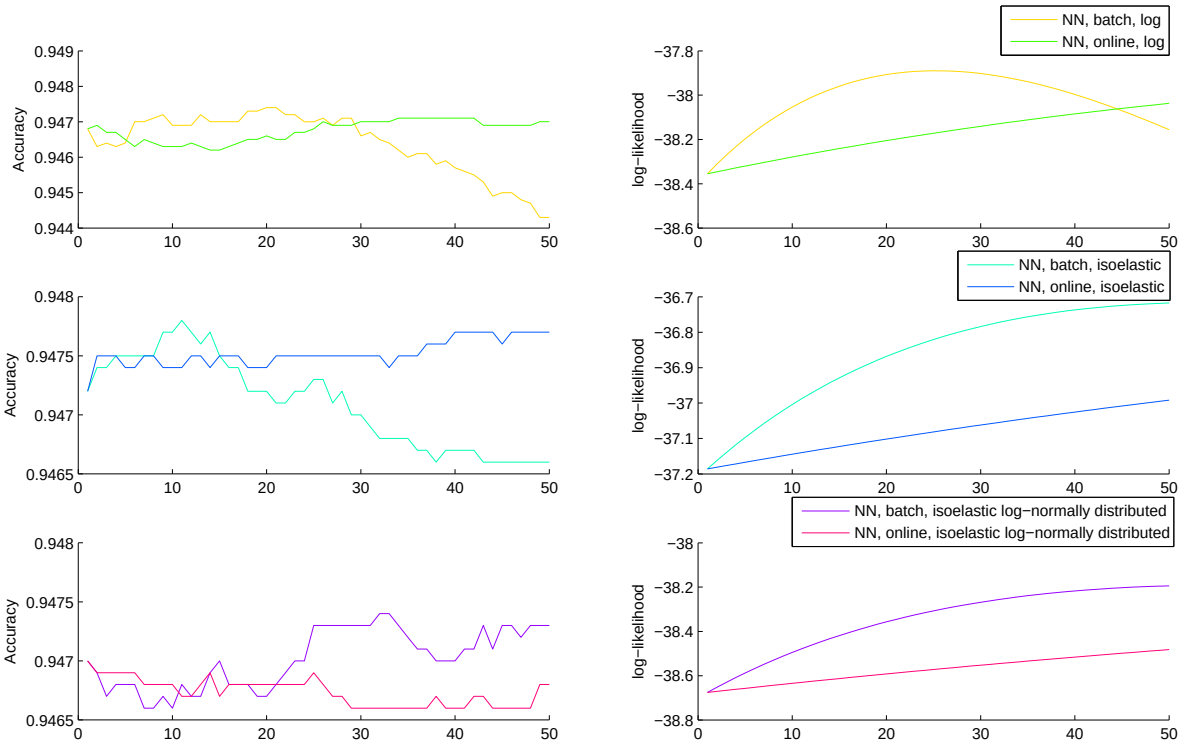
$$
\lim_{\eta\to 0} s_{ik}^* = \begin{cases} \frac{W_i}{c_k} & \text{if } P_i(k) > c_k, \\ 0 & \text{if } P_i(k) = c_k, \\ -\frac{W_i}{1-c_k} & \text{if } P_i(k) < c_k. \end{cases}
$$

Appendix E

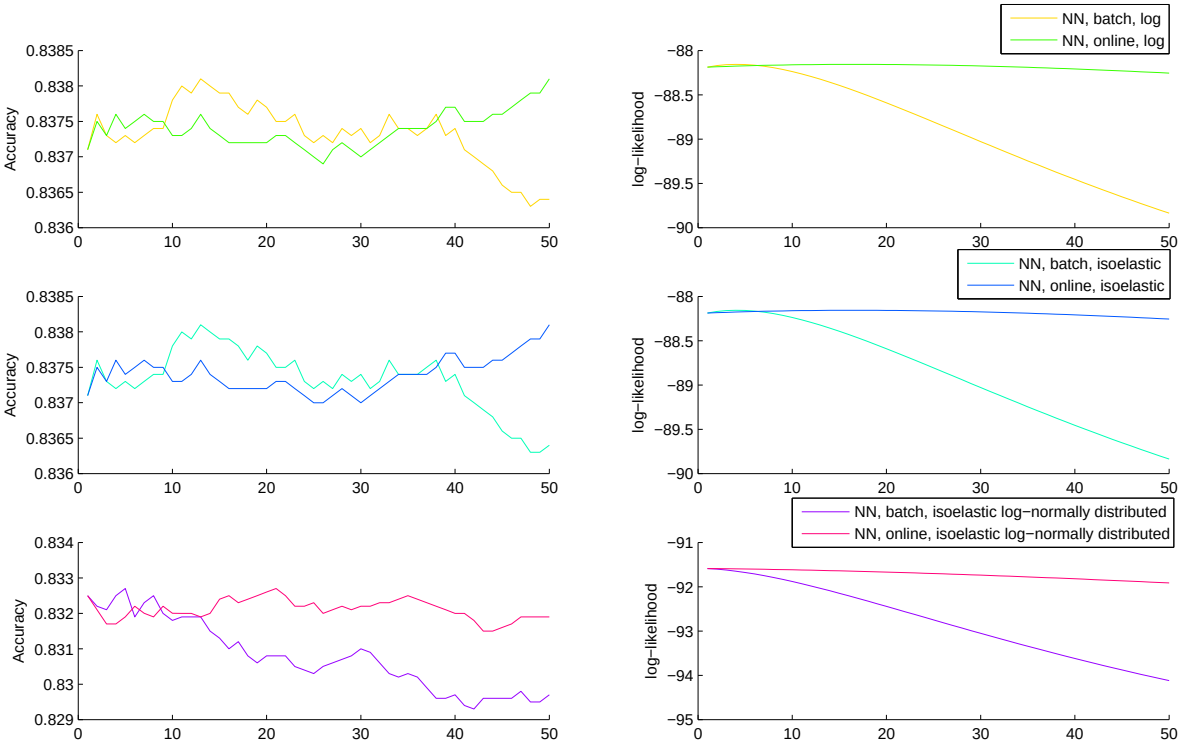
Additional figures

Some figures that did not fit in the main part of the thesis are:

- Figure E.1 extends Figure 5.8.
- Figure E.2 extends Figure 5.4.
- Figure E.3 extends Figure 5.5.
- Figure E.4 extends Figure 5.6.
- Figure E.5 extends Figure 5.7.

