A New Framework for Machine Learning

Christopher M. Bishop

Microsoft Research, Cambridge, U.K
Christopher.Bishop@microsoft.com
http://research.microsoft.com/~cmbishop

Abstract. The last five years have seen the emergence of a powerful new framework for building sophisticated real-world applications based on machine learning. The cornerstones of this approach are (i) the adoption of a Bayesian viewpoint, (ii) the use of graphical models to represent complex probability distributions, and (iii) the development of fast, deterministic inference algorithms, such as variational Bayes and expectation propagation, which provide efficient solutions to inference and learning problems in terms of local message passing algorithms. This paper reviews the key ideas behind this new framework, and highlights some of its major benefits. The framework is illustrated using an example large-scale application.

1 Introduction

In recent years the field of machine learning has witnessed an important convergence of ideas, leading to a powerful new framework for building real-world applications. The goal of this paper is to highlight the emergence of this new viewpoint, and to emphasize its practical advantages over previous approaches. This paper is not, however, intended to be comprehensive, and no attempt is made to give accurate historical attribution of all the many important contributions. A much more detailed and comprehensive treatment of the topics discussed here, including additional references, can be found in [5].

The new framework for machine learning is built upon three key ideas: (i) the adoption of a Bayesian viewpoint, (ii) the use of probabilistic graphical models, and (iii) the application of fast, deterministic inference algorithms. In Section 2 we give a brief overview of the key concepts of Bayesian statistics, illustrated using a simple curve-fitting problem. We then discuss the use of probabilistic graphical models in Section 3. Inference and learning problems in graphical models can be solved efficiently using local message-passing algorithms, as described in Section 4. The new framework for machine learning is then illustrated using a large-scale application in Section 5, and finally in Section 6 we give some brief conclusions.

2 Bayesian Methods

The Bayesian interpretation of probabilities provides a consistent, indeed optimal, framework for the quantification of uncertainty [2,3,13]. In pattern recognition

J.M. Zurada et al. (Eds.): WCCI 2008 Plenary/Invited Lectures, LNCS 5050, pp. 1–24, 2008.

and machine learning applications, uncertainty arises both through noise processes on the observed variables, as well as through the unknown values of latent variables and model parameters. The adoption of a Bayesian viewpoint therefore provides a principled formalism through which all sources of uncertainty can be addressed consistently. In principle, it involves no more than the systematic application of the sum rule and the product rule of probability.

Machine learning models can be divided into *parametric* and *non-parametric*, according to whether or not they are based on models having a prescribed number of adjustable parameters. Most applications to date have been built using parametric models, and indeed this will be the focus of this paper. However, many of the same points emphasized here apply equally to non-parametric techniques.

Consider a model governed by a set of parameters which we group into a vector $\mathbf{w}$. If we denote the training data set by $\mathcal{D}$, then a central quantity is the conditional probability distribution $p(\mathcal{D}|\mathbf{w})$. When viewed as a function of $\mathbf{w}$ this is known as the *likelihood function*, and it plays a central role both in conventional (frequentist) and Bayesian approaches to machine learning. In a frequentist setting the goal is to find an estimator $\mathbf{w}^\star$ for the parameter vector by optimizing some criterion, for example by maximizing the likelihood. A significant problem with such approaches is *over-fitting* whereby the parameters are tuned to the noise on the data, thereby degrading the generalization performance.

In a Bayesian setting we express the uncertainty in the value of $\mathbf{w}$ through a probability distribution $p(\mathbf{w})$. This captures everything that is known about the value of $\mathbf{w}$, aside from the information provided by the training data, and is usually known as the prior distribution. The contribution from the training data is expressed through the likelihood function, and this can be combined with the prior using Bayes' theorem to give the posterior distribution

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}. \tag{1}$$

Here the denominator is given by

$$p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})\,\mathrm{d}\mathbf{w} \tag{2}$$

and can be viewed as the normalization factor which ensures that the posterior distribution $p(\mathbf{w}|\mathcal{D})$ in (1) integrates to one. It also plays a central role in model selection, as we shall discuss shortly.

In order to illustrate the use of Bayesian methods in machine learning, we consider the problem of fitting a set of noisy data points using a polynomial function. Although this example involves only a single input variable, a single output variable, and a simple parametric model, it captures most of the important concepts underpinning real-world applications of more sophisticated multivariate non-linear models.

The polynomial function itself can be written in the form

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{3}$$

where M is the order of the polynomial.

