Nowcasting U.S. Business Cycle Turning Points with Vector Quantization

by

Andrea Giusto
Dalhousie University

and

Jeremy Piger
University of Oregon

Working Paper No. 2013-02

October 2013

DEPARTMENT OF ECONOMICS

DALHOUSIE UNIVERSITY
6214 University Avenue
PO Box 15000
Halifax, Nova Scotia, CANADA
B3H 4R2

Nowcasting U.S. Business Cycle Turning Points with Vector Quantization[*]

Andrea Giusto[†] Jeremy Piger[‡]

September 2013

Abstract: We propose a non-parametric classification algorithm known as Learning Vector Quantization (LVQ) for the purpose of identifying new U.S. business cycle turning points in real time. LVQ is widely used for classification in many other contexts, including production quality monitoring and voice recognition. The algorithm is well suited for real-time classification of economic data to expansion and recession regimes due to its ability to easily incorporate missing data and a large number of economic indicators, both of which are features of the real-time environment. It is also computationally simple to implement as compared to popular parametric alternatives. We present Monte Carlo evidence demonstrating the potential advantages of the LVQ algorithm over a misspecified parametric statistical model. We then evaluate the real-time ability of the algorithm to quickly and accurately establish new business cycle turning points in the United States over the past five NBER recessions. The algorithm's performance is competitive with commonly used alternatives.

Keywords: turning point, classification, reference cycle, expansion, recession

JEL Classification Numbers: C14, C53, E32, E37

[*]We thank seminar participants at Dalhousie University for helpful comments.
[†]Department of Economics, Dalhousie University, (andrea.giusto@dal.ca)
[‡]Department of Economics, University of Oregon, (jpiger@uoregon.edu)

1 Introduction

A traditional view of the U.S. business cycle is that of alternating phases of expansion and recession, where expansions correspond to widespread, persistent growth in economic activity, and recessions consist of widespread, relatively rapid, decline in economic activity. Timely identification of the turning points between these phases, referred to as peaks and troughs, is of considerable importance to policymakers, financial markets, firms, and individuals. A substantial literature has focused on the prediction of future turning points using a variety of leading economic and financial time series, with some limited success.[1] In these studies, the problem is to use data measured for month t and earlier to predict whether month $t + h$ will be a peak or a trough. A smaller literature has focused on the $h = 0$ horizon, so that some coincident data is used to predict turning points. Such predictions are referred to as "nowcasts".[2]

The problem of nowcasting business cycle turning points is of considerable interest because turning points are often not identified until many months after they occur. The official chronology of business cycle turning points in the United States is provided by the National Bureau of Economic Research's (NBER) Business Cycle Dating Committee, which has historically announced new turning points with a lag of between 5-19 months. Statistical models improve on the NBER's timeliness considerably, with little difference in the timing of the turning point dates established.[3] However, these models still generally identify turning points only after several months have passed. As a recent example of this, Hamilton (2011) surveys a wide range of statistical models that were in place to nowcast business cycle turning points, and finds that such models did not send a definitive signal regarding the December 2007 NBER peak until late 2008.[4] One important reason for these identification lags are data

[1]For a recent summary of this literature, see Katayama (2008).
[2]The prefix "now" refers to the use of data measured for month t to evaluate whether month t is a recession or expansion month. Note that because of data reporting lags, a nowcaster will usually be doing her analysis regarding month t in a month subsequent to month t.
[3]See, e.g., Chauvet and Piger (2008) and Chauvet and Hamilton (2006).
[4]The NBER didn't announce the December 2007 peak until December 1, 2008.

reporting lags, as many key coincident indicators are only released after a 1-2 month lag. Another factor is the need for negative or positive data to accumulate over several months before a definitive turning point signal is uncovered.

The essence of real-time turning point identification is a problem of classification. Given a set of observations on economic indicators, we wish to determine which of two "classes" these observations came from, where the classes are expansion and recession. Much of the literature interested in nowcasting business cycle turning points has focused on parametric statistical models to link the observed data to the classes. For example, Chauvet (1998) proposes a dynamic factor model with Markov-switching (DFMS) to identify expansion and recession phases from a group of coincident indicators, and Chauvet and Hamilton (2006) and Chauvet and Piger (2008) evaluate the performance of variants of this DFMS model to identify NBER turning points in real time. Fossati (2012) alternatively evaluates the real-time nowcasting performance of a dynamic probit specification linking NBER expansion and recession phase indicators to observed data. If the nowcaster knows the true data generating process (DGP) linking the observed data to the classes, then such parametric models allow for the construction of an optimal Bayesian classifier of a month's data as belonging to either the expansion or recession class.

However, in the absence of knowledge of the true DGP, non-parametric methods may be more robust, as they do not rely on a specification of the DGP. Of particular interest are non-parametric classification algorithms based on artificial neural networks, which have been successfully utilized for real-time classification in a large number of existing studies outside of economics. Somewhat surprisingly however, application of these algorithms to the problem of real-time classification of macroeconomic data to expansion or recession phases, and the resulting identification of business cycle turning points, are rare. The only study we are aware of in this regard is Qi (2001), who considers the ability of an artificial neural network to produce out-of-sample forecasts of U.S. business cycle turning points from 1972 to 1995. However, this study does not consider nowcasts, nor does it incorporate real-time, unrevised, data when forming out-of-sample predictions.[5]

In this paper, we use a non-parametric classification algorithm known as Learning Vector Quantization (LVQ) to classify economic indicators as arising from either expansion or recession regimes in real time.[6] LVQ is defined as a special case of an artificial neural network, and is a simple and flexible approach to project a high dimensional number of potentially nonlinear relationships onto a lower dimensional array. It is widely used in real-time classification problems in a number of fields and applications, including production quality monitoring, power load forecasting, speech recognition, odor classification, intrusion detection, beer grading, and email spam detection. LVQ takes both historical data and its classification as an input, which is then used to train the algorithm. For this historical classification we use the available NBER chronology. Based on this training, new data that has not yet been classified by the NBER is then labeled as arising from the expansion or recession regime, which provides real-time tracking of new business cycle turning points. Our focus is on the ability of the algorithm to produce timely and accurate identification of new NBER business cycle turning points.

In addition to the potential advantages of a non-parametric approach when the DGP is unknown, LVQ has computational advantages over parametric methods that have been frequently used to nowcast business cycles based on a group of coincident indicators. First, the algorithm is very simple to implement, generally only taking seconds of computer time.

[5]A substantial number of studies have used non-parametric algorithms to establish historical, in-sample, chronologies of business cycle turning points. For example, Harding and Pagan (2006) develop a non-parametric algorithm to identify peaks and troughs in individual economic time series, and then aggregate these into a single turning point chronology. They show that this algorithm applied to U.S. data can reasonably replicate the historical chronology provided by the NBER. Stock and Watson (2012) derive non-parametric estimators of turning points and associated sampling distributions. They apply their approach to a large cross section of monthly U.S. data to establish turning point dates over the 1959-2010 period. Vishwakarma (1994) uses an algorithm based on an artificial neural network to produce a historical classification of U.S. data into expansion and recession regimes over the period from 1965 to 1989. More recently, Fushing et al. (2010) use the Hierarchical Factor Segmentation algorithm proposed in Fushing et al. (2006), to produce a historical chronology of business cycle dates in both the United States and 22 other OECD countries. The objective of our study differs from these in that we take the historical NBER chronology as given, and focus on identifying turning points in end-of-sample data that has not yet been classified by the NBER.