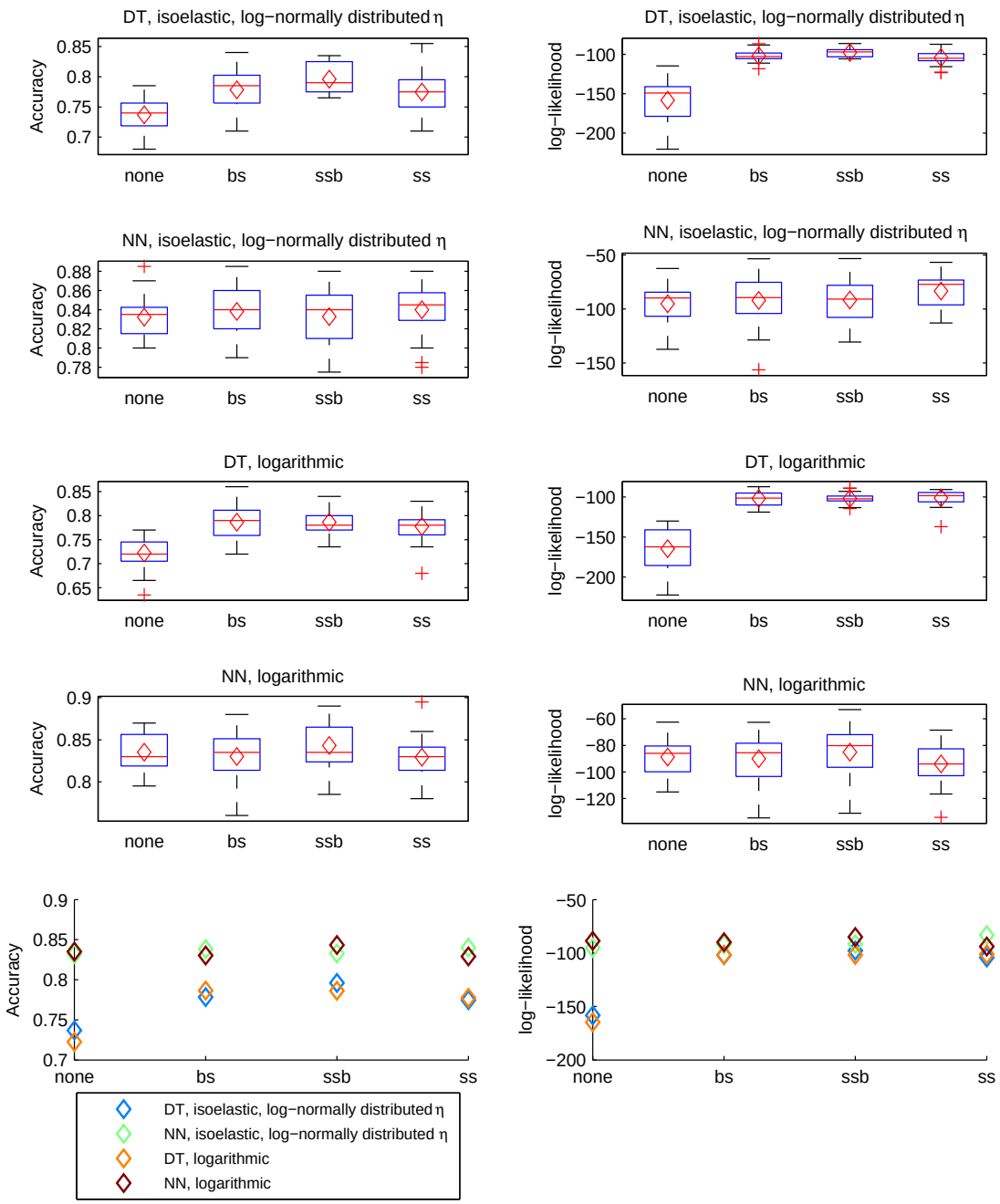


(a) image



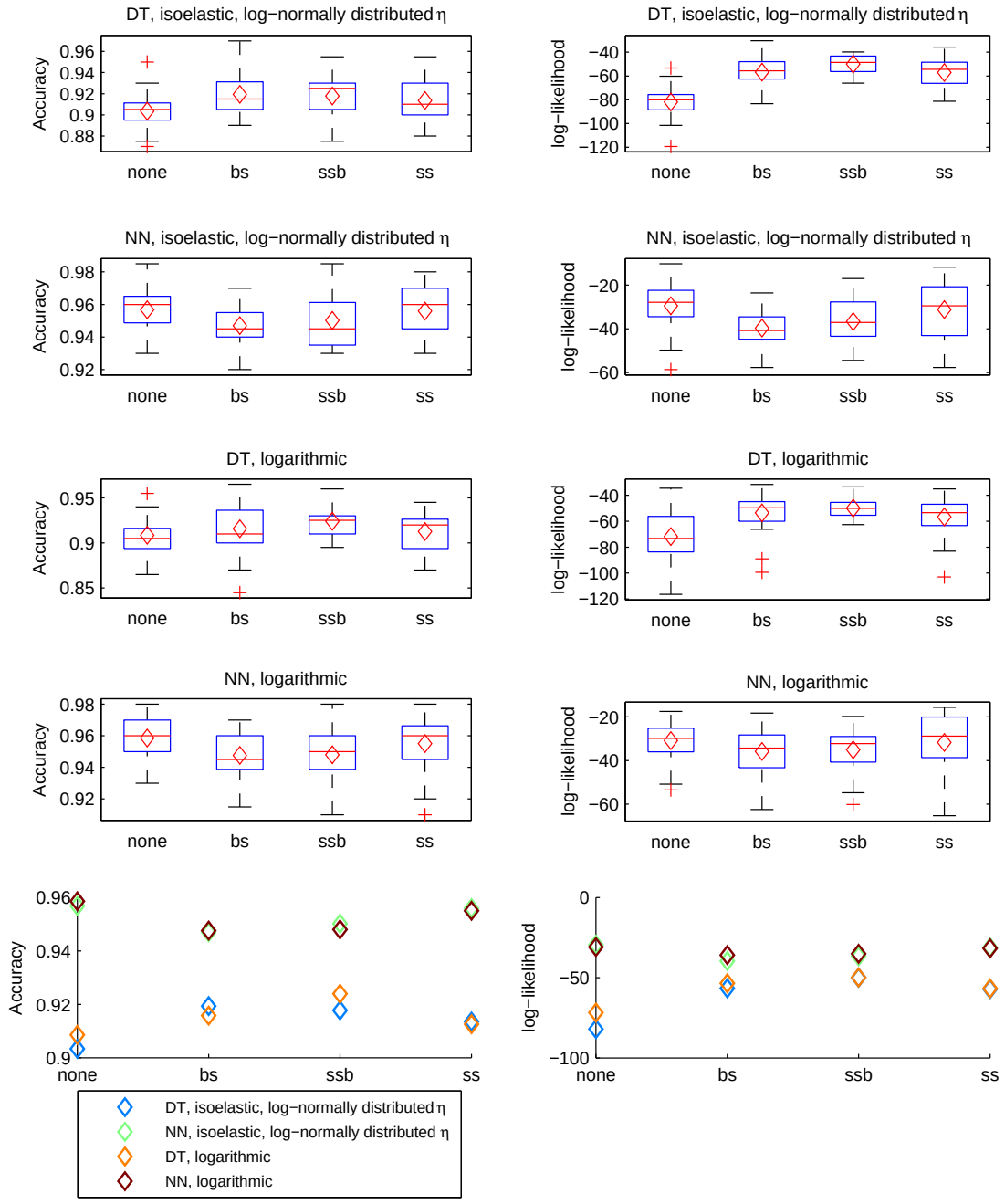
(b) waveform

Figure E.1: Accuracy and log-likelihood as a function of the number of training epochs for batch and online training. Extension of Figure 5.8, with results for neural networks.