First we consider briefly a conventional, non-Bayesian, approach to this problem. Figure 1 shows the training data and the function from which the data is generated, along with the result of fitting several polynomials of different order by minimizing the sum-of-squares error between the polynomial predictions and the data point, defined by

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 \tag{4}$$

where t_n denotes the training set target value corresponding to an input value of x_n

It can be seen that if the order of the polynomial is too low ($M = 0$, 1) then the result is a poor representation of the underlying sinusoidal curve. Equally if the order of the polynomial is too high ($M = 9$) then the result is again poor due to over-fitting. The best approximation arises from a model of intermediate complexity ($M = 3$). This is confirmed by looking at the root-mean-square error, defined by

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N} \tag{5}$$

on both the training set and an independent test set, as shown in Figure 2. The best generalization performance (i.e. the smallest test set error) occurs for models of intermediate complexity.

Now consider a Bayesian approach to this problem. If we assume that the data has Gaussian noise, then the likelihood function takes the form

$$p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}\left(t_n | y(x_n, \mathbf{w}), \beta^{-1}\right) \tag{6}$$

where β is the precision (inverse variance) of the noise process. Here $\mathcal{N}\left(t|\mu, \sigma^2\right)$ denotes a Gaussian distribution over the variable t, with mean μ and variance σ^2.

For simplicity we consider a Gaussian prior distribution of the form

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\} \tag{7}$$

where α is the precision of the distribution. Using Bayes' theorem (1) it is then straightforward to evaluate the posterior distribution over $\mathbf{w}$, which also takes the form of a Gaussian.

The posterior distribution is not itself of interest, but it plays a crucial role in making predictions for new input values. These predictions are governed by

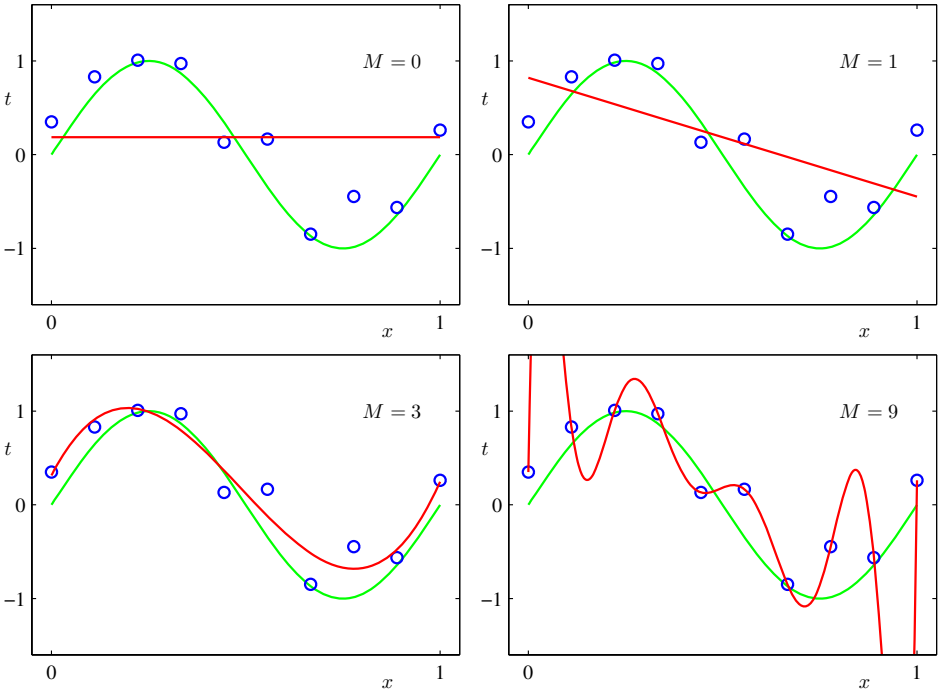

Fig. 1. Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t. The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x, without knowledge of the green curve. The red curves show the result of fitting polynomials of various orders M using least squares.

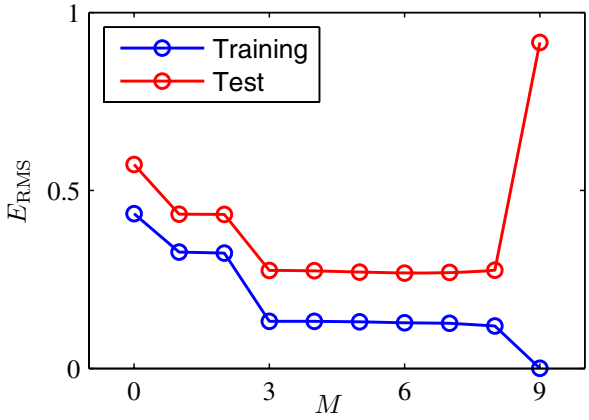

Fig. 2. Graphs of the root-mean-square error, defined by (5), evaluated on the training set and on an independent test set for various values of M

