Chapter 2
Supervised Learning

Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany

Abstract Supervised learning accounts for a lot of research activity in machine learning and many supervised learning techniques have found application in the processing of multimedia content. The defining characteristic of supervised learning is the availability of annotated training data. The name invokes the idea of a 'supervisor' that instructs the learning system on the labels to associate with training examples. Typically these labels are class labels in classification problems. Supervised learning algorithms *induce* models from these training data and these models can be used to classify other unlabelled data. In this chapter we ground or analysis of supervised learning on the theory of risk minimization. We provide an overview of support vector machines and nearest neighbour classifiers – probably the two most popular supervised learning techniques employed in multimedia research.

2.1 Introduction

Supervised learning entails learning a mapping between a set of *input* variables $\mathscr{X}$ and an *output* variable $\mathscr{Y}$ and applying this mapping to predict the outputs for unseen data. Supervised learning is the most important methodology in machine learning and it also has a central importance in the processing of multimedia data. In this chapter we focus on kernel-based approaches to supervised learning. We review support vector machines which represent the dominant supervised learning technology these days – particularly in the processing of multimedia data. We also review nearest neighbour classifiers which can (loosely speaking) be considered

Pádraig Cunningham
University College Dublin, Dublin, Ireland, e-mail: padraig.cunningham@ucd.ie

Matthieu Cord
LIP6, UPMC, Paris, France, e-mail: matthieu.cord@lip6.fr

Sarah Jane Delany
Dublin Institute of Technology, Dublin, Ireland, e-mail: sarahjane.delany@comp.dit.ie

a kernel-based strategy. Nearest neighbour techniques are popular in multimedia because the emphasis on similarity is appropriate for multimedia data where a rich array of similarity assessment techniques is available.

To complete this review of supervised learning we also discuss the ensemble idea, an important strategy for increasing the stability and accuracy of a classifier whereby a single classifier is replaced by a *committee* of classifiers.

The chapter begins with a summary of the principles of statistical learning theory as this offers a general framework to analyze learning algorithms and provides useful tools for solving real world applications. We present basic notions and theorems of statistical learning before presenting some algorithms.

2.2 Introduction to Statistical Learning

2.2.1 Risk Minimization

In the supervised learning paradigm, the goal is to infer a function $f : \mathscr{X} \rightarrow \mathscr{Y}$, the classifier, from a sample data or training set $\mathscr{A}_n$ composed of pairs of (input, output) points, $\mathbf{x}_i$ belonging to some feature set $\mathscr{X}$, and $y_i \in \mathscr{Y}$:

$$\mathscr{A}_n = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)) \in (\mathscr{X} \times \mathscr{Y})^n.$$

Typically $\mathscr{X} \subset \mathbb{R}^d$, and $y_i \in \mathbb{R}$ for regression problems, and y_i is discrete for classification problems. We will often use examples with $y_i \in \{-1, +1\}$ for binary classification.

In the statistical learning framework, the first fundamental hypothesis is that the training data are independently and identically generated from an unknown but fixed joint probability distribution function $P(\mathbf{x}, y)$. The goal of the learning is to find a function f attempting to model the dependency encoded in $P(\mathbf{x}, y)$ between the input $\mathbf{x}$ and the output y. $\mathscr{H}$ will denote the set of functions where the solution is sought: $f \in \mathscr{H}$.

The second fundamental concept is the notion of error or *loss* to measure the agreement between the prediction $f(\mathbf{x})$ and the desired output y. A loss (or *cost*) function $L : \mathscr{Y} \times \mathscr{Y} \rightarrow \mathbb{R}^+$ is introduced to evaluate this error. The choice of the loss function $L(f(\mathbf{x}), y)$ depends on the learning problem being solved. Loss functions are classified according to their regularity or singularity properties and according to their ability to produce convex or non-convex criteria for optimization.

In the case of pattern recognition, where $\mathscr{Y} = \{-1, +1\}$, a common choice for L is the misclassification error:

$$L(f(\mathbf{x}), y)) = \frac{1}{2}|f(\mathbf{x}) - y|.$$

This cost is singular and symmetric. Practical algorithmic considerations may bias the choice of L. For instance, singular functions may be selected for their ability to provide *sparse* solutions. For unsupervised learning developed in Chap. 3.6, the

problem may be expressed in a similar way using a loss function: $L_u : \mathscr{Y} \rightarrow \mathbb{R}^+$ defined by: $L_u(f(\mathbf{x})) = -\log(f(\mathbf{x}))$.

The loss function L leads to the definition of the *risk* for a function f, also called the *generalization error*:

$$R(f) = \int L(f(\mathbf{x}), y) \mathrm{d}P(\mathbf{x}, y). \tag{2.1}$$

In classification, the objective could be to find the function f in $\mathscr{H}$ that minimizes $R(f)$. Unfortunately, it is not possible because the joint probability $P(\mathbf{x}, y)$ is unknown.

From a probabilistic point of view, using the input and output random variable notations X and Y, the risk can be expressed as

$$R(f) = \mathbb{E}(L(f(X), Y))$$

which can be rewritten in two expectations:

$$R(f) = \mathbb{E}[\mathbb{E}(L(f(X), Y)|X)]$$

offering the possibility to separately minimize $\mathbb{E}(L(f(\mathbf{x}), Y)|X = \mathbf{x})$ with respect to the scalar value $f(\mathbf{x})$. This expression is connected to the expected utility function introduced in Sect. 1.6.2 of this book. The resulting function is called the Bayes estimator associated with the risk R.

The learning problem is expressed as a minimization of R for any classifier f. As the joint probability is unknown, the solution is inferred from the available training set $\mathscr{A}_n = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n))$.

There are two ways to address this problem. The first approach, called generative-based, tries to approximate the joint probability $P(X, Y)$, or $P(Y|X)P(X)$, and then compute the Bayes estimator with the obtained probability. The second approach, called discriminative-based, attacks the estimation of the risk $R(f)$ head on.

Some interesting developments on probability models and estimation may be found in the Chap. 1. We focus in the following on the discriminative strategies, offering nice insights into learning theory.

2.2.2 Empirical Risk Minimization

This strategy tackles the problem of risk minimization by approximating the integral given in (2.1), using a data set $\mathscr{S}_n \in (\mathscr{X} \times \mathscr{Y})^n$ that can be the training set $\mathscr{A}_n$ or any other set:

$$R_{\text{emp}}(f, \mathscr{S}_n) = \frac{1}{n} \sum_{i=I}^{n} L(f(\mathbf{x}_i), y_i). \tag{2.2}$$

This approximation may be viewed as a Monte Carlo integration of the (2.1) as described in Chap. 1. The question is to know if the empirical error is a good approximation of the risk R.

According to the law of large numbers, there is a point-wise convergence of the empirical risk for f to $R(f)$ (as n goes to infinity). This is a motivation to minimize R_{emp} instead of the true risk over the training set $\mathscr{A}_n$; it is the principle of *empirical risk minimization* (ERM):

$$f_{\text{ERM}} = \operatorname*{Arg\,min}_{f \in \mathscr{H}} R_{\text{emp}}(f, \mathscr{A}_n) = \operatorname*{Arg\,min}_{f \in \mathscr{H}} \left(\frac{1}{n} \sum_{i=I}^{n} L(f(\mathbf{x}_i), y_i) \right).$$

However, it is not true that, for an arbitrary set of functions $\mathscr{H}$, the empirical risk minimizer will converge to the minimal risk in the class function $\mathscr{H}$ (as n goes to infinity). There are classical examples where, considering the set of all possible functions, the minimizer has a null empirical risk on the training data, but an empirical risk or a risk equal to 1 for a test data set. This shows that learning is impossible in that case. The *no free lunch* theorem [51] is related to this point.

A desirable property for the minimizers is consistency, which can be expressed in terms of probability as the data size n goes to infinity: [45, 46]:

$$\text{ERM consistent iff } \forall \varepsilon > 0 \quad \lim_{n \to \infty} P\Big(\sup_{f \in \mathscr{H}} (R(f) - R_{\text{emp}}(f, \mathscr{A}_n)) > \varepsilon \Big) = 0.$$

Thus, the learning crucially depends on the set of functions $\mathscr{H}$, and this dependency may be expressed in terms of uniform convergence that is theoretically intriguing, but not so helpful in practice. For instance, characterizations of the set of functions $\mathscr{H}$ may be useful. A set of functions with smooth decision boundaries is chosen underlying the smoothness of the decision function in real world problems. It is possible to restrict $\mathscr{H}$ by imposing a constraint of regularity to the function f. This strategy belongs to regularization theory. Instead of minimizing the empirical risk, the following regularized risk is considered:

$$R_{\text{reg}}(f) = R_{\text{emp}}(f, \mathscr{A}_n) + \lambda \Omega(f),$$

where $\Omega(f)$ is a functional introducing a roughness penalty. It will be large for functions f varying too rapidly.