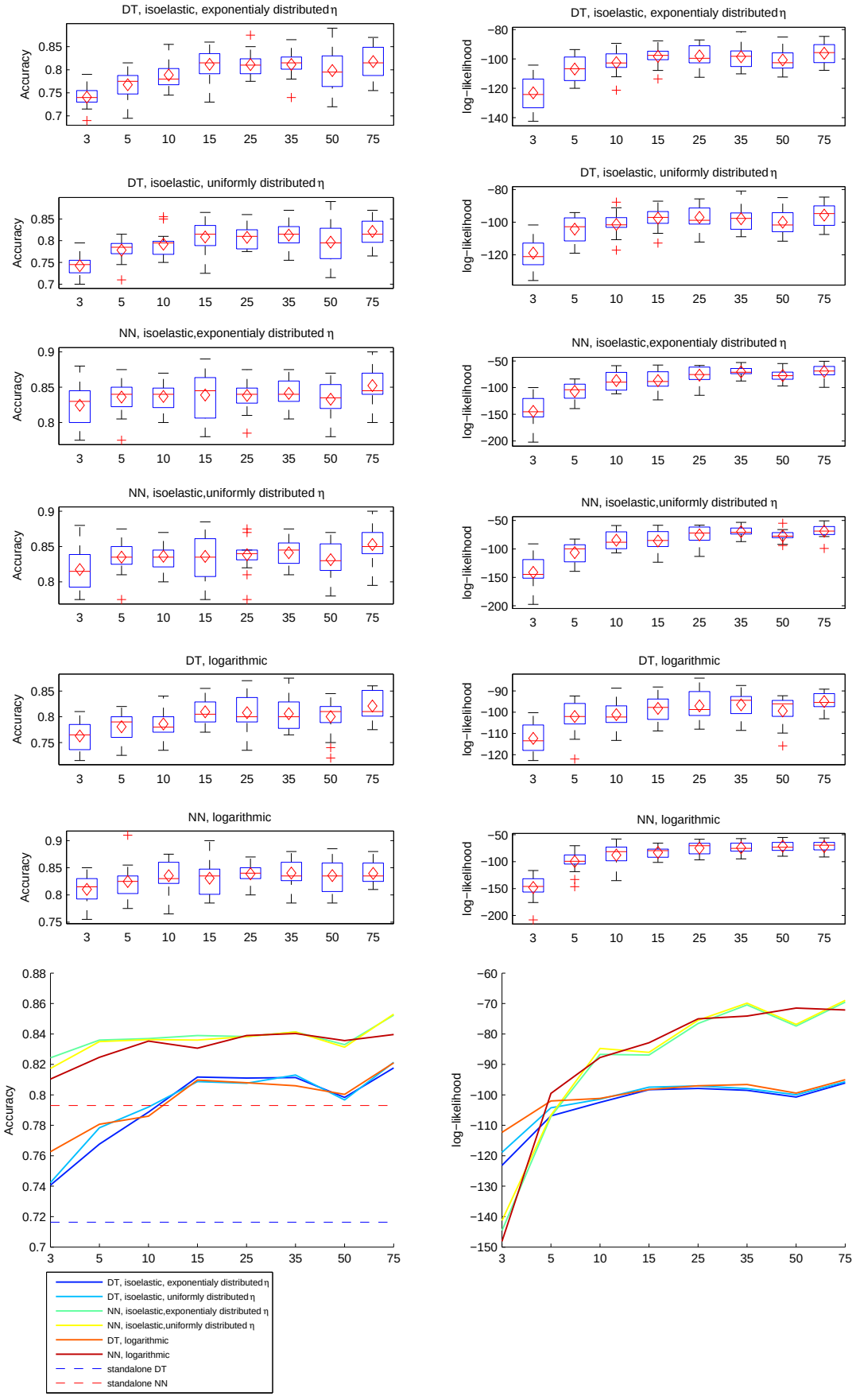
(a) waveform

Figure E.2



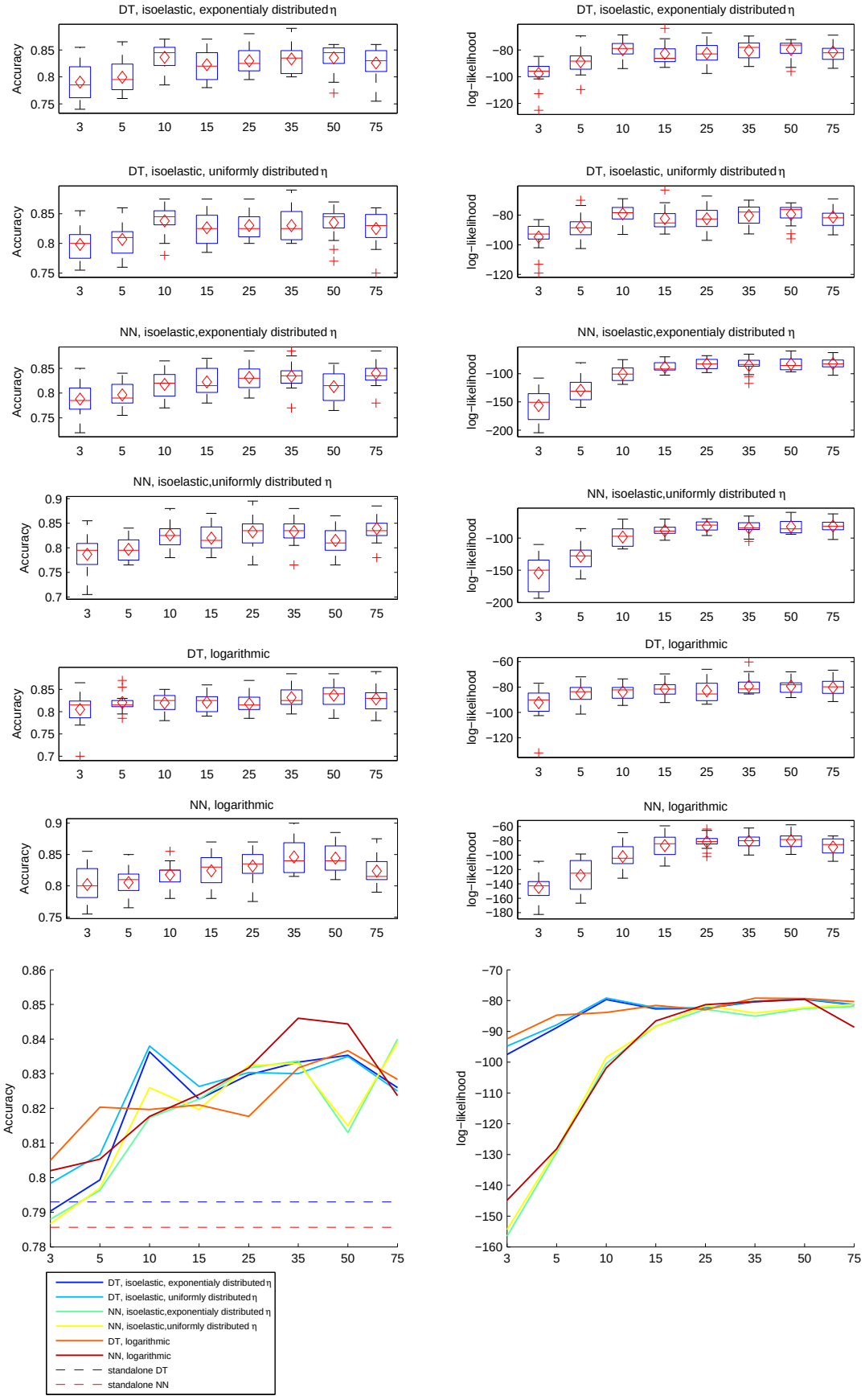
(b) image

Figure E.2: Accuracy and log-likelihood for different sampling methods. Extension of Figure 5.4, with box plots.