the predictive distribution, which is obtained from the sum and product rules of probability in the form

$$p(t|x,\mathcal{D}) = \int p(t|x,\mathbf{w})p(\mathbf{w}|\mathcal{D})\,\mathrm{d}\mathbf{w} = \mathcal{N}\left(t|m(x),s^2(x)\right) \tag{8}$$

where the mean and variance are given by

$$m(x) = \beta\boldsymbol{\phi}(x)^{\mathrm{T}}\mathbf{S}\sum_{n=1}^{N}\boldsymbol{\phi}(x_n)t_n \tag{9}$$

$$s^2(x) = \beta^{-1} + \boldsymbol{\phi}(x)^{\mathrm{T}}\mathbf{S}\boldsymbol{\phi}(x). \tag{10}$$

Here the matrix $\mathbf{S}$ is given by

$$\mathbf{S}^{-1} = \alpha\mathbf{I} + \beta\sum_{n=1}^{N}\boldsymbol{\phi}(x_n)\boldsymbol{\phi}(x_n)^{\mathrm{T}} \tag{11}$$

where $\mathbf{I}$ is the unit matrix, and we have defined the vector $\boldsymbol{\phi}(x)$ with elements $\phi_i(x) = x^i$ for $i = 0,\ldots,M$.

Figure 3 shows a plot of the predictive distribution when the training set comprises $N = 4$ data points. Note that the variance of the predictive distribution is itself a function of the input variable x. In particular, the uncertainty in the predictions is smallest in the neighbourhood of the training data points. This intuitively pleasing result follows directly from the adoption of a Bayesian treatment.

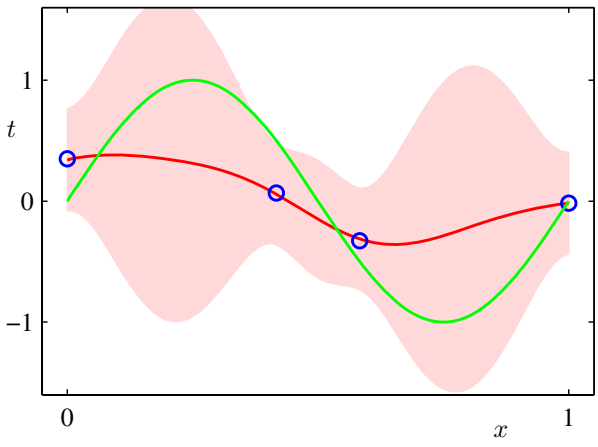

Fig. 3. Plot of the predictive distribution (8) with $N = 4$ training data points. The red curve shows the mean of the predictive distribution, while the red shaded region spans one standard deviation either side of the mean.

As can be seen from (10), the variance of the predictive distribution comprises two terms. The first corresponds to the noise on the training data and represents an irreducible level of uncertainty in predicting the value of t for a new input value x. The second term represents the uncertainty in predictions arising from the uncertainty in the model parameters $\mathbf{w}$. If we observe more data points, this latter uncertainty will decrease, as can be seen in Figure 4.

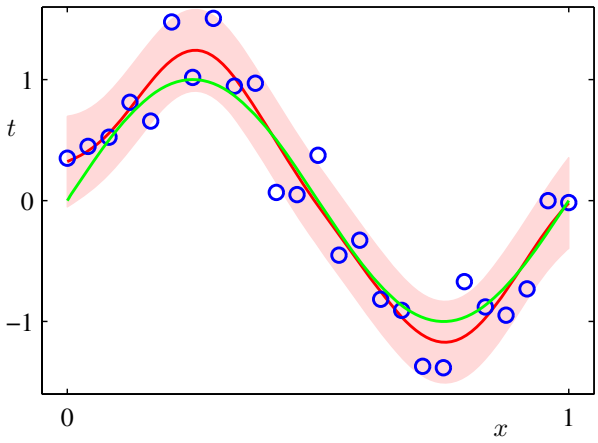

Fig. 4. As in Figure 3 but with $N = 25$ data points. The residual uncertainty in the predictive distribution is due mainly to the noise on the training data.

