Chapter 15
Sparse Representations for Speech Recognition

Tara N. Sainath, Dimitri Kanevsky, David Nahamoo, Bhuvana Ramabhadran and Stephen Wright

Abstract This chapter presents the methods that are currently exploited for sparse optimization in speech. It also demonstrates how sparse representations can be constructed for classification and recognition tasks, and gives an overview of recent results that were obtained with sparse representations.

15.1 Introduction

Sparse representation techniques for machine learning applications have become increasing popular in recent years [1, 2]. Since it is not obvious how to represent speech as a sparse signal, sparse representations have received attention only recently from the speech community [3], where they were proposed originally as a way to enforce exemplar-based representations. Exemplar-based approaches have also found a place in modern speech recognition [4] as an alternative way of modeling observed data. Recent advances in computing power and improvements in machine learning algorithms have made such techniques successful on increasingly complex speech tasks. The goal of exemplar-based modeling is to establish a generalization

T. N. Sainath (✉) · D. Kanevsky · D. Nahamoo · B. Ramabhadran
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
e-mail: tsainath@us.lbm.com

D. Kanevsky
e-mail: kanevsky@us.lbm.com

D. Nahamoo
e-mail: nahamoo@us.lbm.com

B. Ramabhadran
e-mail: bhuvana@us.lbm.com

S. Wright
University of Wisconsin, Madison, WI, USA
e-mail: swright@us.lbm.com

A. Y. Carmi et al. (eds.), *Compressed Sensing & Sparse Filtering*,
Signals and Communication Technology, DOI: 10.1007/978-3-642-38398-4_15,

from the set of observed data such that accurate inference (classification, decision, recognition) can be made about the data yet to be observed the " unseen" data. This approach selects a subset of exemplars from the training data to build a local model for every test sample, in contrast with the standard approach, which uses all available training data to build a model before the test sample is seen.

Exemplar-based methods, including k-nearest neighbors (kNN) [1], support vector machines (SVMs) and sparse representations (SRs) [3], utilize the details of actual training examples when making a classification decision. Since the number of training examples in speech tasks can be very large, such methods commonly use a small number of training examples to characterize a test vector, that is, a *sparse representation*. This approach stands in contrast to such standard regression methods as ridge regression [5], nearest subspace [6], and nearest line [6] techniques, which utilize information about *all* training examples when characterizing a test vector.

An SR classifier can be defined as follows. A dictionary $H = [h_1; h_2 \ldots; h_N]$ is constructed using individual examples of training data, where each $h_i \in Re^m$ is a feature vector belonging to a specific class. H is an over-complete dictionary, in that the number of examples n is much greater than the dimension of each h_i (that is, $m \ll N$). To reconstruct a signal y from H, SR requires that equation $y \approx H\beta$, but imposes a sparseness condition on β, meaning that it requires only small number of examples from H to describe y. A classification decision can be made by looking at the values of β coefficients for columns in H belonging to the same class.

The goal of this chapter is to explain how sparse optimization methods can be exploited in speech, how sparse representation can be constructed for classification and recognition tasks, and to give an overview of results obtained using sparse representation.

15.1.1 Chapter Organization

The remainder of the chapter is organized as follows. The second section deals with mathematical aspects of sparse optimization. We describe two SR methods: approximate Bayesian compressive sensing (ABCS) [7] and convex hull extended Baum-Welch (CHEBW) [8]. We discuss too their relation with the Extended Baum-Welch (EBW) optimization framework [9].

The third section is concerned with a variety of different sparseness techniques employing different types of regularization [2, 3]. Following [10] we explore what type of sparseness regularization should be employed. Typically sparseness methods such as LASSO [11] and Bayesian compressive sensing (BCS) [12] use an l_1 sparseness constraint. Other possibilities include the Elastic Net [13], which uses a combination of an l_1 and l_2 (Gaussian prior) constraint, and ABCS [3], which uses an l_1^2 constraint, known as a Semi-Gaussian prior. We analyze the difference in the spareness objectives for the above methods and we compare the performance of these methods for phonetic classification in TIMIT.

In the fourth section, we explore the application of ABCS to phoneme classification task in TIMIT. The benefit of this Bayesian approach is that it allows us to build compressive sensing (CS) on top of other Bayesian classifiers, for example a Gaussian mixture model (GMM). It was shown, following [3], that the CS technique allows attaining an accuracy of 80.01 %, outperforming the GMM, kNN, and SVM methods.

In the fifth section, we describe a novel exemplar-based technique for classification problems, in which for every new test sample the classification model is re-estimated from a subset of relevant samples of the training data. We formulate the exemplar-based classification paradigm as a SR problem and explore the use of convex hull constraints to enforce both regularization and sparsity. Finally, we utilize the EBW optimization technique to solve the SR problem, and apply our proposed methodology for the TIMIT phonetic classification task, showing statistically significant improvements over common classification methods.

In the sixth section, following [14], we explore the use of exemplar-based SR to map test features into the linear span of training examples. Given these new SR features, we train a Hidden Markov Model (HMM) and perform recognition. On the TIMIT corpus, we show that applying the SR features on top of our best discriminatively trained system yields a reduction in phonetic error rate (PER) from 19.9 % to 19.2 %. In fact, after applying model adaptation we reduce the PER further to **19.0** %, which was the best result on TIMIT reported in 2011. Furthermore, on a large vocabulary 50-h broadcast news task, we achieve a reduction in word error rate (WER) of **0.3** %.

In the seventh section, following [15], we discuss using SRs to create a new set of sparse representation phone identification features (S_{pif}). We describe the S_{pif} features for both small and large vocabulary tasks. On the TIMIT corpus [16], we show that the use of SR in conjunction with our best context-dependent (CD) HMM system allows for a 0.7 % absolute reduction in phonetic error rate (PER), to 23.8 %. Furthermore, on a 50-h Broadcast News task [17], we achieve a reduction in word error rate (WER) of 0.9 − 17.8 %, using the SR features on top of our best discriminatively trained HMM system.

In the eighth section we describe how one can improve sparse exemplar modeling for speech tasks via enhancing exemplar-based posteriors.

15.2 Sparse Optimization

Recent studies have shown that sparse signals can be recovered accurately using fewer observations than the Nyquist/Shannon sampling principle would imply. The emergent theory that brought this insight to light is known as compressive sensing (CS) [22, 23]. Problems of reconstructing signals from compressive sensing data can be represented in several equivalent ways. One such formulation is the following optimization problem:

$$\min_{\beta} \| y - H\beta \|_2 \quad \text{subject to} \quad \| \beta \|_1 \leq \epsilon, \tag{15.1}$$

where y is an m-dimensional vector, x is an N-dimensional vector, H is an $m \times N$ matrix. The parameter ϵ controls the sparsity of the recovered solution. Provided H satisfies certain properties, the signal β can be reconstructed even when the number of observations m is much less than the dimension N of the ambient space in which β resides. In fact, the required number of observations m is related more strongly to the number of nonzeros in β.

This formulation can be generalized to handle other types of sparse and regularized optimization. We can write

$$\min_{\beta} f(\beta) \quad \text{subject to} \quad \phi(\beta) \leq \epsilon, \tag{15.2}$$

where f and ϕ are typically convex functions mapping $\mathbb{R}^n$ to $\mathbb{R}$. Typically, f is a loss function or maximum likelihood function, while the regularization function ϕ is typically nonsmooth, and chosen so as to induce the desired type of structure in β. As noted above, the popular choice $\phi(\beta) = \|\beta\|_1$ induces sparsity into β. An alternative to (15.2) is the following weighted formulation:

$$\min_{\beta} f(\beta) + \lambda\phi(\beta), \tag{15.3}$$

for some parameter $\lambda \geq 0$. It can be shown that (15.2) and (15.3) are equivalent: Under certain assumptions on f and ϕ, the solution of (15.2) for some value of $\epsilon > 0$ is identical to the solution of (15.3) for some value of $\lambda \geq 0$, and vice versa.

We can generalize the formulations (15.2) and (15.3) further by considering nonconvex loss functions f and regularization functions ϕ, and adding an explicit constraint on the values of β. Nonconvex f arise in, for example, deep belief networks, in which the outputs are highly nonconvex functions of the parameters in the network. Nonconvex regularizers ϕ such as SCAP and MCP are sometimes used to avoid biasing effects associated with the use of convex penalties. Explicit constraints such as nonnegativity ($\beta \geq 0$) and simplex ($\beta \geq 0$ and $\sum_{i=1}^{n} \beta_i = 1$) are common in many settings.

Many algorithms have been proposed to solve (15.2) and (15.3), many of which exploit the particular structure of f and ϕ in various applications. One general approach that has been applied successfully in several settings is the *prox-linear* approach in which f in (15.3) is replaced by a linear approximation and a prox-term that discourages the new iterate β^{k+1} from being moved too far from the current iterate β^k. The subproblem to be solved at each iteration is:

$$\beta^{k+1} = \arg\min_{\beta} \nabla f(\beta^k)^T(\beta - \beta^k) + \frac{1}{2\alpha_k}\|\beta - \beta^k\|_2^2 + \lambda\phi(\beta), \tag{15.4}$$

where α_k is a positive parameter that plays the role of a line-search parameter. If the new iterate does not give satisfactory descent in the objective function of (15.3), we can decrease α_k and recompute a more conservative alternative value of β^{k+1}, repeating as necessary.

The approach based on (15.4) is potentially useful when (a) the gradient $\nabla f(\cdot)$ can be computed at reasonable cost and (b) the subproblem (15.4) can be solved efficiently. Both situations typically hold in compressed sensing, under the formulation (15.3) with $f(\beta) = \|H\beta - y\|_2^2$ and $\phi(\cdot) = \|\cdot\|_1$. In this situation, the solution of (15.4) can be computed in $O(n)$ operations.

In the remainder of this chapter, we consider two fundamental methods for sparse optimization: an extended Baum-Welch (EBW) method (which can be expressed via a line-search $\mathcal{A}$-function (LSAF)) and an Approximate Bayesian Compressive Sensing (ABCS) algorithm, which is also closely related to EBW. The LSAF derivation is closely related to the prox-linear approach described above; in fact, the $\mathcal{A}$-function can be thought of as a generalization of the simple quadratic approximation to f that is used in (15.4).

Both EBW and ABCS have been applied to speech classification and recognition problems, as we discuss in subsequent sections.

15.2.1 An EBW Compressed Sensing Algorithm

The Extended Baum-Welch (EBW) technique was introduced initially for estimating the discrete probability parameters of multinomial distribution functions of HMM speech recognition problems under the Maximum Mutual Information discriminative objective function [24]. Later, in [25], EBW was extended to estimating parameters of Gaussian Mixture Models (GMMs) of HMMs under the MMI discriminative function for speech recognition problems. In [9] the EBW technique was generalized to the novel Line Search A-functions (LSAF) optimization technique. A simple geometric proof was provided to show that LSAF recursions result in a growth transformation (that is, the value of the original function increases for the new parameters values). In [26] it was shown that a discrete version of EBW invented in more than 24 years ago can be also represented using $\mathcal{A}$-functions. This connection allowed a convergence proof for a discrete EBW to be developed [26].

15.2.2 Line Search A-Functions

Let $f(x) : \mathcal{U} \subset \mathbb{R}^n \to \mathbb{R}$ be a real valued differentiable function in an open subset $\mathcal{U}$. Let $\mathbf{A}_f = \mathbf{A}_f(x, y) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be twice differentiable in $x \in \mathcal{U}$ for each $y \in \mathcal{U}$. We define $\mathbf{A}_f$ as an $\mathcal{A}$-function for f if the following properties hold.

1. $\mathbf{A}_f(x, y)$ is a strictly convex or strictly concave function of x for any $y \in \mathcal{U}$. (Recall that twice differentiable function is strictly concave or convex over some domain if its Hessian function is positive or negative definite in the domain, respectively.)
2. Hyperplanes tangent to manifolds defined by $z = g_y(x) = \mathbf{A}_f(x, y)$ and $z = f(x)$ at any $x = y \in \mathcal{U}$ are parallel to each other, that is,

$$\nabla_x \mathbf{A}_f(x, y)|_{x=y} = \nabla_x f(x) \tag{15.5}$$

It was shown in [9] that a general optimization technique can be constructed based on $\mathcal{A}$-function. We formulated a growth transformation such that the next step in the parameter update that increases $f(x)$ is obtained as a linear combination of the current parameter values and the value $\tilde{x}$ that optimizes the $\mathcal{A}$-function, for which $\nabla_x \mathbf{A}_f(x, y)|_{x=\tilde{x}} = 0$. More precisely, we stated that $\mathcal{A}$-function gives a set of iterative update rules with the following "growth" property: let x_0 be some point in $\mathcal{U}$ and $\mathcal{U} \ni \tilde{x}_0 \neq x_0$ be a solution of $\nabla_x A(x, x_0)|_{x=\tilde{x}_0} = 0$. Defining

$$x_1 = x(\alpha) = \alpha \tilde{x}_0 + (1 - \alpha)x_0, \tag{15.6}$$

we have for sufficiently small $|\alpha| \neq 0$ that $f(x(\alpha)) > f(x_0)$, where $\alpha > 0$ if $A(x, x_0)$ concave and $\alpha < 0$ if $A(x, x_0)$ convex. The technique of generating $\tilde{x}$ in this way and performing the line search is termed "Line Search A-Function" (LSAF).

15.2.3 Discrete EBW

Here we show that discrete EBW can be described using the LSAF framework. Our descriptiong ins limited to the case of a single distribution, but the technique generalizes readily to several distributions.

Let the simplex $\mathcal{S}$ be defined as

$$\mathcal{S} := \{\beta : \beta \in \mathbb{R}^n, \beta_i \geq 0, i = 1, \ldots n, \sum \beta_i = 1\},$$

and suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function on some subset $X \subset \mathcal{S}$. We wish to solve the following maximization problem for a function $f(\beta)$:

$$\max \ f(\beta) \ \text{ subject to } \ \beta \in \mathcal{S}. \tag{15.7}$$

Let $\beta \in X$ and define $a_i^k := \frac{\partial f(\beta^k)}{\partial \beta_i^k}$, $i = 1, \ldots n$. For any $D \in \mathbb{R}$ and $\beta^k \in \mathbb{R}^n$ such that $\sum_{j=1}^n a_j^k \beta_j^k + D \neq 0$, we define a recursion $T_D : \mathbb{R}^n \to \mathbb{R}^n$ as follows:

$$\beta_i^{k+1} = T_D(\beta^k) = \frac{a_i^k \beta_i^k + D\beta_i^k}{\sum_{j=1}^n a_j^k \beta_j^k + D}. \tag{15.8}$$

It was shown in [27] that for sufficiently large D, we have $f(\beta^{k+1}) > f(\beta^k)$, unless $\beta^{k+1} = \beta^k$.

