Fuzzy Systems and Data Mining VIII
A.J. Tallón-Ballesteros (Ed.)

doi:10.3233/FAIA220373

Lattice Linear Discriminant Analysis for Shape Constrained Classification

Geng DENG [a,1], Yaoguo XIE [a], Xindong WANG [a] and Qiang FU [a]
[a] *Corporate Model Risk, Wells Fargo*

Abstract. Recently shape constrained classification has gained popularity in the machine learning literature in order to exploit extra model information besides raw data features. In this paper, we present a new Lattice Linear Discriminant Analysis (Lattice-LDA) classifier, which allows to take shape constraints of data inputs, such as monotonicity and convexity/concavity. Lattice-LDA constructs a nonparametric nonlinear discriminant hyperplane for classification, using an additive format of 1-D lattice functions (piecewise linear functions). Moreover, the new classifier features in taking complex shape constraints including combinations of shapes or S-shape. We optimize the model parameters using the Adaptive Moment Estimation (Adam) algorithm embedding stepwise projections which guarantee feasibility of the shape constraints. Through simulation and real-world examples, we demonstrate that the new classifier could accurately recover the nonlinear marginal effect functions and improve classification accuracy when additional shape information is present.

Keywords. Shape constrained classification, Lattice method, Linear Discriminant Analysis, Monotonic inference

1. Introduction

In machine learning, classification is the problem to use an object's features to identify which group or class it belongs to. The classification problem has been discussed long in history and can be traced to Fisher's Linear Discriminant Analysis (LDA) [1]. In many areas of applied science, for classification problems, it typically exhibits shape restricted relationships, which may include monotonicity and unimodality, between input features and the target. For instance, in business and econometrics area, generally the demand functions are monotonically decreasing in prices, cost functions are monotonically increasing and may be concave in input prices, and utility functions of risk averse agents are concave. Conventional classifiers, such as standard LDA and logistic regression, is not able to utilize this domain-specific knowledge to improve accuracy of model and may also make it difficult to interpret. Considering various types of shape constraints information, monotonicity is commonly seen and also widely used in practice. Traditional classifiers, which may not accommodate the aforementioned shape information, could yield

[1]Corresponding Author: Geng Deng, Wells Fargo, 1753 Pinnacle Dr, McLean, VA 22102, USA ; E-mail: geng.deng@wellsfargo.com.
Disclaimer: The opinions in this paper are strictly those of the authors and do not represent the views of Wells Fargo & Company, or any of their subsidiaries or affiliates.

data over-fitting issues, and make the results counter-intuitive. In recent years, researches regarding classification problems with monotonicity have gained increasing attentions. For instance, a thorough review of modern monotonic classifiers, which includes monotonic tree methods, support vector machine (SVM), monotonic neural network (NN), etc. are given in [2].

As a supervised learning tool, LDA has applications in classification and dimensionality reduction [3]. Since LDA was developed by Fisher in 1936, it was generalized to handle various classification problems. Hastie et al. [4] applied optimal scoring to obtain nonparametric versions of discriminant analysis. Later, Shao et al. [5] developed sparse LDA framework, and Wang et al. [6] proposed a semi-supervised LDA for dimension reduction and classifications. Robust LDA methodologies were developed to handle the outliers and enhance the robustness to noise [7, 8]. More recently, LDA and several extensions were also applied in image recognitions [9, 10, 11]. To handle non-linearity, Generalized Discriminant Analysis (GDA) using kernel function operator was proposed in [12, 13, 14]. The underlying theory is analogous to the kernel transformation in SVM as the GDA method maps the input features to high dimensional space. However, similar to SVM, it does not allow domain-specific shape information, and may cause overfitting issue. Thus, an improved methodology to incorporate monotonic or more generic shape information is preferred in practical area.

Monotonic classification [2] has applications to many classifiers, k-Nearest Neighbor (k-NN) [15], tree methods [16], SVM [17], and neural networks (NN) [18]. [19] developed the isotonic regression, which was one of the early examples of shape-restricted inference. After that, in data analytics area, researches have developed various methods and algorithms to apply the monotonic constraints. For example, Ben-David et al. [20] introduced Ordinal Learning Method (OLM), where the authors applied a monotonic rule based object ordering procedure. Gutierrez et al. [21] proposed an ordinal classification/regression where target can take values from an ordered categories set. Kotlowski and Slowiński [22] also presented an analysis using a nonparametric ordinal classification which can accommodate monotonicity constraints. Moreover, in finance area, Potharst and Feelders [23] provided a monotonically constrained tree method to model house pricing problem.

Motivated by the flexibility of lattice approach, we introduce Lattice-LDA, a novel extension of LDA using the lattice based discriminant hyperplane. The new classifier directly supports simple shape constraints such as monotonicity and convexity/concavity for unimodal input. We also present examples which highlight the classifier's ability to process more complex shape constraints such as the S-shape. The key to our approach is to develop a nonlinear discriminant hyperplane using an additive model of 1-D lattice functions (which is essentially a piecewise linear function). Estimation of the lattice parameters uses the celebrated Adaptive Moment Estimation (Adam) Stochastic Gradient Descent (SGD) algorithm which includes a sequential projection algorithm to map to solution to shape constrained feasible regions.

The lattice based model is a nonparametric approach, where the lattice itself is fully defined by the lattice grid and the values at the grid. The shape constraint can be considered as a model regularization to improve robustness of fit. Garcia and Gupta [24] postulate the original lattice regression, including a range of applications [25]. Google has recently published the TensorFlow Lattice (TFL) library [26] which is a lattice based predictive model taking shape constrained features, in addition to the well known Ten-

sorFlow library. Our Adam plus sequential projection algorithm is motivated by the the same idea as in TFL.