As an aside, suppose we make a frequentist point estimate of the model parameters by maximizing the posterior distribution. Equivalently we can maximize the logarithm of the posterior distribution, which takes the form

$$\ln p(\mathbf{w}|\mathcal{D}) = -\frac{\beta}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 - \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \text{const.} \tag{12}$$

which we recognise as the negative of the standard sum-of-squares error function with a quadratic weight penalty (regularization term). Thus we see how a conventional frequentist approach arises as a particular approximation to a Bayesian treatment.

We have seen that a Bayesian approach naturally makes predictions in the form of probability distributions over possible values, conditioned on the observed input variables. This is substantially more powerful than simply making point predictions as in conventional (non-Bayesian) machine learning approaches [5, pages 44–46].

Another major advantage of the Bayesian approach is that it automatically addresses the question of model complexity and model comparison. In conventional approaches based on a point estimate of the model parameters, it is common to optimize the model complexity to achieve a balance between too simple

a model (which performs poorly on both training data and test data) and one which is too complex (which over-fits the training data and makes poor predictions on test data). This is usually addressed using data hold-out techniques such as cross-validation, in which part of the training data is kept aside in order to compare models of different complexity and to select the one which has the best generalization performance. Such cross-validations methods are wasteful of valuable training data, and are often computationally expensive due to the need for multiple training runs.

The Bayesian view of model comparison involves the use of probabilities to represent uncertainty in the choice of model, along with a consistent application of the sum and product rules of probability. Suppose we wish to compare a set of L models $\{\mathcal{M}_i\}$ where $i = 1, \ldots, L$. Here a model refers to a parametric representation for the probability distribution over the observed data $\mathcal{D}$, along with a prior distribution for the parameters. We shall suppose that the data is generated from one of these models but we are uncertain which one. Our uncertainty in the choice of model is expressed through a prior probability distribution $p(\mathcal{M}_i)$. Given a training set $\mathcal{D}$, we then wish to evaluate the posterior distribution

$$p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{M}_i)p(\mathcal{D}|\mathcal{M}_i). \tag{13}$$

The prior allows us to express a preference for different models. Let us simply assume that all models are given equal prior probability. The interesting term is the *model evidence* $p(\mathcal{D}|\mathcal{M}_i)$ which expresses the preference shown by the data for different models. It is sometimes also called the *marginal likelihood* because it can be viewed as a likelihood function over the space of models, in which the parameters have been marginalized out. For a model governed by a set of parameters $\mathbf{w}$, the model evidence is given, from the sum and product rules of probability, by

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i)p(\mathbf{w}|\mathcal{M}_i)\,\mathrm{d}\mathbf{w}. \tag{14}$$

Recall that this term arises as the normalization factor in Bayes' theorem (1) for the parameters.

We can gain insight into Bayesian model comparison, and understand how the marginal likelihood can favour models of intermediate complexity, by considering Figure 5. Here the horizontal axis is a one-dimensional representation of the space of possible data sets, so that each point on this axis corresponds to a specific data set. We now consider three models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ of successively increasing complexity. Imagine running these models generatively to produce example data sets, and then looking at the distribution of data sets that result. Any given model can generate a variety of different data sets since the parameters are governed by a prior probability distribution, and for any choice of the parameters there may be random noise on the target variables. To generate a particular data set from a specific model, we first choose the values of the parameters from their prior distribution $p(\mathbf{w})$, and then for these parameter values we sample the data from $p(\mathcal{D}|\mathbf{w})$. A simple model (for example, based on a first order polynomial) has little variability and so will generate data sets that are fairly

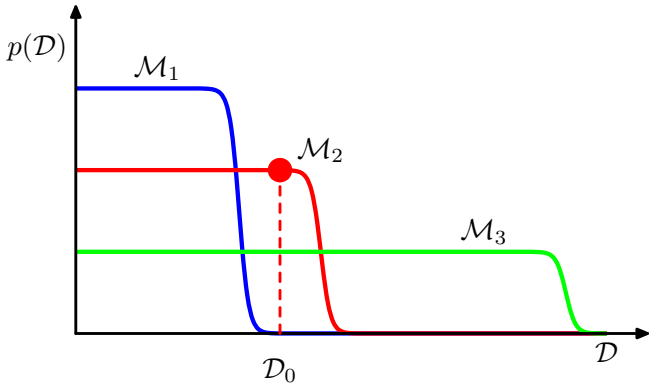

Fig. 5. Schematic illustration of the distribution of data sets for three models of different complexity, in which $\mathcal{M}_1$ is the simplest and $\mathcal{M}_3$ is the most complex. Note that the distributions are normalized. In this example, for the particular observed data set $\mathcal{D}_0$, the model $\mathcal{M}_2$ with intermediate complexity has the largest evidence.

similar to each other. Its distribution $p(\mathcal{D})$ is therefore confined to a relatively small region of the horizontal axis. By contrast, a complex model (such as a ninth order polynomial) can generate a great variety of different data sets, and so its distribution $p(\mathcal{D})$ is spread over a large region of the space of data sets. Because the distributions $p(\mathcal{D}|\mathcal{M}_i)$ are normalized, we see that the particular data set $\mathcal{D}_0$ can have the highest value of the evidence for the model of intermediate complexity. Essentially, the simpler model cannot fit the data well, whereas the more complex model spreads its predictive probability over too broad a range of data sets and so assigns relatively small probability to any one of them.