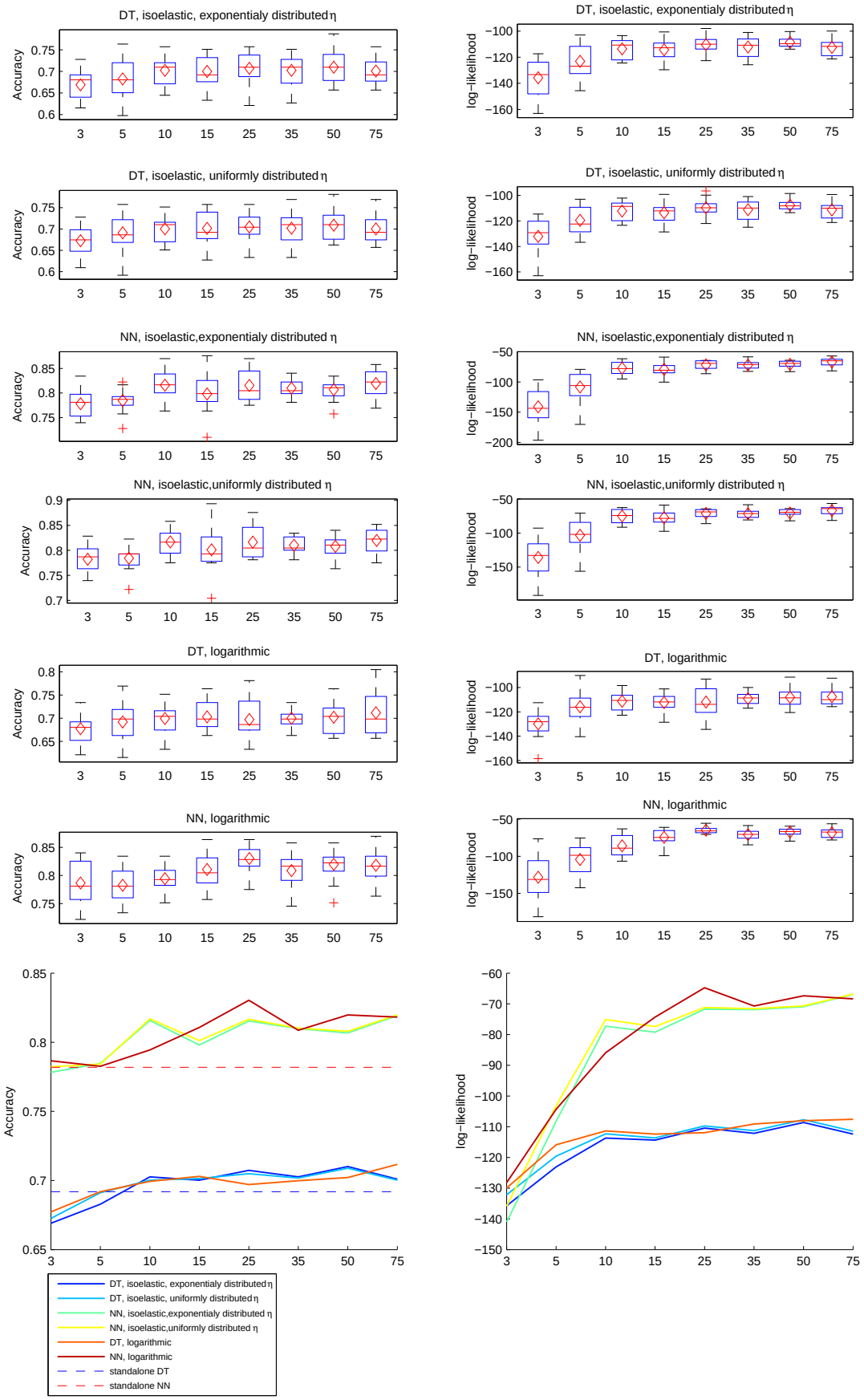
(a) waveform

Figure E.3



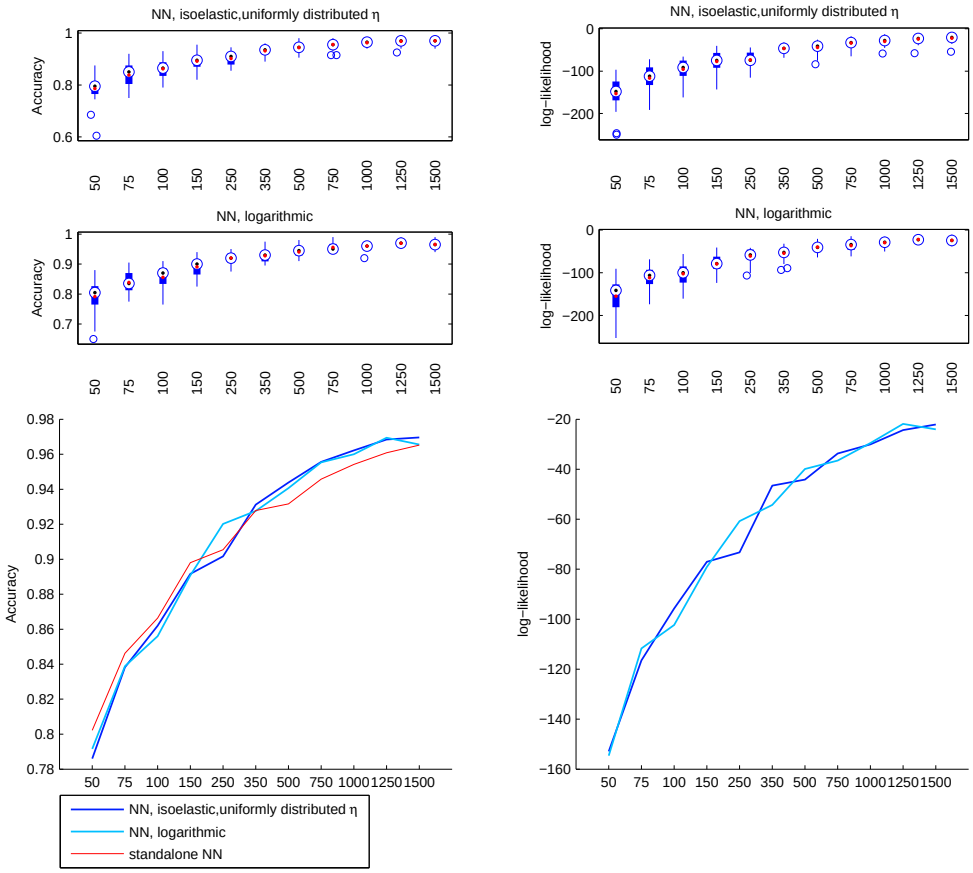
(b) magic

Figure E.3

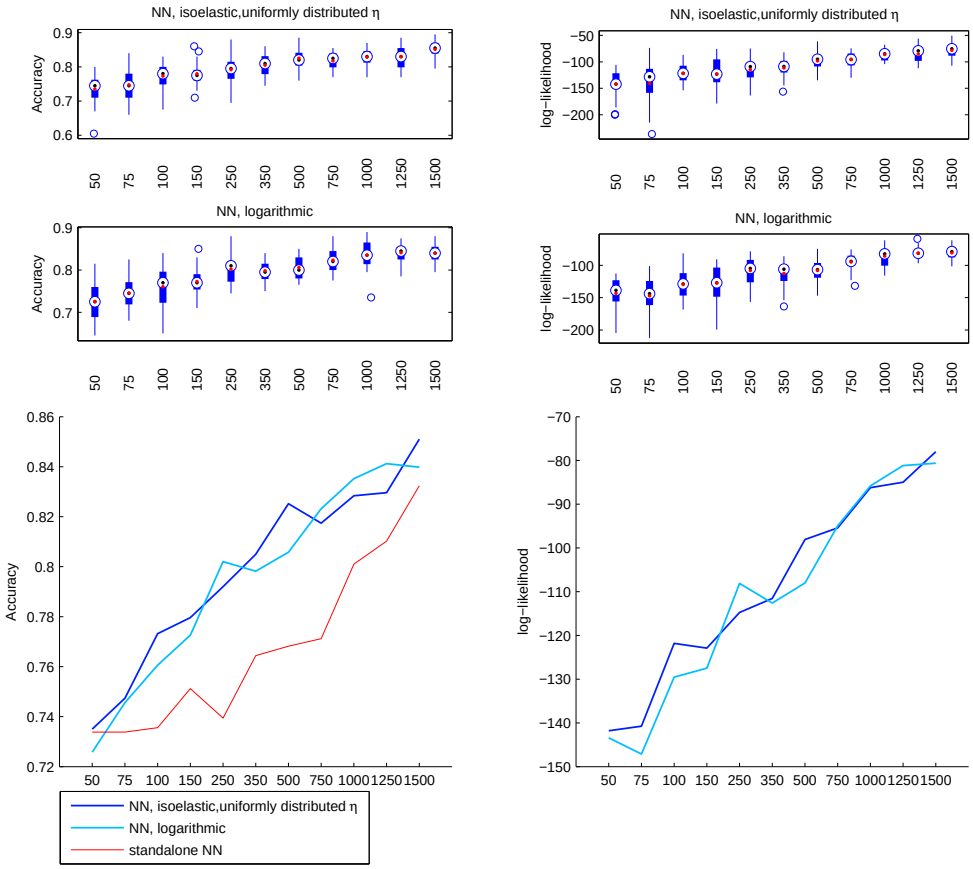


(c) vehicle

Figure E.3: Accuracy and log-likelihood as a function of the number of agents in the market. Extension of Figure 5.5, with box plots.