An $\mathcal{A}$-function $\mathbb{A}_f$ for the function f in (15.7) that is differentiable in some compact neighborhood $\mathcal{U} \subset X$ of a point $\beta_0 \in \mathcal{S}$ is given as:

$$\mathbb{A}_f(\beta_0, \beta) = \sum (c_i + \beta_{0i} D) \log \beta_i, \tag{15.9}$$

where $c_i = c_i(\beta_0) = \beta_{0i} \frac{\partial f(\beta)}{\partial \beta_i}|_{\beta=\beta_0} = \beta_{0i} a_i(\beta_0)$ and D is any number such that $a_i(\beta) + D > 0$ for all i and any $\beta \in \mathcal{U}$. (Existence of D is guaranteed by differentiability of f in $\mathcal{U}$ and compactness of $\mathcal{U}$.) To show that the function $\mathbb{A}_f(\beta_0, \beta)$ in (15.9) is an $\mathcal{A}$-function, one needs to check (15.5) as follows. Replace $\beta_n = 1 - \sum \beta_i$ in (15.7), (15.9), that is, consider the functions $g(\beta') = f(\beta_1, ..., \beta_{n-1}, 1 - \sum_1^{n-1} \beta_i)$, $\mathbb{A}_g(\beta_0; \beta') = \mathbb{A}_f(\beta_0, \{\beta_1, ..., \beta_{n-1}\}, 1 - \sum_1^{n-1} \beta_j)$ where $\beta' = \{\beta_1, ...\beta_{n-1}\}$. We have

$$\frac{\partial \mathbb{A}_g(\beta_0, \beta')}{\partial \beta_i}|_{\beta_i=\beta_{0i}} = a_i(\beta_0) \frac{\partial f(\beta)}{\partial \beta_i}|_{\beta_i=\beta_{0i}} + D\beta_{0i} \frac{\partial \log \beta_i}{\partial \beta_i}|_{\beta_i=\beta_{0i}} +$$
$$D(1 - \sum_1^{n-1} \beta_{0i}) \frac{\partial \log(1 - \sum_1^{n-1} \beta_i)}{\partial \beta_i}|_{\beta=\beta_0} = \frac{\partial g(\beta')}{\partial \beta'}|_{\beta_i'=\beta_{0i}'}.$$

It can be shown that adding a quadratic penalty $C\beta^T\beta$ to the objective function $f(\beta)$ is equivalent to substituting the term D with $D + 2C$ in the discrete EBW recursion (15.8). Moreover, for sufficiently large C, the function $f(\beta) + C\beta^T\beta$ is concave in a simplex $\mathcal{S}$. Therefore, it achieves its maximum on the boundary of the of the simplex $\mathcal{S}$. This fact implies that for sufficiently large D, the EBW recursion enforces a sparse solution.

Discrete EBW methods can be applied to optimization of objective functions with fractional norm constraints, as suggested in [28]. We have

$$\max\ f(\{\beta_i\}) \quad \text{subejct to} \quad \|\beta\|_q = 1 \quad \text{and} \quad \beta_{\mathrm{i}} \geq 0,\ \mathrm{i} = 1, 2, \ldots, \mathrm{n}, \tag{15.10}$$

where $\| \beta \|_q := (\sum \beta_i^q)^{1/q}$. Setting

$$\gamma_i = \beta_i^{1/q}, \quad g(\{\gamma_i\}) = f(\{\beta_i\}), \tag{15.11}$$

transforms the problem (15.10) into a discrete EBW problem for which the recursion (15.8) could be applied. In [26], this optimization method with fractional norm constraints was applied to TIMIT classification tasks.

15.2.4 An ABCS Compressed Sensing Algorithm

Following [29], we describe the approximate Bayesian CS (ABCS) method. The key idea behind this algorithm is based on an approximate sparseness promoting prior which is a sort of mixture of Gaussian and Laplace distributions. ABCS is a variant of the algorithm in [30] and [31]. In what follows we gradually develop this underlying concept and a few others which form the core of the new method.

15.2.4.1 Bayesian Estimation

The Bayesian estimation methodology provides a convenient representation for dealing with complex observation models. In this work, however, we restrict ourselves to the conventional linear model used in CS theory

$$y_k = H\beta + n_k \tag{15.12}$$

where y_k, $H \in \mathbb{R}^{m\times N}$, and n_k denote the kth $\mathbb{R}^m$-valued observation, a fixed sensing matrix, and the observation noise of which the pdf $p(n_k)$ is known, respectively. The sought-after random parameter (the signal) β is a $\mathbb{R}^N$-valued vector for which the prior pdf $p(\beta)$ is given. Following this, the complete statistics of β conditioned on the entire observation set consisting of k elements, $\mathscr{Y}_k = [y_1, \ldots, y_k]$ can be sequentially computed via the Bayesian recursion

$$p(\beta \mid \mathscr{Y}_k) = \frac{p(y_k \mid \beta)p(\beta \mid \mathscr{Y}_{k-1})}{\int p(y_k \mid \beta)p(\beta \mid \mathscr{Y}_{k-1})d\beta} \tag{15.13}$$

where the likelihood $p(y_k \mid \beta) = p_{n_k}(y_k - H\beta)$. One can rarely obtain a closed-form analytic expression of the posterior pdf (15.13), so approximation techniques are often used. One well-known example in which (15.13) does admit a closed-form solution is given by the following theorem, which plays a fundamental role in this work. (This is a well known result in estimation theory which is revisited here for completeness.)

Theorem 1 (Gaussian pdf Update). *Assume that $p(\beta \mid \mathscr{Y}_{k-1})$ is a Gaussian pdf of which the first two statistical moments are given by $\hat{\beta}_{k-1} \in \mathbb{R}^n$ and $P_{k-1} \in \mathbb{R}^{n\times n}$, that is $p(\beta \mid \mathscr{Y}_{k-1}) = \mathcal{N}(\beta \mid \hat{\beta}_{k-1}, P_{k-1})$. Assume also that the observation y_k satisfies the linear model (15.12) where n_k is a $\mathbb{R}^m$-valued zero-mean Gaussian random variable $n_k \sim \mathcal{N}(0, R)$ that is statistically independent of β. Then the Bayesian recursion (15.13) yields $p(\beta \mid \mathscr{Y}_k) = \mathcal{N}(\beta \mid \hat{\beta}_k, P_k)$ where*

$$\hat{\beta}_k = \hat{\beta}_{k-1} + P_{k-1}H^T\left(HP_{k-1}H^T + R\right)^{-1}\left[y_k - H\hat{\beta}_{k-1}\right] \tag{15.14a}$$

$$P_k = \left[I - P_{k-1} H^T \left(H P_{k-1} H^T + R \right)^{-1} H \right] P_{k-1} \tag{15.14b}$$

The initial values of the above quantities are set according to the Gaussian prior $p(\beta) = \mathcal{N}(\beta \mid \hat{\beta}_0, P_0)$.

The proof of this statement can be found in [29]. Note that the quantity P_k in Theorem 1 is the estimation error covariance, i.e.,

$$P_k := E\left[(\beta - \hat{\beta}_k)(\beta - \hat{\beta}_k)^T \mid \mathcal{Y}_k \right]$$

where $\beta - \hat{\beta}_k$ is the estimation error of the unbiased estimator $\hat{\beta}_k$.

15.2.5 Sparseness-Promoting Semi-Gaussian Priors

Compressed sensing was embedded in the framework of Bayesian estimation by utilizing sparseness promoting priors such as Laplace and Cauchy [32]. Here we consider a different type of prior that facilitates the application of the closed-form recursion of Theorem 1. The sparseness-promoting prior used here is termed "semi-Gaussian" (SG) owing to its form

$$p(\beta) = c \exp\left(-\frac{1}{2} \frac{\| \beta \|_1^2}{\sigma^2} \right). \tag{15.15}$$

The motivation for using a SG prior can be motivated by analyzing the characteristics of the SG constraint $\| \beta \|_1^2 = (\sum_i |\beta_i|)^2$ and the Laplacian constraint $\| \beta \|_1 = (\sum_i |\beta_i|)$. We can denote the SG density function as proportional to $p_{semi-gauss} \propto exp(- \| \beta \|_1^2)$ and the Laplacian density function proportional to $p_{laplace} \propto exp(- \| \beta \|_1)$. When $\| \beta \|_1 < 1$, it is straightforward to see that $p_{semi-gauss} > p_{laplace}$. When $\| \beta \|_1 = 1$, the density functions are the same, and when $\| \beta \|_1 > 1$ then $p_{semi-gauss} < p_{laplace}$. Therefore the semi-Gaussian density is more concentrated than the Laplacian density in the convex area inside $\| \beta \|_1 < 1$. Given the sparseness constraint $\| \beta \|_q$, as the fractional norm q goes to 0, the density becomes concentrated at the coordinate axes and the problem of solving for β becomes a non-convex optimization problem where the reconstructed signal has the least mean-squared-error (MSE). Intuitively, we expect the solution using the semi-Gaussian prior to behave closer to the non-convex solution.

This observation is further illustrated in Fig. 15.1, in which the level maps are shown for Laplace, semi-Gaussian, and Gaussian pdfs in the 2-dimensional case. The embedding of the prior (15.15) within the Gaussian variant of the Bayesian recursion in Theorem 1 is not straightforward. This follows from the fact that the restrictions under which Theorem 1 is derived involve a purely Gaussian prior and a

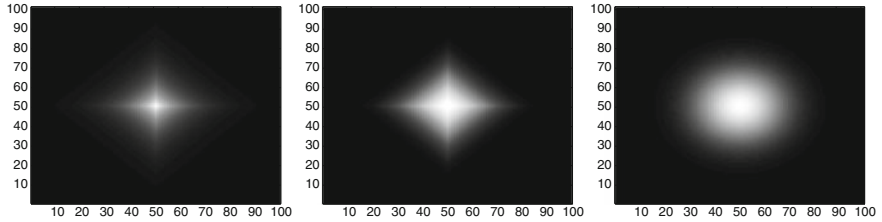

Fig. 15.1 Laplace, semi-Gaussian, and Gaussian pdfs in $\mathbb{R}^2$

likelihood pdf that is based on a deterministic sensing matrix H,

$$p(y_k \mid \beta) \propto \exp\left(-\frac{1}{2}(y_k - H\beta)^T R^{-1}(y_k - H\beta)\right). \tag{15.16}$$

Theorem 1 provides an exact recursion for computing the Gaussian posterior based exclusively on the factors composing the above likelihood: the observation y_k, the sensing matrix H and the observation noise covariance R. This fact has motivated the following approach which allows enforcing an approximate semi-Gaussian prior without changing the fundamental structure of the underlying update equations as obtained in Theorem 1.

15.2.6 Approximate Semi-Gaussian Prior

We introduce a state-dependent matrix $\hat{H} \in \mathbb{R}^{1\times N}$ whose entries are $\hat{H}^i = \text{sign}(\beta^i)$, $i = 1, 2, \ldots, N$ (that is, $\hat{H}^i = +1$ and $\hat{H}^i = -1$ for $\beta^i > 0$ and $\beta^i < 0$, respectively). The semi-Gaussian prior can be expressed based on (15.16) while replacing H and R with $\hat{H}$ and σ, respectively, and assuming a fictitious observation $y = 0$, that is

$$p(\beta) = p(y = 0 \mid \beta, \hat{H}, \sigma) \propto \exp\left(-\frac{1}{2}\frac{(0 - \hat{H}\beta)^2}{\sigma^2}\right) \tag{15.17}$$

The only difficulty in using (15.14a) for enforcing the semi-Gaussian prior (15.17) is the dependency of $\hat{H}$ on β. We recall that Theorem 1 relies on possibly varying a deterministic H as opposed to the formulation in (15.17). This problem can be alleviated by letting

$$\hat{H}^i = \text{sign}(\hat{\beta}_k^i), \quad i = 1, 2, \ldots, N, \tag{15.18}$$

that is, by substituting the conditional mean instead of the actual β. This modification renders $\hat{H}$ a $\mathscr{Y}_k$-measurable quantity, as it depends on $\hat{\beta}_k$ which is a function of the entire observation set. This fact clearly does not affect the expressions in Theorem 1 as the derivations are conditioned on $\mathscr{Y}_k$ (see [29]). Applying this

approximation facilitates the implementation of Theorem 1 based on the likelihood (15.17). Hence, an additional processing stage is needed to apply the approximate sparseness-promoting prior:

$$\hat{\beta}_{k+1} = \left[I - \frac{P_k \hat{H}^T \hat{H}}{\hat{H} P_k \hat{H}^T + \sigma^2} \right] \hat{\beta}_k \tag{15.19a}$$

$$P_{k+1} = \left[I - \frac{P_k \hat{H}^T \hat{H}}{\hat{H} P_k \hat{H}^T + \sigma^2} \right] P_k. \tag{15.19b}$$

This stage is implemented after the usual processing of the observations set $\mathscr{Y}_k$ (see (15.14)), where the initial covariance is taken as $P_0 \to \infty$.

At this point, a natural question is raised concerning the validity of the approximation suggested above. The following theorem, proved in [29], bounds the discrepancy between the exact posterior which uses the semi-Gaussian prior (15.15) and the approximate posterior in terms of the estimation error covariance $\hat{P}_k$.

Theorem 2 *Denote $\hat{p}(\beta \mid \mathscr{Y}_k)$ the Gaussian posterior pdf obtained by using the approximate semi-Gaussian prior technique, and let $p(\beta \mid \mathscr{Y}_k)$ be the posterior pdf obtained by using the exact semi-Gaussian prior (15.15). Then*

$$\mathrm{KL}\left(\hat{p}(\beta \mid \mathscr{Y}_k) \parallel p(\beta \mid \mathscr{Y}_k)\right) = \mathcal{O}\left(\sigma^{-2} \max\left\{\mathrm{Tr}(\hat{P}_k), \mathrm{Tr}(\hat{P}_k)^{1/2}\right\}\right), \tag{15.20}$$

where KL *and* Tr *denote the Kullback-Leibler divergence and the matrix trace operator, respectively.*

In practical applications for speech classification and recognition tasks, it was observed that the classification and recognition accuracy is not affected if ones computes a term P_k in (15.19b) only once, then fixes this term for all subsequent iterations. This trick provides a significant speed up without significant degradation of accuracy.