[6]LVQ algorithms and related extensions are described in detail in Kohonen (2001).

By contrast, estimation of the parameters of the DFMS model or a dynamic probit specification is generally achieved via computationally intensive Bayesian techniques. Frequentist approaches to estimation can also be used, but these require approximations as the exact likelihood function is generally not available for these models. Also, the LVQ algorithm can be easily modified to incorporate data series that arrive with different reporting lags, as well as data series of mixed reporting frequency (e.g. monthly vs. quarterly), both of which are real-world features of the nowcasting environment. Finally, the LVQ algorithm can be implemented when there is a large number of indicators with little increase in computational complexity.

A summary of our empirical analysis is as follows. We first conduct a Monte Carlo study of the potential benefits of using the LVQ algorithm over a misspecified parametric statistical model. This analysis suggests these benefits are potentially large, in that classification accuracy can be significantly improved, particularly in the less frequently occurring classes. We then turn to evaluating the LVQ algorithm's ability to classify U.S. macroeconomic data into expansion and recession phases in real time, with particular emphasis on the timely identification of business cycle turning points. As the data to be classified, we use four monthly coincident series highlighted by the NBER as providing information regarding business cycle phases, and we consider an analyst using the LVQ algorithm in each month from December 1976 to August 2013 to identify new business cycle turning points. Importantly, we use a real-time vintage data set for this evaluation, which accurately reflects the information that would have been available to the analyst in real time. Also, as the economic indicators we consider are released with different reporting lags, we consider nowcasts produced at different points during the month, allowing us to investigate the importance of different pieces of the monthly data flow for assessing business cycle conditions. This updating of inferences about the business cycle regime is simple to implement because of the ease with which unbalanced panels are incorporated into the LVQ algorithm.

The rest of the paper proceeds as follows. Section 2 provides a general description of the LVQ algorithm. Section 3 reports evidence from a Monte Carlo simulation designed to compare the real-time classification performance of the LVQ algorithm to a Bayesian classifier. Section 4 describes the performance of the model in identifying new business cycle turning points. Section 5 concludes.

2 Classification Using Learning Vector Quantization

2.1 Data Classification

The issue of automatic data classification has long been studied in statistics, where the traditional approach is based on the calculation of conditional probabilities. Consider the problem of assigning an m-dimensional real vector $x \in \mathbb{R}^m$ to a class belonging to the set $\{C_k\}_{k=1}^K$. Let $p(C_k)$ denote the *a priori* probability that a vector belongs to class k and let $p(x|x \in C_k)$ be the probability that sample x is observed if it belongs to class C_k. The statistical approach to classification uses the *discriminant functions*

$$\delta_k = p(x|x \in C_k)p(C_k)$$

where pattern x is assigned to class C_k where $C_k = \max_k\{\delta_k\}$. This approach has several theoretical advantages including the property that the probability of misclassification is minimized. From a practical point of view though, the desirable theoretical properties of this classification rule may not be exploitable because of a lack of knowledge of the probabilities that govern the DGP. The usual approach taken in practical statistical pattern classification consists of deriving a good estimate of the discriminant functions over the entire relevant domain of x (or a subset of it that depends on the particular context).

To make things more concrete, consider the data illustrated in panel (a) of Figure 1 which shows two random samples of equal size drawn from two independent bivariate normal distributions. In terms of the notation used above we have that $x \in \mathbb{R}^2$, the set of classes is $\{C_1, C_2\}$, the *a priori* probability of each class is $\frac{1}{2}$, and the conditional probabilities are calculated as follows

$$p(x|x \in C_k) = \frac{1}{2\pi|\Sigma_k|^{1/2}} \exp\Big(-\tfrac{1}{2}(x-\mu_k)'\Sigma_k^{-1}(x-\mu_k)\Big), \quad k = 1, 2, \tag{1}$$

where μ_k and Σ_k denote respectively the mean and variance-covariance matrix of the two bivariate normal distributions. As is clear from Figure 1, this simple classification problem is non-trivial since the two classes have overlapping areas. To classify the samples we define a separation manifold between the two classes determined by the discriminant functions. The separating manifold partitions $\mathbb{R}^m$ into subsets over which each discriminant function is maximal. Panel (b) of Figure 1 shows the separating manifold for these data. Once the boundary between the two regions has been established, it is immediate to build a Bayesian classification system that assigns class 1 to new data points that lie below the curve and class 2 to those points lying above it. Notice that the optimality of this classifier system does not imply perfection in its classification ability, but rather only that the probability of an erroneous classification is minimized.

Vector Quantization (VQ) is an alternative strategy used to obtain a separating manifold in the space of interest, but it is based on a completely different principle than the Bayesian classifier. A VQ classifier is based on the definition of certain key points – called *codebook vectors* – in the data space. Once these relevant points are singled out, data is classified to belong to the same class as the closest codebook vector in the Euclidean metric.[7] A key difference between a Bayesian and a VQ classifier is that while the Bayesian strategy seeks to approximate the discriminant functions over the whole space used to represent the data, the VQ classifier may focus on a small region of this space, if this is where most of the action takes place.

Consider, for example, the situation depicted in Panel (a) of Figure 2: two codebook

[7] The Euclidean metric is one of the many possible metrics that one could use. See Kohonen (2001) for more details.

vectors representing two different classes are plotted respectively with a diamond and a crossed square.[8] New data of unknown classification, then, is classified simply by means of a "nearest neighbor" strategy. For example a hypothetical vector x would be classified to belong to class 1, while vector y would be assigned to class 2. In short, a VQ classifier is based on *quantizing* (i.e. approximating) all the salient features of the data into the codebook vectors, which can be thought of as the most representative points of each class. Just like in the Bayesian classifier, the codebook vectors define a separating manifold in the space $\mathbb{R}^m$ through the midplanes between neighboring pairs of vectors. For the data represented in Figure 1, the codebook vectors may be placed as in panel (b) of Figure 2, where the midplanes between neighboring pairs of vectors would approximate the optimal Bayesian separation curve.

The identification of good codebook vectors may seem difficult without prior knowledge of (or assumptions about) the statistical distributions involved. The algorithms described in the next section solve this problem in a surprisingly simple and computationally light manner.