We organize the paper as follows. We first introduce standard LDA theory in Section 2. Next, we describe the Lattice-LDA methodology in Section 3. More specifically, the Adam optimization and projection algorithms are presented in Section 3.3. In Sections 4 and 5, we perform simulation and real data studies to demonstrate that, the Lattice-LDA methodology could produce better classification results compared with the popular classifiers, which include the standard LDA, SVM, and tree methods. Finally, in Section 6, we present the concluding remarks and discuss some future research directions.

2. Standard LDA

The standard Linear Discriminant Analysis (LDA) model was first proposed by Fisher [1] to separate two classes of objects, and was generalized by Rao [27] to address multi-class problems. A further discussion of the LDA theory can be found in the book [3] and the Scikit-learn machine learning library [28]. In the following, we will introduce the notions used and present the LDA algorithm.

Denote training data set as $\{(x_l, y_l)\}, l = 1, 2, \ldots, N$, where each object x_l includes d features: $x_l \in \mathbb{R}^d$, and each label y_l comes from an M-class set: $y_l \in \{0, 1, 2, ..., M-1\}$. In the conventional discriminant analysis, the density of input X of each class M is assumed to follow a multivariate Gaussian distribution $N(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. More specifically, in LDA, all distributions share the same covariance matrix $\boldsymbol{\Sigma}_m = \boldsymbol{\Sigma}$, a homoscedasticity assumption; whereas in the Quadratic Discriminant Analysis (QDA), the covariance matrix $\boldsymbol{\Sigma}_m$ is allowed to differ by class m. LDA applies the Bayesian inference to select the optimal class label corresponding to the highest conditional probability

$$LDA(\boldsymbol{x}) = \arg\max_m p(y = m | X = \boldsymbol{x}) \tag{1}$$

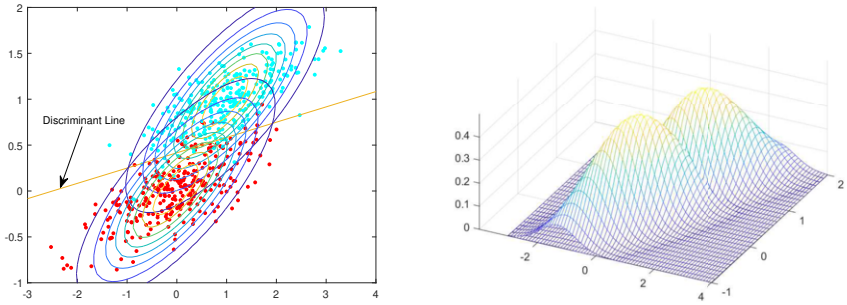
After the variable transformation and simplification process, the solution of LDA classifier is linked to the Gaussian distribution parameters

$$LDA(\boldsymbol{x}) = \arg\max_m \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_m - \frac{1}{2} \boldsymbol{\mu}_m^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_m + \log(\pi_m) \tag{2}$$

where π_m is the prior probability.

For illustration purpose, we analyze Fisher's linear discriminant analysis using a two-class example $y_l \in \{0, 1\}$, which is a 2-D example from Scikit-learn [28]. Because there are two features $d = 2$, the class separating boundary is essentially in a linear line. Figure 1 (a) provides the scatter plot of the source data using blue and red dots, representing the two classes. It also plots the corresponding contours of the conditional distribution, and the linear discriminant line generated by LDA. As for Figure 1 (b), it shows two density plots of the corresponding Gaussian conditional distributions. LDA predicts the label corresponding to the larger of the two densities.

In LDA, the linear discriminant boundary is defined as a linear hyperplane $\boldsymbol{x}^T \boldsymbol{\beta} - \beta_0 = 0$, where the optimal estimators are



(a) (b)

Figure 1. Illustration of LDA analysis (an example from the Scikit-learn): (a) Plots of the source data, contours and LDA discriminant line (b) Two Gaussian conditional distributions corresponding to each class.

$$\hat{\boldsymbol{\beta}} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \tag{3}$$

$$\hat{\boldsymbol{\beta}}_0 = \frac{1}{2}\boldsymbol{\beta}^T(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0) \tag{4}$$

In Fisher's original definition, the coefficients $\boldsymbol{\beta}$ of the separating plane is in proportion to the solution by minimizing the ratio $S(\boldsymbol{\beta})$. It is defined as the between classes variance, divided by the within classes variance.

$$\hat{\boldsymbol{\beta}} \propto \arg\min_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) := \frac{\sigma^2_{between}}{\sigma^2_{within}} = \frac{(\boldsymbol{\beta}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0))^2}{2\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}} \tag{5}$$

As a result, the formula serves as a verification of the optimality of $\hat{\boldsymbol{\beta}}$. If $\hat{\boldsymbol{\beta}}$ is optimal, the gradient of the objective function should satisfy

$$\frac{\partial S(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\Big|_{\hat{\boldsymbol{\beta}}} = 0 \tag{6}$$

3. Lattice-LDA for shape constrained classification

The core component of Lattice-LDA methodology is the lattice based discriminant hyperplane. We construct the hyperplane using an additive model of the 1-D lattice functions, instead of directly using a high dimensional lattice. The optimization algorithm ensures that each of the 1-D function satisfies the corresponding shape constraint.

Recall that in LDA, we have a linear separating hyperplane

$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta} + \beta_0 = \sum_{i=1}^{d} x_i\beta_i + \beta_0 = 0$$

where d is the number of features. The idea of Lattice-LDA is to introduce a nonlinear function $f(\cdot)$, which is defined as an additive form of 1-D nonlinear functions $r_i(x_i)$ of each feature x_i

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} r_i(x_i) + \beta_0 \tag{7}$$

Each function $r_i(x_i)$ is subject to a different shape constraint and, as a result, the linear combination function $f(\cdot)$ also marginally satisfies the shape constraints of feature x_i. Note that, from the marginal convexity or concavity, it is not necessary to imply if the joint effect is convex or concave. In other words, we did not consider the shape information regarding the interactions of features.