One can show that minimizing a regularized risk is equivalent to ERM on a restricted set $\mathscr{F}$ of functions f. Another way to deal with the tradeoff between ERM and constraints on $f \in \mathscr{H}$ is to investigate the characterization of $\mathscr{H}$ in terms of strength or complexity for learning.

2.2.3 Risk Bounds

The idea is to find a bound depending on $\mathscr{H}$, $\mathscr{A}_n$ and δ, such that, for any $f \in \mathscr{H}$, with a probability at least $1 - \delta$:

$$R(f) \leq R_{\text{emp}}(f, \mathscr{A}_n) + B(\mathscr{H}, \mathscr{A}_n, \delta).$$

First, we consider the case of a finite class of functions, $|\mathscr{H}| = N$. Using the Hoeffding inequality (1963), by summing the probability over the whole set, one can show the following result:

$$\forall \delta \in]0,1],\ P\left(\forall f \in \mathscr{H},\ R(f) \leq R_{\text{emp}}(f, \mathscr{A}_n) + \sqrt{\frac{\log(N/\delta)}{2n}}\right) \geq 1 - \delta.$$

This inequality shows how the risk is limited by a sum of two terms, the empirical error and a bound depending on the size of the set of functions and on the empirical risk.

As $\lim_{n\to\infty} \sqrt{\frac{\log(N/\delta)}{2n}} = 0$, we have the result: for any finite class of functions, the ERM principle is consistent for any data distribution. The tradeoff between these two terms is fundamental in machine learning, it is also called the bias/variance dilemma in the literature. It is easy to see that if $\mathscr{H}$ is large, then one can find an f that fits the training data well, but at the expense of undesirable behaviour at other points, such as lack of smoothness, that will give poor performance on test data. This scenario where there is no generalization is termed overfitting. On the other hand, if $\mathscr{H}$ is too small, there is no way to find an f function that correctly fits the training data.

To go one step further, there is an extension to infinite sets of functions. Instead of working on the size of the set, a notion of complexity, the Vapnik–Chervonenkis (VC) dimension, provides a measure of the capacity of the functions to differently label the data (in classification context) [45]: The VC dimension h of a class of functions $\mathscr{F}$ is defined as the maximum number of points that can be learnt exactly by a function of $\mathscr{F}$:

$$\mathrm{h} = \max\left\{|\mathbf{X}|, \mathbf{X} \subset \mathscr{X}, \forall b \in \{-1,+1\}^{|\mathbf{X}|}, \exists f \in \mathscr{F} \mid \forall \mathbf{x}_i \in \mathbf{X}, f(\mathbf{x}_i) = b_i\right\}.$$

A strategy of bounding the risk has been developed in order to produce a new bound, depending on h, n and δ) [45]:

$$B(h,n,\delta) = \sqrt{\frac{h(\log(2n/h)+1) + \log(4/\delta)}{n}},$$

for which we have, for any $f \in \mathscr{H}$, with a probability at least $1-\delta$:

$$R(f) \leq R_{\text{emp}}(f, \mathscr{A}_n) + B(h,n,\delta).$$

Moreover, a nice result due to Vapnik [45] stipulates: *ERM is consistent (for any data distribution) iff the VC dimension h is finite.*

The tradeoff is now between controlling $B(h,n,\delta)$, which increases monotonically with the VC dimension h, and having a small empirical error on training data. The structural risk minimization (SRM) principle introduced by Vapnik exploits this last bound by considering classes of functions embedded by increasing h values.

We illustrate an aspect of the SRM over the family c_γ of linear functions to classify data $\mathscr{X} \subset \mathbb{R}^d$ with a γ margin:

$$\begin{aligned} c_{w,b,\gamma}(\mathbf{x}) &= 1 \text{ if } < \mathbf{x}, \mathbf{w} > +b \geq \gamma \\ c_{w,b,\gamma}(\mathbf{x}) &= -1 \text{ if } < \mathbf{x}, \mathbf{w} > +b \leq -\gamma \end{aligned}$$

for $\mathbf{w} \in \mathbb{R}^d, ||\mathbf{w}|| = 1$. Let $\mathscr{X}$ be included in a ball of radius R, then we have the following bound on the VC dimension h_γ:

$$h_\gamma \leq \left[\frac{R^2}{\gamma^2} \right] + 1.$$

If we can choose between two linear functions that perfectly classify the training data, it makes sense from the last bound to select the one that maximizes γ. Indeed, such a classifier is ensured to have a null empirical error $R_{\text{emp}}(f, \mathscr{A}_n)$ on the training set and to belong to a class of classifiers c_γ with the hardest bounding on h_γ (hence on $B(h_\gamma, n, \delta)$). This rule is used to build the famous support vector machines classifiers.

2.3 Support Vector Machines and Kernels

Support vector machines (SVM) are a type of learning algorithm developed in the 1990s. They are based on results from statistical learning theory introduced by Vapnik [45] described previously. These learning machines are also closely connected to kernel functions [37], which are a central concept for a number of learning tasks.

The kernel framework and SVM are used now in a variety of fields, including multimedia information retrieval (see for instance [42, 48] for CBIR applications), bioinformatics and pattern recognition.

We focus here on the introduction of SVM as linear discriminant functions for binary classification. A complete introduction to SVM and kernel theory can be found in [10] and [36].

2.3.1 Linear Classification: SVM Principle

To introduce the basic concepts of these learning machines, we start with the linear support vector approach for binary classification. We assume here that both classes are linearly separable. Let $(\mathbf{x}_i)_{i\in[1,N]}$, $\mathbf{x}_i \in \mathbb{R}^p$ be the feature vectors representing the training data and $(y_i)_{i\in[1,N]}$, $y_i \in \{-1, 1\}$ be their respective class labels. We can define a hyperplane by $< \mathbf{w}, \mathbf{x} > +b = 0$ where $\mathbf{w} \in \mathbb{R}^p$ and $b \in \mathbb{R}$. Since the classes are linearly separable, we can find a function f, $f(\mathbf{x}) =< \mathbf{w}, \mathbf{x} > +b$ with

$$y_i f(\mathbf{x}_i) = y_i(< \mathbf{w}, \mathbf{x}_i > +b) > 0, \ \forall i \in [1, N]. \tag{2.3}$$

The decision function may be expressed as $f_d(\mathbf{x}) = \text{sign}(< \mathbf{w}, \mathbf{x} > +b)$ with

$$f_d(\mathbf{x}_i) = \text{sign}(y_i),\ \forall i \in [1,N].$$

Since many functions realize the correct separation between training data, additional constraints are used. One way described in Sect. 2.2 aims to investigate the generalization properties for any of the candidates. In that context, the larger the margin, the smaller the VC dimension. This is the basic rule to express the SVM optimization: the SVM classification method aims at finding the *optimal* hyperplane based on the maximization of the *margin*[1] between the training data for both classes.

Because the distance between a point $\mathbf{x}$ and the hyperplane is $\frac{yf(\mathbf{x})}{||\mathbf{w}||}$, it is easy to show [10] that the optimization problem may be expressed as the following minimization:

$$\min \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } y_i(< \mathbf{w}, \mathbf{x}_i > +b) \geq 1,\ \forall i \in [1,N]. \tag{2.4}$$

The *support vectors* are the training points for which we have an equality in (2.4). They are all equally close to the optimal hyperplane. One can prove that they are enough to compute the separating hyperplane (hence their name).

This is a convex optimization problem (quadratic criterion, linear inequality constraints). Usually, the dual formulation is favoured for its easy solution with standard techniques. Using Lagrange multipliers, the problem may be re-expressed in the equivalent maximization on α (dual form):

$$\alpha^\star = \underset{\alpha}{\text{argmax}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j > \tag{2.5}$$

$$\text{s.t.} \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \forall i \in [1,N] \quad \alpha_i \geq 0.$$

The hyperplane decision function can be written as

$$f_d(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} y_i \alpha_i^\star < \mathbf{x}, \mathbf{x}_i > +b). \tag{2.6}$$

2.3.2 Soft Margin

The previous method is applicable to linearly separable data. When data cannot be linearly separated, the linear SVM method may be adapted. A soft margin may be used in order to get better efficiency in a noisy situation.

[1] The margin is defined as the distance from the hyperplane of the closest points, on either side.