2.2 LVQ Algorithms

Learning Vector Quantization is an adaptive learning algorithm in which the locations of the codebook vectors are established through adjustments of decreasing magnitude. We will consider two variants of the same basic strategy, called LVQ1 and LVQ2. Let X be a collection of N observations $x_n \in \mathbb{R}^m, n = 1, \ldots, N$ for which the classification in the set $\{C_k\}_{k=1}^{K}$ is known. Let there be $\bar{N} \in [K, N]$ codebook vectors $m_i \in \mathbb{R}^m, i = 1, \ldots, \bar{N}$ with given initial locations. A strategy for establishing the initial locations is discussed in Section 2.4. Finally, let $g = 1, 2, \ldots, G$ denote iterations of the algorithm and let α^g be a decreasing sequence of real numbers bounded between zero and one. Given the initial location of the $\bar{N}$ codebook vectors, the LVQ1 algorithm makes adjustments to their location through these

[8]The definition of a codebook vector includes the location in the relevant space and the class that codebook vector belongs to, which can be read in the legend.

steps:

Algorithm 1 (LVQ1)

1. *Let $g = 1$ and $n = 1$.*

2. *Identify the codebook vector m_c^g closest to the data point x_n*

$$c = \underset{i \in \{1,\ldots,\bar{N}\}}{\operatorname{argmin}} \{||x_n - m_i^g||\}\,.$$

3. *Adjust the location of the codebook vector with index c according to the following rule:*

$$\begin{cases} m_c^{g+1} = m_c^g + \alpha^g [x_n - m_c^g] & \textit{if } x_n \textit{ and } m_c^g \textit{ belong to the same class} \\ m_c^{g+1} = m_c^g - \alpha^g [x_n - m_c^g] & \textit{otherwise} \end{cases}$$

4. *If $n + 1 \leq N$ let $n = n + 1$ and repeat from step 2. Otherwise let $n = 1$ and $g = g + 1$ and if $g \leq G$ repeat from step 2; stop otherwise.*

The LVQ1 algorithm is very simple. At each iteration a data point is selected, and its nearest codebook vector is identified. If this codebook vector agrees with the actual classification of the data point, its location is moved closer to the data point. If the selected codebook vector does not classify the data point correctly, then it is moved farther from the data point. Figure 3 shows a hypothetical example of these two cases. These calculations are repeated for each point in the data set. When the data has all been used, the weight α^g, which controls the size of the adjustment to the codebook vectors, is decreased and a new iteration is started. It is possible to show that this algorithm converges to an approximation of the Bayesian classifier described above, where the precision of the approximation depends on the number of codebook vectors used – see Kohonen (2001) for additional details.

The LVQ2 algorithm represents a variant to LVQ1 that has better classification ability for certain applications. The only difference between LVQ1 and LVQ2 is that while LVQ1 updates one codebook vector at a time, LVQ2 updates two codebook vectors. More specifically, for each data point and at each iteration, one needs to identify the two codebook vectors closest to the data point, such that one classifies this data correctly while the other does not. In terms of the example illustrated in panel (a) of Figure 3, assuming x belongs to the same class as the codebook vector m_i, the two vectors identified by LVQ2 would be m_c (which correctly classifies x) and m_j. The updating takes place similarly, by moving "closer to the data" the codebook vector that classifies the data correctly, and moving "farther from the data" the codebook vector that does not classify it correctly.

2.3 Class Prediction

The LVQ algorithms described in the previous section require a sample of data $x_n \in \mathbb{R}^m, n = 1, \ldots, N$ for which the classification in the set $\{C_k\}_{k=1}^K$ is known. This sample of data is used to produce codebook vectors via the LVQ algorithms, a process typically referred to as "training" the classifier. Once this training is complete, and the codebook vectors established, the classifier can be used to predict the class of new observations for which the true class is not yet known. This process is analogous to the in-sample parameter estimation and out-of-sample prediction steps employed with parametric statistical models.

The specifics of class prediction is as follows. Suppose we have a new data point, x_{N+1}, for which the classification is unknown. We can use the LVQ classifier to predict the class for this data point by first finding the codebook vector m_c that is closest to x_{N+1} in the Euclidean metric:

$$c = \underset{i \in \{1,\ldots,\bar{N}\}}{\operatorname{argmin}} \left\{ ||x_{N+1} - m_i|| \right\},$$

and then assigning x_{N+1} to the same class as is assigned to codebook vector m_c.

In some applications, the new data point to be classified may be only partially observed. This is particularly true for our application to nowcasting business cycle turning points, since relevant data is released at different times and with varying reporting lags. In such cases, class prediction is achieved simply by finding the codebook vector m_c that is closest to x_{N+1} in the Euclidean metric for the elements of x_{N+1} that are observed.

2.4 Initialization of Codebook Vectors Using Self-Organizing Maps

We require $\bar{N}$ codebook vectors to initialize the LVQ algorithms. This initialization could affect the final placement of codebook vectors, and is thus not innocuous. In this section we describe a data-based strategy for the initialization of codebook vectors based on an algorithm known as a self-organizing map (SOM). A SOM is a tool to represent data of dimensionality $m > 2$ on a two dimensional grid of nodes, in a topology-preserving manner.[9] Each node of the SOM is linked to a key point in the m-dimensional space of interest and it is used to represent all the data points that are closest to this key point in the relevant metric. As an example, suppose we have three points $x, y, z \in \mathbb{R}^m$ such that x and y are closer than x and z. Suppose also that point x and y are represented by the nodes A and B respectively in Figure 4. A SOM then will map point z to a node that is not closer to A than node B (for example node C would do.) The similarity between the codebook vectors used in VQ and the key point linked to each node of a SOM is evident. Precisely this similarity allows us to initialize the LVQ algorithms by first constructing a SOM representing the data, and then initializing the codebook vectors at the key points identified by the SOM. A concise description of the SOM algorithm follows.

Let X be a collection of N observations $x_n \in \mathbb{R}^m, n = 1, \ldots, N$, and let $\bar{N} < N$ denote the number of desired nodes in the map. Each node represents a point $m_i \in \mathbb{R}^m$. Let $s = 1, 2, \ldots, S$ denote iterations of the algorithm, and $m_i^s \in \mathbb{R}^m$ is the point attached to node i at iteration s. A SOM representation of the data is then obtained through the application of the following algorithm.

[9] The relevant topology is the one that is generated by the relevant notion of distance used by a particular application. Throughout the paper we always use the Euclidean metric.

Algorithm 2 (SOM)

1. *Set $s = 1$, $n = 1$. Initialize the points assigned to each node by randomly drawing m_i^1, $i = 1, \cdots, \bar{N}$, from X.*

2. *Identify the map's node c that represents the data sample x_n, i.e.*

$$c = \underset{i \in \{1,\ldots,\bar{N}\}}{\operatorname{argmin}} \{||x_n - m_i^s||\}$$

3. *Adjust the points attached to each of the nodes using the following formula*

$$m_i^{s+1} = m_i^s + h^s(c,i)[x_n - m_i^s], \ i = 1, \ldots, \bar{N},$$

   where $\{h^s(c,i)\}_{s=1}^S$ is a sequence of positive "neighborhood functions," defined over the map's nodes and satisfying the following conditions: (1) $h^s(c,i) \to 0$ as $s \to \infty$; (2) $h^s(c,i)$ is non-increasing in the distance between nodes c and i.