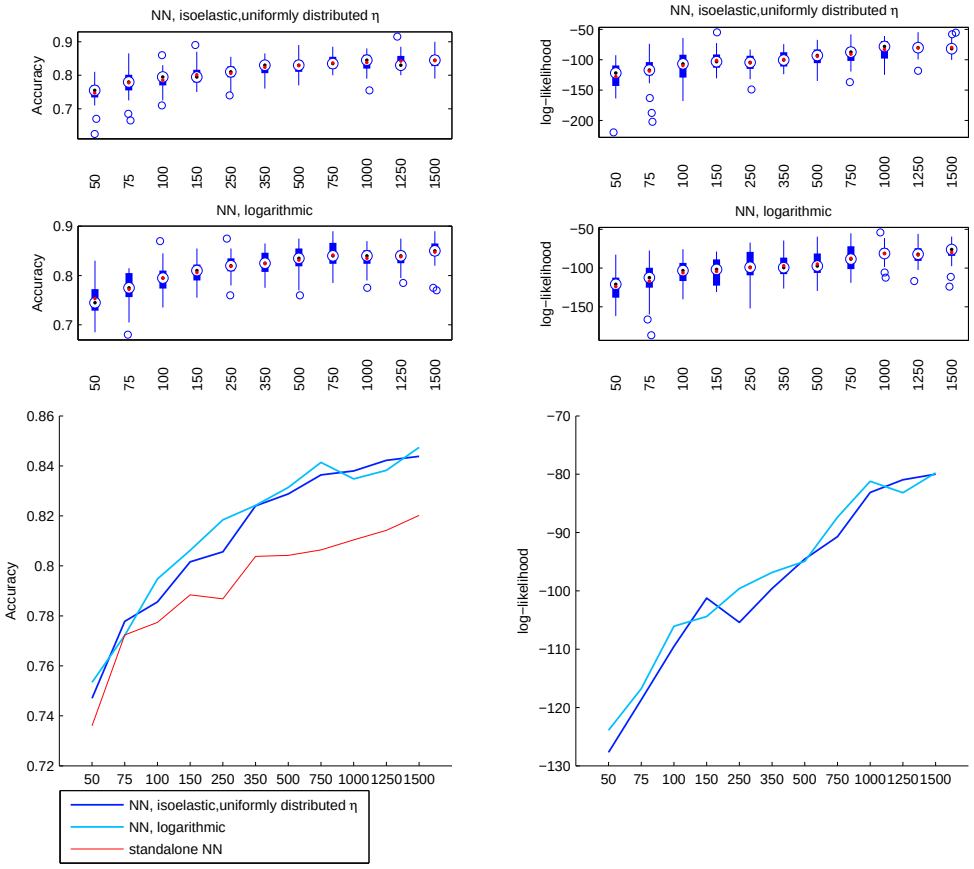


(a) waveform



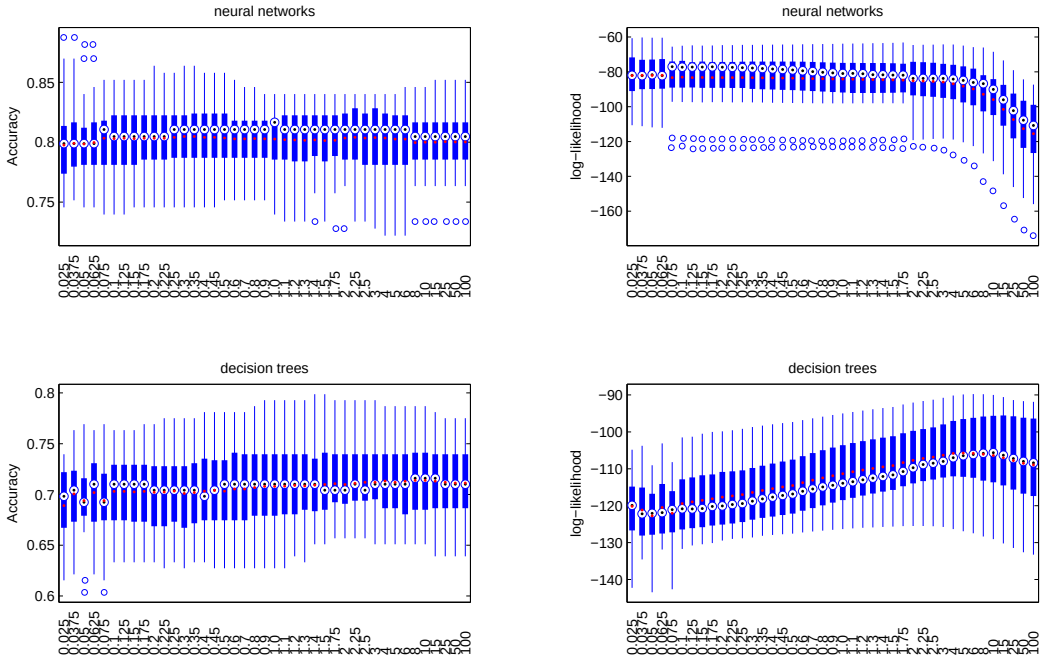
(b) magic

Figure E.4

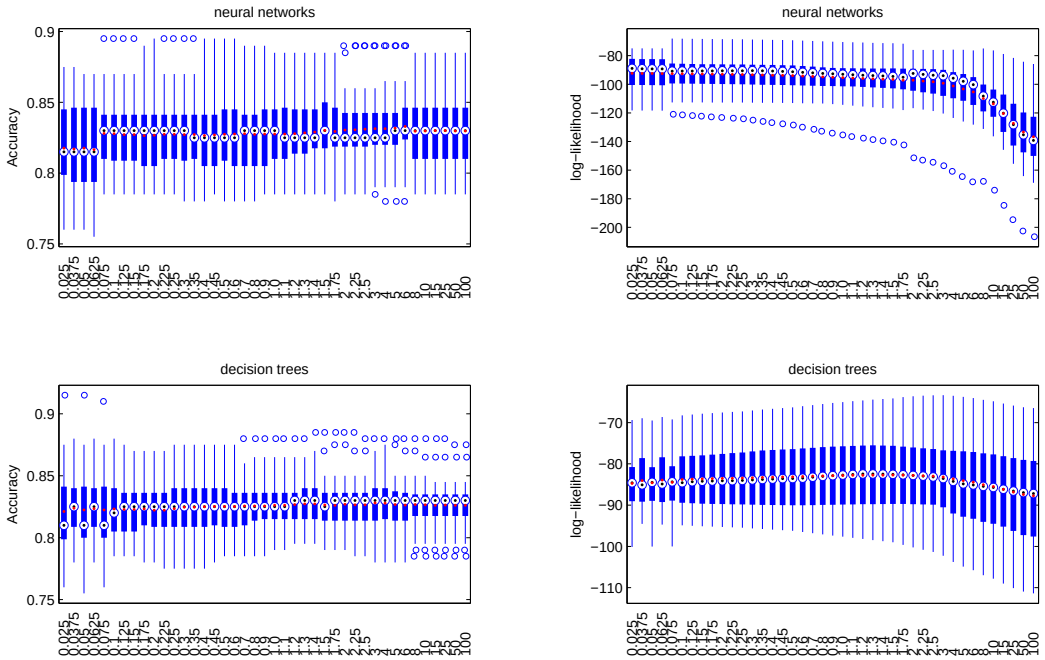


(c) image

Figure E.4: Accuracy and log-likelihood as a function of the number of points in the training set. Extension of Figure 5.6, with box plots.