In order to carry out the optimization, the constraints of (2.3) are relaxed using slack variables ξ_i:

$$y_i f(\mathbf{x}_i) = y_i(<\mathbf{w},\mathbf{x}_i> +b) > 1-\xi_i, \text{ with } \xi_i \geq 0 \ \forall i \in [1,N]. \tag{2.7}$$

The minimization may be then expressed as follows:

$$\min \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \tag{2.8}$$

with a constant $C > 0$, subject to the constraints (2.7).
The dual representation is obtained in a similar way:

$$\alpha^\star = \underset{\alpha}{\text{argmax}} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j <\mathbf{x}_i,\mathbf{x}_j> \tag{2.9}$$

subject to

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \quad \text{and} \quad \forall i \in [1,N] \quad 0 \leq \alpha_i \leq C.$$

It is a very simple adaptation of the original algorithm by introducing a bound C [47]. The constant C is used to tune the tradeoff between having a large margin and few classification errors.

2.3.3 Kernel-Based Classification

The linear SVM classifier previously described finds linear boundaries in the input feature space. To obtain more general decision surfaces, the feature space may be mapped into a larger space before undertaking linear classification. Linear boundaries in the enlarged space translate to non-linear boundaries in the original space.

Let us denote the induced space $\mathscr{H}$ via a map Φ:

$$\begin{aligned}\Phi : \mathbb{R}^p &\rightarrow \mathscr{H} \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}).\end{aligned}$$

In the SVM framework, there is no assumption on the dimensionality of $\mathscr{H}$, which could be very large, and sometimes infinite. We just suppose that $\mathscr{H}$ is equipped with a dot product. Maximizing (2.9) now requires the computation of dot products $<\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)>$. In practice, $\Phi(\mathbf{x})$ is never computed due to the *kernel trick*: kernel functions $k(.,.)$, respecting some conditions (positive definite, Mercer's conditions[2]), are introduced such that

$$k(\mathbf{x},\mathbf{y}) = <\Phi(\mathbf{x}),\Phi(\mathbf{y})> .$$

[2] We are dealing with the class of kernels k that correspond to dot product in the enlarged space.

Using these functions allows us to avoid to explicitly compute $\Phi(\mathbf{x})$. Everything about the linear case also applies to non-linear cases, using a suitable kernel k instead of the Euclidean dot product. The resulting optimization problem may be expressed as follows:

$$\begin{aligned} \alpha^\star &= \underset{\alpha}{\operatorname{argmax}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{with} &\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \forall i \in [1,N] \quad 0 \leq \alpha_i \leq C. \end{cases} \end{aligned} \tag{2.10}$$

Thanks to the optimal $\alpha^\star$ value, the decision function is

$$f_d(\mathbf{x}) = \operatorname{sign}(\sum_{i=1}^{N} y_i \alpha_i^\star k(\mathbf{x}, \mathbf{x}_i) + b). \tag{2.11}$$

Some popular choices for k in the SVM literature are, for $\mathscr{X} \subset \mathbb{R}^d$:

- Gaussian radial basis function kernel:

$$k(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\frac{1}{2}\left(\frac{||\mathbf{x}-\mathbf{y}||}{\sigma}\right)^2} \tag{2.12}$$

- Sigmoid kernel: $k(\mathbf{x}, \mathbf{y}) = \tanh(\eta_1 < \mathbf{x}, \mathbf{y} > + \eta_2)$
- Polynomial kernel: $k(\mathbf{x}, \mathbf{y}) = < \mathbf{x}, \mathbf{y} >^d$
- Rational kernel: $k(\mathbf{x}, \mathbf{y}) = 1 - \frac{||\mathbf{x}-\mathbf{y}||^2}{||\mathbf{x}-\mathbf{y}||^2 + b}$

For multicategory problems, the linear SVM approach may be extended using many linear machines to create boundaries consisting of sections of hyperplanes. When linear discrimination is not effective, an appropriate nonlinear mapping can be found.

For density estimation problems, adaptation of binary SVM machines is also proposed. The one-class SVM method estimates the density support of a vector set $(\mathbf{x}_i)_{i \in [1,N]}$.

2.4 Nearest Neighbour Classification

The principle underlying nearest neighbour classification is quite straightforward, examples are classified based on the class of their nearest neighbours. It is often useful to take more than one neighbour into account so the technique is more commonly referred to as k-nearest neighbour (k-NN) Classification where k nearest neighbours are used in determining the class. Since the training examples are needed at run-time, i.e. they need to be in memory at run-time, it is sometimes also called memory-based classification. Because induction is delayed to run-time, it is considered a *lazy*

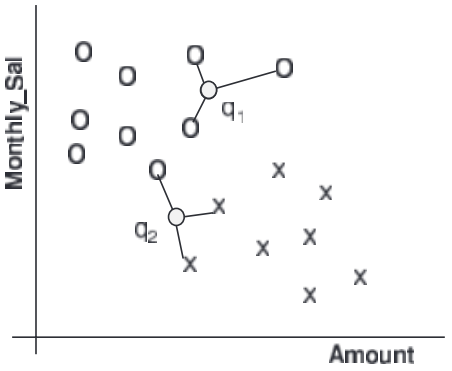

Fig. 2.1 A simple example of 3-nearest neighbour classification

learning technique. Because classification is based directly on the training examples it is also called example-based classification or case-based classification.

The basic idea is as shown in Fig. 2.1 which depicts a 3-nearest neighbour classifier on a two-class problem in a two-dimensional feature space. In this example the decision for q_1 is straightforward – all three of its nearest neighbours are of class O so it is classified as an O. The situation for q_2 is a bit more complicated at it has two neighbours of class X and one of class O. This can be resolved by simple majority voting or by distance-weighted voting (see below).

So k-NN classification has two stages; the first is the determination of the nearest neighbours and the second is the determination of the class using those neighbours.

Let us assume that we have a training data set D made up of $(\mathbf{x}_i)_{i\in[1,|D|]}$ training samples. The examples are described by a set of features F and any numeric features have been normalized to the range [0,1]. Each training example is labelled with a class label $y_j \in Y$. Our objective is to classify an unknown example $\mathbf{q}$. For each $\mathbf{x}_i \in D$ we can calculate the distance between $\mathbf{q}$ and $\mathbf{x}_i$ as follows:

$$d(\mathbf{q},\mathbf{x}_i) = \sum_{f\in F} w_f \delta(\mathbf{q}_f, \mathbf{x}_{if}). \tag{2.13}$$

There are a large range of possibilities for this distance metric; a basic version for continuous and discrete attributes would be

$$\delta(\mathbf{q}_f, \mathbf{x}_{if}) = \begin{cases} 0 & f \text{ discrete and } \mathbf{q}_f = \mathbf{x}_{if} \\ 1 & f \text{ discrete and } \mathbf{q}_f \neq \mathbf{x}_{if} \\ |\mathbf{q}_f - \mathbf{x}_{if}| & f \text{ continuous} \end{cases}. \tag{2.14}$$

The k nearest neighbours are selected based on this distance metric. Then there are a variety of ways in which the k nearest neighbours can be used to determine the class of $\mathbf{q}$. The most straightforward approach is to assign the majority class among the nearest neighbours to the query.

It will often make sense to assign more weight to the nearer neighbours in deciding the class of the query. A fairly general technique to achieve this is distance-weighted voting where the neighbours get to vote on the class of the query case with votes weighted by the inverse of their distance to the query:

$$\text{Vote}(y_j) = \sum_{c=1}^{k} \frac{1}{d(\mathbf{q}, \mathbf{x}_c)^n} 1(y_j, y_c). \tag{2.15}$$

Thus the vote assigned to class y_j by neighbour $\mathbf{x}_c$ is 1 divided by the distance to that neighbour, i.e. $1(y_j, y_c)$ returns 1 if the class labels match and 0 otherwise. In (2.15) n would normally be 1 but values greater than 1 can be used to further reduce the influence of more distant neighbours.

Another approach to voting is based on Shepard's work [38] and uses an exponential function rather than inverse distance, i.e.:

$$\text{Vote}(y_j) = \sum_{c=1}^{k} \mathrm{e}^{-\frac{d(\mathbf{q}, \mathbf{x}_c)}{h}} 1(y_j, y_c). \tag{2.16}$$

This simplicity of k-NN classifiers makes them an attractive option for use in analysing multimedia content. In this section we consider three important issues that arise with the use of k-NN classifiers. In the next section we look at the core issue of similarity and distance measures and explore some *exotic* (dis)similarity measures to illustrate the generality of the k-NN idea. In Sect. 2.4.3 we look at computational complexity issues and review some speed-up techniques for k-NN. In Sect. 4.4 we look at dimension reduction – both feature selection and sample selection. Dimension reduction is of particular importance with k-NN as it has a big impact on computational performance and accuracy. The section concludes with a summary of the advantages and disadvantages of k-NN.