In this paper, we mainly study five types of shape constraints, including linear, monotone increasing/decreasing, and convex/concave, as shown in Table 1. In Lattice-LDA, the model is component-wise shape-restricted, which implies each feature x_i satisfies one of the specific shape constraints. These five shapes are simple and fundamental types of shapes, which can be readily expanded to more complex combinations of the shapes such as convex increasing/decreasing or S-shape.

Table 1. Supported shape constraints

Shape #	Shape type	Shape label
1	Linear	l
2	Monotone increasing	in
3	Monotone decreasing	de
4	Convex	cvx
5	Concave	ccv

In the following subsections, we first describe how to construct the 1-D function $r_i(x_i)$ in (7) by applying the lattice transformation, formulate the Lattice-LDA optimization problem, and then solve it using the Adam SGD method.

3.1. Lattice transformation

The concept of lattice transformation is originally introduced in Lattice regression [24, 25]. The lattice regression model essentially interpolates a parametric function on a regular grid of knots. When the function values on the grid knots are defined, the function in the lattice cell is simply calculated via weighted average of the vertex values of the cell. The lattice transformation maps the original data to a higher dimensional sparse matrix of the weights. Instead of processing the source data, the transformed weight matrix effectively becomes the new model input. Google's TFL also adopts the lattice transformation and the 1D lattice function as a fundamental model component.

The lattice transformation is designed in generic high dimensional problems, however, a high dimensional lattice of $(x_1, x_2, \ldots, x_d)$ is expensive to construct. To simplify the approach, we apply a 1-D lattice transformation to each function $r_i(x_i)$ in Equation

(7).[2] Suppose that there are K_i knots in the x_i dimension, where the knots are defined as

$$X_{1,i} \leq X_{2,i} \leq \ldots \leq X_{K_i,i}$$

$X_{1,i}$ and $X_{K_i,i}$ represent to the minimum and maximum bounds of the domain of x_i. The location of the knots is customizable but is typically set as (equal spaced) quantiles of the data and unchanged throughout the model estimation. As a result, each 1-D lattice of x_i is completely defined using K_i knot values or K_i parameters.

3.1.1. Piecewise linear approximation of $r(x)$

The 1-D lattice is essentially a piecewise linear function.[3] Without loss of generality, we drop the subscript i in this subsection, reducing the function form to $r(x)$. In this section, we will describe how to approximate the function $r(x)$ using an interpolation of look-up table values on the knots. Figure 2 provides an illustration of the piecewise linear, or 1-D lattice approximation.

The pre-specified K knots $X_1, X_2, \ldots, X_K$ define $K-1$ buckets in the entire domain of x. Suppose x is a data point in one bucket $[X_k, X_{k+1})$, it can be expressed as the weighted average of the precedent and subsequent knots

$$x = w_k X_k + w_{k+1} X_{k+1}$$

where the linear weights are simply computed as

$$w_k = \frac{X_{k+1} - x}{X_{k+1} - X_k} \text{ and } w_{k+1} = 1 - w_k$$

Let a parameter vector $\boldsymbol{\beta}$ be the knot values of the piecewise linear function, or $\beta_k = r(X_k)$. The values of the parameter $\boldsymbol{\beta}$ regulate the shape of the piecewise linear function. The function value of $r(x)$ is therefore the weighted average of the knot values

$$r(x) = w_k \beta_k + w_{k+1} \beta_{k+1}$$

In order for the formula to be consistent across all buckets, we introduce a weight vector $\boldsymbol{\phi}(x)$ of length K for any given x, and include zero weights in the other buckets

$$\boldsymbol{\phi}(x) = (0, 0, \ldots, w_k, w_{k+1}, 0, \ldots, 0) \tag{8}$$

such that

$$x = \boldsymbol{\phi}(x)^T (X_1, X_2, \ldots, X_K)$$

and define the piecewise function $r(x)$ of the entire domain as

$$r(x) = \boldsymbol{\phi}(x)^T \boldsymbol{\beta} \tag{9}$$

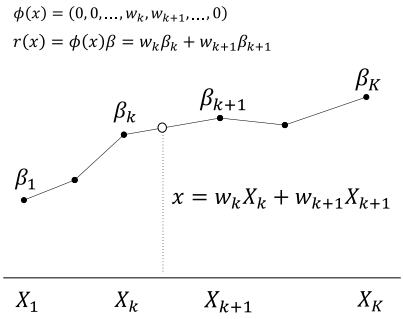


Figure 2. Piecewise linear function construction of $r(x)$ (equivalent to one-dimensional lattice)

The weight vector $\boldsymbol{\phi}(x)$ is the lattice transformation or mapping function.

Note that the vector β defines the look-up table values on each knots $r(X_i) = \beta_i$. To make function $r(x)$ satisfy domain-knowledge shape constraints is essentially achieved by altering the values of β. For example, for a monotone increasing piecewise linear function $r(x)$, $\boldsymbol{\beta}$ needs to be monotone increasing:

$$\beta_k \leq \beta_{k+1}, k = 1, 2, \ldots, K-1$$

The values in $\boldsymbol{\beta}$ can be altered in a similar way such that the vector meets the more complex shape restrictions, including convex or concave shapes. Using the knot values $\boldsymbol{\beta}$ to control the shape of the function is a feature of the lattice transformation.