4. *If $n + 1 \leq N$ let $n = n + 1$ and repeat from step 2. Otherwise let $n = 1$ and $s = s + 1$ and if $s \leq S$ repeat from step 2; stop otherwise.*

Although the LVQ and SOM algorithms appear very similar, there is a fundamental difference between them.[10] Unlike the LVQ algorithm, SOMs do not use any information on the classification of the data (hence the adjective "self-organizing"). Consequently, a SOM does not attach a class to its key points in $\mathbb{R}^m$. However, the LVQ algorithms requires that a class be attached to the initial codebook vectors. Thus, before the SOM key points can be used to initialize the codebook vectors for the LVQ algorithms, we must assign classes to the nodes of the SOM. There are at least two approaches to achieve this. First one might map the data on the SOM and then assign to each node the same classification as the majority

[10]This similarity originates from a common source of inspiration. The adaptive adjustment strategy used by these algorithms is originally inspired by the biology of the human brain, and both are examples of artificial neural networks. See Kohonen (2001) for details.

of the data points it represents. Alternatively one can first split the data according to the classification information, and then run the SOM algorithm separately for each class. We follow this second strategy in the empirical analysis that follows.

A SOM representation of the data is not unique. Specifically, depending on the random location of the initial points in Step 1 of the SOM algorithm, one may obtain different maps representing the same data. As it turns out, some iterations of Algorithm 2 lead to more useful maps than others, and this poses the problem of automatically selecting "good" maps. We deal with this issue in two ways. First, it is intuitively obvious that a SOM that maps the available data to only a small set of the available nodes is inefficient, as many nodes of such a map are unused. Maps that suffer from such inefficiency tend to be characterized by a "large" average distance of the key points in $\mathbb{R}^m$ that each node represents. For this reason, a widely accepted rule-of-thumb for the design of good SOMs consists of ensuring that the distance between any two key points in $\mathbb{R}^m$ is smaller than half the standard deviation of the data. We employ this heuristic in the empirical analysis below. Second, for the analysis of real-time nowcasting of business cycle turning points, we base our empirical results on multiple runs of the LVQ algorithm with SOM initialization, where the SOM algorithm is initialized with different random vectors for each run. We then choose a single class prediction based on the prevalence of the class predictions across these multiple runs. Additional details are provided in Section 4.2 below.

2.5 Implementation of LVQ with SOM Initialization

In the above algorithms, there are several settings that must be chosen for implementation. These include the number of nodes used in the SOM, $\bar{N}$, which in turn determines the number of codebook vectors in the LVQ, the specification of the neighborhood function in the SOM, $h^s(c,i)$, the form of the weight function, α^g, the number of algorithm iterations, G and S, and the choice of LVQ1 or LVQ2. In our implementation, we focus our analysis on the LVQ1 algorithm. We then calibrate the other settings according to the recommendations given in Kohonen (2001). Specifically, we use the R package “kohonen,” with associated commands “olvq1”, “lvq1” and “som” to implement the LVQ1 and SOM algorithms. For additional details, see Kohonen (2001).

3 Monte-Carlo Analysis of Robustness

Because the LVQ classifier requires no assumption about the probability distributions of the classes of interest, it is robust to misspecification of these distributions. This is a potentially important advantage over the Bayesian classifier, the performance of which is dependent on the validity of the assumed model specification. The possibility of misspecification is especially salient for nowcasting and forecasting macroeconomic conditions, where structural changes in the economic environment create an important source of model misspecification.[11]

Because the relative performance of the two classifiers depends on many factors – including the nature of any misspecification for the Bayesian classifier, as well as the correlation structure between the data points generated by the two different classes – it is not possible to say whether LVQ will outperform a misspecified Bayesian classifier under general conditions. In this section, we instead use Monte Carlo simulations to document the relative performance of the two classifiers for the specific task of determining whether simulated macroeconomic data comes from an expansion or recession phase. We perform experiments both where the Bayesian classifier is based on a correctly specified model, and is thus optimal, and where the Bayesian classifier is based on a misspecified model.

[11] Timmermann (2006) discusses the importance of structural breaks in generating forecast failures in macroeconomic forecasting models. In the specific context of dating business cycles, Kim and Nelson (1999) show that the autoregressive Markov-switching model of Hamilton (1989) must be modified to account for structural breaks before it is able to accurately identify U.S. recession and expansion phases after 1984.

3.1 Data Generating Processes for Simulated Data

In the Monte Carlo experiments we consider the case of four time series random variables, whose time t values are collected in the vector x_t, $t = 1, \ldots, T$. We assume that x_t is generated from one of two classes, labeled expansion and recession. The class in operation at time t is indexed by S_t, where $S_t = 0$ denotes expansion and $S_t = 1$ denotes recession. We consider two alternative DGPs for the expansion and recession classes. For the first, labeled DGP1, x_t is *i.i.d.* Gaussian in both classes, and the classes evolve independently across time:

DGP1:

$$\begin{aligned} x_t | (S_t = 0) &\sim i.i.d.\ N(\mu_0, \Sigma_0) \\ x_t | (S_t = 1) &\sim i.i.d.\ N(\mu_1, \Sigma_1) \\ S_t &\sim Bernoulli(p). \end{aligned}$$

For the second, labeled DGP2, x_t has an i.i.d. multivariate t distribution within classes:

DGP2:

$$\begin{aligned} x_t | (S_t = 0) &\sim i.i.d.\ t(\mu_0, \Sigma_0, v) \\ x_t | (S_t = 1) &\sim i.i.d.\ t(\mu_1, \Sigma_1, v) \\ S_t &\sim Bernoulli(p). \end{aligned}$$

We calibrate these DGPs to four monthly coincident series that are highlighted by the NBER in their decisions about the timing of U.S. business cycle turning points. These are: 1) growth in non-farm payroll employment (E), 2) growth in the industrial production index (I), 3) growth in real personal income excluding transfer receipts (P), and 4) growth in real manufacturing and trade sales (M). We collect monthly data on these four time series from February 1967 to July 2013, and use the NBER's recession chronology over this sample period to define expansion and recession months. We set (μ_0, Σ_0) and (μ_1, Σ_1) to the sample mean and contemporaneous sample variance-covariance matrix of these series during NBER expansion and recession phases respectively. The parameter p is set to the proportion of recession months in the sample, and the degree of freedom parameter in DGP2 (v) is set to 3.

3.2 Classification

For each of the DGPs discussed above, we apply two classifiers, a Bayesian classifier and the LVQ1 classifier with SOM initialization. In evaluating the discriminant function for the Bayesian classifier, we assume that DGP1 generated the data in all cases. Thus, for those simulations where DGP2 is the true DGP, the Bayesian classifier is misspecified.

For each Monte Carlo experiment, we generate 10,000 samples of x_t of size $T = 200$ from one of the DGPs given above. For each of these samples, the first $T_1 = 100$ data points are used to estimate the parameters required to apply the Bayesian classifier, and as a training sample to obtain codebook vectors for the LVQ algorithm.[12] Over these 100 periods the class indicator S_t is taken as known. The class of the second $T_2 = 100$ data points is then predicted by the two classifiers.

We report the performance of the Bayesian and LVQ classifier in the Monte Carlo simulations using the confusion matrix, which gives the rates of correct and incorrect classification of each class. We also report the Matthews Correlation Coefficient (MCC) as a summary measure of classifier performance, defined as:

$$\textbf{MCC:} \quad \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

where TP and TN are the number of correct classifications of $S_t = 1$ and $S_t = 0$ respectively,

[12]As the assumed DGP for the Bayeisan classifier is DGP1, it requires an estimate of μ_i and Σ_i, $i = 0, 1$. These are taken as the sample mean and variance of the observations from each class.

and FP and FN are the number of incorrect classifications of $S_t = 1$ and $S_t = 0$. The MCC ranges between -1 and 1, with 1 indicating perfect classification ability and 0 indicating classification ability no better than random classification.[13] It is preferred to other summary measures, such as the proportion of correct classifications, when one class occurs more frequently than the other.