15.2.7 ABCS Representations via LSAF

We recall the ℓ_1-constrained problem (15.1), modified slightly by the use of a weighted data-fitting term

$$\min \parallel y - H\beta \parallel_R^2 \quad \text{subject to} \quad \parallel \beta \parallel_1 \leq \epsilon.$$

In many practical application it is useful to add an l_2 regularization term to this formulation, to yield

$$\min \parallel y - H\beta \parallel_R^2 + \parallel \beta - \beta_0 \parallel_{P_0}^2 \quad \text{subject to} \quad \parallel \beta \parallel_1 \leq \epsilon.$$

Using $\| y - H\beta \|_R^2 + \| \beta - \beta_0 \|_{P_0}^2 = \| \beta - \beta_1 \|_{P_1}^2$ we can represent this problem as

$$\min \| \beta - \beta_1 \|_{P_1}^2 \quad \text{subject to} \quad \| \beta \|_1 \leq \epsilon,$$

where P_1 is assumed to be positive-definite. We can now represent (15.1) by

$$\min F(\beta) := \| \beta - \beta_1 \|_{P_1}^2 + \| \beta \|_1^i / \sigma^2, \tag{15.21}$$

and define the $\mathcal{A}$-function as:

$$\mathbb{A}(\beta, \beta^*) = \| \beta - \beta^* \|_{P_1}^2 + \{\text{sign}(\beta^*)\beta\}^i / \sigma^2, \tag{15.22}$$

where $i = 1$ (Laplacian) or $i = 2$ (squared l_1 norm). In [26] we show that $\mathbb{A}(\beta, \beta^*)$ is $\mathcal{A}$-function of $F(\beta)$. According to the definition of the $\mathcal{A}$-function, we consider $\mathbb{A}(\beta, \beta^*)$ and $F(\beta)$ in an open domain where they are both differentiable and construct an update of parameters when the extremum of $\mathbb{A}(\beta, \beta^*)$ belongs to this domain. Our open domain excludes the origin $\beta = 0$. If some coordinates of β approach 0 we can remove them by reducing the dimension of the problem. Using LSAF, we have the recursion:

$$\beta_k = \alpha \tilde{\beta}_{k-1} + (1 - \alpha)\beta_{k-1}.$$

The ABCS algorithm corresponds to a squared l_1-norm. Analysis of various regularization penalties for speech classification problems is given in Sect. 15.3. The ABCS method gives a solution of (15.21) via the recursion: $\tilde{\beta}_{k-1} = \arg\max_\beta A(\beta, \beta_{k-1})$. Numerical experiments show that for a suitable choice of α, the parameter β_k converges to a solution of (15.21) more rapidly than the one obtained through the ABCS recursion. One can expect that LSAF with appropriate choices of α is more efficient than the ABCS.

15.3 An Analysis of Sparseness and Regularization in Exemplar-Based Methods for Speech Classification

Following [10] we describe and compare a variety of different sparseness techniques, which employ different types of regularization, and that have been explored for speech tasks [2, 3]. Firstly, we describe the main framework behind exemplar-based classification. Then we give a brief description of the TIMIT corpus. Next we discuss how sparseness can be useful in classification tasks. Finally, we compare the performance of different sparseness methods for classification.

15.3.1 Classification Based on Exemplars

The goal of classification is to use training data from k different classes to determine the best class to assign to test vector y. First, let us consider taking all training examples n_i from class i and concatenate them into a matrix H_i as columns, in other words $H_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}] \in \mathbb{R}^{m \times n_i}$, where $x \in \mathbb{R}^m$ represents a feature vector from the training set of class i with dimension m. Given sufficient training examples from class i, [6] shows that a test sample y from the same class can be represented as a linear combination of the entries in H_i weighted by β, that is:

$$y = \beta_{i,1} x_{i,1} + \beta_{i,2} x_{i,2} + \ldots + \beta_{i,n_i} x_{i,n_i} \tag{15.23}$$

However, since the class membership of y is unknown, we define a matrix H to include training examples from all k classes in the training set, in other words the columns of H are defined as $H = [H_1, H_2, \ldots, H_k] = [x_{1,1}, x_{1,2}, \ldots, x_{k,n_k}] \in \mathbb{R}^{m \times N}$. Here N is the total number of all training examples from all classes. We can then write test vector y as a linear combination of all training examples, in other words $y = H\beta$. We can solve this linear system for β and use information about β to make a classification decision. Specifically, large entries of β should correspond to the entries in H with the same class as y. Thus, one proposed classification decision approach [3] is to compute the l_2 norm for all β entries within a specific class, and choose the class with the largest l_2 norm support.

15.3.2 Exemplar-Based Methods

Various types of exemplar-based classifiers can be cast in the framework of representing the test vector y as a linear combination of training examples H, subject to a constraint on β. Below, we review a few popular techniques that are based on the following optimization problem for various values of q and α

$$\min_{\beta} \| y - H\beta \|_2 \quad \text{s.t.} \quad \| \beta \|_q^{\alpha} \leq \epsilon \tag{15.24}$$

1. Ridge regression (RR) methods [5] use information about all training examples in H to make a classification decision about y, in contrast to a nearest-neighbor (NN) approach to exemplar-based classification, which uses information about just 1 training example. Specifically, the RR method looks to project y into the linear space of all training examples and solves for the β which minimizes (15.24) for $q = 2, \alpha = 2$. The term $\| \beta \|_2^2 \leq \epsilon$ is an l_2 norm on β (i.e. a Gaussian constraint) but does not enforce any sparseness.
2. Sparse representations: like RR methods, sparse representation (SR) techniques (i.e., [3, 6], project y into the linear span of examples in H, but constrain β to be sparse. Specifically, SR methods solve for β by minimizing (15.24), given

various settings for α and q. For example, in a probabilistic setting $q = 1, \alpha = 1$ leads to a Laplacian constraint, whereas $q = 1, \alpha = 2$ leads to a Semi-Gaussian constraint. The remainder of this section is focused on comparing the RR method to various SR methods with different types of regularizations.

15.3.3 Description of TIMIT

We analyze the behavior of various exemplar-based methods on the TIMIT [16] corpus. The corpus contains over 6,300 phonetically rich utterances divided into three sets, namely the training, development, and core test set. For testing purposes, the standard practice is to collapse the 48 trained labels into a smaller set of 39 labels. All methods are tuned on the development set and all experiments are reported on the core test set.

The complete experimental setup, as well as the features used for classification, are similar to [3]. First, we represent each frame in our signal by a 40 dimensional discriminatively trained Space Boosted Maximum Mutual Information (fBMMI) feature. We split each phonetic segment into thirds, taking the average of these frame-level features around 3rds, and splice them together to form a 120 dimensional vector. This allows us to capture time dynamics into each segment. Then, at each segment, segmental feature vectors to the left and right of this segment are joined together and a Linear Discriminative Analysis (LDA) transform is applied to project 200 dimensional feature vector down to 40 dimensions.

Similar to [3], we find a neighborhood of closest points to y in the training set using a kd-tree. These k neighbors become the entries of H. We explore classification performance for different sizes of H. In what follows, we explore the following two questions, using TIMIT to provide experimental results to support our framework.

- Why and when is sparseness important for exemplar-based methods?
- If sparseness is used, what type of regularization constraint should be utilized?

15.3.4 Why Sparse Representations?

We will motivate the difference between the RR and SR methods further with the following example. Let us consider a 2×7 matrix

$$H = [h_1, h_2, h_3, h_4, h_5, h_6, h_7] = \begin{bmatrix} 0.2 & 0.1 & 0.4 & 0.3 & -0.6 & 0.6 & -0.6 \\ 0.2 & 0.3 & 0.35 & 0.3 & 0.1 & 0.3 & 0.4 \end{bmatrix},$$

where first three columns h_1, h_2, h_3 are "training" utterances that belong to a class C_1 and last four columns are "training" utterances that belong C_2. Assume also that a vector $y = [0.29; 0.29]$ is "test" data that belong to a class C_1. thus will

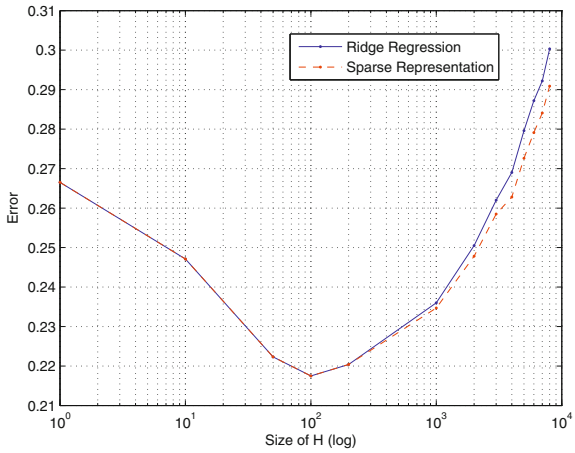

Fig. 15.2 Error for RR and SR methods for varied H

include the outlier points of C_2. Solving (15.24) with $q = 2, \alpha = 2$ (i.e., the RR method) produces the vector $\beta \approx [0.12; 0.15; 0.21; 0.18; -0.05; 0.1220.08]$ and the best class is C_2. However using the SR method in (15.24) (for example, using ABCS method with a SG constraint as explained in Sect. 15.2) produces a vector $\beta \approx [0.00; 0.01; 0.77; 0.00; 0.00; 0.00; 0.03]$ with the support located at the third entry in H. In this case, the C_1 is identified as the correct class. Thus, by using a subset of examples in H, the classification decision for SR and RR can be vastly different, particularly in the case of outliers.

To analyze the behavior of the SR and RR methods in a practical speech example, we explore phonetic classification on TIMIT as the size of H is varied from 1 to 10,000. A plot of the error rate for the two methods for varied H is shown Fig. 15.2. For this figure, we again used the ABCS SR method. First notice that as the size of H increases up to 1,000 the error rates of the RR and SR both decrease, showing the benefit of including multiple training examples when making a classification decision. Also notice that there is no difference in error between the RR and SR techniques, suggesting that regularization does not provide any extra benefit. However, as the size of H increases past 1,000 and there are more number of training examples for each class, the SR method performs better than the RR method, demonstrating the advantage of using sparseness to select only a few examples in H to explain y rather than all examples in H.

15.3.5 What Type of Regularization?

Now that we have motivated the use of regularization, in this section we analyze different forms of regularization. As illustrated by (15.24), with $q = 1$, a sparse representation solution can be formulated by finding the β which minimizes the

residual error $\| y - H\beta \|_2$, subject to a regularization $\| \beta \|_q \leq \epsilon$ on β. There are four common types of regularizations on β.

1. If $q = 2$ and $\alpha = 2$, then the regularization becomes $\| \beta \|_2 \leq \epsilon$. This constraint can be modeled as a Gaussian prior. Common techniques which impose an l_2 constraint on β include Ridge Regression [5]. The effect of the l_2 norm is to spread values of entries in β equally. Therefore the optimization problem (15.24) for $q = 2$ tries to find a balance between keeping the residual $\| y - \beta \|_2$ small and trying to keep all the entries in the vector β to be non-zero.
2. If $q = 1$ and $\alpha = 1$, then the regularization becomes $\| \beta \|_1 \leq \epsilon$. This constraint can be modeled as a Laplacian prior. Common techniques which impose an l_1 constraint on β include LASSO [11] and Bayesian Compressive Sensing (BCS) [12]. The Lasso problem can be formulated as follows:

$$\min_{\beta} \| y - H\beta \|_2 + \lambda \| \beta \|_1, \tag{15.25}$$

   as in (15.3), where λ controls the weight of the l_1 norm. The Least Angle Regression (LARS) ([33]) solves LASSO through a forward stepwise regression, computing point estimates of β at each step. The effect of the l_2 norm is to spread values of entries in β equally. Therefore the optimization problem (15.24) for $q = 2$ tries to find a balance between keeping the residual $\| y - \beta \|_2$ small while at the same time preventing all the entries in β from vanish. In contrast, the norm l_1 tries to enforce sparsity in β while keeping the residual $\| y - H\beta \|_2$ small.
   Bayesian Compressive sensing [12] can be formulated in a fashion similar to (15.25). BCS introduces a probabilistic framework to estimate the spareness parameters required for signal recovery. This technique limits the effort required to tune the sparseness constraint and also provides complete statistics for the estimate of β.
3. Many techniques also impose a combination of an l_1 and l_2 constraint on β. These methods include the popular Elastic Net [13]. The Elastic Net [13] method imposes a mixture of an l_1 and l_2 constraints, i.e.,

$$\min_{\beta} \| y - H\beta \|_2 + \lambda_1 \| \beta \|_1 + \lambda_2 \| \beta \|_2^2 . \tag{15.26}$$

   Here λ_1 and λ_2 are weights controlling the l_1 and l_2 constraint. In the elastic net formulation the l_1 term enforces the sparsity of solution, whereas the l_2 penalty ensures democracy among groups of correlated variables. The second term has also a smoothing effect that stabilizes the obtained solution.
4. The previously described ABCS explores the use of a semi-Gaussian prior and solves for β in a Bayesian framework. The ABCS essentially solves

$$\min_{\beta} \| y - H\beta \|_2 + \lambda_1 (\beta - \beta_0)^T P_0^{-1} (\beta - \beta_0) + \lambda_2 \| \beta \|_1^2 . \tag{15.27}$$

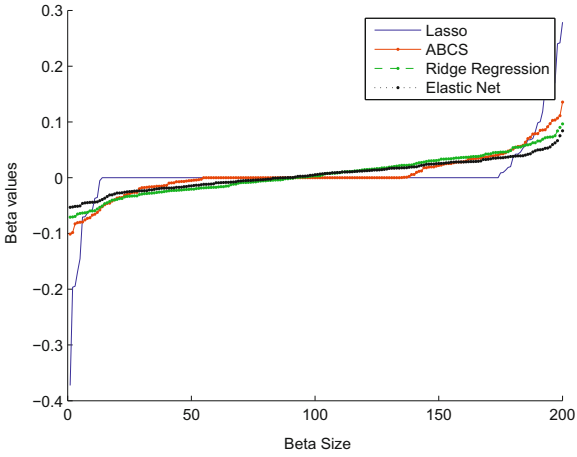

Fig. 15.3 Plot of β for different regularization constraints

Visualization of Sparsity

We analyze the difference in β coefficients for different sparseness methods. For a randomly selected classification frame y in TIMIT and an H of size 200, we solve (15.24) for β. Figure 15.3 plots the sorted 200 β coefficients for four different techniques employing different reguliarizations, namely Ridge Regression, Lasso, Elastic Net and ABCS. The plot shows that the β coefficients for the RR method are the least sparse, as we would expect. In addition, the LASSO technique has the sparsest β values. The sparsity of the Elastic Net and ABCS techniques methods are in between RR and LASSO, with ABCS being more sparse than Elastic Net due to the Semi-Gaussian constraint in ABCS, which is more sparse than the l_1 constraint in the Elastic Net.