3.2. The new Lattice-LDA model

We have formulated each shape-restricted 1-D lattice as $r_i(x_i) = \boldsymbol{\phi}_i(x_i)^T \boldsymbol{\beta}^{(i)}$, where $\boldsymbol{\beta}^{(i)}$ are the vectors of knot values defined in Section 3.1.1 (subscript i has been dropped). Without loss of generality, we assume that the first d_1 indexes are linear type shapes, which do not require to a piecewise linear formulation

$$r_i(x_i) = x_i \beta^{(i)} \text{ , or } \phi(x_i) = x_i, i = 1, 2, \ldots, d_1$$

The remaining terms $r_i(x_i), i = d_1 + 1, \ldots, d$ are shape constrained piecewise linear functions. Therefore, when merging all components together, it results the nonlinear hyperplane defined by

$$0 = f(\boldsymbol{x}, \boldsymbol{\beta}, \beta_0) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\beta} + \beta_0, \text{ s.t. } g_i(\boldsymbol{\beta}^{(i)}) \geq 0, i = d_1 + 1, \ldots, d, \tag{10}$$

where $\boldsymbol{\phi}(\boldsymbol{x})$ concatenates all components of $\boldsymbol{\phi}_i(x_i), i = 1, 2, \ldots, d$. $g_i(\cdot) \geq 0$ is a conceptual form of constraint that requires each knot value vector $\boldsymbol{\beta}^{(i)}$ to satisfy the corresponding shape constraints.

[2]We may also introduce a shape restricted terms using 2-D lattice to explicitly capture the pairwise interactions of features. However, the approach is significantly more challenging than just stacking the 1-D lattices, which remains as a future research topic.

[3]In TensorFlow Lattice, the piecewise linear function is referred as a calibrator.

After applying the lattice transformation on input $\boldsymbol{x}$ to $\boldsymbol{\phi}(\boldsymbol{x})$, which is a matrix of weights, the original input with d dimensions is transformed to a much larger dimension: $d_1 + \sum_{i=d_1+1}^{d} K_i$ number of covariates. Given the hyperplane defined by $\{\boldsymbol{x} \,|\, f(\boldsymbol{x}, \boldsymbol{\beta}, \beta_0) = 0\}$, the lattice LDA's objective function is the variance ratio function $S(\boldsymbol{\beta})$, the same as the standard LDA. Lattice-LDA requires additional shape constraints $g_i(\boldsymbol{\beta}^{(i)}) \geq 0, i = d_1 + 1, \ldots, d$.

The optimization problem setup of the new shape-constrained lattice LDA is therefore formulated as a new constrained optimization problem

$$\hat{\boldsymbol{\beta}} \propto \arg\min_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) \tag{11}$$

$$\text{subject to: } g_i(\boldsymbol{\beta}^{(i)}) \geq 0, i = d_1 + 1, \ldots, d$$

3.3. Adam SGD method with stepwise projection

In this subsection, we apply the celebrated Adam SGD method [29] with stepwise projection to solve the Lattice-LDA optimization problem (11). The constraints $g_i(\boldsymbol{\beta}^{(i)}) \geq 0$ are shape constraints specifying the shape of each individual feature. As mentioned before, the Adam method plus projection is also motivated by the algorithm implemented in TFL [26].

Adam is a SGD optimizer that combines features of its two predecessors: AdaGrad and RMSProp. It is a popular optimizer applied in various machine learning tools. Denote the loss function as $L(\boldsymbol{\beta})$. At the tth iteration, the standard gradient descent algorithm updates the iterate

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \alpha \nabla L(\boldsymbol{\beta}_t)$$

where α is the learning rate and $\nabla L(\boldsymbol{\beta}_t)$ is the gradient of the loss function. Due to the randomness of the gradient estimation, instead of directly using the gradient, the Adam optimizer updates the first and second moments m_t and v_t using the following moving average approach:

$$m_{t+1} \leftarrow b_1 m_t + (1 - b_1) \nabla L(\boldsymbol{\beta}_t)$$

$$v_{t+1} \leftarrow b_2 v_t + (1 - b_2)(\nabla L(\boldsymbol{\beta}_t))^2$$

where b_1 and b_2 are exponential decaying rates. The bias-corrected first and second moments are then estimated as follows:

$$\hat{m}_{t+1} = m_{t+1} / (1 - b_1^{t+1})$$

$$\hat{v}_{t+1} = v_{t+1} / (1 - b_2^{t+1})$$

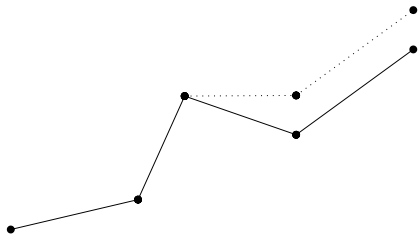


Figure 3. Projection of a non-monotone solution as a monotone increasing solution

Finally, iterate $\boldsymbol{\beta}_{t+1}$ is updated using a dynamic learning rate

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \alpha \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \varepsilon}$$

where ε is a stability control parameter.

In order to satisfy the constraint $g_i(\boldsymbol{\beta}^{(i)}) \geq 0$, we adopt the a shape projection algorithm at each iteration. The new iterate update step is expressed as follows

$$\boldsymbol{\beta}_{t+1} = \text{proj}\left(\boldsymbol{\beta}_t - \alpha \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \varepsilon}\right), \text{ s.t. } g_i(\boldsymbol{\beta}^{(i)}) \geq 0 \tag{12}$$

There are two basic types of projection algorithms for the monotone shapes and convex/concave shapes:

- Monotone increasing or decreasing types
  Figure 3 illustrates an example of the projection of an increasing type. To transform a shape constraint into a monotone increasing shape, consider a given vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_K)$ and simply compute the step difference between the values $\Delta\beta_k = \beta_{k+1} - \beta_k, k = 1, 2, \ldots, K-1$. Then, convert the difference values to nonnegative (or nonpositive) and recalculate the vector based on the first value, $\hat{\boldsymbol{\beta}} = \text{proj}(\boldsymbol{\beta})$ where

  $$\hat{\beta}_k = \beta_1 + \sum_{i=1}^{k-1} \min / \max(\Delta\beta_i, 0)$$

  "max" should be applied in monotone increasing types and "min" in monotone decreasing types.
- Convex or concave types
  The algorithm that projects a vector as convex (or concave) is based on Dykstra's alternating projections algorithm. In addition to the knot value difference $\Delta\beta_k$, define the step difference of the knot locations as $\Delta X_k = X_{k+1} - X_k, k = 1, 2, \ldots, K-1$. The new $\hat{\beta}$ is alternatively updated in a set of two steps

  $$\Delta\hat{\beta}_k = \min / \max\left(\Delta\beta_k, \frac{\Delta X_k(\Delta X_k + \Delta X_{k+1})}{\Delta\beta_k + \Delta\beta_{k+1}}\right), k = 1, 3, \ldots$$

$$\Delta\hat{\beta}_k = \max / \min\left(\Delta\beta_k, \frac{\Delta X_{k+1}(\Delta X_k + \Delta X_{k+1})}{\Delta\beta_k + \Delta\beta_{k+1}}\right), k = 2, 4, \ldots$$

Use "min and max" for the convex type and "max and min" for the concave type. That way, the project value becomes $\hat{\boldsymbol{\beta}} = \text{proj}(\boldsymbol{\beta})$ which is calculated using the following formula

$$\hat{\beta}_k = \beta_1 + \sum_{i=1}^{k-1} \Delta\hat{\beta}_k$$

Note that several rounds of value updates are required when applying Dykstra's alternating projection algorithm. As pointed out by the authors in TFL [26], Dykstra's algorithm would give proper projection with respect to L2 norm but approaches it from "wrong" side, so it would need to do approximate projections which project strictly into feasible space. To move towards the feasible region defined by the constraints, the default number of projection times of Dykstra's alternative algorithm is set to 8.

From our experience, we observe that around 10 rounds of the Dykstra's updates would provide good approximations with respect to the convex or concave shape constraints. In the optimization process, we set a large value (50) as the maximum number of rounds of updates in the following simulation and real data analysis.

4. Simulation studies

In this section, we will assess the performance of the Lattice-LDA classifier using three simulation examples. Our main interest is to examine the model's ability to recover underlying marginal effect functions and the model goodness of fit. For illustration purposes, the target attribute y comprises two labels $\{-1, 1\}$. The marginal effect functions, or the responses of y to each input variable x, are subject to shape constraints. The Lattice-LDA classifier is implemented in MATLAB version R2019b, on a 2.70-GHz Intel Core i7-10850H CPU with 32-GB RAM running Windows 10.

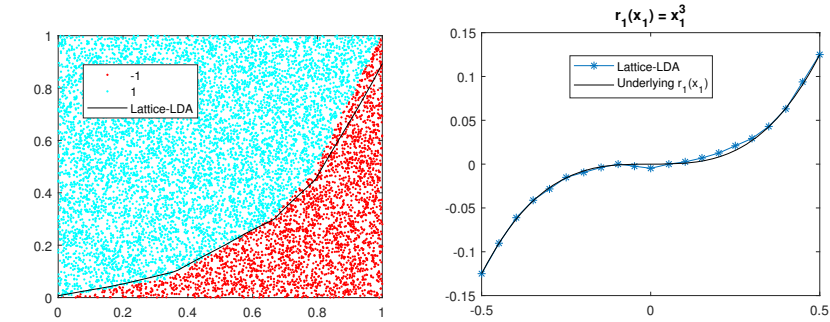
Table 2 summarizes the simulation settings and hyperparameters for Lattice-LDA model in the three examples. We use 19 equally-spaced percentiles, which are the 0.05 to 0.95 percentiles, plus the minimum and maximum points to be knot set for each feature.

Table 2. Simulation settings for examples 1-3

#	n	x distribution	$r(x)$	Shape type	Shape label
1	5,000	$x_1 \sim$ Unif(0, 1)	NA	linear	l
		$x_2 \sim$ Unif(0, 1)	NA	concave	ccv
2	50,000	$x_1 \sim$ Unif(-0.5, 0.5)	x_1^3	S-shape	ccv, cvx
3	50,000	$x_1 \sim$ Unif(-0.5, 0.5)	$2x_1^2$	convex	cvx
		$x_2 \sim$ Unif(0, 1)	$\exp(x_2) - 1$	convex	cvx
		$x_3 \sim$ Unif(0, 1)	$\log(x_3)/4 + 2$	concave	ccv