2.4.1 Similarity and Distance Metrics

While the terms *similarity metric* and *distance metric* are often used colloquially to refer to any measure of affinity between two objects, the term *metric* has a formal meaning in mathematics. A metric must conform to the following four criteria (where $d(x, y)$ refers to the distance between two objects x and y):

1. $d(x, y) \geq 0$; non-negativity
2. $d(x, y) = 0$ only if $x = y$; identity
3. $d(x, y) = d(y, x)$; symmetry
4. $d(x, z) \leq d(x, y) + d(y, z)$; triangle inequality

It is possible to build a k-NN classifier that incorporates an affinity measure that is not a proper metric, however, there are some performance optimizations to the basic k-NN algorithm that require the use of a proper metric [3, 34]. In brief, these techniques can identify the nearest neighbour of an object without comparing that object to every other object but the affinity measure must be a metric, in particular it must satisfy the triangle inequality.

The basic distance metric described in (2.13) and (2.14) is a special case of the Minkowski distance metric – in fact it is the 1-norm (L_1) Minkowski distance. The general formula for the Minkowski distance is

$$MD_p(\mathbf{q},\mathbf{x}_i) = \left(\sum_{f \in F} |\mathbf{q}_f - \mathbf{x}_{if}|^p \right)^{\frac{1}{p}}. \tag{2.17}$$

The L_1 Minkowski distance is the Manhattan distance and the L_2 distance is the Euclidean distance. It is unusual but not unheard of to use p values greater than 2. Larger values of p have the effect of giving greater weight to the attributes on which the objects differ most. To illustrate this we can consider three points in 2D space; $A = (1,1), B = (5,1)$ and $C = (4,4)$. Since A and B differ on one attribute only the $MD_p(A,B)$ is 4 for all p, whereas $MD_p(A,C)$ is 6, 4.24 and 3.78 for p values of 1, 2 and 3, respectively. So C becomes the nearer neighbour to A for p values of 3 and greater.

The other important Minkowski distance is the L_∞ or Chebyshev distance:

$$MD_\infty(\mathbf{q},\mathbf{x}_i) = \max_{f \in F} |\mathbf{q}_f - \mathbf{x}_{if}|.$$

This is simply the distance in the dimension in which the two examples are most different; it is sometimes referred to as the chessboard distance as it is the number of moves it takes a chess king to reach any square on the board.

In the remainder of this section we will review a selection of other metric distances that are important in multimedia analysis.

2.4.2 Other Distance Metrics for Multimedia Data

The Minkowski distance defined in (2.17) is a very general metric that can be used in a k-NN classifier for any data that is represented as a feature vector. When working with image data a convenient representation for the purpose of calculating distances is a colour histogram. An image can be considered as a grey-scale histogram H of N levels or bins where h_i is the number of pixels that fall into the interval represented by bin i (this vector h is the feature vector). The Minkowski distance formula (2.17) can be used to compare two images described as histograms. L_1, L_2 and less often L_∞ norms are used.

Other popular measures for comparing histograms are the Kullback–Leibler divergence (2.18) [24] and the χ^2 statistic (2.19) [33]:

$$d_{KL}(H,K) = \sum_{i=1}^{N} h_i \log\left(\frac{h_i}{k_i}\right), \tag{2.18}$$

$$d_{\chi^2}(H,K) = \sum_{i=1}^{N} \frac{h_i - m_i}{h_i}, \tag{2.19}$$

where H and K are two histograms, h and k are the corresponding vectors of bin values and $m_i = \frac{h_i + k_i}{2}$.

While these measures have sound theoretical support in information theory and in statistics they have some significant drawbacks. The first drawback is that they

are not metrics in that they do not satisfy the symmetry requirement. However, this problem can easily be overcome by defining a modified distance between x and y that is in some way an average of $d(x,y)$ and $d(y,x)$ – see [33] for the Jeffrey divergence which is a symmetric version of the Kullback–Leibler divergence.

A more significant drawback is that these measures are prone to errors due to bin boundaries. The distance between an image and a slightly darker version of itself can be great if pixels fall into an adjacent bin as there is no consideration of adjacency of bins in these measures.

2.4.2.1 Earth Mover Distance

The earth mover distance (EMD) is a distance measure that overcomes many of these problems that arise from the arbitrariness of binning. As the name implies, the distance is based on the notion of the amount of effort required to convert one image to another based on the analogy of transporting *mass* from one distribution to another. If we think of two images as distributions and view one distribution as a mass of earth in space and the other distribution as a hole (or set of holes) in the same space then the EMD is the minimum amount of work involved in filling the holes with the earth.

In their analysis of the EMD Rubner et al. argue that a measure based on the notion of a *signature* is better than one based on a histogram. A signature $\{\mathbf{s}_j = \mathbf{m}_j, w_{\mathbf{m}_j}\}$ is a set of j clusters where $\mathbf{m}_j$ is a vector describing the mode of cluster j and $w_{\mathbf{m}_j}$ is the fraction of pixels falling into that cluster. Thus a signature is a generalization of the notion of a histogram where boundaries and the number of partitions are not set in advance; instead j should be 'appropriate' to the complexity of the image [33].

The example in Fig. 2.2 illustrates this idea. We can think of the clustering as a quantization of the image in some colour space so that the image is represented by a set of cluster modes and their weights. In the figure the source image is represented in a 2D space as two points of weights 0.6 and 0.4; the target image is represented by three points with weights 0.5, 0.3 and 0.2. In this example the EMD is calculated to be the sum of the amounts moved (0.2, 0.2, 0.1 and 0.5) multiplied by the distances they are moved. Calculating the EMD involves discovering an assignment that minimizes this amount.

For two images described by signatures $S = \{\mathbf{m}_j, w_{\mathbf{m}_j}\}_{j=1}^{n}$ and $Q = \{\mathbf{p}_k, w_{\mathbf{p}_k}\}_{k=1}^{r}$ we are interested in the work required to transfer from one to the other for a given flow pattern $\mathbf{F}$:

$$\mathrm{WORK}(S,Q,\mathbf{F}) = \sum_{j=1}^{n}\sum_{k=1}^{r} d_{jk} f_{jk}, \tag{2.20}$$

where d_{jk} is the distance between clusters $\mathbf{m}_j$ and $\mathbf{p}_k$ and f_{jk} is the flow between $\mathbf{m}_j$ and $\mathbf{p}_k$ that minimizes overall cost. An example of this in a 2D colour space is shown in Fig. 2.2. Once the transportation problem of identifying the flow that

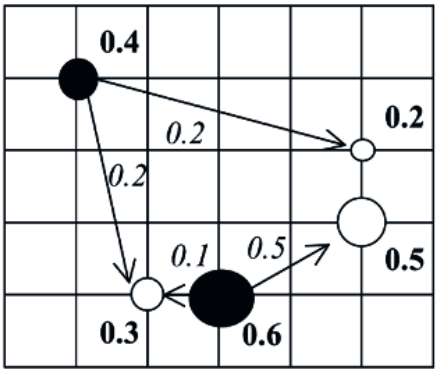

Fig. 2.2 An example of the EMD between two 2D signatures with two points (clusters) in one signature and three in the other (based on example in [32])

minimizes effort is solved (using dynamic programming) the EMD is defined to be

$$\mathrm{EMD}(S,Q) = \frac{\sum_{j=1}^{n}\sum_{k=1}^{r} d_{jk} f_{jk}}{\sum_{j=1}^{n}\sum_{k=1}^{r} f_{jk}}. \tag{2.21}$$

Efficient algorithms for the EMD are described in [33]; however, this measure is expensive to compute with cost increasing more than linearly with the number of clusters. Nevertheless it is an effective measure for capturing similarity between images.

2.4.2.2 Compression-Based Dissimilarity

In recent years the idea of basing a similarity metric on compression has received a lot of attention [21, 28]. Indeed Li et al. [28] refer to this as *The* similarity metric. The basic idea is quite straightforward; if two documents are very similar then the compressed size of the two documents concatenated together will not be much greater than the compressed size of a single document. This will not be true for two documents that are very different. Slightly more formally, the difference between two documents A and B is related to the compressed size of document B when compressed using the codebook produced when compressing document A.

The theoretical basis of this metric is in the field of Kolmogorov complexity, specifically in conditional Kolmogorov complexity:

$$d_{Kv}(x,y) = \frac{Kv(x|y) + Kv(y|x)}{Kv(xy)}, \tag{2.22}$$

where $Kv(x|y)$ is the length of the shortest program that computes x when y is given as an auxiliary input to the program and $Kv(xy)$ is the length of the shortest program that outputs y concatenated to x. While this is an abstract idea it can be approximated using compression

$$d_C(x,y) = \frac{C(x|y) + C(y|x)}{C(xy)}. \tag{2.23}$$

$C(x)$ is the size of data x after compression, and $C(x|y)$ is the size of x after compressing it with the compression model built for y. If we assume that $Kv(x|y) \approx Kv(xy) - Kv(y)$ then we can define a normalized compression distance

$$d_{NC}(x,y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))}. \tag{2.24}$$

It is important that $C(.)$ should be an appropriate compression metric for the data. Delany and Bridge [12] show that compression using Lempel–Ziv (GZip) is effective for text. They show that this compression-based metric is more accurate in k-NN classification than distance-based metrics on a bag-of-words representation of the text.