TIMIT Results

Table 15.1 shows the results comparing various sparseness methods on TIMIT for a size of $H = 200$. As one can see from the table, the three methods which combine a sparseness constraint with and l_2 norm, namely ABCS, Elastic Net and CSP, all achieve statistically the same accuracy. The two methods which use the l_1 norm, namely BCS and LASSO, have slightly lower accuracy, showing the decrease in accuracy when a high degree of sparseness is enforced. Thus, it appears that using a combination of a sparsity constraint on β, coupled with an l_2 norm, does not force unnecessary sparseness and offers the best performance.

Table 15.1 Accuracies for different sparseness methods

Method	PER
LASSO	74.40
BCS	73.58
Elastic net	**77.89**
ABCS	**77.80**
CSP	**77.55**

15.4 ABCS for Classification

In this section we follow [3] and describe application of ABCS for Timit classification tasks. We perform classification as described in Sect. 15.3.1 solving (15.23) for $\alpha = 2$ and $q = 1$ via (15.14a), (15.14b), (15.19a), and (15.19b). We compute the l_2 norm for all β entries within a specific class and choose the class with the largest l_2 norm support. Pooling together all training data from all classes into H will make the columns of H large (i.e., can be greater than 100,000 for TIMIT), and will make solving for β intractable. Therefore, to reduce the size of N and make ABCS problem more solvable, for each y, we find a neighborhood of closest points to y in the training set using a kd-tree [35]. These k neighbors become the entries of H. k is chosen to be in the large to ensure that β is sparse and all training examples are not chosen from the same class.

Constants P_0 and β_0 must be chosen to initialize the ABCS algorithm. Recall that β_0 and the diagonal elements of P_0 all correspond to a specific class. We choose β_0 to be 0 since we do not have a very confident estimate of β and we assume its sparse around 0 anyways. We choose to initialize a diagonal P_0 where the entries corresponding to a particular class are proportional to the GMM posterior for that class. The intuition behind this is that the larger the initial P_0, the more weight is given to examples in H belonging to this class in ABCS. Therefore, the GMM posterior picks out the most likely supports, and ABCS provies an addition step by using the actual training data to refine these supports.

15.4.1 Nonlinear Compressive Sensing

The traditional CS implementation represents y as a linear combination of samples in H. Many pattern recognition algorithms, such as SVMs [36] have shown better performance can be achieved by a nonlinear mapping of the feature set to a higher dimensional space. After this mapping, a weight vector w is found which projects all dimensions within a particular feature vector to a single dimension where different classes are linearly separable. We can think of this weight vector w as selecting some linear combination of dimensions within a feature vector to make it linearly separable. The goal of CS is to find a linear combination of actual features, not dimensions within a feature vector. Therefore, we introduce nonlinearity into CS, by

constructing H such that the entries of H themselves are nonlinear. For example, one such nonlinearity is to square all the elements within H. That is if we define $H_{lin} = [x_{1,1}, x_{1,2}, \ldots, x_{k,n_k}]$, then H^2 is defined as $H^2 = [x_{1,1}^2, x_{1,2}^2, \ldots, x_{k,n_k}^2]$ and similarly H^3 would take cubed products of each of the x entries. We could also take products between different x_i as $H_{inner} = [x_{1,1}x_{1,2}, x_{1,1}x_{1,3} \ldots, x_{k,8}x_{k,n_k}]$. We then take a specific nonlinear H_{nonlin} and combine it with the linear H_{lin} to form a new $H_{tot} = [H_{lin}, H_{nonlin}]$ and use ABCS to solve for β. In Sect. 5.1 , we discuss the performance of the ABCS algorithm for different choices of nonlinear H.

15.4.2 Experiments

Classification experiments are conducted on TIMIT [16] acoustic phonetic corpus as described in Sect. 15.3.3. First, we analyze the performance of the CS classifier for different choices of linear and nonlinear H as described in Sect. 3.4. Next, we compare the performance of CS with three other standard classifiers used on this task, namely a Gaussian Mixture Model (GMM), Support Vector Machine (SVM) [36] and k-nearest Neighbors (kNN) classifier [35]. The parameters of each classifier were optimized for each feature set on the development set. Specifically, we found that modeling each phone as a 16-component GMM was appropriate. The kernel type and parameters within this kernel were optimized for the SVM. In addition, the number of k closest neighbors for kNN was also learned. And finally, for CS the size of H_{lin} was optimized to be 200 examples from the kd-tree. In addition to compute H_{nonlin}, 100 columns were randomly chosen from H_{lin} to compute each type of nonlinear H.

Performance for Different H

Table 15.2 shows the accuracy on the development set for different choices of H using Mel-frequency cepstral coefficients (MFCC) features. Notice that the nonlinear CS-$H_{lin}H^2$ method offers improvements over the linear CS-H_{lin} method. Taking $H_{lin}H^2H^3$ offers addition improvements, though overtraining occurs when higher order features past H^3 are used. Furthermore, there is very little difference between squaring individual entries of H (i.e. $H_{lin}H^2$) or taking products between differ-

Table 15.2 Accuracy for different H using MFCC features

Method	Dev-MFCC
CS-H_{lin}	76.64
CS-$H_{lin}H^2$	76.84
CS-$H_{lin}H^2H_{inner}$	76.53
CS-$H_{lin}H^2H^3$	**76.89**
CS-$H_{lin}H^2H^3H^4$	76.86

Table 15.3 Accuracy for different classifiers on TIMIT testcore set

Method	MFCC	fBMMI
GMM	74.19	78.31
kNN	73.69	79.58 (=)
SVM	76.20 (=)	78.38
CS-$H_{lin}H^2H^3$	**76.44**	**80.01**

ent entries of H (i.e., $H_{lin}H_{inner}$). While not shown here, similar trends were also observed for fBMMI features. Since the CS-$H_{lin}H^2H^3$ method offers the best performance of the CS methods, we will report the results for this classifier in subsequent sections.

Comparing Different Classifiers

Table 15.3 compares the performance of the CS classifier with the GMM, kNN and SVM methods for both MFCC and fBMMI features. Classifiers which are not statistically significant from the CS classifier, as confirmed by McNemar's Test, are also indicated by '='. First, notice that when MFCC features are used, CS outpeforms both then kNN and GMM methods, and offers similar performance to the SVM. When discriminative features are used, the GMM technique is closely matched to the SVM though CS is able provide further gains over these two methods. This is one of the benefits of CS—a discriminative non-parametric classifier built on top of the GMM.

Analysis of Results

To better understand the gains achieved by the CS classifier compared to the other three techniques, Fig. 15.4 plots the relative difference in error rates within 6 broad phonetic classes (BPCs) for CS compared to the three other methods. First, notice that CS offers improvements over the GMM in all BPCs, again confirming its benefit of a non-parametric discriminative classifier on top of the GMM. Secondly, while the SVM technique offers improvements over the CS method in the vowel/semivowel class, the CS method significantly outperforms the SVM in the weak fricative, stop and closure classes. Finally, the CS method offers slight improvements over the kNN method in the nasal, strong fricative and stop classes, while kNN offers slight improvements in the vowel, weak fricative and closure classes. Thus, we can see that with the exception of the GMM, the gains from CS do not come from it outperforming the kNN and SVM techniques within all BPCs, but only within certain BPCs.

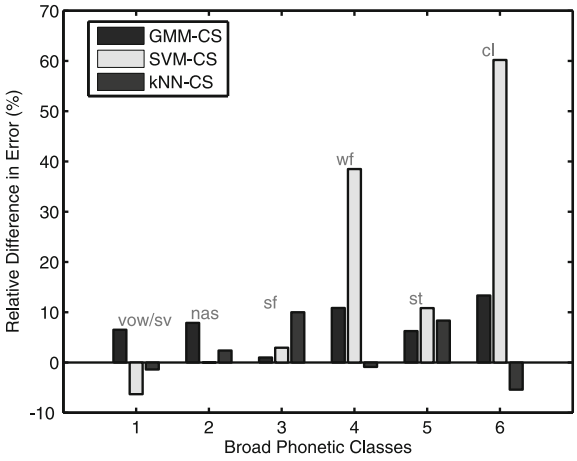

Fig. 15.4 Relative difference in error rates between CS and other methods

15.5 A Convex Hull Approach to Sparse Representations

A typical SR formulation in (15.24) does not constrain β to be positive and normalized, which can result in the projected training points $h_i\beta_i \in H\beta$ being reflected and scaled. While this can be desirable when data variability exists, allowing for too much flexibility when data variability is minimized can reduce the discrimination between classes. Driven by this intuition, below we present two examples where data variability is minimized, and demonstrate how SRs manipulate the feature space, thus leading to classification errors.

First, consider two clusters in a 2-dim space as shown in Fig. 15.5a with sample points $\{a_1, a_2, \ldots, a_6\}$ belonging to Class 1 and $\{b_1, b_2, \ldots, b_6\}$ belonging to Class 2. Assume that points a_i and b_i are concatenated into a matrix $H = [h_1, h_2, \ldots, h_{12}] = [a_1, \ldots, a_6, b_1, \ldots b_6]$, with a specific entry being denoted by $h_i \in H$. In a typical SR problem, given a new point y indicted in Fig. 15.5b, we project y into the linear span of training examples in H by trying to solve:

$$\arg\min \| \beta \|_0 \qquad \texttt{s.t.} \qquad y = H\beta = \sum_{i=1}^{12} h_i\beta_i \tag{15.28}$$

As shown in Fig. 15.5a, the best solution will be obtained by setting all $\beta_i = 0$ except for $\beta_8 = -1$, corresponding to the weight on point b_2. At this point $|\beta|_0$ takes the lowest value of 1 and $y = -b_2$, meaning it is assigned to Class 2. The SR method misclassifies point y, as it is clearly in Class 1, because it puts no constraints on the β values. Specifically, in this case, the issue arises from the possibility of β entries taking negative values.

Second, consider two clusters in a 2-dimensional space as shown in Fig. 15.5b with sample points belonging to Class 1 and 2. Again, we try to find the best representation

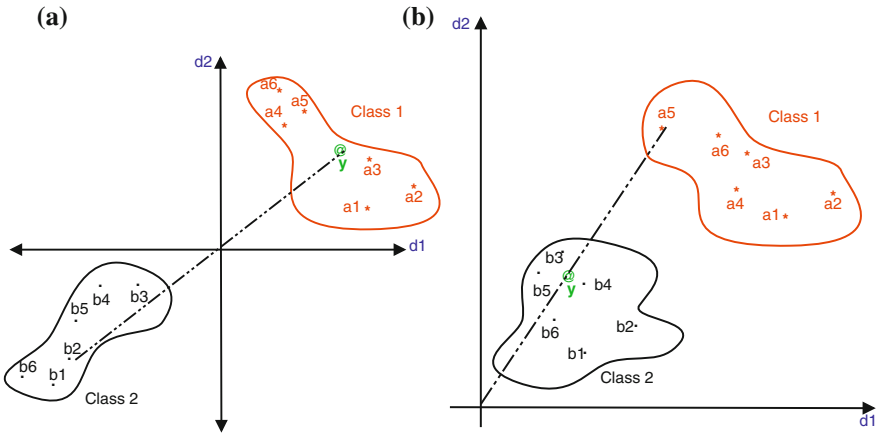

Fig. 15.5 **a** Reflective issue with negative β, **b** Scaling issue with unnormalized β

for test point y by solving (15.28). The best solution will be obtained by setting all $\beta_i = 0$ except for $\beta_5 = 0.5$. At this value, $|\beta|_0$ will take the lowest possible value of 1 and $y = 0.5 \times a_5$. This leads to a wrong classification decision as y clearly is a point in Class 2. The misclassification is due to having no constraint on the β elements. Specifically, in this case, the issue arises from total independence between the β values and no normalization criteria as a way to enforce dependency between the β elements. If we enforce β to be positive and normalized, then training points $h_i \in H$ form a convex hull. Mathematically speaking, a convex hull of training points H is defined by the set of all convex combinations of finite subsets of points from H, in other words a set of points that satisfy the following: $\sum_{i=1}^{n} h_i \beta_i$. Here n is any arbitrary number and the β_i components are positive and sum to 1.

Since many classification techniques can be sensitive to outliers, we examine the sensitivity of our convex hull SR method. Consider two clusters shown in Fig. 15.6 with sample points in Classes 1 and 2. Again, given point y, we try to find the best representation for y by solving (15.28), where now we will use a convex hull approach to solve, putting extra positivity and normalization constraints on β.

As shown in Fig. 15.6, if we project y onto the convex hulls of Class 1 and Class 2, the distance from y to the convex hull of Class 1 (indicated by r_1) is less than the distance from y to the convex hull of Class 2 (i.e. r_2). This leads to a wrong classification decision as y clearly is a point in Class 2. The misclassification is due to the effect of outliers a_1 and a_4, which create an inappropriate convex hull for Class 1.

However, all-data methods, such as GMMs, are much less susceptible to outliers, as a model for a class is built by estimating the mean and variance of training examples belonging to this class. Thus, if we include the the distance between the projection of y onto the two convex hulls of Class 1 and Class 2, as well as the distance between this projection and the means m_i of Class 1 and 2 (distance indicated by q_1 and q_2)

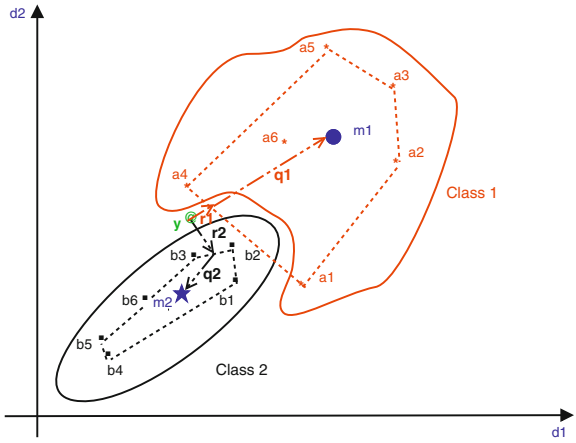

Fig. 15.6 Outliers effect

respectively, then test point y is classified correctly. Thus combining purely exemplar-based distances (r_i) with GMM-based distances (q_i), which are less susceptible to outliers, provides a more robust measure.