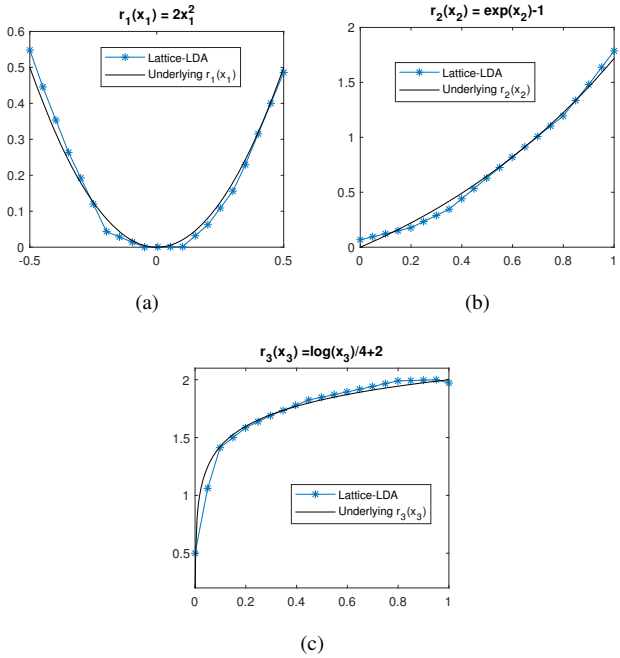
1. A convex increasing function.
   The first experiment is a 2-D example in a box domain. Both x_1 and x_2 are sampled from the standard uniform distribution in $(0,1)$. We then separate the data into two groups by a convex increasing curve, as shown in Figure 4 (a). Even the underlying curve is "convex increasing", we only specify shape constraints in the model as x_1 = "l" (linear), and x_2 = "ccv" (concave), and the setting meets our purpose. Figure 4 (a) also plots the reconstructed piecewise linear discriminant curve, a convex piecewise linear function. As observed, the approximation of the underlying nonparametric curve is precise in the interval (0.2, 0.8), where the samples are dense, but less accurate the edges of the x domain.
2. A concave to convex S-shape
   The second example is a 1-D example with x_1 generated from a uniform distribution in $(-0.5, 0.5)$. y is from a binomial distribution with the probability p to be an S-shape function $p = r_1(x_1) = x_1^3$. We also normalize the value p so that the range is $(0,1)$. In order to handle the S-shape function, we break x_1 into two additive components, where the $x_{1,pos} = \max(x_1, 0)$, and $x_{1,neg} = \min(x_1, 0)$ such that the probability p can be expressed as $p = r_1(x_1) = r_{1,pos}(x_{1,pos}) + r_{1,neg}(x_{1,neg})$. Both the new functions $r_{1,pos}$ and $r_{1,neg}$ are modeled in Lattice-LDA, and the specified shapes for the $x_{1,neg}$ and $x_{1,neg}$ are "ccv" (concave) and "cvx" (convex). Figure 4 (b) shows the combined piecewise linear approximation result, where two curve components are scaled and concatenated. It can be observed that Lattice-LDA is able to approximate the actual functional form $r_1(x_1) = x_1^3$ precisely.
3. An additive model
   In the third experiment, we generate a 3-D example, with x_1, x_2, x_3 from the uniform distributions in Table 2. The probability p to generate the binary response y is defined by an additive model $p = \sum_{j=1}^{3} r_j(x_j)$, where the underlying functions $r_1(x_1) = 2x_1^2$, $r_2(x_2) = \exp(x_2) - 1$, and $r_3(x_3) = \log(x_3)/4 + 2$. Similar to the previous experiment, we also standardize the vector p_i so that the range is (0, 1). The specified shapes for x_1, x_2, x_3 according to the underlying functions are "cvx" (convex), "cvx" (convex) and "ccv" (concave).
   Figure 5 presents the reconstructed piecewise linear functions for the three features. Again, we scale the model fitted curve for each feature to have the same range as the actual functions. Based on the results, it shows Lattice-LDA approximates the two convex functions well. For the concave function $r_3(x_3)$, the approximation is slightly less accurate when in the interval $x_3 < 0.1$, this is due to the fact that the knots are spare in that data range.

In the three examples, Lattice-LDA can recover the underlying parametric or nonparametric functions with high accuracy, yielding better classification than conventional linear classifier. It is noted that increasing the sample size or using more dense selection of knots would improve the goodness of model fit, but it would also require the optimization process to take a longer time (and more iterations), which may cause overfitting in the data. Based on our experience, 10 to 20 knots are sufficient for the Lattice-LDA method to maintain the classification accuracy.



(a) (b)

Figure 4. (a) Example 1: A convex increasing separating line; (b) Piecewise linear spline approximation $\hat{r}_1(x_1)$ vs. actual $r_1(x_1) = x_1^3$ for Example 2.



(a) (b)

(c)

Figure 5. Piecewise linear spline approximation $\hat{r}_i(x_i)$ vs. actual $r_i(x_i)$ for Example 3 (a) $r_1(x_1) = 2x_1^2$; (b) $r_2(x_2) = \exp(x_2) - 1$; (c) $r_3(x_3) = \log(x_3)/4 + 2$;

5. Real world data analysis

In the following, we compare Lattice-LDA with a set of classifiers using several real examples. We will compare Lattice-LDA regarding predicting accuracy with the following classifiers: standard LDA, SVM (with Gaussian kernel), Classification tree, Classification tree by applying Adaptive Boosting [30], and PM-SVM [31]. We are also interested in recovering approximation functions of each feature, which could help better under-

stand the underlying relationship between outputs and features. Note that in the examples, the PM-SVM method was applied with a Randomised Conjunctive (CJ1) algorithm to generate the constraint set [17]. We sourced the data from the UCI Machine Learning Repository [32]. Summarized statistics of the data that we used are given in Table 3. Note that entries with missing values were not included in the analysis.

Table 3. Description of data sets used in experiments.

Data set	# Obs	# Features	# Original Classes	Source
Pima	768	8	2	UCI
Wisc Breast Cancer (WBC)	683	9	2	UCI

We have applied 10-fold cross-validations on each dataset. In each run, it randomly selects 90% data as training sample, remaining 10% data are used as test sample. For a fair comparison, we used the same training/test and CV partitions for each of the classifiers. To measure the model performance, we used average accuracy, which equals one minus the mean value of mis-classification rate. The mis-classification rate is calculated using a 0/1 loss function in the cross validations. We reported the model outputs, including the average accuracy and the standard deviation, in Table 4. Note that for the tree method with Adaptive Boosting, we implemented 100 learning cycles for each training set.

From Table 4, Lattice-LDA has better performance than all the other classification methods regarding the average accuracy results. In addition, we have observed that Lattice-LDA not only gives higher accuracy than standard LDA and SVM (including PM-SVM), but also more precise than the tree methods, which are known as typical nonlinear classifiers.

Table 4. Classification performance based on average accuracy and standard deviation of 0/1 loss.