2.4.3 Computational Complexity

Computationally expensive metrics such as the earth mover's distance and compression-based (dis)similarity metrics focus attention on the computational issues associated with k-NN classifiers. Basic k-NN classifiers that use a simple Minkowski distance will have a time behaviour that is $O(|D||F|)$ where D is the training set and F is the set of features that describe the data, i.e. the distance metric is linear in the number of features and the comparison process increases linearly with the amount of data. The computational complexity of the EMD and compression metrics is more difficult to characterize but a k-NN classifier that incorporates an EMD metric is likely to be $O(|D|n^3 \log n)$ where n is the number of clusters [33].

For these reasons there has been considerable research on editing down the training data and on reducing the number of features used to describe the data (see Sect. 4.4). There has also been considerable research on alternatives to the exhaustive search strategy that is used in the standard k-NN algorithm. Here is a summary of four of the strategies for speeding up nearest neighbour retrieval:

- **Case-retrieval nets:** k-NN retrieval is widely used in case-based reasoning and case-retrieval nets (CRNs) are perhaps the most popular technique for speeding up the retrieval process. Again, the cases are pre-processed, but this time to form a network structure that is used at retrieval time. The retrieval process is done by *spreading activation* in this network structure. CRNs can be configured to return exactly the same cases as k-NN [26, 27].
- **Footprint-based retrieval:** As with all strategies for speeding up nearest neighbour retrieval, footprint-based retrieval involves a preprocessing stage to organize the training data into a two-level hierarchy on which a two-stage retrieval process operates. The preprocessing constructs a competence model which identifies 'footprint' cases which are landmark cases in the data. This process is not

guaranteed to retrieve the same cases as k-NN but the results of the evaluation of speed-up and retrieval quality are nevertheless impressive [40].

- **Fish & Shrink:** This technique requires the distance to be a true metric as it exploits the triangle inequality property to produce an organization of the case-base into candidate neighbours and cases excluded from consideration. Cases that are remote from the query can be bounded out so that they need not be considered in the retrieval process. Fish & Shrink can be guaranteed to be equivalent to k-NN [34].
- **Cover trees for nearest neighbour:** This technique might be considered the state-of-the-art in nearest neighbour speed-up. It uses a data structure called a cover tree to organize the cases for efficient retrieval. The use of cover trees requires that the distance measure is a true metric; however, they have attractive characteristics in terms of space requirements and speed-up performance. The space requirement is $O(n)$ where n is the number of cases; the construction time is $O(c^6 n \log n)$ and the retrieval time is $O(c^{12} \log n)$ where c is a measure of the inherent dimensionality of the data [3].

These techniques involve additional preprocessing to construct data structures that are used to speed-up retrieval. Consequently they are more difficult to implement than the standard k-NN algorithm. As emphasized at the beginning of this section, the alternative speed-up strategy is to reduce the dimension of the data – this is covered in the next section.

2.4.4 Instance Selection and Noise Reduction

An area of instance-based learning that has prompted much recent research is case-base editing, which involves reducing the number of cases in the training set while maintaining or even improving performance.

2.4.4.1 Early Techniques

Case-base editing techniques have been categorized by Brighton and Mellish [6] as competence preservation or competence enhancement techniques. Competence preservation corresponds to redundancy reduction, removing superfluous cases that do not contribute to classification competence. Competence enhancement is effectively noise reduction, removing noisy or corrupt cases from the training set. Figure 2.3 illustrates both of these, where cases of one class are represented by stars and cases of the other class are represented by circles. Competence preservation techniques aim to remove internal cases in a cluster of cases of the same class and can predispose towards preserving noisy cases as exceptions or border cases. Noise reduction on the other hand aims to remove noisy or corrupt cases but can remove exceptional or border cases which may not be distinguishable from true noise, so a balance of both can be useful.

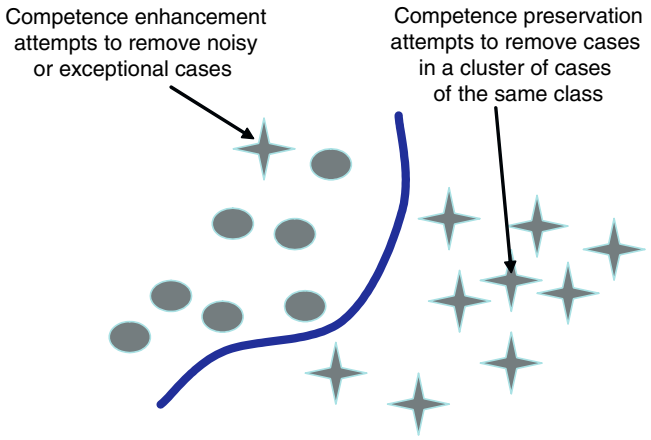

Fig. 2.3 Case-base editing techniques demonstrating competence preservation and competence enhancement

Editing strategies normally operate in one of two ways; *incremental* which involves adding selected cases from the training set to an initially empty edited set, and *decremental* which involves contracting the training set by removing selected cases.

An early competence preservation technique is Hart's condensed nearest neighbour (CNN) [18]. CNN is an incremental technique which adds to an initially empty edited set any case from the training set that cannot be classified correctly by the edited set. This technique is very sensitive to noise and to the order of presentation of the training set cases, in fact CNN by definition will tend to preserve noisy cases. Ritter et al. [31] reported improvements on the CNN with their selective nearest neighbour (SNN) which imposes the rule that every case in the training set must be closer to a case of the same class in the edited set than to any other training case of a different class. Gates [16] introduced a decremental technique which starts with the edited set equal to the training set and removes a case from the edited set where its removal does not cause any other training case to be misclassified. This technique will allow for the removal of noisy cases but is sensitive to the order of presentation of cases.

Competence enhancement or noise reduction techniques start with Wilson's edited nearest neighbour (ENN) algorithm [50], a decremental strategy, which removes cases from the training set which do not agree with their k nearest neighbours. These cases are considered to be noise and appear as exceptional cases in a group of cases of the same class.

Tomek [41] extended this with his repeated ENN (RENN) and his *all k-NN* algorithms. Both make multiple passes over the training set, the former repeating the ENN algorithm until no further eliminations can be made from the training set and the latter using incrementing values of k. These techniques focus on noisy or exceptional cases and do not result in the same storage reduction gains as the competence preservation approaches.

Later editing techniques can be classified as hybrid techniques incorporating both competence preservation and competence enhancement stages. Aha et al. [1] presented a series of instance-based learning algorithms to reduce storage requirements

and tolerate noisy instances. IB2 is similar to CNN adding only cases that cannot be classified correctly by the reduced training set. IB2's susceptibility to noise is handled by IB3 which records how well cases are classified and only keeps those that classify correctly to a statistically significant degree. Other researchers have provided variations on the IBn algorithms [7, 8, 52].

2.4.4.2 Competence-Based Case-Base Editing

More recent approaches to case-base editing build a competence model of the training data and use the competence properties of the cases to determine which cases to include in the edited set. Measuring and using case competence to guide case-base maintenance was first introduced by Smyth and Keane [39] and developed by Zu and Yang [53]. Smyth and Keane [39] introduce two important competence properties, the *reachability* and *coverage* sets for a case in a case-base. The *reachability set* of a case c is the set of all cases that can successfully classify c, and the *coverage set* of a case c is the set of all cases that c can successfully classify. The coverage and reachability sets represent the local competence characteristics of a case and are used as the basis of a number of editing techniques.

McKenna and Smyth [29] presented a family of competence-guided editing methods for case-bases which combine both incremental and decremental strategies. The family of algorithms is based on four features;

(i) An *ordering policy* for the presentation of the cases that is based on the competence characteristics of the cases,
(ii) An *addition rule* to determine the cases to be added to the edited set,
(iii) A *deletion rule* to determine the cases to be removed from the training set and
(iv) An *update policy* which indicates whether the competence model is updated after each editing step.

The different combinations of ordering policy, addition rule, deletion rule and update policy produce the family of algorithms.