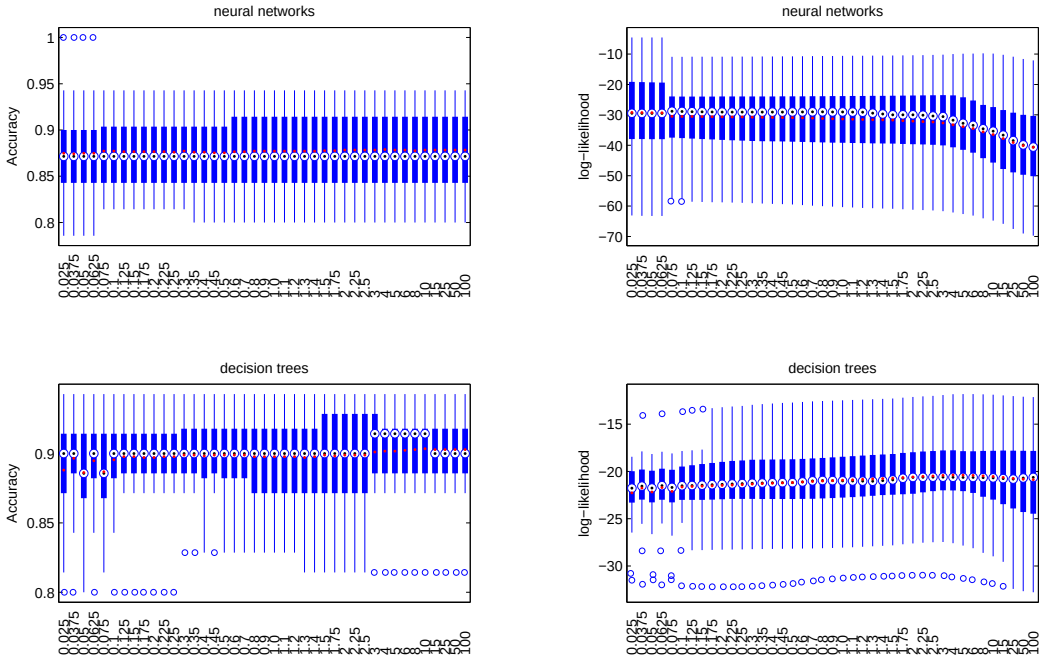


(a) vehicle

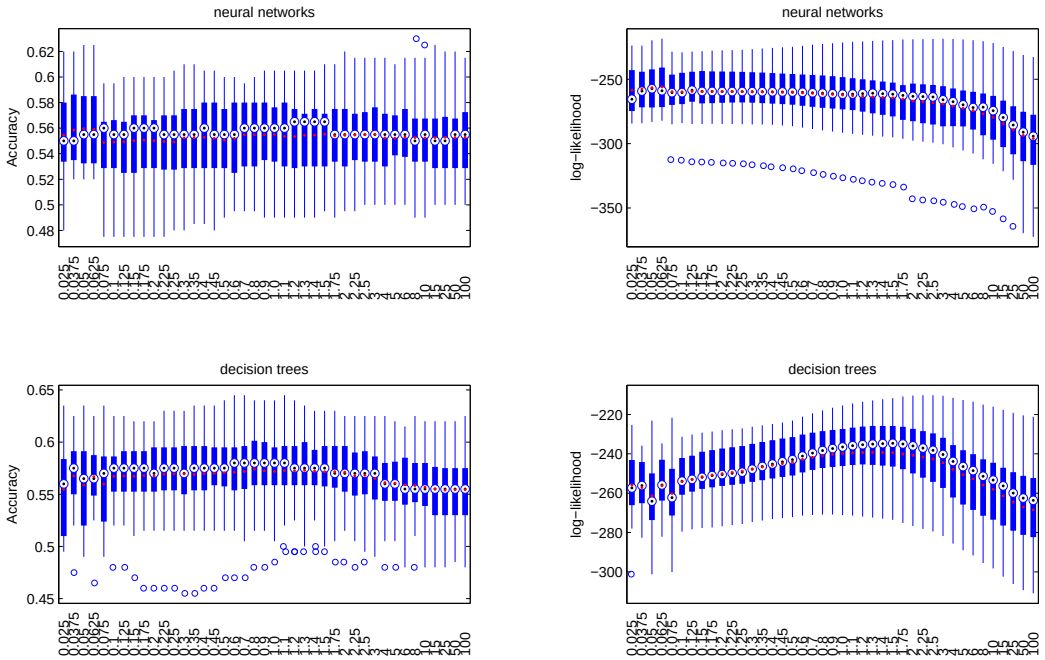


(b) magic

Figure E.5

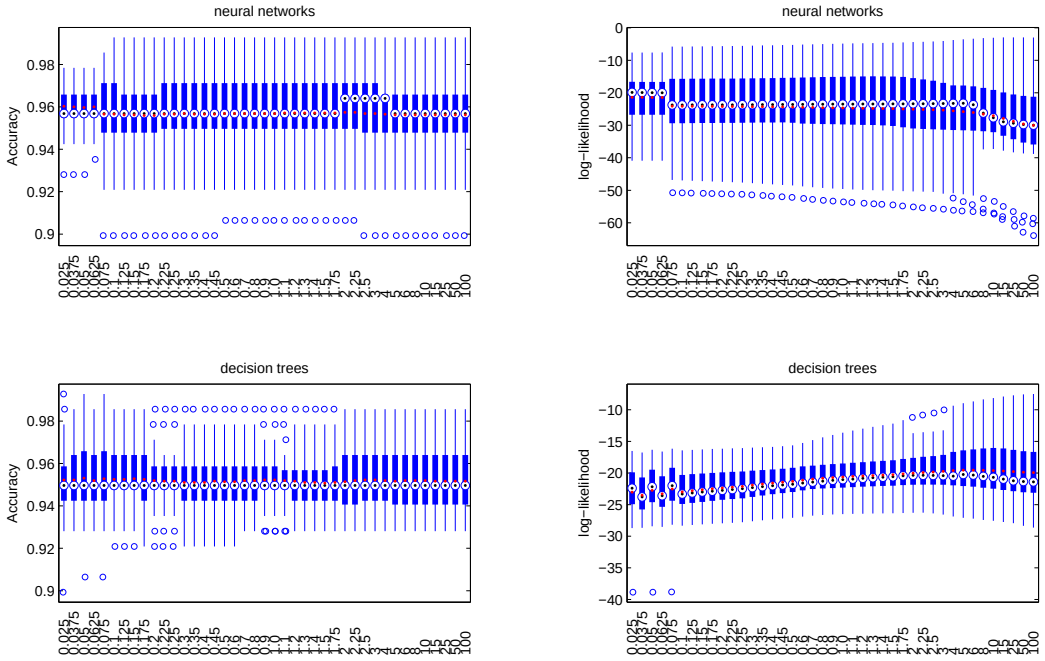


(c) ionosphere

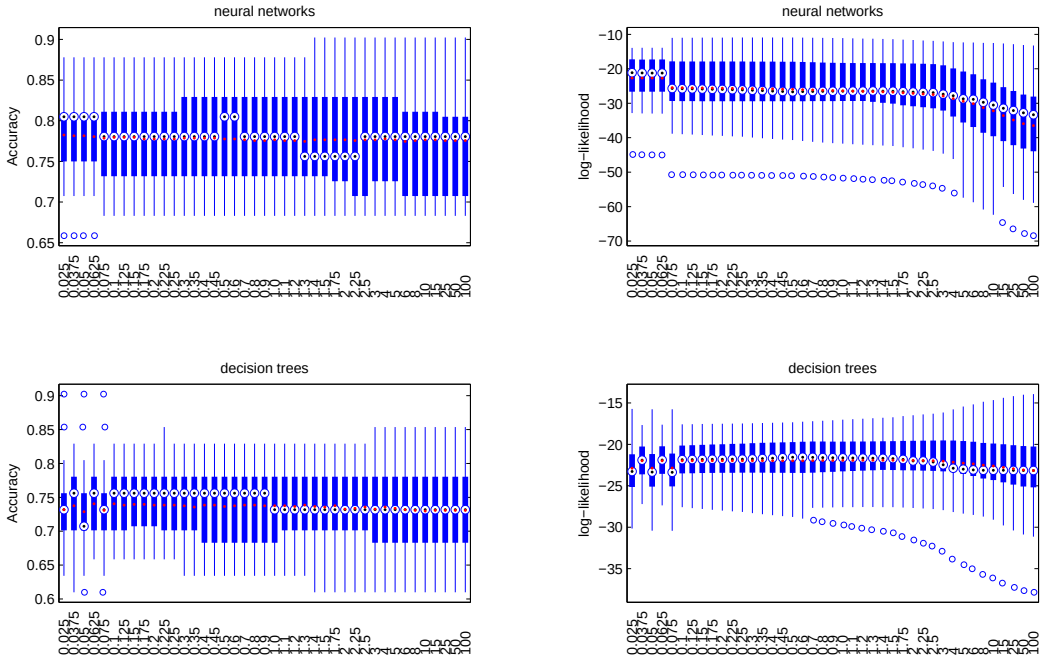


(d) yeast

Figure E.5



(e) breast cancer



(f) sonar

Figure E.5: Accuracy and log-likelihood as a function of η. Extension of Figure 5.7, with more data sets.

Bibliography

[ArrDeb54] Kenneth J. Arrow, Gerard Debreu, *Existence of an equilibrium for a competitive economy*, Econometrica: Journal of the Econometric Society, 22(3):265290, 1954.

[BarLay11] Adrian Barbu, Nathan Lay, *An Introduction to Artificial Prediction Markets*, `arXiv:1102.1465v3 [stat.ML]`.

[BelKor07] Robert M. Bell, Yehuda Koren, *Lessons from the Netflix prize challenge*, ACM SIGKDD Explorations Newsletter, 9(2):7579, 2007.

[BerForNelRie08] Joyce Berg, Robert Forsythe, Forrest Nelson, Thomas Rietz, *Results from a Dozen Years of Election Futures Markets Research*, Handbook of Experimental Economics Results, 2008, vol. 1, pages 742-751.

[Bis06] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[Bre01] Leo Breiman, *Random Forests*, Machine Learning 45(1): 5-32, 2001.

[CapSomBruŻyc09] Valerio Cappellini, Hans-Jürgen Sommers, Wojciech Bruzda, Karol Życzkowski, *Random bistochastic matrices*, Journal of Physics A-mathematical and Theoretical, vol. 42, no. 36, 2009.

[Die00] Thomas G. Dietterich, *Ensemble Methods in Machine Learning*, MCS 2000.

[Fun01] Glenn Fung, *A Comprehensive Overview of Basic Clustering Algorithms.*

[HasTibFri09] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2009.