3.3 Results

Table 1 gives the results of the Monte Carlo experiment where DGP1 generates the data. Again, in this baseline case, the Bayesian classifier is based on a correctly specified DGP, and is the optimal classifier. As the confusion matrix shows, in this case the Bayesian classifier correctly classifies 96.7% and 56.8% of expansion and recession months respectively. The LVQ classifier does worse than the Bayesian classifier for classifying expansions (86.2%) but is better at classifying recessions (66.7%). Overall, the MCC prefers the Bayesian classifier.

Table 2 shows the results of the Monte Carlo experiment when DGP2 generates the data. Again, in this case the Bayesian classifier is misspecified in that it assumes a Gaussian distribution inside of each class, when the true distribution is multivariate t with three degrees of freedom. These results show that the deterioration of the Bayesian classifier is substantial for this form of misspecification. In particular, the Bayesian classifier does far worse at classifying recessions under DGP2, with the rate of correct classification falling substantially to only 35.9%. In contrast, the LVQ classifier appears much more robust to the change in the data generating process, with its rate of correct classification in recessions falling only a few percentage points, and remaining above 60%. The relative deterioration of the Bayesian classifier is reflected in the MCC, which now ranks the two classifiers as essentially equal. Thus, the advantage of the Bayesian classifier over LVQ disappears completely due to the misspecification introduced by moving from DGP1 to DGP2.

[13]An MCC of -1 indicates complete disagreement between the classifier and the actual classes. Of course, such a classifier could be inverted to obtain perfect classification.

4 Real-Time Identification of Turning Points

In this section we evaluate the performance of the LVQ1 classifier with SOM initialization for real-time identification of new U.S. business cycle turning points over a period from December 1976 to August 2013, a period over which there have been five complete NBER recession episodes, and thus five peaks and troughs. We evaluate both the accuracy and speed with which turning points are identified, with the eventual NBER chronology serving as the standard for accuracy.

4.1 Data to be Classified

The data series we use as inputs to the algorithm are the same monthly series as those used to calibrate the Monte Carlo experiments above, namely: 1) growth in non-farm payroll employment (E), 2) growth in the industrial production index (I), 3) growth in real personal income less transfer receipts (P), and 4) growth in real manufacturing and trade sales (M). Again, these series are the four monthly series highlighted by the NBER in their decisions regarding U.S. business cycle classification. Also, they are the focus of the existing literature interested in nowcasting U.S. business cycle turning points.[14]. The LVQ algorithm can easily be extended to consider a larger number of indicators, with little increase in computational complexity. However, to maintain comparability with studies using alternative approaches to identify turning points, we restrict our attention to these four series.

We consider an analyst applying the LVQ algorithm in real time each month from December 1976 to August 2013. To accurately replicate the data that would have been available to this analyst, we use a dataset containing each release, or vintage, of these four coincident variables over each month in our evaluation period. Specifically, we collect a time series for each variable as it would have appeared as of its monthly release, beginning with data released in December 1976, and ending with data released in August 2013. For each vintage, the sample collected begins in January 1959 and ends with the most recent data available

[14]See, e.g., Chauvet and Piger (2008), Camacho et al. (2012), and Fossati (2012)

for that vintage. To collect this vintage dataset, we begin with the dataset used in Chauvet and Piger (2008), which was collected partly from the Federal Reserve Bank of Philadelphia real-time data archive described in Croushore and Stark (2001), and partly from past releases of government statistical agencies. This dataset was then extended to include data for the recent "Great Recession" and beyond using data releases from the Bureau of Economic Analysis and the Federal Reserve Bank of St. Louis ALFRED database.

NBER recession and expansion phases are persistent, with the estimated probability over the 1967-2013 sample of remaining in an expansion equal to 0.98, and the corresponding probability for recessions equal to 0.9. To incorporate the information this persistence provides into the classification problem, we classify a vector of data that contains both current and one lag of the monthly series mentioned above. The vector to be classified is then:

$$x_t = (E_t, I_t, P_t, M_t, E_{t-1}, I_{t-1}, P_{t-1}, M_{t-1})',$$

4.2 Real-Time Classification Environment

The real-time classification environment is as follows. We assume an analyst applies the LVQ algorithm at three dates in each month to a sample of data that would have been available as of that date. The analyst is attempting to assess whether a new business cycle turning point has occurred in the recent past, and is doing so during month $T+1$. The analyst first forms her inferences immediately following the release of the employment series for month T, which occurs on the first Friday of month $T+1$. At this point, the analyst would have E available for month T, I and P available for month $T-1$, and M available for month $T-2$. We refer to this as a beginning of month (BOM) nowcast. We next consider the analyst forming inferences immediately following the release of the industrial production series for month T, which occurs near the middle of month $T+1$. We refer to this as a middle of month (MOM) nowcast. Finally, we consider the analyst forming inferences immediately following the release of the personal income series for month T and the manufacturing and trade sales series for month $T-1$, which occurs at the end of month $T+1$. We refer to this as an end of month (EOM) nowcast. At each of these points, the analyst would be able to form an inference about the business cycle phase in operation from the beginning of the sample period to month T. However, the analyst will have differing amounts of information available to form this inference.

For each of the BOM, MOM, and EOM timing, the LVQ classifier is trained on a sample that extends from the beginning of the sample through month $T-j$, over which the NBER classification is assumed known. In our analysis, j varies across data vintages, and is set using the following assumptions: 1) The date of a new peak or trough is assumed to be known once it is announced by the NBER. 2) A new peak will be announced by the NBER within twelve months of it occurring, where twelve months is the longest historical lag the NBER has taken in announcing a new business cycle peak.[15] This assumption allows us to update the knowledge of the NBER phase during long expansions, despite the lack of any official NBER announcements over these periods. 3) Once the date of a new turning point is announced by the NBER, the new NBER business cycle phase (expansion or recession) is assumed to last at least six months.[16] After training, the analyst would then attempt to assess whether a new turning point has occurred over the period from $T-j+1$ through T. That is, over this period the classification is assumed unknown, and the analyst uses the LVQ classifier to predict the classification.

As an example of the classification environment, suppose an analyst is forming an inference during December 1980, so that period T is November 1980. At this point, the most recent NBER announcement was made on June 3, 1980, and was an announcement of a business cycle peak in January 1980. Since we assume this new recession phase will last a minimum of six months, the NBER classification is then known through July 1980, with the period February 1980 through July 1980 classified as recession months. The value of j in this

[15]The NBER announcement dates of past turning points are available at http://nber.org/cycles/main.html.