Brighton and Mellish [6] also use the coverage and reachability properties of cases in their iterative case filtering (ICF) algorithm. ICF is a decremental strategy contracting the training set by removing those cases c, where the number of other cases that can correctly classify c is higher that the number of cases that c can correctly classify. This strategy focuses on removing cases far from class borders. After each pass over the training set, the competence model is updated and the process repeated until no more cases can be removed. ICF includes a preprocessing noise reduction stage, effectively RENN, to remove noisy cases. McKenna and Smyth compared their family of algorithms to ICF and concluded that the overall best algorithm of the family delivered improved accuracy (albeit marginal, 0.22%) with less than 50% of the cases needed by the ICF edited set [29].

Wilson and Martinez [49] present a series of reduction technique (RT) algorithms, RT1, RT2 and RT3 which, although published before the definitions of coverage and reachability, could also be considered to use a competence model. They define the set of associates of a case c which is comparable to the coverage set of

McKenna and Smyth except that the associates set will include cases of a different class from case c whereas the coverage set will only include cases of the same class as c. The RTn algorithms use a decremental strategy. RT1, the basic algorithm, removes a case c if at least as many of its associates would still be classified correctly without c. This algorithm focuses on removing noisy cases and cases at the centre of clusters of cases of the same class as their associates which will most probably still be classified correctly without them. RT2 fixes the order of presentation of cases as those furthest from their nearest unlike neighbour (i.e. nearest case of a different class) to remove cases furthest from the class borders first. RT2 also uses the original set of associates when making the deletion decision, which effectively means that the associate's competence model is not rebuilt after each editing step which is done in RT1. RT3 adds a noise reduction preprocessing pass based on Wilson's noise reduction algorithm.

Wilson and Martinez [49] concluded from their evaluation of the RTn algorithms against IB3 that RT3 had a higher average generalization accuracy and lower storage requirements overall but that certain data sets seem well suited to the techniques while others were unsuited. Brighton and Mellish [6] evaluated their ICF against RT3 and found that neither algorithm consistently outperformed the other and both represented the 'cutting edge in instance set reduction techniques'.

2.4.5 k-NN: Advantages and Disadvantages

k-NN is very simple to understand and easy to implement. So it should be considered in seeking a solution to any classification problem. Some advantages of k-NN are as follows (many of these derive from its simplicity and interpretability):

- Because the process is transparent, it is easy to implement and debug.
- In situations where an explanation of the output of the classifier is useful, k-NN can be very effective if an analysis of the neighbours is useful as explanation.
- There are some noise reduction techniques that work only for k-NN that can be effective in improving the accuracy of the classifier [13].
- Case-retrieval nets [26] are an elaboration of the memory-based classifier idea that can greatly improve run-time performance on large case-bases.

These advantages of k-NN, particularly those that derive from its interpretability, should not be underestimated. On the other hand, some significant disadvantages are as follows:

- Because all the work is done at run-time, k-NN can have poor run-time performance if the training set is large.
- k-NN is very sensitive to irrelevant or redundant features because all features contribute to the similarity (see (2.13)) and thus to the classification. This can be ameliorated by careful feature selection or feature weighting.
- On very difficult classification tasks, k-NN may be outperformed by more *exotic* techniques such as support vector machines or neural networks.

2.5 Ensemble Techniques

2.5.1 Introduction

The key idea in ensemble research is in many situations a committee of classifiers will produce better results than a single classifier – better in terms of stability and accuracy. This is particularly the case when the component classifiers are unstable as is the case with neural networks and decision trees. While the use of ensembles in machine learning research is fairly new, the idea that aggregating the opinions of a committee of experts will increase accuracy is not new. The Condorcet jury theorem states that:

If each voter has a probability p of being correct and the probability of a majority of voters being correct is P, then $p > 0.5$ *implies* $P > p$*. In the limit, P approaches* 1*, for all* $p > 0.5$*, as the number of voters approaches infinity.*

This theorem was proposed by the Marquis of Condorcet in 1784 [9] – a more accessible reference is by Nitzan and Paroush [30]. We know now that P will be greater than p only if there is diversity in the pool of voters. And we know that the probability of the ensemble being correct will only increase as the ensemble grows if the diversity in the ensemble continues to grow as well. Typically the diversity of the ensemble will plateau as will the accuracy of the ensemble at some size between 10 and 50 members.

In ML research it is well known that ensembling will improve the performance of unstable learners. Unstable learners are learners where small changes in the training data can produce quite different models and thus different predictions. Thus, a ready source of diversity is to train models on different subsets of the training data. This approach has been applied with great success in eager learning systems such as neural networks [17] or decision trees [4]. This research shows that, for difficult classification and regression tasks, ensembling will improve the performance of unstable learning techniques such as neural networks and decision trees. Ensembling will also improve the accuracy of more stable learners such as k-NN or Naïve Bayes classifiers; however, these techniques are relatively stable in the face of changes in training data so other sources of diversity must be employed. Perhaps the most popular choice for stable classifiers is to achieve diversity by training different classifiers on using different feature subsets [11, 19, 20].

Krogh and Vedelsby [23] have shown that the reduction in error due to an ensemble is directly proportionate to the diversity or ambiguity in the predictions of the components of the ensemble as measured by variance. It is difficult to show such a direct relationship for classification tasks but it is clear that the uplift due to the ensemble depends on the diversity in the ensemble members.

Colloquially, we can say that if the ensemble members are more likely on average to be right, and when they are wrong they are wrong at different points, then their decisions by majority voting are more likely to be right than that of individual members. But they must be more likely on average to be right and when they are wrong they must be wrong in different ways.

2.5.2 Bias–Variance Analysis of Error

In their seminal paper on the error analysis of zero–one loss functions Kohavi and Wolpert [22] develop the idea that the error of a classifier can be divided into three components.

- **Intrinsic 'target noise':** This noise is inherent in the learning problem and is effectively a lower bound on the error that can be achieved by the classifier. It reflects shortcomings in the potential of the available features to capture the phenomenon.
- **Bias:** This captures how the average guess of the learning algorithm (overall possible training sets of the given training set size) matches the target.
- **Variance:** This quantifies how much the learning algorithm 'bounces around' for the different training sets of a given size.

In this analysis the intrinsic error is something we cannot do anything about. However, in adjusting a classifier to reduce error, it is useful to be mindful of what aspect of error we are trying to reduce. It is well known that while neural nets can fit a model to training data very well they are unstable learners. In terms of the bias–variance decomposition of error a neural net has low bias but high variance. By contrast a logistic regression model will have low variance error but a high bias. This is because logistic regression models are simpler than neural networks. It is normally the case that attempts to reduce bias will increase variance and vice versa; thus there is a *tension* between these two types of error.

It should also be clear from the definition of the variance component of error given above that a process that averages the output of several models should have the effect of reducing error due to model variance. This is what happens in bagging as explained in Sect. 2.5.3. The boosting approach to building ensembles (Sect. 2.5.5) is even better in that it can reduce *both* the variance and bias components of error.

2.5.3 Bagging

The simplest way to generate an ensemble of unstable classifiers such as neural nets or decision trees is to use bootstrap aggregation, more commonly known as bagging [4]. The basic idea for a bagging ensemble is shown in Fig. 2.4; given a set of training data D and a query sample $\mathbf{q}$ the key steps are as follows:

1. For an ensemble of S members, generate S training sets T_i $(i = 1,S)$ from D by bootstrap sampling, i.e. sampling with replacement. Often $|D_i| = |D|$.
2. For each D_i; let Dv_i be the set of training examples not selected in D_i (this set can be used as a validation set to control the overfitting of the ensemble member trained with D_i). Dv_i is often called the 'out-of-bag' (OOB) data.
3. Train S classifiers $f_i(,D_i)$ using the D_i training sets. The validation sets Dv_i can be used to control overfitting.

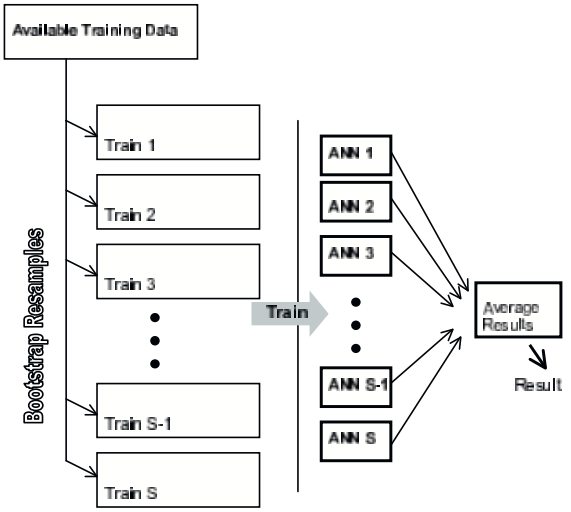

Fig. 2.4 An overview of a bagging ensemble

4. Generate S predictions for $\mathbf{q}$ using the S classifiers $f_i(,D_i)$.
5. Aggregate these S predictions $f_i(\mathbf{q},D_i)$ to get a single prediction for $\mathbf{q}$ using some aggregation function.