15.5.1 Convex Hull Formulation

In our sparse representations convex hull (SR-CH) formulation, first we seek to project test point y into the convex hull of H. After y is projected into the convex hull of H, we compute how far this projection (which we call $H\beta$) is from the Gaussian means[1] of all classes in H. The full convex hull formulation, which tries to find the optimal β to minimize both the exemplar and GMM-based distances [8]. Here $N_{classes}$ represents the number of unique classes in H, and $\| H\beta - \mu_t \|_2^2$ is the distance from $H\beta$ to the mean μ_t of class t,

$$\arg\min_{\beta} \| y - H\beta \|_2^2 + \sum_{t=1}^{N_{classes}} \| H\beta - \mu_t \|_2^2 \quad \texttt{s.t.} \quad \sum_i \beta_i = 1 \text{ and } \beta_i \geq 0 \tag{15.29}$$

In our work, we associate these distance measures with probabilities. Specifically, we assume that y satisfies a linear model as $y = H\beta + \zeta$ with observation noise $\zeta \sim N(0, R)$. This allows us to represent the distance between y and $H\beta$ using the term $p(y|\beta)$

$$p(y|\beta) \propto \exp(-1/2(y - H\beta)^T R^{-1}(y - H\beta)) \tag{15.30}$$

[1] Note that the Gaussian means we refer to in this work are built from the original training data, not the projected $H\beta$ features.

which we will refer to as the exemplar-based term.

We also explore a probabilistic representation for the $\sum_{t=1}^{N_{classes}} \parallel H\beta - \mu_t \parallel_2^2$ term. Specifically, we define the GMM-based term $p_M(\beta)$, by seeing how well our projection of y onto the convex hull of H, as represented by $H\beta$, is explained by each of the $N_{classes}$ GMM models. We score $H\beta$ against the GMM from each of the classes and sum the scores (in log-space) from all classes. This is given more formally as (log-space)

$$\log p_M(\beta) = \sum_{t=1}^{N_{classes}} \log p(H\beta|GMM_t) \tag{15.31}$$

where $p(H\beta|GMM_t)$ indicates the score from GMM t. Given the exemplar-based term $p(y|\beta)$ and GMM-based term $p_M(\beta)$, the total objective function we would like to maximize is given in the log-space by

$$\max_{\beta} F(\beta) = \{\log p(y|\beta) + \log p_M(\beta)\} \quad \texttt{s.t.} \quad \sum_i \beta_i = 1 \quad \texttt{and} \quad \beta_i \geq 0 \tag{15.32}$$

Equation (15.32) can be solved using a variety of optimization methods. We use a technique widely employed in speech recognition, namely the Extended Baum-Welch transformations (EBW) [24], to solve this problem. In [37], it is shown that EBW optimization technique can be used to maximize objective functions which are differentiable and satisfy constraints given in (15.32) (see also Sect. 15.2.3 and the recursion (15.8)). In [8], we provide a closed-form solution for β_k^i given the exemplar-based term (15.30) and a GMM-based term (15.31).

The parameter D in (15.8) controls the growth of the objective function. We explore setting D to a small value to ensure a large jump in the objective function. However, for a specific choice of D if we see that the objective function value has decreased when estimating β^k, i.e. $F(\beta^k) < F(\beta^{k-1})$, or one of the β_i^k components is negative, then we double the value of D and use this to estimate a new value of β^k in (15.8). We continue to increase the value of D until we guarantee a growth in the objective function, and all β_i components are positive. This strategy of setting D is similar to other applications in speech where the EBW transformations are used [38]. The process of iteratively estimating β continues until there is very little change in the objective function value.

15.5.2 Convex Hull Classification Rule

Because we are trying to solve for β which maximizes the objective function (15.32), it seems natural to also explore a classification rule which defines the best class as that which maximizes this objective function. Using (15.32) with the exemplar-based term (15.30) and the GMM-based term (15.31), the objective-function linked

classification rule for the best class t^* is given by

$$t^* = \max_t \{\log p(y|\delta_t(\beta)) + \log p(H\delta_t(\beta)|GMM_t)\} \tag{15.33}$$

where $\delta_t(\beta)$ is a vector which is only non-zero for entries of β corresponding to class t.

15.5.3 Experiments

We compare the performance of our SR-CH method to other standard classifiers used on the TIMIT task, including the GMM, SVM, kNN and ABCS sparse representation methods. For the GMM, we explored training it via a maximum likelihood objective function, and a discriminative BMMI objective function [38]. The parameters of each classifier were optimized for each feature set on the development set. We compare SR-CH to this method. Note that for the ABCS classification rule, the best class is defined as that which has the maximum l_2 norm of β entries.

Algorithmic Behavior

As discussed in Sect. 15.5.1, for an appropriate choice of D, the objective function of the SR-CH method is guaranteed to increase on each iteration. To observe this behavior experimentally on TIMIT, we chose a random test phone segment y, and solve $y = H\beta$ using the SR-CH algorithm. Figure 15.7 plots the value of the objective function at each iteration. Notice that the objective function increases rapidly until about iteration 30 and then increases slower, experimentally confirming growth.

We also analyze the sparsity behavior for the SR-CH method. For a randomly chosen test segment y, Fig. 15.7 plots the sparsity level (defined as the number of

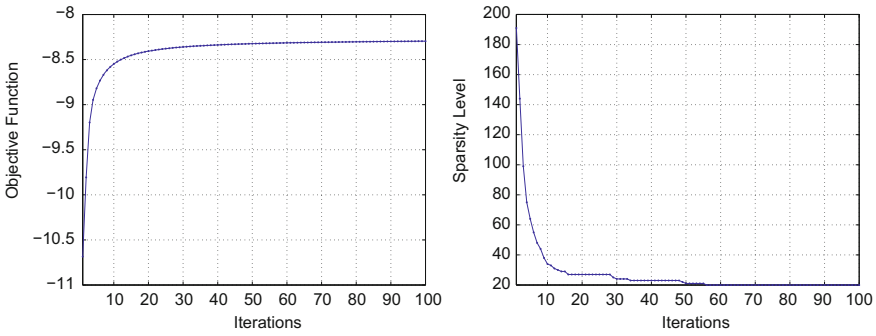

Fig. 15.7 *Left* Iterations versus objective function. *Right* Iterations versus sparsity

Table 15.4 Accuracy of sparse representation methods

Method	Accuracy
SR-CH (exemplar-only)	**83.86**
ABCS (exemplar-only)	78.16

non-zero β coefficients), for each iteration of the SR-CH algorithm. Notice that as the number of iterations increases, the sparsity level continues to decrease and eventually approaches 20. Our intuitive feeling is that the normalization and positive constraints on β in the convex hull formulation allow for this sparse solution. Recall that all β coefficients are positive and the sum of the β coefficients is small (i.e., $\sum_i \beta_i = 1$). Given that the initial β values are chosen to be uniform, and the fact we seek to find a β to maximize (15.32), then naturally only a few β elements will dominate and most β values would evolve to be close to zero.

Comparison with ABCS

To explore the constraints on β in the CH framework, we compare SR-CH to ABCS, an SR method which puts no positive and normalization constraints on β. To fairly analyze the different β constraints in the SR-CH and ABCS methods, we compare both methods only using the exemplar terms, since the GMM-based terms for the two are different. Table 15.4 shows that SR-CH method offers improvements over ABCS on the fBMMI feature set, experimentally demonstrating that constraining β values to be positive and normalized, and not allowing data in H to be reflected and shifted, allows for improved classification accuracy.

GMM-Based Term

In this section we analyze the behavior of SR-CH when using the exemplar-term only versus including the additional model-based term given in (15.31). Table 15.5 shows the classification accuracy on the development set with the fBMMI features. Notice that including the additional $H\beta$ GMM modeling term over the exemplar-based term offers a slight improvement in classification accuracy, demonstrating that including the GMM term allows for a slightly better classifier.

Table 15.5 SR-CH accuracy, TIMIT development set

SR-CH GMM-based term	Accuracy
Exemplar term only	83.86
Exemplar term+ $H\beta$ GMM term	**84.00**

Table 15.6 Classification accuracy, TIMIT core test set

Method	Accuracy fBMMI	Accuracy SA+fBMMI
SR-CH (Ex. + GMM)	**82.87**	**85.14**
ABCS (Ex. + GMM)	81.37	83.22
kNN	81.30	83.56
GMM—BMMI trained	80.82	82.84
SVM	80.79	82.62
GMM—ML trained	79.75	82.02

Comparison with Other Techniques

Table 15.6 compares the classification accuracy of the SR-CH method on the TIMIT core test set to other common classification methods. Note that for ABCS, the best numbers for this method, which include the exemplar and GMM-based terms, are reported. Results are provided for the fBMMI and SA+fBMMI feature sets. Notice that SR-CH outperforms the GMM, kNN and SVM classifiers. In addition, enforcing β to be positive allows for improvements over ABCS. A McNemar's Significance Test indicates that the SR-CH result is statistically significant from other classifiers with a 95 % confidence level. The classification accuracy of 82.87 % achieved in [8] was in 2011 the best number on the TIMIT phone classification task reported when discriminative features are used, beating the previous best single-classifier number of 82.3 % reported in [39]. Finally, when using SA + fBMMI features, the SR-CH method achieves an accuracy of over 85 %.

Accuracy Versus Size of Dictionary

One disadvantage of many exemplar-based methods is that as the number of training exemplars used to make a classification decision increases, the accuracy deteriorates significantly. For example, in the kNN method, this implies that the number of training examples from each class used during voting increases. Similarly, for SR methods, this is equivalent to the size of H growing. Parametric-based classification approaches such as GMMs do not suffer from a degradation in performance for increased training data size.

Figure 15.8 shows the classification error versus number of training-exemplars (i.e. size of H) for different classification methods. Note that the GMM method is trained with all of the training data, and is just shown here as a reference. In addition, since the feature vectors in H have dimension 120, and for our SR methods we assume H is over-complete, we only report results on SR methods when the number of examples in H is larger than 120.

First, observe that the error rates for the two purely exemplar-based methods, namely kNN and ABCS with no model term, increase exponentially as the size of H grows. However, the SR-CH exemplar-only methodology is much more robust

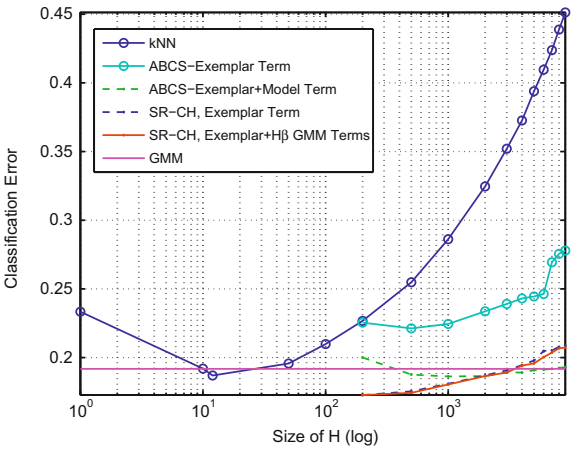

Fig. 15.8 Classification error vs. size of H

with respect to increased size of H, demonstrating the value of the convex hull regularization constraints. Including the extra GMM term into the SR-CH method improves the accuracy slightly. However, the SR-CH method still performs poorly compared to the ABCS technique which uses the GMM-based term. One explanation for this behavior is that GMM term for ABCS is capturing the probability of the data y given the GMM model, and thus the accuracy of the ABCS method eventually approaches the GMM accuracy. However, in SR-CH we capture the probability of $H\beta$ given the GMM. This is one drawback of SR-CH compared to ABCS for large H that we hope to address in the future.

15.6 Sparse Representation Features

In this section, we explore the use of a sparse representation exemplar-based technique [14] to create a new set of features while utilizing the benefits of HMMs to efficiently compare scores across frames. This is in contrast to previous exemplar-based methods which try to utilize the decision scores from the exemplar-based classifiers themselves to generate probabilities ([1, 2]). In our SR approach, given a test vector y and a set of exemplars h_i from the training set, which we put into a dictionary $H = [h_1; h_2 \ldots; h_n]$, we represent y as a linear combination of training examples by solving $y = H\beta$ subject to a spareness constraint on β. The feature $H\beta$ can be thought of as mapping test sample y back into the linear span of training examples in H. We will show that the frame classification accuracy is higher for the SR method[2] compared to a GMM, showing that not only does the $H\beta$ representation move test features closer to training, but also moves these features closer to the

[2] Using SRs to compute accuracy is described in [14].

correct class. Given these new set of $H\beta$ features, we train up an HMM on these features and perform recognition.

A speech signal is defined by a series of feature vectors, $Y = \{y^1, y^2 \dots y^n\}$, for example Mel-Scale Frequency Cepstral Coefficients (MFCCs). For every test sample $y^t \in Y$, we choose an appropriate H^t and then solve $y^t = H^t\beta^t$ to compute a β^t via ABCS. Then given this β^t, a corresponding $H^t\beta^t$ vector is formed. Thus a series of $H\beta$ vectors is created at each frame as $\{H^1\beta^1, H^2\beta^2 \dots H^n\beta^n\}$. The sparse representation features are created for both training and test. An HMM is then trained given this new set of features and recognition is performed in this new feature space.

15.6.1 Measure of Quality

We can measure how well y assigns itself to different classes in H by looking at the residual error between y and the $H\beta$ entries corresponding to a specific class [6]. Ideally, all nonzero entries of β should correspond to the entries in H with the same class as y and the residual error will be smallest within this class. More specifically, let us define a selector $\delta_i(\beta) \in \mathbb{R}^N$ as a vector whose entries are non-zero except for entries in β corresponding to class i. We then compute the residual error for class i as $\| y - H\delta_i(\beta) \|_2$. The best class for y will be the class with the smallest residual error. Mathematically, the best class i^* is defined as

$$i^* = \min_i \| y - H\delta_i(\beta) \|_2 . \tag{15.34}$$

15.6.2 Choices of Dictionary H

Success on the sparse representation features depends heavily on a good choice of H. Pooling together all training data from all classes into H will make the columns of H large (typically millions of frames), and will make solving for β intractable. Therefore, in this section we discussion various methodologies to select H from a large sample set. Recall that H is selected for each frame y, and then β is found using ABCS, in order to create an $H\beta$ feature for each frame.

- **Seeding H from Nearest Neighbors**: For each y, we find a neighborhood of closest points to y in the training set. These k neighbors become the entries of H. We refer the reader to [3] for a discussion on choosing the number of k neighbors for SRs. A set of $H\beta$ features is created for both training and test, but H is always seeded with data from training data. To avoid overtraining of $H\beta$ features on the training set, we require that only when creating $H\beta$ features on training, samples be selected from training that are of a different speaker than the speaker corresponding to frame y. While this kNN approach is computationally feasible on small-vocabulary tasks, using a kNN for large vocabulary tasks can be computationally expensive.

To address this, we discuss other choices for seeding H below, tailored to large vocabulary applications.