[16]These assumptions regarding lags of knowledge of the NBER classification are meant to be a reasonable description of reality. However, they are not essential for our results, which are similar for other, more conservative, assumptions.

case is then four months, and the analyst would be searching for a new business cycle trough over the period from August 1980 through November 1980. As a second example, suppose the analyst was doing her analysis in December of 2005, so that period T is November 2005. Here, the most recent NBER announcement was made on July 17, 2003, and announced the November 2001 trough in economic activity. In this case, the value of j would be set to 12 months, with the period December 2001 through November 2004 classified as expansion months. The analyst would then be searching for a new business cycle peak over the period from December 2004 to November 2005.

Note that for the BOM, MOM and EOM nowcasts, the most recent data to be classified will be incomplete. For example, as discussed above, for the BOM nowcast, the end-of-sample data vector, x_T will be as follows:

$$x_T = (E_t, E_{t-1}, I_{t-1}, P_{t-1})'.$$

Here, this vector is missing the time T values of I, P and M, and the time $T-1$ values of M. To classify this vector we follow the procedure outlined in Section 2.3, and classify it according to the closest codebook vector in the dimensions on which x_T is not missing.

To determine whether a new turning point has occurred over the prediction period, we follow Chauvet and Piger (2008) and require three consecutive months classified as a new business cycle phase prior to calling a new turning point.[17] This enforces a prior belief that business cycle phases are persistent, which is consistent with the NBER definition of a business cycle phase. Specifically, if the most recent known NBER classification was a business cycle trough (peak), we identify a new business cycle peak (trough) if the LVQ classifies three consecutive months over the period from $T-j+1$ through T as recession (expansion) months. Once a new turning point is identified, the date of this new turning

[17]In conducting the LVQ classification, we again utilize the R package "kohonen." However, because the outcome of the classifier has a random element due to the random initialization of the nodes in the SOM, we base our predicted classification on twenty realizations of the classifier. We then consider a month to be classified as either an expansion or recession if at least 80% of the classifier runs return this classification.

point is established as the month prior to the string of consecutive months that is classified as belonging to the new business cycle phase. For example, suppose the most recent business cycle turning point identified was a business cycle trough, and months $T-2$ through month T are classified as recession months, but month $T-3$ is classified as an expansion month. In this case, a new business cycle peak would be established as occurring in month $T-3$.[18]

4.3 Results

Table 3 presents the results of the turning point classification experiment, with the top panel showing results for NBER business cycle peaks and the bottom panel showing results for NBER business cycle troughs. The first column of the table shows the monthly dates of NBER turning points, while the remaining columns show the monthly turning point dates established in real time by the LVQ algorithm applied for the BOM, MOM and EOM timing respectively. Comparing the NBER dates to the dates established by the LVQ algorithm then quantifies the accuracy of the dates established by the LVQ algorithm. The number in parenthesis next to each LVQ turning point date shows the number of days it would have taken, following the end of the official NBER turning point month, to establish the LVQ turning point month. This then measures the timeliness of the LVQ algorithm for detecting NBER turning points. Comparing across the BOM, MOM, and EOM columns provides information regarding the added value of new data as it arrives over the month. The table also reports information about the average speed and accuracy of the LVQ algorithm, as well as providing any instances of turning points established by the LVQ algorithm that didn't correspond to NBER turning points.

Beginning with the accuracy of the LVQ algorithm, it is clear from the table that the LVQ algorithm is very accurate at replicating NBER peak and trough dates. For peaks,

[18]This decision rule uses the convention that a business cycle trough represents the last month of a recession and a business cycle peak represents the last month of an expansion. Instead, the NBER represents these turning points as inflection points that don't necessarily belong to either expansion or recession phases. However, previous statistical classification exercises have confirmed that associating peak months with expansions and trough months with recessions provides the best fit (see e.g. Chauvet and Piger (2008)).

the average absolute difference of the turning point date established by the LVQ algorithm and the NBER date is 0.2 for the BOM nowcasts, 0.6 for MOM nowcasts and 0.4 for the EOM nowcasts. For troughs, the LVQ algorithm is less accurate than for peaks, but still very accurate, with the dates established by the LVQ algorithm always within one month of the corresponding NBER dates. Each of the BOM, MOM and EOM nowcasts produce one "false positive" recession episode which begins in late 1991 and ends in early 1992. This episode corresponds to the well documented jobless recovery that followed the 1990-1991 NBER recession.

These results suggest that the LVQ algorithm is an accurate technique to establish the dates of NBER turning points in real time. Having established this, the primary question of interest is the timeliness with which this accurate identification is achieved. Focusing on the BOM column, the LVQ algorithm establishes business cycle peaks with an average delay of 126 days, and business cycle troughs with an average delay of 192 days. This is very competitive relative to documented alternatives. For example, the NBER business cycle dating committee has announced peak and trough dates with an average lag time of 224 days for peaks and 446 days for troughs.

Admittedly, the NBER dating committee is not concerned with timeliness, and has instead historically focused on putting a premium on establishing a correct date. Thus, a more relevant comparison is to alternative statistical approaches used in the literature. To this end, we compare the results from the LVQ algorithm to other statistical approaches in use to identify the beginning and end of the recent 2007-2009 recession, as summarized in Hamilton (2011). To keep the analysis comparable, we focus on alternative approaches that use the same four coincident series that we use here. The LVQ-BOM procedure would have established the business cycle peak for the most recent recession on June 6, 2008, which is 157 days following the end of the official NBER peak month, December 2007. By contrast, Hamilton (2011) reports that other real-time nowcasting models in use at the time did not identify a business cycle peak until the Fall of 2008 or Winter of 2009. Overall, the timeliness of the LVQ algorithm seems promising.

The average lag time for the MOM nowcasts is slightly slower than the BOM nowcasts, while the EOM nowcasts are in turn slower to identify turning points than the MOM nowcasts. In one sense this pattern is to be expected, as the BOM, MOM, and EOM nowcasts are formed at increasingly later points in the month. Note however that this is not mechanically true, as the nowcasts formed later in the month are based on additional information, and thus may identify a turning point that a nowcast formed earlier in the same month would miss. Indeed, there are a few specific turning points detailed in Table 2 for which the EOM nowcast identifies a turning point a few days earlier than the BOM nowcast. However, for the majority of turning points, the BOM nowcasts are more timely. One might expect that the tradeoff for the increased timeliness of the BOM nowcasts would be a loss in accuracy, since by forming nowcasts earlier in the month the analyst uses less data than if they had waited. However, this does not seem to be the case in practice. The average accuracy of the BOM nowcasts are equal to or better than the MOM and EOM nowcasts, and there are no individual turning points where the MOM or EOM nowcast improves on the accuracy of the BOM turning point date. Also, waiting until later in the month to form nowcasts does not eliminate the single false positive identified by the LVQ algorithm. This suggests that the extra data gained by waiting later in month $T+1$ to form nowcasts, which is data on IP and PIX for month T, and data on MTS for month $T-1$, does not provide any noticeable improvement in the performance of the algorithm for establishing business cycle turning points.

5 Conclusion

Non-parametric algorithms are commonly used in many disciplines to classify data as belonging to one of a set of classes. We have proposed a particularly salient algorithm, known as Learning Vector Quantization, for the purposes of classifying economic data into expansion and recession regimes. Of particular interest is the ability of the algorithm to accurately and quickly identify U.S. business cycle turning points in real time.