The formula for this ensemble prediction would be

$$f_E(\mathbf{q},D) = F(f_1(\mathbf{q},D_1), f_2(\mathbf{q},D_2), ..., f_S(\mathbf{q},D_S)). \tag{2.25}$$

The simplest approach to aggregate a regression ensemble is by averaging and the simplest for a classification ensemble is by averaging or by *weighted* averaging:

$$f_E(\mathbf{q},D) = \sum_{i=1}^{S} w_i \times f_i(\mathbf{q},D_i), \tag{2.26}$$

where $\sum_{i=1}^{S} w_i = 1$.

Provided there is diversity in the ensemble (and the bootstrap resampling should deliver this), the predictions of the ensemble $f_E(\mathbf{q},D)$ will be more accurate than the predictions from the ensemble members $f_i(\mathbf{q},D_i)$.

It is important to note that 'bagging' (i.e. sub-sampling the training data) is only a source of diversity if the classifier is unstable. For stable classifiers such as k-NN or Naïve Bayes sub-sampling the training data will not produce diverse ensemble members. In that situation an alternative strategy is to sub-sample the features rather than the examples. This can be quite effective when the data is described by a large number of features and there is redundancy in this representation.

2.5.3.1 Quantifying Diversity

Krogh and Vedelsby [23] have shown that the following very simple relationship holds for regression ensembles:

$$E = \bar{E} - \bar{A}. \tag{2.27}$$

This says that the reduction in error due to an ensemble is directly proportionate to the diversity or ambiguity in the predictions of the components of the ensemble as measured by variance of the predictions. Unfortunately there appears to be no intuitive means of quantifying diversity in classification that has the same direct relationship to the reduction in error due to the ensemble. Nevertheless several measures of diversity have been evaluated to assess their ability to quantify the reduction in error in an ensemble [25, 43].

The evaluation presented by Tsymbal et al. [43] found the following four measures to be effective:

- **Plain disagreement:** This is a pair-wise measure that quantifies the proportion of instances on which a pair of classifiers i and j disagree:

$$\text{div\_plain}_{i,j} = \frac{1}{|D|} \sum_{k=1}^{|D|} \text{Diff}(C_i(\mathbf{x}_k), C_j(\mathbf{x}_k)), \tag{2.28}$$

  where D is the data set, $C_i(\mathbf{x}_k)$ is the class assigned to instance k by classifier i and Diff() returns 1 if its two arguments are different.
- **Fail/non-fail disagreement:** This is the percentage of test instances for which the classifiers make different predictions but for which one of them is correct:

$$\text{div\_dis}_{i,j} = \frac{N^{01} + N^{10}}{|D|}, \tag{2.29}$$

  where N^{01} is the number of instances correctly classified by classifier j and not correctly classified by classifier i.
- **Entropy:** A non-pairwise measure based on entropy was proposed by Cunningham and Carney [11]:

$$\text{div\_ent} = \frac{1}{|D|} \sum_{i=1}^{|D|} \sum_{k=1}^{l} -\frac{N_k^i}{S} \log \frac{N_k^i}{S}, \tag{2.30}$$

  where S is the number of classifiers in the ensemble, l the number of classes and N_k^i the number of classifiers that assign class k to instance i.
- **Ambiguity:** This measure adapts the variance-based diversity from regression problems [23] for use in classification. An l class classification task is considered as l pseudo-regression problems. The diversity of the classification ensemble can then be calculated as the average ambiguity over these pseudo-regression tasks for each of the instances:

$$\begin{aligned} \text{div\_amb} &= \frac{1}{l|D|} \sum_{i=1}^{l} \sum_{j=1}^{|D|} \text{Ambiguity}_{i,j} \\ &= \frac{1}{l|D|} \sum_{i=1}^{l} \sum_{j=1}^{|D|} \sum_{k=1}^{S} \left(\text{Is}(C_k(\mathbf{x}_j) = i) - \frac{N_i^j}{S} \right)^2, \end{aligned} \tag{2.31}$$

where Is() is a truth predicate.

The entropy and ambiguity measures quantify the diversity of the ensemble as a whole. The two 'disagreement' measures are pair-wise measures that quantify the disagreement between a pair of base-classifiers; for these, the total ensemble diversity is the average of the disagreement of all the pairs of classifiers in the ensemble.

The evaluation presented by Tsymbal et al. [43] showed that these four diversity measures were well correlated with the uplift in accuracy due to the ensemble with *div_dis* showing slightly better correlation than the other three.

2.5.4 Random Forests

In Sect. 2.5.3 we mentioned that the two most popular strategies for introducing diversity into an ensemble are sub-sampling the examples (bagging) and sub-sampling the features (feature selection). Breiman has used both of these ideas in the random forest strategy for developing ensembles [5]. It is worth looking at random frorests in some detail because some important additional benefits emerge from the random forest strategy – these are described at the end of this section.

As the name suggests, a random forest is an ensemble of decision trees. The two sources of diversity are manifest in the algorithm as follows:

1. As in bagging, for each ensemble member the training set D is sub-sampled with replacement to produce a training set of size $|D|$.
2. Where F is the set of features that describes the data, $m << |F|$ is selected as the number of features to be used in the feature selection process. At each stage (i.e. node) in the building of a tree m features are selected at random to be the candidates for splitting at that node.

In order to ensure diversity among the component trees no pruning is employed as would be normal in building decision trees. The OOB data can be used to assess the generalization accuracy of the ensemble members. It is normal when building a random forest to generate many more ensemble members that would be used in bagging – 100 or even 1000 trees might be built. The effort expended on building these trees has the added benefit of providing an analysis of the data. It is worth highlighting three of these benefits here:

- **Estimation of generalization error:** The OOB data directly offers an estimate of the generalization error of the component trees. However, it can also be used to get an estimate of the generalization error of the complete ensemble that is unbiased [5]. Since each case is out-of-bag in roughly 1/3 of trees, the majority vote from those trees can be taken as the class the ensemble would predict for that case. The error on these aggregated out-of-bag predictions is an unbiased estimate of the error of the random forest as a whole.
- **Case proximities:** When many trees are being built, an interesting statistic to track is the frequency with which cases (both training and OOB) are located at the same leaf node. Every leaf node in every tree is examined and an $|D| \times |D|$ matrix is maintained where cell (i, j) is incremented each time cases i and j share

the same leaf node. If the matrix entries are divided by the number of trees we have a proximity measure that is *in tune* with the classification algorithm (the random forest).

- **Variable importance:** The basic ideas in assessing the importance of a variable using a random forest is to look at the impact of randomly permuting values of that variable in OOB cases and reclassifying these permuted cases. If the error increases significantly when a variable is *noised* in this way then that variable is important. If the error does not increase then that variable is not useful for the classification.

In recent years there has been a shift of emphasis in ML research as increases in computing power remove concerns about computational aspects of algorithms. Instead we are interested in useful ways to *spend* the significant computational resources available [15]. The random forest idea is a useful innovation in this regard.

2.5.5 Boosting

Boosting is a more deliberate approach to ensemble building. Instead of building the component classifiers all at once as in bagging, in boosting, the classifiers are built sequentially. The principle can be explained with reference to Fig. 2.5. This figure shows a true decision surface and a decision surface that has been learned by the classifier. The errors due to the discrepancy between the true decision surface and the learned one are highlighted. The idea with boosting is to focus on these errors in building subsequent classifiers. It is because of this that boosting reduces the bias component of error as well as the variance component. In boosting, classifiers are built sequentially with subsequent classifiers focusing on training examples that have not been learned well by earlier classifiers. This is achieved by adjusting the sampling distribution of the training data. The details as outlined by Schapire [35] are as follows (where $(x_1, y_1), ..., (x_{|D|}, y_{|D|})$ with $x_i \in \mathbf{X}, y_i \in Y = \{-1, +1\}$ is the available training data):