- **Using a Trigram Language Model**: Ideally only a small subset of Gaussians are typically evaluated at a given frame, and thus training data belonging to this small subset can be used to seed H. To determine these Gaussians at each frame, we decode the data using a trigram language model (LM), and find the best aligned Gaussian at each frame. For each Gaussian, we compute the 4 other closest Gaussians to this Gaussian. Here closeness is defined by finding Gaussian pairs which have the smallest Euclidean distance between their means. After we find the top five Gaussians at a specific frame, we seed H with the training data aligning to these top five Gaussians. Since this still typically amounts to thousands of training samples in H, we must sample this further. Our method for sampling is discussed in Sect. 15.6.3. We also compare seeding H using the top 10 Gaussians rather than top five.
- **Using a Unigram Language Model**: One problem with using a trigram LM is that this decode is actually the baseline system we are trying to improve upon. Therefore, seeding H with frames related to the top aligned Gaussian is essentially projecting y back down to the same Gaussian which initially identified it. Thus to increase variability between the Gaussians used to seed H and the best aligned Gaussian from the trigram LM decode, we explore using a unigram LM to find the best aligned Gaussian at each frame. Again, given the best aligned Gaussian, the four closest Gaussians to this are found and data from these five Gaussians is used to seed H.
- **Using no Language Model Information**: To further weaken the effect of the LM, we explore seeding H using only acoustic information. Namely, at each frame we find the top five scoring Gaussians. H is seeded with training data aligning to these Gaussians.
- **Enforcing Unique Phonemes**: Another problem with seeding H by finding the five closest Gaussians relative to the best aligned Gaussian is that all of these Gaussians could come from the same phoneme (i.e. phoneme “AA"). Therefore, we explore finding the five closest Gaussians relative to the best aligned such that the phoneme identities of these Gaussians are unique (i.e. “AA", “AE", “AW", etc.). H is then seeded by from frames aligning to these five Gaussians.
- **Using Gaussian Means**: The above approaches of seeding H use actual examples from the training set, which is computationally expensive. To address this, we investigate seeding H from Gaussian means. Namely, at each frame we use a trigram LM to find the best aligned Gaussian. Then we find the 499 closest Gaussians to this top Gaussian, and use the means from these 500 Gaussians to seed H.

15.6.3 Choice of Sampling

As discussed above, if we seed H using all training data belonging to specific Gaussians, this amounts to thousands of training examples in H. We explore two different approaches to sampling a subset of this data for seeding H.

- **Random Sampling**: For each gausssian we want to select training data from, we explore randomly sampling N training examples from the total set of training frames that aligned to this Gaussian. This process is repeated for each of the closest five Gaussians. We reduce the size of N as the "closeness" decreases. For example, for the closest 5 Gaussians, the number of data points N chosen from each Gaussian is 200, 100, 100, 50 and 50 respectively.
- **Sampling Based on Cosine Similarity**: While random sampling offers a relatively quick approach to select a subset of training examples, it does not guarantee that we select "good examples" from this Gaussian which actually are close to frame y. Alternatively, we explore splitting training points aligning to a Gaussian as being $1\sigma, 2\sigma$, etc. away from the mean of the Gaussian. Here σ is chosen to be the total number of training points aligned to this Gaussian, divided by number of samples N we want to sample from this Gaussian. Then within each σ set, we find the training point which has the closest cosine similarity to the test point y. This is repeated for all $1\sigma, 2\sigma$, etc. values. Again the number of samples taken from each Gaussian reduces as "closeness" decreases.

15.6.4 Experiments

The small vocabulary recognition experiments in this paper are conducted on the TIMIT phonetic corpus [16]. Similar to [40], acoustic models are trained on the training set, and results are reported on the core test set. The initial acoustic features are 13-dimensional MFCC features. The large vocabulary experiments are conducted on an English broadcast news transcription task [17]. The acoustic model is trained on 50 h of data from the 1996 and 1997 English Broadcast News Speech Corpora. Results are reported on 3 h of the EARS Dev-04f set. The initial acoustic features are 19-dimensional PLP features.

Both small and large vocabulary experiments utilize the following recipe for training acoustic models [40]. First, a set of CI HMMs are trained, either using information from the phonetic transcription (TIMIT) or from flat-start (broadcast news). The CI models are then used to bootstrap the training of a set of CD triphone models. In this step, at each frame, a series of consecutive frames surrounding this frame are joined together and a Linear Discriminative Analysis (LDA) transform is applied to project the feature vector down to 40 dimensions. Next, vocal tract length normalization (VTLN) and feature space Maximum Likelihood Linear Regression (fMLLR) are used to map the features into a canonical speaker space. Then, a set of discriminatively trained features and models are created using the boosted Maximum

Mutual Information (BMMI) criterion. Finally, the set of models is adapted using MLLR.

We create a set of $H\beta$ features from a set of fBMMI features. We choose this level as these features offer the highest frame accuracy relative to LDA, VTLN, or fMLLR features, allowing us to further improve on the accuracy with with the $H\beta$ features. A set of $H\beta$ features are created at each frame from the fBMMI features for both training and test. A new ML HMM is trained up from these new features and used for both training and test. Since $H\beta$ features create a linear combination of the discriminatively trained fBMMI features, we argue that some discrimination can be lost. Therefore, we explore applying another fBMMI transformation to the $H\beta$ features before applying model space discriminative training and MLLR.

In what follows we present results using $H\beta$ features on both small and large vocabulary tasks.

15.6.5 Sparsity Analysis

We first analyze the β coefficients obtained by solving $y = H\beta$ using ABCS [3]. For two randomly selected frames y, Fig. 15.9 shows the β coefficients corresponding to 200 entries in H for TIMIT and 500 entries for Broadcast News. Notice that for both datasets, the β entries are quite sparse, illustrating that only a few samples in H are used to characterize y. As [6] discusses, this sparsity can be thought of as a form of discrimination, as certain examples are selected as "good" in H while jointly assigning zero weights "bad" examples in H. We have seen advantages of the

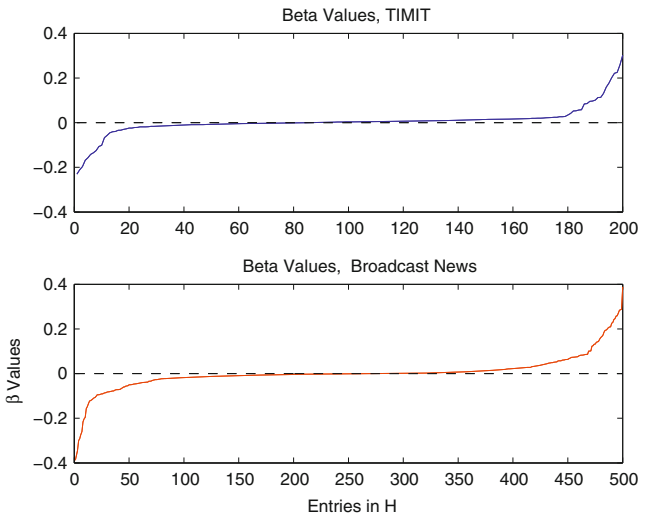

Fig. 15.9 β coefficients on TIMIT and broadcast news

Table 15.7 Frame accuracy on TIMIT testcore set

Classifier	frame Accuracy
GMM	70.4
Sparse representations	**71.7**

SR approach for classification, even on top of discriminatively trained y features, compared to a GMM [3]. We will also re-confirm this behavior in Sect. 15.6.6. The extra benefit of SRs on top of discriminatively trained fBMMI features, coupled with an exemplar-based nature of SRs, motivates us to further explore its behavior for recognition tasks.

15.6.6 TIMIT Results

Frame Accuracy

The success of $H\beta$ first relies on the fact that the β vectors give large support to correct classes and small support to incorrect classes (as demonstrated by Fig. 15.9) when computing $y = H\beta$ at each frame. Thus, the classification accuracy per frame, computed using (15.34), should ideally be high. Table 15.7 shows the frame accuracy for the GMM and SR methods.

Notice that the SR technique offers significant improvements over the GMM method, again confirming the benefit of exemplar-based classifiers.

Error Rate for $H\beta$ Features

Table 15.8 shows the recognition performance of $H\beta$ features on TIMIT. Due to the small vocabulary nature of TIMIT, we only explore seeding H from nearest neighbors. Notice that creating a set of $H\beta$ features in the fBMMI space offers a 0.7 % absolute improvement in PER. Given the small vocabulary nature of TIMIT, no gain was found applying another fBMMI transform to the baseline or $H\beta$ features. After applying BMMI and MLLR to both feature sets, the $H\beta$ features offer a 0.5 % improvement in PER over the baseline system. This shows that using exemplar-based SRs to produce $H\beta$ features not only moves test features closer to training, but also moves the feature vectors closer to the correct class, resulting in a decrease in PER.

Table 15.8 WER on TIMIT

Baseline system	PER	$H\beta$ System	PER
fBMMI	19.9	$H\beta$	**19.2**
+BMMI +MLLR	19.5	+BMMI +MLLR	**19.0**

15.6.7 Broadcast News Results

Selection of H

Table 15.9 shows the WER for the $H\beta$ features for different H choices discussed in Sect. 15.6.2. Note that the baseline fMMI system as a WER of 21.1 %. The following can be observed:

- There is little difference in WER when sampling is done randomly or using cosine similarity. For speed efficiencies, we use random sampling for H selection methods.
- There is little difference between using 5 and 10 Gaussians.
- Seeding H using nearest neighbors is worse than using the trigram LM. On broadcast news, we find that a kNN has lower frame-accuracy than a GMM, a result similarly observed in the literature for large vocabulary corpora [1]. This lower frame accuracy translates into a higher WER when H is seeded with nearest neighbors.
- Seeding H from unique Gaussians provides too much variability of phoneme classes into the $H\beta$ feature, also leading to a higher WER.
- Using a unigram LM to reduce the link between the Gaussians used to seed H and the best aligned Gaussian from the trigram LM decode offers a slight improvement in WER over the trigram LM.
- Utilizing no LM information results in a very high WER.
- Using Gaussian means to seed H reduces the computation to create $H\beta$ without a large increase in WER.

WER for $H\beta$ Features

Table 15.10 shows the performance of $H\beta$ features on the Broadcast News task. Creating a set of $H\beta$ features at the fBMMI space offers a WER of 21.1 % which is comparable to the baseline system. However, after applying an fBMMI transform to the $H\beta$ features we achieve a WER of 20.2 %, a 0.2 % absolute improvement when another fBMMI transform is applied to the original fBMMI features. Finally,

Table 15.9 WER of $H\beta$ features for different H

H selection method	WER
Trigram LM, random sampling, top 5 Gaussians	21.2
Trigram LM, cosine similarity sampling, top 5 Gaussians	21.3
Trigram LM, top 10 Gaussians	21.3
Nearest neighbor, 500	21.4
Trigram LM, 5 unique Gaussians	21.6
Unigram LM, top 5 Gaussians	**21.1**
No LM information, top 5 Gaussians	22.7
Gaussian means, top 500 Gaussians	21.4

Table 15.10 WER on broadcast news

Baseline system	WER	$H\beta$ system	WER
fBMMI	21.1	$H\beta$	21.1
+fBMMI	20.4	+fBMMI	**20.2**
+BMMI +MLLR	19.0	+BMMI +MLLR	**18.7**

after applying BMMI and MLLR to both feature sets, the $H\beta$ features offer a WER of 18.7 %, a 0.3 % absolute improvement in WER over the baseline system. This demonstrates again that using information about actual training examples to produce a set of features which are mapped closer to training and have a higher frame accuracy than GMMs improves accuracy for large vocabulary as well.

15.7 SR Phone Identification Features (S_{pif})

In this section, we review the use of SR for classification and use this framework to create our S_{pif} features. Let us, first, describe how we can use β to create a set of S_{pif} vectors. First, define matrix $H_{phnid} = [p_{1,1}, p_{1,2}, \ldots, p_{w,n_w}] \in \mathbb{R}^{r \times N}$, which has the same number of columns N as the original H, but a different number of rows r. Recall that each $x_{i,j} \in H$ has a corresponding class label i. We define each $p_{i,j} \in H_{phnid}$ corresponding to feature vector $x_{i,j} \in H$ to be a vector with zeros everywhere except at the index i corresponding to class of $x_{i,j}$. Figure 15.10 shows the H_{phnid} corresponding to H, where each $p_{i,j}$ becomes a phone identification vector with a value of 1 corresponding to the class of $x_{i.j}$. Here r, the dimension of each $p_{i,j}$, is equivalent to the total number of classes.

Once β is found by solving $y = H\beta$, we use this same β to select important classes within the new dictionary H_{phnid}. Specifically, let us define a new feature vector S_{pif}, as $S_{pif} = H_{phnid}\beta^2$, where each element of β is squared, i.e., $\beta^2 = \{\beta_i^2\}$. Notice that we are using β^2, as this is similar to the $\| \delta_i(\beta) \|_2$ classification rule given by (15.34). Each row i of the S_{pif} vector roughly represents the l_2 norm of β entries for class i.

A speech signal is defined by a series of feature vectors, $Y = \{y^1, y^2 \ldots y^n\}$, for example Mel-Scale Frequency Cepstral Coefficients (MFCCs). For every test sample $y^t \in Y$, we solve $y^t = H^t\beta^t$ to compute a β^t. Then given this β^t, a corresponding

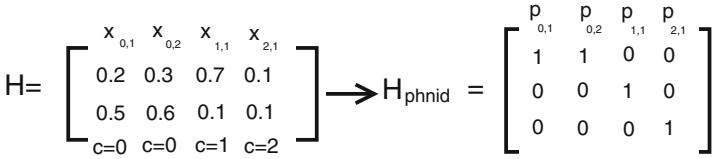

Fig. 15.10 H_{phnid} corresponding to H

S^t_{pif} vector is formed. Since β^t at each sample represents a weighting of entries in H^t that best represent test vector y^t, this makes it difficult to compare β^t values and the S^t_{pif} vectors across frames. Therefore, to ensure that the values can be compared across samples, the S^t_{pif} vectors are normalized at each sample. Thus, the new $\bar{S}^t_{pif}$ at sample t is computed as $\bar{S}^t_{pif} = \frac{S^t_{pif}}{\|S^t_{pif}\|_1}$. A series of S_{pif} vectors is created as $\{\bar{S}^1_{pif}, \bar{S}^2_{pif} \ldots \bar{S}^n_{pif}\}$, and are used for recognition.

15.7.1 Construction of Dictionary H

Success of SRs depends on a good choice of H. In [14], various methods for seeding H from a large sample set were explored. Below we summarize the main techniques used in this work to select H.

Seeding *H* from Nearest Neighbors

For each y, we find a neighborhood of closest points to y from all examples in the training set. These k neighbors become the entries of H. While this approach works well on small-vocabulary tasks, it is computationally expensive for large data sets.