[Hin02] Goffrey E. Hinton, *Training products of experts by minimizing contrastive divergence*, Neural Computation, 14:1771-1800, 2002.

[Hor91] Kurt Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, vol. 4(2), 1991.

[Lay09] Nathan Lay, *Supervised Aggregation of Classifiers using Artificial Prediction Markets.* MSc Thesis, buying State University, 2009.

[LayBar10] Nathan Lay, Adrian Barbu, *Supervised Aggregation of Classifiers using Artificial Prediction Markets*, ICML 2010.

[LIBSVM] Machine Learning and Data Mining Group at National Taiwan University, www.csie.ntu.edu.tw/∼cjlin/libsvm/.

[Man06] Charles F. Manski, *Interpreting the predictions of prediction markets*, Economics Letters, Elsevier, vol. 91(3), pages 425-429, June.

[Min02] Thomas P. Minka, *Bayesian model averaging is not model combination*, MIT Media Lab note, 2002.

[Netlab] Non-linearity and Complexity Research Group at Aston University, www.aston.ac.uk/ncrg.

[Pet05] Michael Peters, *Lecture notes to Econ304 - Honours Microeconomics at the University of British Columbia*, 2005.

[Sch03] Robert E. Schapire, *The boosting approach to machine learning: An overview*, Nonlinear Estimation and Classification, Springer, 2003.

[Sto11] Amos Storkey, *Machine Learning Markets*, AISTATS 2011.

[StoMilGer11] Amos Storkey, Jonathan Millin, Krzysztof Geras, *Inhomogeneous Agents in Machine Learning Markets*, submitted to NIPS 2011.

[WolZit04] Justin Wolfers, Eric Zitzewitz, *Prediction markets*, Journal of Economic Perspectives, pages 107-126, 2004.

[WolZit07] Justin Wolfers, Eric Zitzewitz, *Interpreting Prediction Market Prices as Probabilities*, 2007.