Data set	Lattice-LDA	LDA	SVM (Gaussian)	Tree	Tree (AdaBoost)	PM-SVM
Pima	0.7748	0.7721	0.6707	0.7058	0.7566	0.7748
	$\pm$ 0.0488	$\pm$ 0.0369	$\pm$ 0.0579	$\pm$ 0.0460	$\pm$ 0.0488	$\pm$ 0.0428
WBC	0.9693	0.9590	0.9664	0.9267	0.9605	0.9605
	$\pm$ 0.0262	$\pm$0.0299	$\pm$ 0.0257	$\pm$ 0.0354	$\pm$ 0.0217	$\pm$ 0.0256

Figure 6 shows the bar charts of all eight features in the Pima (Pima Indians Diabetes) example. We have assigned 1 (orange) to the decision attribute "class = test positive", and -1 (blue) to "class = test positive". Overall, about 35% of the patients are tested positive to diabetes. We break each feature into 10 equal width buckets, and the bucket numbers shown in the figure present the right end point of the bucket.

Figure 7 presents the marginal shape approximation functions $r_i(x_i)$ in the Pima data based on the proposed Lattice-LDA. As one can see, Pregnancies, BMI have convex shapes, Glucose, Skin Thickness and Diabetes Predigree Function have linear increasing shapes, and Blood Pressure and Insulin have linear decreasing shapes, which are consistent with the trend in the bar charts. One interesting feature is Age, which exhibits a

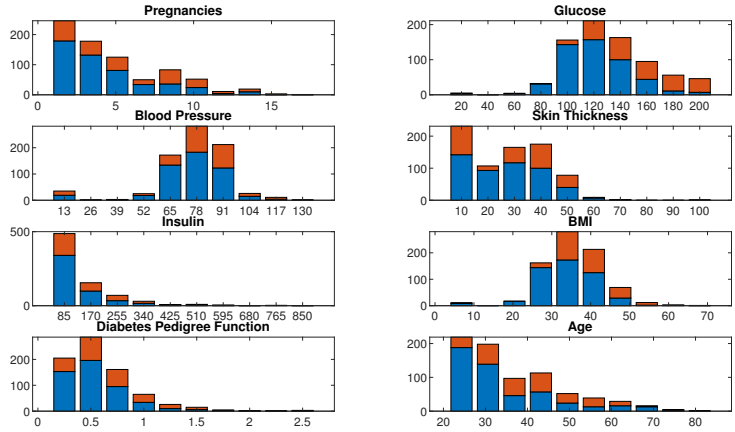


Figure 6. Bar plot of each feature in the Pima example: Orange: "test positive" (1), Blue: "test negative" (-1).

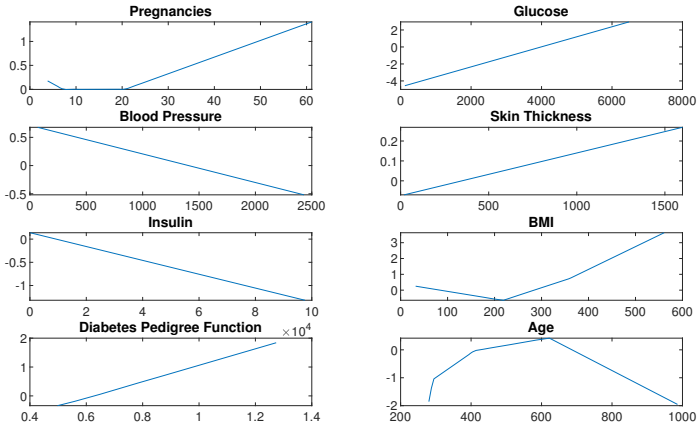


Figure 7. Plot of each marginal approximation function $\hat{r}_i(x_i)$ in the Pima example

concave shape, this is due to the fact that majority of the data have age 70 or less, and the Lattice-LDA methodology is affected more by outliers in the dataset.

For the second example, Figure 8 shows the bar charts of all nine features in the Wisconsin Breast Cancer (WBC) results. We assigned 1 (orange) to the decision attribute "class = malignant", and -1 (blue) to "class = benign". Overall, around 35% of the attributes are 1.

Figure 9 shows the marginal shape approximation functions $r_i(x_i)$ in the WBC example using Lattice-LDA. According to the results, all features reflect a monotone increasing or convex shapes, which is consistent with the trend in the bar charts. More specifically, Clump Thickness has a convex shape, whereas Cell Shape Uniformity, Bland Chromatin and Normal Nucleoli reflect increasing shapes. As for the rest of the features, they all consist of linear increasing shapes.

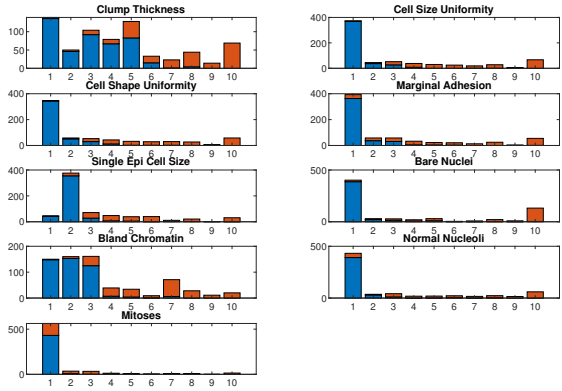


Figure 8. Bar plot of each feature in the WBC example: Orange: "malignant" (1), Blue: "benign" (-1).

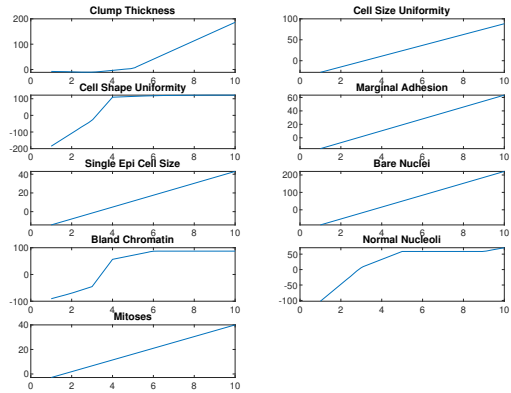


Figure 9. Plot of each marginal approximation function $\hat{r}_i(x_i)$ in the WBC example

From the real data analysis, one may also concern the model interpretability, which is another advantage of the Lattice-LDA method. Lattice-LDA model accommodates nonlinear relationship among inputs features, thus, evaluating the monotonicity and convexity/concavity of the features is intuitive, performed simply by checking the fitted marginal effect function $r_i(x_i)$. According to the results in the WBC example as shown in Figure 9, the Cell Shape Uniformity feature not only shows an increasing trend, but also produces a steeper effect when the value is between three and four than the less than three region, and shows more flattened effect when the value is greater than four. In addition, by using more knots, one can fit the shape constrained features more precisely. However, it requires more computational resources when the knots are increasing, and it may also yield an overfitting to data. Based on our analysis, 10 to 20 knots are sufficient for the Lattice-LDA method to maintain the classification accuracy.