Using a Language Model

In speech recognition, when an utterance is scored using a set of HMMs (which have output distributions given by Gaussians), typically evaluating only a small subset of these Gaussians at a given frame allows for a large improvement in speed without a reduction in accuracy [41]. Using this fact, we use training data belonging to a small subset of Gaussians to seed H. To determine these Gaussians at each frame, we decode the data using a language model (LM), and find the best aligned Gaussian at each frame. For each Gaussian, we compute the four other closest Gaussians to this Gaussian. After we find the top five Gaussians at a specific frame, we seed H with the training data aligning to these top five Gaussians. We explore using both a trigram and unigram LMs to obtain the top Gaussians.

Using a Lattice

Seeding H as suggested above is similar to finding the best H at the frame level. However, the goal of speech recognition is to recognize words, and therefore we explore seeding H using information related to competing word hypotheses. Specifically, we create a lattice of competing word hypotheses and obtain the top Gaussians at each

frame from the Gaussian alignments of the lattice. Gaussians to the best Gaussian are found and data from these five Gaussians is used to seed H.

15.7.2 Reducing Sharpness Estimation Error

As described in Sect. 15.7.1, for computational efficiency, S_{pif} features are created by first pre-selecting a small amount of data for dictionary H. This implies that only a few classes are present in H and only a few S_{pif} posteriors are non-zero, something we will define as feature sharpness. Feature sharpness by itself is advantageous—for example if we were able to correctly predict the right class at each frame and capture this in S_{pif} the WER would be close to zero. However, because we are limited by the amount of data that can be used to seed H, incorrect classes may have their probabilities boosted over correct classes, something we will refer to as sharpness estimation error. In this section, we explore various techniques to smooth out the sharp S_{pif} features and reduce estimation error.

Choice of Class Identification

The S_{pif} vectors are defined based on the class labels in H. We explore two choice of class labels in this paper. First, we explore using monophone class labels. Second, we investigate labeling classes in H by a set of context independent (CI) triphones. While using triphones increases the dimension of the S_{pif} vector, the elements in the vector are less sharp now since β values for a specific monophone are more likely to be distributed within the three different triphones of this monophone.

Posterior Combination

Another technique to reduce feature sharpness is to combine S_{pif} posteriors with posteriors coming from an HMM system, a technique which is often explored when posteriors are created using Neural Nets [17]. Specifically, let us define $h^j(y_t)$ as the output distribution for observation y_t and state j of an HMM system. In addition, define $S^j_{pif}(y_t)$ as the S_{pif} posterior corresponding to state j. Note that the number of S_{pif} posteriors could be less than the number of HMM states, so the same S_{pif} posterior could map to multiple HMM states. For example, the S_{pif} posterior corresponding to phone "aa" could map to HMM states "aa-b-0", "aa-m-0", etc. Given the HMM and S_{pif} posteriors, the final output distribution $b^j(y_t)$ is given by Eq. 15.35, where λ is a weight on the S_{pif} posterior stream, selected on a held-out set.

$$b^j(y_t) = h^j(y_t) + \lambda S^j_{pif}(y_t) \tag{15.35}$$

S_{pif} Feature Combination

As we will show in Sect. 15.7.5, S_{pif} features created using different methodologies to select H offer complementary information. For example, S_{pif} features created when H is seeded with a lattice have higher frame accuracy and incorporate more sequence information than when H is seeded using a unigram or trigram LM. However, S_{pif} features created from lattice information are much sharper compared to features created with a uni/trigram LM. Thus, we explore combining different S_{pif} features. If we denote S^{tri}_{pif}, S^{uni}_{pif} and S^{lat}_{pif} as being created from the three different H selection methodologies, we combine these features to produce a new S^{comb}_{pif} feature as given by Eq. 15.36. Weights $\{\alpha, \beta, \gamma\}$ are chosen on a held-out set with the constraint that $\alpha + \beta + \gamma = 1$.

$$S^{comb}_{pif} = \alpha S^{tri}_{pif} + \beta S^{uni}_{pif} + \gamma S^{lat}_{pif} \tag{15.36}$$

15.7.3 Experiments

The small vocabulary recognition experiments are conducted on TIMIT [16]. Similar to [14], acoustic models are trained on the training set, and results are reported on the core test set. The initial acoustic features are 13-dimensional MFCC features. The large vocabulary experiments are conducted on an English broadcast news transcription task [17]. The acoustic model is trained on 50 h of data from the 1996 and 1997 English Broadcast News Speech Corpora. Results are reported on 3 h of the EARS Dev-04f set. The initial acoustic features are 19-dimensional PLP features.

Both corpora utilize the following recipe for training. First, a set of CI HMMs are trained, either using information from the phonetic transcription (TIMIT) or from flat-start (Broadcast News). The CI models are then used to bootstrap the training of a set of CD triphone models. In this step, given an initial set of MFCC or PLP features, a set of LDA features are created. After the features are speaker adapted, a set of discriminatively trained features and models are created using the boosted Maximum Mutual Information (BMMI) criterion. Finally, models are adapted via MLLR.

On TIMIT, we explore creating S_{pif} features from both LDA and fBMMI features, while for Broadcast news, we only create S_{pif} features after the fBMMI stage. The initial LDA/fBMMI features are used for both y and H to solve $y = H\beta$ and crate S_{pif} features at each frame. In this work, we explore the ABCS method. Once series of S_{pif} vectors are created, an HMM is built on the training features.

15.7.4 TIMIT Results

Frame Accuracy

The success of S_{pif} first relies on the fact that the classification accuracy per frame, computed using Eq. 15.34, should ideally be high. Table 15.11 shows the classification accuracy for the GMM and SR methods,[3] for both LDA and fBMMI feature spaces. Notice that the SR technique offers significant improvements over the GMM method.

Recognition Results: Class Identification

Table 15.12 shows the phonetic error rate (PER) at the CD level for different class identification choices. Since only a kNN is used to seed H on TIMIT, we will call the feature S_{pif}^{knn}. We have also listed results for other CD-ML trained systems reported in the literature on TIMIT. Notice that smoothing out sharpness error of the S_{pif} features by using triphones rather than monophones results in a decrease in error rate. The S_{pif}-triphone features outperform the LDA features and also offer the best result of all methods on TIMIT at the CD level for ML trained systems.

We further explore S_{pif} features created after the fBMMI stage. Table 15.13 shows that the performance is now worse than the fBMMI system. Because the fBMMI features are already discriminative in nature and offer good class separability, S_{pif} features created in this space are too sharp, explaining the increase in PER.

Recognition Results: Posterior Combination

We explore reducing feature sharpness by combining S_{pif} posteriors with HMM posteriors, as shown in Table 15.14. We observe that on TIMIT, combining posteriors from two different feature streams has virtually no impact in recognition accuracy compared to the baseline fBMMI system, indicating there is little complementarity between the two systems. Because gains were not observed with posterior combination, further S_{pif} feature combination was not explored.

Table 15.11 Frame accuracy on TIMIT testcore set

Classifier	Frame Acc. (LDA)	Frame Acc. (fBMMI)
GMM	61.5	70.4
SR	**64.0**	**71.7**

[3] We have not included the accuracy of the HMM since this takes into account sequence information which both the GMM and SR methods do not.

Table 15.12 PER on TIMIT core test set—CD ML trained systems

System	PER (%)
S_{pif}^{knn} monophones, IBM CD HMM (this paper)	25.1
Monophone HTMs [42]	24.8
Baseline LDA features, IBM CD HMM	24.5
Heterogeneous measurements [43]	24.4
S_{pif}^{knn} triphones, IBM CD HMM (this paper)	**23.8**

Table 15.13 PER on TIMIT core test set—fMMI level

Features	PER
Baseline fBMMI features	19.4
S_{pif}^{knn} triphones	20.7

Table 15.14 PER on TIMIT core test set—posterior combination

Features	PER
Baseline fBMMI Features	19.4
S_{pif}^{knn}, Posterior Combination	19.4

15.7.5 Broadcast News

In this section we explore the S_{pif} features on Broadcast News.

Recognition Results: Choice of H and Class Identity

Table 15.15 shows the frame accuracy and WER on Broadcast news for different choice of H and class identity. We also quantify the sharpness estimation error between the different S_{pif} methods. We define "sharpness" of a S_{pif} vector by calculating the entropy from the non-zero probabilities of the feature. The sharper the S_{pif} feature, the lower the entropy. A very sharp S_{pif} feature that emphasizes the incorrect class for a frame will lead to a classification error. Therefore, we measure sharpness error by the average entropy of all misclassified S_{pif} frames. Please note that sharpness is only measured for monophone S_{pif} features. Using triphone S_{pif} smooths out class probabilities since the feature dimension is increased. However, it is difficult to quantifiably compare feature sharpness for the monophone and triphone S_{pif} features since the correct phone labels and dimensions are of the two features are different.

First, notice the trend between frame accuracy and entropy in Table 15.15. S_{pif}^{uni} features have a low frame accuracy and hence a low WER. While S_{pif}^{lat} features have a very high frame accuracy, they have a higher entropy on misclassified frames compared to S_{pif}^{tri} and S_{pif}^{uni}, and hence have a high WER. S_{pif}^{tri} features created from a trigram LM offer the best tradeoff between feature sharpness and accuracy, and achieve a WER close to the baseline. However, if feature sharpness is reduced by

Table 15.15 WER on broadcast news, class identification

Features	Frame Acc.	S_{pif} Entropy Error Frames	WER
Baseline fBMMI, ML training	–	–	19.4
S^{tri}_{pif} monophones	70.3	2.27	19.5
S^{uni}_{pif} monophones	68.3	2.23	29.0
S^{lat}_{pif} monophones	77.2	0.86	21.6
S^{tri}_{pif} triphones	–	–	19.8

using triphone S^{tri}_{pif} features, we see now on a word recognition task that the WER increases slightly.

Oracle Results of Reducing Estimation Error

We motivate the need for reducing sharpness error, with the following oracle experiment. Given the S^{tri}_{pif}-monophone features, x % of the frames which are misclassified are corrected to have a probability of 1 at the correct phone index and 0 elsewhere. Table 15.16 shows the results when 1 %, 3 %, and 5 % of the misclassified S_{pif} features are corrected. Notice that just by correcting a small % of misclassified features, the WER reduces significantly. This motivates us to explore different techniques to reduce S_{pif} sharpness in the next section.

Recognition Results: Posterior and S_{pif} Combination

In this section, we explore reducing sharpness through posterior and S_{pif} combination. Table 15.17 shows the baseline results for the fBMMI and S_{pif}-monphone features at 18.7 % and 19.5 % respectively. The frame accuracies and entropies of misclassified frames for various S_{pif} combination features are also listed. Note that the frame accuracy is only reported on the S_{pif} feature and does not include frame accuracy after posterior combination.

First, notice that through posterior combination, we reduce the WER by 0.5 % absolute from 18.7 % to 18.2 %, showing the complementarity between the fBMMI and S_{pif} feature spaces. Second, by doing additional S_{pif} feature combination, we

Table 15.16 WER on broadcast news, oracle results

Features	Frame accuracy	WER
S^{tri}_{pif} 0 % cheating	70.3	19.5
S^{tri}_{pif} 1 % cheating	71.4	19.4
S^{tri}_{pif} 3 % cheating	73.7	18.8
S^{tri}_{pif} 5 % cheating	76.1	17.6

Table 15.17 WER on broadcast news, posterior and S_{pif} combination

Features	Frame Acc.	S_{pif} Ent.	WER
Baseline fBMMI features, BMMI training + MLLR	–	–	18.7
S_{pif}^{tri} monophones	70.3	2.27	19.5
S_{pif}^{tri}, posterior combination	70.3	2.27	**18.2**
$\alpha S_{pif}^{tri} + \beta S_{pif}^{uni} + \gamma S_{pif}^{lat}$, posterior combination	76.3	2.29	**17.8**

are able to increase the frame accuracy from 70.3 % to 76.3 %, without a reduction in S_{pif} entropy as it increases slightly from 2.27 to 2.29. This results in a further decrease in WER of 0.4 % absolute from 18.2 % to 17.8 %, indicating the importance of reducing feature sharpness, particularly for misclassified S_{pif} frames.

15.8 Enhancing Exemplar-Based Posteriors for Speech Recognition Tasks

When errors occur in exemplar modeling, this results in wrong classes having their probabilities over-emphasized, something we will refer to as feature or posterior sharpness. In general, it can be argued that a more desired methodology for enhancing the posteriors is the one that simultaneously improves the frame accuracy and reduces the erratic sharpness across the frames. Given that through a NN transformation we have enhanced the posteriors by improving the frame error rate, we explore a new technique to smooth the posteriors. Specifically, we explore a technique similar to the tied mixture approach [20] where new posteriors are modeled as a tied mixture of the NN posteriors. Specifically, given feature o_t and a set of NN posterior scores $p(s_i|o_t)$ for all classes $i \in L$, we can estimate the posterior for state s_j as given by

$$p(s_j|o_t) = \sum_{i=1}^{L} p(s_i|o_t)p(s_j|o_t, s_i) \tag{15.37}$$

As in the tied mixture approach [20], a tying is invoked such that the term $p(s_j|o_t, s_i)$ for a given i is independent of o_t, which reduces (15.37) to

$$p(s_j|o_t) = \sum_{i=1}^{L} p(s_i|o_t)p(s_j|s_i) \tag{15.38}$$

where $p(s_j|s_i)$ is a set of mixing coefficients. Mixing NN posteriors from different classes helps to smooth over sharp posterior distributions [20].