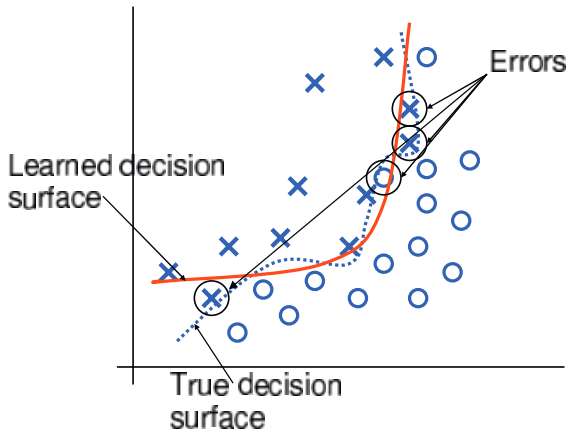

Fig. 2.5 Difficult to classify examples near the decision surface

1. Initialize the sampling distribution $P_1(i) = 1/|D|$
2. For $t = 1, ..., T$:
   a. Train classifier using distribution P_t.
   b. Hypothesis this classifier represents is $h_t : \mathbf{X} \rightarrow \{-1, +1\}$.
   c. Estimate error of this classifier as $\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} P_t(i)$.
   d. Let $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$.
   e. Update
$$P_{t+1}(i) = \frac{P_t(i)}{Z_t} \times \begin{cases} e^{-\alpha t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha t} & \text{if } h_t(x_i) \neq y_i \end{cases}, \tag{2.32}$$
   where Z_t is a normalization factor set to ensure that P_{t+1} will be a distribution.
   f. Continue training new classifiers while $\varepsilon_t < 0.5$.
3. This ensemble of classifiers can be used to produce a classification as follows:
$$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)). \tag{2.33}$$

Clearly, the role that classification error plays in this algorithm means that this formulation only works for classification problems. However, extensions to the boosting idea for regression problems exist; two representative examples are AdaBoost.R2 by Drucker [14] and BEM by Avnimelech and Intrator [2].

ssec: It is generally the case that the generalization error of a classifier will be improved by building an ensemble of such classifiers and aggregating the results from these component classifiers. Even if the reduction in error is modest (of the order of a couple of per cent) and the computational cost increases by an order of magnitude it will probably still be worth it as the computational resources are available.

The simplest strategy for building an ensemble is bagging where diverse component classifiers are built by sub-sampling the training cases or sub-sampling the features. Boosting will often produce even better improvements on error than bagging as it has the potential to reduce the bias component of error in addition to the variance component. Finally, random forests are an interesting strategy for building ensembles that can provide some useful insights into the data in addition to providing a very effective classifier.

2.6 Summary

Supervised learning is the dominant methodology in machine learning. Supervised learning techniques are more powerful than unsupervised techniques because the availability of labelled training data provides clear criteria for model optimization. In the analysis presented here risk minimization is presented as the appropriate criteria to optimize in supervised learning and SVMs are presented as a learning model

that implements risk minimization. This overview of supervised learning techniques for multimedia data analysis is completed with descriptions of nearest neighbour classifiers and ensembles, two other techniques that are widely used in multimedia data analysis.

Readers interested in exploring the theoretical underpinnings of these learning techniques should explore PAC learning theory (probably approximately correct) as presented by Valiant [44] and the work of Vapnik on risk minimization and SVMs [45, 46]. The theoretical foundations of boosting as presented by Schapire [35] are also worth exploration.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. R. Avnimelech and N. Intrator. Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, 11(2):483–497, 1999.
3. A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of 23rd International Conference on Machine Learning (ICML 2006)*, 2006.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
6. H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6(2):153–172, 2002.
7. C. Brodley. Addressing the selective superiority problem: Automatic algorithm/mode class selection. In *Proceedings of the 10th International Conference on Machine Learning (ICML 93)*, pages 17–24. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
8. R. M. Cameron-Jones. Minimum description length instance-based learning. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 368–373. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.
9. Marquis J. A. Condorcet. Sur les elections par scrutiny. *Histoire de l'Academie Royale des Sciences*, 31–34, 1781.
10. N. Cristianini and J. Shawe-Taylor. An introduction to support vector machines. Cambridge University Press, Cambridge, 2000.
11. P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In Ramon López de Mántaras and Enric Plaza, editors, *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 31–June 2, 2000, Proceedings*, pages 109–116. Springer, New York, 2000.
12. S.J. Delany and D. Bridge. Feature-based and feature-free textual cbr: A comparison in spam filtering. In D. Bell, P. Milligan, and P. Sage, editors, *Proceedings of the 17th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'06)*, pages 244–253, 2006.
13. S.J. Delany and P. Cunningham. An analysis of case-base editing in a spam filtering system. In *7th European Conference on Case-Based Reasoning*. Springer Verlag, New York, 2004.
14. H. Drucker. Improving regressors using boosting techniques. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8–12, 1997*, pages 107–115. Morgan Kaufmann, San Francisco, CA, USA, 1997.
15. S. Esmeir and S. Markovitch. Anytime induction of decision trees: An iterative improvement approach. In *AAAI*. AAAI Press, Menlo Park, CA, USA, 2006.
16. G. W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.

17. L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
18. P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.
19. T. K. Ho. Nearest neighbors in random subspaces. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR '98 and SPR '98, Sydney, NSW, Australia, August 11–13, 1998, Proceedings*, pages 640–648. Springer, New York, 1998.
20. T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
21. E. J. Keogh, S. Lonardi, and C. Ratanamahatana. Towards parameter-free data mining. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *KDD*, pages 206–215. ACM, New York, Ny, USA, 2004.
22. R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero–one loss functions. In *ICML*, pages 275–283. Morgan Kaufmann, 1996.
23. A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pages 231–238. MIT Press, Cambridge, MA, USA, 1994.
24. S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.
25. L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.
26. M. Lenz and H-D. Burkhard. Case retrieval nets: Basic ideas and extensions. In *KI - Kunstliche Intelligenz*, pages 227–239, 1996.
27. M. Lenz, H.-D.Burkhard, and S. Brückner. Applying case retrieval nets to diagnostic tasks in technical domains. In Ian F. C. Smith and Boi Faltings, editors, *EWCBR*, volume 1168 of *Lecture Notes in Computer Science*, pages 219–233. Springer, New York, 1996.
28. M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.
29. E. McKenna and B. Smyth. Competence-guided editing methods for lazy learning. In W. Horn, editor, *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence*, pages 60–64. IOS Press, The Netherlands 2000.
30. S.I. Nitzan and J. Paroush. *Collective Decision Making*. Cambridge University Press, Cambridge, 1985.
31. G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.
32. Y. Rubner, L. J. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pages 661–668, 1997.
33. Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
34. J.W. Schaaf. Fish and Shrink. A next step towards efficient case retrieval in large-scale case bases. In I. Smith and B. Faltings, editors, *European Conference on Case-Based Reasoning (EWCBR'96*, pages 362–376. Springer, New York, 1996.
35. R. E. Schapire. A brief introduction to boosting. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31–August 6, 1999. 2 Volumes, 1450 pages*, pages 1401–1406. Morgan Kaufmann, San Francisco, CA, USA, 1999.
36. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
37. J. Shawe-Taylor and N. Cristianini. *Kernel methods for Pattern Analysis*. Cambridge University Press, Cambridge ISBN 0-521-81397-2, 2004.
38. R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237:1317–1228, 1987.

39. B. Smyth and M. Keane. Remembering to forget: A competence preserving case deletion policy for cbr system. In C. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI (1995)*, pages 337–382. Morgan Kaufmann, San Francisco, CA, USA, 1995.
40. B. Smyth and E. McKenna. Footprint-based retrieval. In Klaus-Dieter Althoff, Ralph Bergmann, and Karl Branting, editors, *ICCBR*, volume 1650 of *Lecture Notes in Computer Science*, pages 343–357. Springer, New York, 1999.
41. I. Tomek. An experiment with the nearest neighbor rule. *IEEE Transactions on Information Theory*, 6(6):448–452, 1976.
42. S. Tong. *Active Learning: Theory and Applications*. PhD thesis, Stanford University, 2001.
43. A. Tsymbal, M. Pechenizkiy, and P. Cunningham. Diversity in random subspacing ensembles. In Yahiko Kambayashi, Mukesh K. Mohania, and Wolfram Wöß, editors, *DaWaK*, volume 3181 of *Lecture Notes in Computer Science*, pages 309–319. Springer, New York, 2004.
44. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.
45. V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.
46. V. N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.
47. K. Veropoulos. Controlling the sensivity of support vector machines. In *International Joint Conference on Artificial Intelligence (IJCAI99)*, Stockholm, Sweden, 1999.
48. L. Wang. Image retrieval with svm active learning embedding euclidean search. In *IEEE International Conference on Image Processing*, Barcelona, September 2003.
49. D. Wilson and T. Martinez. Instance pruning techniques. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 403–411. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
50. D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421, 1972.
51. D. H. Wolpert The lack of a priori distinctions between learning algorithms. In *Neural Computation*, 7, pages 1341–1390, 1996.
52. J. Zhang. Selecting typical instances in instance-based learning. In *Proceedings of the 9th International Conference on Machine Learning (ICML 92)*, pages 470–479. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.
53. J. Zu and Q. Yang. Remembering to add: competence preserving case-addition policies for case-base maintenance. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 234–239. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.