Overall, from both the simulation and real data analysis, it supports the idea that, one can improve predicting accuracy by incorporating prior knowledge of shape information

into the classifiers. It could also help to give better interpretability on input features.

6. Conclusion

In this research, we have introduced a novel Lattice-LDA classifier that captures the shape-restricted constraints of input features. It provides an intuitive way to formulate and interpret classification problems. We have applied the 1-D lattice transformation to each input feature, and approximated the marginal function by piecewise linear functions with given set of knots. Solving the shape restricted Lattice-LDA model utilizes the popular Adam SGD and customizable projection algorithms to map to solution to be feasible.

The benefit of using shape information of inputs is evidenced in the classification accuracy. Setting proper shape constraint is an effective regularization approach to correcting model bias and generating robust predictions. For the sake of the analysis, we have presented simulation and real-world examples to evaluate the Lattice-LDA performance. Based on the results, the shape constraints regularize the model response, preventing the model from overfitting. The marginal function also helps interpret the responses of each feature.

Finally, all the shape information is built into Lattice-LDA throughout the input dimensions. Lattice-LDA does not consider any interaction effects or enforce joint shape constraints of the features. Adding higher dimensional lattice to the model remains a future research direction. There are other model enhancement yet to be researched - one direction is to introduce additional smoothing regularization control of the lattice.

References

[1] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[2] J. Ramón Cano, P.A. Gutiérrez, B. Krawczyk, M. Wozniak, and S. García. Monotonic classification: an overview on algorithms, performance measures and data sets. *Arxiv.org*, abs/1811.07155, 2018. URL http://arxiv.org/abs/1811.07155.

[3] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.

[4] Trevor Hastie, Robert Tibshirani, and Andreas Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American statistical association*, 89:1255–1270, 1994.

[5] Jun Shao, Yazhen Wang, Xinwei Deng, and Sijian Wang. Sparse linear discriminant analysis by thresholding for high dimensional data. *The Annals of statistics*, 39(2): 1241–1265, 2011.

[6] Sheng Wang, Jianfeng Lu, Xingjian Gu, Haishun Du, and Jingyu Yang. Semi-supervised linear discriminant analysis for dimension reduction and classification. *Pattern Recognition*, 57:179–189, 2016.

[7] Fujin Zhong and Jiashu Zhang. Linear discriminant analysis based on l1-norm maximization. *IEEE Transactions on Image Processing*, 22(8):3018–3027, 2013.

[8] Jie Wen, Xiaozhao Fang, Jinrong Cui, Lunke Fei, Ke Yan, Yan Chen, and Yong Xu. Robust sparse linear discriminant analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 29:390–403, 2018.

[9] Simon JD Prince and James H Elder. Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[10] Lin Wu, Chunhua Shen, and Anton Van Den Hengel. Deep linear discriminant analysis on fisher networks: A hybrid architecture for person re-identification. *Pattern Recognition*, 65:238–250, 2017.

[11] Peng Hu, Dezhong Peng, Yongsheng Sang, and Yong Xiang. Multi-view linear discriminant analysis network. *IEEE Transactions on Image Processing*, 28(11): 5352–5365, 2019.

[12] Bernhard Scholkopft and Klaus-Robert Mullert. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pages 41–48, 1999.

[13] Volker Roth and Volker Steinhage. Nonlinear discriminant analysis using kernel functions. In *NIPS*, volume 12, pages 568–574, 1999.

[14] Cheong Hee Park and Haesun Park. Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition. *SIAM journal on matrix analysis and applications*, 27:87–102, 2005.

[15] W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. *ECML/PKDD Lecture Notes in Computer Science*, 5211:301–316, 2008.

[16] Y. Qian, H. Xu, J. Liang, and J.Wang B. Liu. Fusing monotonic decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 27(10):2717–2728, 2015.

[17] C. Chen and S.T. Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[18] H. Daniels and M. Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

[19] M.J. Best and N. Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47:425–439, 1990.

[20] A. Ben-David, L. Sterling, and Y.H. Pao. Learning and classification of monotonic ordinal concepts. *Comput. Intell.*, 5:45–49, 1989.

[21] P. Gutierrez, M. Perez-Ortiz, J. Sanchez-Monedero, F. Fernandez-Navarro, and C. Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146, 2015.

[22] W. Kotlowski and R. Slowiński. On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2576–2589, 2012.

[23] R. Potharst and A.J. Feelders. Classification trees for problems with monotonicity constraints. *ACM SIGKDD Explorations Newsletter*, 4(1):1–10, 2002.

[24] E. K. Garcia and M. R. Gupta. Lattice regression. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2009.

[25] E. K. Garcia, R. Arora, and M. R. Gupta. Optimized regression for efficient function evaluation. *IEEE Trans. Image Processing*, 21(9):4128–4140, 2012.

[26] Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander van Esbroeck. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, 17(109):1–47, 2016.

[27] R. C. Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society, Series B*, 10(2):159–203, 1948.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[30] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[31] C. Bartley, W. Liu, and M. Reynolds. Effective monotone knowledge integration in kernel support vector machine. In *Proceedings of the 12th international conference on advanced data mining and applications*, pages 3–18. Springer, 2016.

[32] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.