In this section we look to learn a set of mixing coefficients $p(s_j|s_i)$ to mix state based posteriors from different states. More formally, we will refer to the $NN - S_{pif}$ posteriors $p(s_i|o_t)$ as a. If we assume there are L states, then the posterior probability $a_t(l)$ at time t for state l satisfies the following properties:

$$a_t(l) \geq 0 \quad \texttt{and} \quad \sum_{l=1}^{L} a_t(l) = 1 \tag{15.39}$$

Given state l and a set of $k = \{1, \ldots, L\}$ NN posteriors for this state l, we define a mixing coefficient $p(s_j|s_i)$ as $b(l, k)$, which satisfies the following properties:

$$b(l, k) \geq 0 \quad \texttt{and} \quad \sum_{k=1}^{L} b(l, k) = 1 \tag{15.40}$$

Our objective is to learn a set of mixing coefficients $b(l, k)$ via maximum likelihood. In this paper, we explore maximizing an objective function which linearly interpolates the original posteriors a, similar to the tied mixture approach [20]. Specifically, consider all frames aligned to a state l from $t = 1$ to T_l. We can define the mixed posterior for a specific frame t as

$$c_t(l) = \sum_{k=1}^{L} b(l, k) a_t(k) \tag{15.41}$$

It is easy to see that $c_t(l)$ satisfies (15.40) and is a posterior. The objective function of this posterior across all frames in the training data aligned to state l is given by

$$f_l(b) = \prod_{t=1}^{T_l} c_t(l) = \prod_{t=1}^{T_l} (\sum_{k=1}^{L} b(l, k) a_t(k)) \tag{15.42}$$

Because (15.42) is a polynomial with positive coefficients, the Baum-Welch update equation can be used to iteratively solve for $b(l, k)$ which maximizes the above objective function. The recursive update equation for $b(l, k)$ is given by

$$b(l, k) := \frac{b(l, k) \nabla_{b(l,k)} f_l(b)}{\sum_{j=1}^{L} b(l, j) \nabla_{b(l,j)} f_l(b)} \tag{15.43}$$

Here the gradient of the objective function $f_l(b)$ is

$$\nabla_{b(l,k)} f_l(b) = \sum_{t=1}^{T_l} f_l(b) \frac{a_t(k)}{\sum_{i=1}^{L} b(l, i) a_t(i)} \tag{15.44}$$

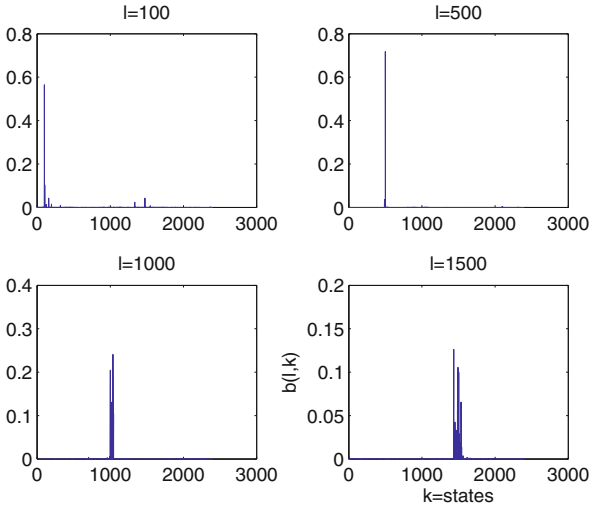

Fig. 15.11 Mixing coefficient examples

Substituting the gradient (15.44) into the update formula (15.43) yields the following update for $b(l,k)$

$$b(l,k) := \frac{1}{T_l} \sum_{t=1}^{T_l} \frac{b(l,k)a_t(k)}{\sum_{i=1}^{L} b(l,i)a_t(i)} \tag{15.45}$$

This equation shows that the mixing coefficients $b(l,k)$ learned for state l effectively take a linearly weighted average of posterior coefficients a over all training frames aligned to state l.

Note that (15.45) assumes an initial value of $b(l,k)$. We assume that the initial $b(l,k)$ is uniformly distributed as $1/L$ where L is the number of states. $b(l,k)$ is iteratively updated using (15.45) until the change in the objective function value between iterations is below a specified threshold.

Once $b(l,k)$ is learned, given state l, and the $NN - S_{pif}$ posteriors (denoted by a), a new posterior for state l is computed by taking a weighted average of the NN posteriors and mixing coefficients. This new posterior, denoted by $NN - S_{pif} - Post^{(l)}$ for state l is given by

$$NN - S_{pif} - Post^{(l)} = \sum_{k=1}^{L} b(l,k)a_t(k) \tag{15.46}$$

Figure 15.11 plots the mixing coefficients $b(l,k)$ for states $l = 100, 500, 1,000,$ and $1,500$. We can observe that for all states, the non-zero mixing coefficients are clustered together, and thus come from context-dependent states which are similar to each other, for example states which map to the same monophone.

15.8.1 Results

The following experiments were conducted as described in Sect. 15.7.3.

Using S_{pif} Features As Output Probabilities

First, we explore the performance of S_{pif} posteriors when used as output probabilities directly in an HMM system. Table 15.18 shows that the performance of the S_{pif} posteriors is worse than the baseline GMM/HMM system trained on fBMMI features, illustrating the problem with deriving exemplar-based posterior features which are not learned through a discriminative process linked to WER. Furthermore, combining S_{pif} and GMM posteriors in tandem does not offer improvements over the baseline GMM/HMM system.

Enhancing Using Neural Networks

Second, we explore the performance of training a NN with S_{pif} features as input, and then again using the $NN - S_{pif}$ probabilities as output probabilities in an HMM system. Table 15.19 shows that the $NN - S_{pif}$ features offers a 1.3 % absolute reduction in WER over using S_{pif} features alone. This illustrates the importance of enhancing S_{pif} posteriors with a NN to create a set of posteriors better aligned the PER objective in speech. Furthermore, the PER of 19.0 % is better than the GMM/HMM system trained with fBMMI features [21], as well as a NN trained with fBMMI features [44]. This demonstrates the benefit of exemplar-based features over standard speech features (i.e. fBMMI).

Table 15.18 PER on TIMIT core test set, S_{pif} features

Features	PER
GMM/HMM fBMMI	**19.5**
S_{pif} posteriors	20.3
Tandem: S_{pif} + GMM	19.5

Table 15.19 PER on TIMIT core test set, NN enhancement

Features	PER
S_{pif}	20.3
$NN - S_{pif}$	**19.0**
GMM/HMM—fBMMI + BMMI + MLLR [21]	19.4
NN—fBMMI [44]	19.4

Table 15.20 PER on TIMIT Core Test set, posterior smoothing

Features	PER
$NN - S_{pif}$	19.0
$NN - S_{pif} - Post$	**18.7**

Smoothing with Posterior Modeling

Finally, we explore smoothing out $NN - S_{pif}$ posteriors through tied mixtures as discussed in this section. Again, mixed posteriors $NN - S_{pif} - Post$ are used as output probabilities in an HMM system. Table 15.20 shows that using posterior modeling, we can obtain a small improvement of 0.3 % absolute over the $NN - S_{pif}$ posteriors. illustrating the value of reducing posterior sharpness through tied mixture smoothing.

Error Analysis

Figure 15.12 shows the breakdown of error rates for the GMM/HMM, $NN - S_{pif}$ and $NN - S_{pif} - Post$ methods within six BPCs, namely vowels/semivowels, nasals, strong fricatives, weak fricatives, stops and closures/silence. Here the error rate was calculated by counting the number of insertions, deletions and substitutions that occur for all phonemes within a particular BPC. The $NN - S_{pif}$ method offers improvements over the GMM/HMM system in all classes except nasals and closures. Furthermore, we can see the gains with the $NN - S_{pif} - Post$ method are coming due to better modeling in the vowel, weak fricative and closure classes.

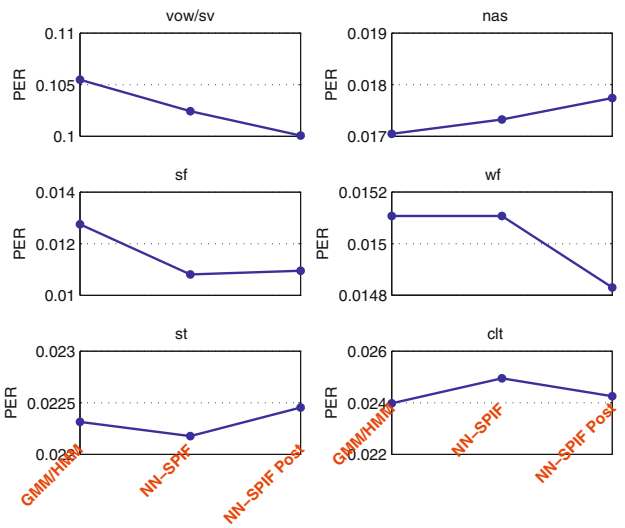

Fig. 15.12 Error rates within 6 BPCs for various methods

References

1. Deselaers T, Heigold G, Ney H (2007) Speech recognition with state-based nearest neighbour classifiers. In: Proceedings of the interspeech.
2. Gemmeke JF, Virtanen T (2010) Noise robust exemplar-based connected digit recognition. In: Proceedings of the ICASSP.
3. Sainath TN, Carmi A, Kanevsky D, Ramabhadran B (2010) Bayesian compressive sensing for phonetic classification. In: Proceedings of the ICASSP.
4. De Wachter M, Demuynck K, Van Compernolle D, Wambacq P (2003) Data driven example based continuous speech recognition. In: Proceedings of the european conference on speech communication and technology.
5. Tychonoff A, Arseny V (1977) Solution of ill-posed problems. Winston and Sons, Washington
6. Wright J, Yang A, Ganesh A, Sastry SS, Ma Y (2009) Robust face recognition via sparse representation. IEEE Trans Pattern Anal Mach Intell 31: 210–227
7. Carmi A, Gurfil P, Kanevsky D, Ramabhadran B (2009) ABCS: approximate bayesian compressive sensing. Technical Report Human Language Technologies, IBM
8. Sainath TN, Nahamoo D, Kanevsky D, Ramabhadrans B, Shah PM (2011) A convex hull approach to sparse representations for exemplar-based speech recognition. In: Proceedings of the ASRU.
9. Sainath T, Ramabhadran B, Olsen P, Kanevsky D, Nahamoo D (2011) A-Functions: a generalization of extended baum-welch transformations to convex optimization. In: Proceedings of the ICASSP.
10. Kanevsky D, Sainath TN, Ramabhadran B, Nahamoo D (2010) An analysis of sparseness and regularization in exemplar-based methods for speech classification. In: Proceedings of the interspeech.
11. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J Roy Stat Soc Ser B (Methodol.) 58(1):267–288
12. Ji S, Xue Y, Carin L (2008) Bayesian compressive sensing. IEEE Trans Signal Process 56:2346–2356
13. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Statist Soc B 67:301–320
14. Sainath TN, Ramabhadran B, Nahamoo D, Kanevsky D, Sethy A (2010) Exemplar-based sparse representation features for speech recognition. In: Proceedings of the interspeech.
15. Sainath TN, Nahamoo D, Ramabhadran B, Kanevsky D, Goel V, Shah PM (2011) Exemplar-based sparse representation phone identification features. In: Proceedings of the ICASSP.
16. Lamel L, Kassel R, Seneff S (1986) Speech database development: design and analysis of the acoustic-phonetic corpus. In: Proceedings of the DARPA speech recognition, workshop.
17. Kingsbury B (2009) Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling In: Proceedings of the ICASSP.
18. De Wachter M, Matton M, Demuynck K, Wambacq P, Cools R, Van Compernolle D (2007) Template based continuous speech recognition. IEEE Trans Audio Speech Lang Process 15(4):1377–1390
19. Sainath TN, Ramabhadran B, Nahamoo D, Kanevsky D, Sethy A (2012) Enhancing exemplar-based posteriors for speech recognition tasks. In: Proceedings of the interspeech.
20. Bellegarda J, Nahamoo D (1990) Tied mixture continuous parameter modeling for speech recognition. IEEE Trans Acous Speech Signal Process 38(12):2033–2045
21. Sainath TN, Ramabhadran B, Picheny M, Nahamoo D, Kanevsky D (2011) Exemplar-based sparse representation features: From TIMIT to LVCSR. IEEE Trans Acous Speech and Signal Process 19(8):2598–2613
22. Candes EJ, Romberg J, Tao T (2006) Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans Inf Theory 52:489–509
23. Candes EJ (2006) Compressive sampling. Proceedings of the international congress of mathematicians, European Mathematical Society, Madrid, Spain

24. Gopalakrishnan PS, Kanevsky D, Nahamoo D, Nadas A (1991) An inequality for rational functions with applications to some statistical estimation problems. IEEE Trans. Information Theory 37(1): 107–113
25. Povey D (2003) Discriminative training for large vocabulary speech recognition. Ph.D. thesis, Cambridge University.
26. Sainath T, Ramabhadran B, Olsen P, Kanevsky D, Nahamoo D (2011) Convergence of line search a-function methods. In: Proceedings of the interspeech.
27. Kanevsky D (2005) Extended baum transformations for general functions, II", Technical Report, RC23645(W0506–120). Human Language Technologies, IBM
28. Carmi A, Gurfil P, Kanevsky D Ramabhadran B (2009) Extended compressed sensing: filtering inspired methods for sparse signal recovery and their nonlinear variants. Technical Report, RC24785, Human Language Technologies, IBM.
29. Carmi A, Gurfil P, Kanevsky D, Ramabhadran B (2009) ABCS: Approximate bayesian compressed sensing. Technical Report, RC24816, Human Language Technologies, IBM.
30. Carmi A, Gurfil P, Kanevsky D (April 2010) Methods for signal recovering using kalman filtering with embedded pseudo-measurement norms and quasi-norms. IEEE Trans Signal Process 58(4):2405–2409
31. Horesh L, Gurfil P, Ramabhadran B, Kanevsky D, Carmi A, Sainath TN (2010) Kalman filtering for compressed sensing. In: Proceedings of the information fusion, Edinburgh.
32. Ji S, Xue Y, Carin L (June 2008) Bayesian compressive sensing. IEEE Trans Signal Process 56:2346–2356
33. Efron B, Hassie B, Johnstone T, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–451
34. Carmi A, Gurfil P (2009) Convex feasibility programming for compressed sensing. Technical Report, Technion
35. Mount D, Arya S (2006) ANN: A library for approximate nearest neighbor searching. Software available at http://www.cs.umd.edu/ mount/ANN/
36. Chang C, Lin C (2001) LIBSVM: A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm
37. Kanevsky D (2004) Extended baum transformations for general functions. In: Proceedings of the ICASSP.
38. Povey D, Kanevsky D, Kingsbury B, Ramabhadran B, Saon G, Visweswariah K (2008) Boosted MMI for model and feature space discriminative training. In: Proceedings of the ICASSP.
39. Chang H, Glass J (2007) Hierarchical large-marging gaussian mixture models for phonetic classification. In: Proceedings of the ASRU.
40. Sainath TN, Ramabhadran B, Picheny M (2009) An exploration of large vocabulary tools for small vocabulary phonetic recognition. In: Proceedings of the ASRU.
41. Saon G, Zweig G, Kingsbury B, Mangu L, Chaudhari U (2003) An architecture for rapid decoding of large vocabulary conversational speech. In: Proceedings of the eurospeech.
42. Deng L, Yu D (2007) Use of differential cepstra as acoustic features in hidden trajectory modeling for phonetic recognition. In: Proceedings of the ICASSP.
43. Halberstat A, Glass J (1998) Heterogeneous measurements and multiple classifiers for speech recognition. In: Proceedings of the ICSLP.
44. Mohamad A, Sainath TN, Dahl G, Ramabhadrans B, Hinton GE, Picheny M (2011) Deep belief networks using discriminative features for phone recognition. In: Proceedings of the ICASSP.