Using a Monte Carlo simulation, we have demonstrated the potential advantages of the LVQ algorithm relative to a Bayesian classifier when the latter is based on a misspecified data generating process for the observed data. We then evaluate the real-time performance of the LVQ algorithm for identifying business cycle turning points in the United States over the past 35 years and five recessions. The LVQ algorithm identified the timing of all five recessions over this period accurately. Also, the speed at which the LVQ algorithm identified these recessions was promising. For example, the LVQ alorighm would have identified the December 2007 peak in economic activity by early June 2008, several months ahead of the statistical tracking procedures reviewed by Hamilton (2011) as being in use at the time.

Our analysis here applied the LVQ algorithm to four monthly coincident series that have received substantial identification in the literature as tracking variables for business cycle turning points. However, the LVQ algorithm can be applied to larger datasets, such as that in Stock and Watson (2010), with little increase in computational complexity. Datasets with mixed frequency, as in Aruoba et al. (2009), can also be incorporated, as well as datasets that are a mix of leading and coincident variables. We intend to pursue such extensions in future research.

References

Aruoba, S., F. Diebold, and C. Scotti (2009). Real-time measurement of business conditions. *Journal of Business and Economic Statistics 27*, 417–427.

Camacho, M., G. Perez-Quiros, and P. Poncela (2012). Markov-switching dynamic factor models in real time. working paper.

Chauvet, M. (1998). An econometric characterization of business cycle dynamics with factor structure and regime switching. *International Economic Review 39*, 969–996.

Chauvet, M. and J. D. Hamilton (2006). Dating business cycle turning points. In P. R. Costas Milas and D. van Dijk (Eds.), *Nonlinear Time Series Analysis of Business Cycles*, pp. 1–53. Elsevier, North Holland.

Chauvet, M. and J. Piger (2008). A comparison of the real-time performance of business cycle dating methods. *Journal of Business and Economic Statistics 26*(1), 42–49.

Croushore, D. and T. Stark (2001). A real-time data set for macroeconomists. *Journal of Econometrics 105*, 111–130.

Fossati, S. (2012). Dating u.s. business cycle with macro factors. working paper.

Fushing, H., S.-C. Chen, T. Berge, and O. Jorda (2010). A chronology of international business cycles through non-parametric decoding. working paper.

Fushing, H., C.-R. Hwang, T.-K. Lee, Y.-C. Lan, and S.-B. Horng (2006). Exploring and reassembling patterns in female bean weevils cognitive processing networks. *Journal of Theoretical Biology 238*(4), 805–816.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica 57*(2), 357–384. ID: 480697643.

Hamilton, J. D. (2011). Calling recessions in real time. *International Journal of Forecasting 27*(4), 1006–1026.

Harding, D. and A. Pagan (2006). Synchronization of cycles. *Journal of Econometrics 132*, 59–79.

Katayama, M. (2008). Improving recession probability forecasts in the u.s. economy. Working Paper.

Kim, C.-J. and C. R. Nelson (1999). Has the u.s. economy become more stable? a bayesian approach based on a markov-switching model of the business cycle. *Review of Economics and Statistics 81*(4), 608–616.

Kohonen, T. (2001). *Self-Organizing Maps.* Berlin: Spring-Verlag.

Qi, M. (2001). Predicting u.s. recessions with leading indicators via neural network models. *International Journal of Forecasting 17*, 383–401.

Stock, J. H. and M. W. Watson (2010). Indicators for dating business cycles: Cross-history selection and comparisons. *American Economic Review Papers and Proceedings 100*(2), 16–19.

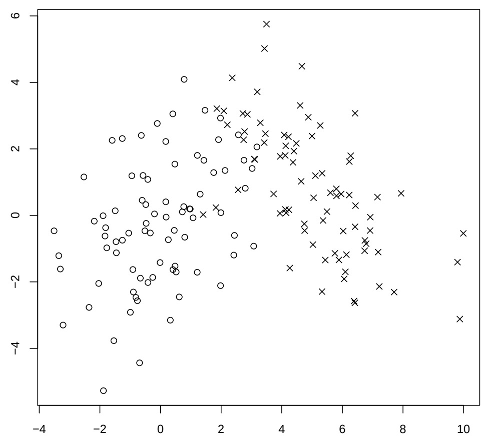
Stock, J. H. and M. W. Watson (2012). Estimating turning points using large data sets. *Journal of Econometrics*, forthcoming.

Timmermann, A. (2006). Forecast combinations. In C. G. Graham Elliott and A. Timmermann (Eds.), *Handbook of Economic Forecasting*, pp. 135–196. Elsevier, North Holland.

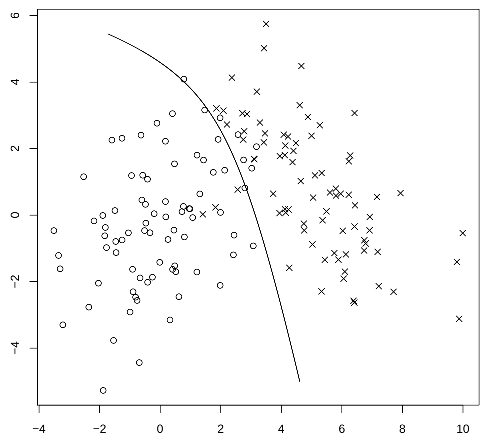
Vishwakarma, K. (1994). Recognizing business cycle turning points by means of a neural network. *Computational Economics 7*, 175–185.

Figure 1

(a)

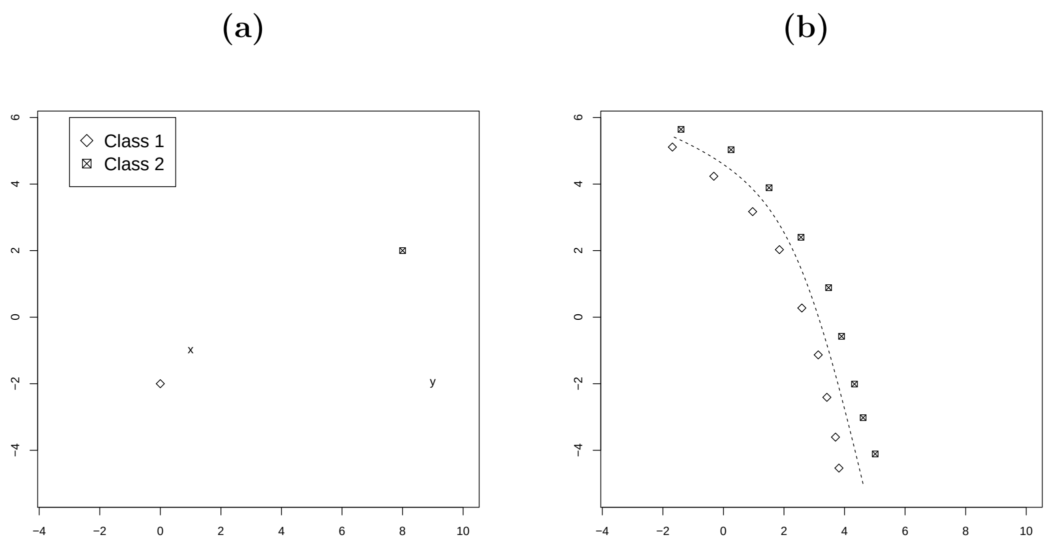


(b)



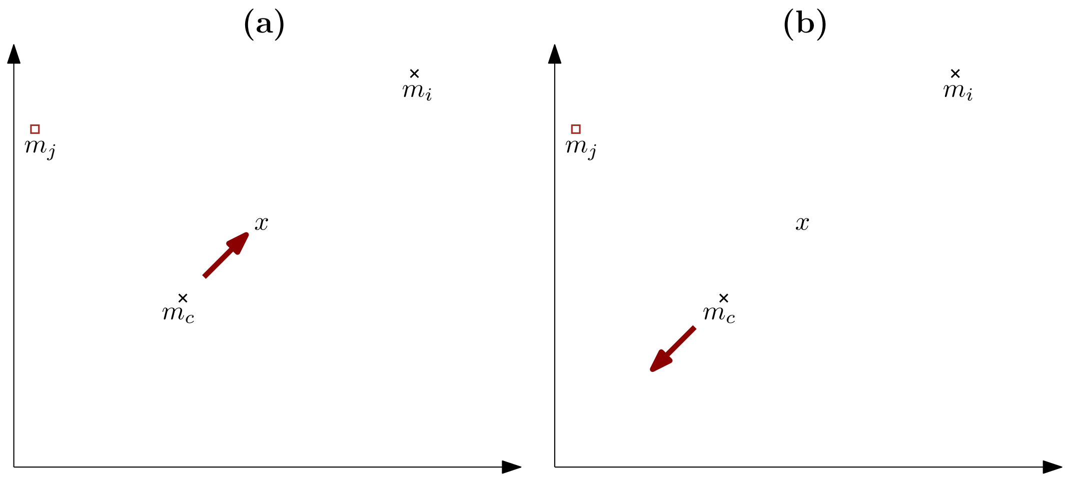
Notes: Bivariate random samples generated from two independent bivariate normal distributions. The data represented with circles has mean $(0, 0)$, variance 3 (along both dimensions), and covariance 1. The data represented with crosses has mean $(5, 1)$, variance 3, and covariance -2. Panel B contains the separation manifold defined by the discriminant function in (1).

Figure 2



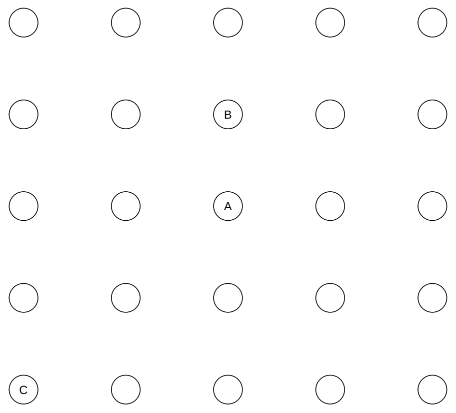
Notes: Panel (a) illustrates the nearest neighbor classification rule. Vector x belongs to class 1, while vector y belongs to class 2. Panel (b) shows hypothetical placement of the codebook vectors for the data shown in Figure 1

Figure 3



Notes: Adjustments to the codebook vector m_c in the two cases in which m_c correctly classifies x (a) or not (b).

Figure 4



Notes: Example of topology of a SOM. Node A is closer to node B than it is to node C. Data that maps to node A is "more similar" to data that maps to node B, than it is to data that maps to node C.

Table 1
Classification Accuracy with Gaussian DGP

	Bayesian Classification		**LVQ Classification**	
	Expansion	Recession	Expansion	Recession
Expansion	96.74%	3.26%	86.17%	13.83%
	(2.41)		(5.89)	
Recession	41.40%	58.60%	33.31%	66.69%
		(14.88)		(13.01)
MCC	0.62		0.48	
	(0.12)		(0.12)	

Notes: This table gives the confusion matrix and the Matthews Correlation Coefficient (MCC) for the Monte Carlo experiment described in Section 3 when the true data generating process is Gaussian inside of each class. In this case the Bayesian classifier is correctly specified. The presample window for estimation and training consists of 100 periods, and the out-of-sample classification window is an additional 100 periods. The average and standard deviation (in parentheses) of the performance metrics over 10000 Monte Carlo repetitions are reported.

Table 2
Classification Accuracy with Student-t DGP

	Bayesian Classfication		**LVQ Classification**	
	Expansion	Recession	Expansion	Recession
Expansion	95.15%	4.85%	81.79%	18.21%
	(3.46)		(6.48)	
Recession	64.13%	35.87%	38.68%	61.32%
		(16.60)		(14.19)
MCC	0.38		0.37	
	(0.16)		(0.12)	

Notes: This table gives the confusion matrix and the Matthews Correlation Coefficient (MCC) for the Monte Carlo experiment described in Section 3 when the true data generating process is multivariate Student-t inside of each class. In this case the Bayesian classifier is misspecified. The presample window for estimation and training consists of 100 periods, and the out-of-sample classification window is an additional 100 periods. The average and standard deviation (in parentheses) of the performance metrics over 10000 Monte Carlo repetitions are reported.

Table 3
U.S. Business Cycle Turning Points Identified in Real-Time

NBER Peak Date	**LVQ-BOM**	**LVQ-MOM**	**LVQ-EOM**
1/1980	1/1980 (126)	1/1980 (133)	1/1980 (152)
7/1981	7/1981 (160)	7/1981 (167)	7/1981 (156)
7/1990	7/1990 (93)	6/1990 (77)	6/1990 (128)
3/2001	3/2001 (96)	3/2001 (107)	3/2001 (121)
12/2007	1/2008 (157)	2/2008 (168)	11/2007 (330)
Average Absolute Error:	0.2 months	0.6 months	0.4 months
Average Days to Identify:	126	130	177
NBER Trough Date	**LVQ-BOM**	**LVQ-MOM**	**LVQ-EOM**
7/1980	7/1980 (126)	7/1980 (137)	8/1980 (157)
11/1982	12/1982 (156)	12/1982 (135)	12/1982 (154)
3/1991	4/1991 (214)	4/1991 (228)	4/1991 (249)
11/2001	12/2001 (307)	12/2001 (320)	12/2001 (303)
6/2009	6/2009 (156)	7/2009 (167)	7/2009 (147)
Average Absolute Error:	0.6 months	0.8 months	1.0 months
Average Days to Identify:	192	197	202
Non-NBER Recessions:	10/91-2/92	10/91-2/92	10/91-2/92

Notes: This table gives NBER-established peak and trough dates for recessions occurring over the sample period November 1976 to April 2010, along with the turning point dates established in real time by the LVQ algorithm over this period. Three versions of the algorithm are reported, one using data available as of the time of the monthly employment release, one using data available as of the time of the monthly industrial production release, and one using data available as of the time of the monthly personal income and manufacturing and trade sales release. These correspond to near the beginning of the month (BOM), middle of the month (MOM), and end of the month (EOM) respectively. The number in parenthesis gives the number of days following the last day of the NBER turning point month for which the LVQ turning point identification would have been available.