Returning to the polynomial regression problem, we can plot the model evidence against the order of the polynomial, as shown in Figure 6. Here we have assumed a prior of the form (7) with the parameter α fixed at $\alpha = 5 \times 10^{-3}$. The form of this plot is very instructive. Referring back to Figure 1, we see that the $M = 0$ polynomial has very poor fit to the data and consequently gives a relatively low value for the evidence. Going to the $M = 1$ polynomial greatly improves the data fit, and hence the evidence in Figure 6 is significantly higher. However, in going to $M = 2$, the data fit is improved only very marginally, due to the fact that the underlying sinusoidal function from which the data is generated is an odd function and so has no even terms in a polynomial expansion. Indeed, Figure 2 shows that the residual data error is reduced only slightly in going from $M = 1$ to $M = 2$. Because this richer model suffers a greater complexity penalty, the evidence actually falls in going from $M = 1$ to $M = 2$. When we go to $M = 3$ we obtain a significant further improvement in data fit, as seen in Figure 1, and so the evidence is increased again, giving the highest overall evidence for any of the polynomials. Further increases in the value of M produce only small improvements in the fit to the data but suffer increasing complexity penalty, leading overall to a decrease in the evidence values. Looking again at

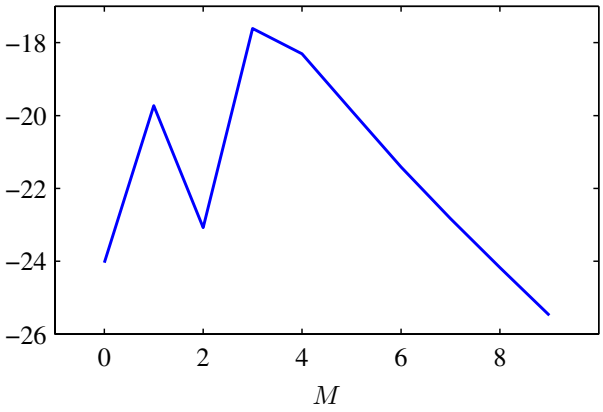

Fig. 6. Plot of the model evidence versus the order of the polynomial, for the simple curve fitting problem

Figure 2, we see that the generalization error is roughly constant between $M = 3$ and $M = 8$, and it would be difficult to choose between these models on the basis of this plot alone. The evidence values, however, show a clear preference for $M = 3$, since this is the simplest model which gives a good explanation for the observed data.

3 Graphical Models

The use of Bayesian methods in machine learning amounts to a consistent application of the sum and product rules of probability. We could therefore proceed to formulate and solve complicated probabilistic models purely by algebraic manipulation. However, it is highly advantageous to augment the analysis using diagrammatic representations of probability distributions, called *probabilistic graphical models*. These offer several useful properties:

1. They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
2. Insights into the properties of the model can be obtained by inspection of the graph.
3. Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

A graph comprises *nodes* (also called *vertices*) connected by *links* (also known as *edges* or *arcs*). In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relationships between these variables.

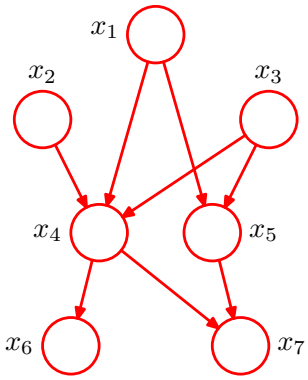

Fig. 7. Example of a directed graph describing the joint distribution over variables $x_1, \ldots, x_7$. The corresponding decomposition of the joint distribution is given by (16).

There are two main types of graphical model in widespread use, corresponding to directed graphs (in which the links have a directionality indicated by arrows) and undirected graphs (in which the links are symmetrical). In both cases the graph expresses the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables, but the relationship between the graph and the factorization is different for the two types of graph.

Consider first the case of directed graphs, also known as *Bayesian networks* or *belief networks*. An example is shown in Figure 7. If there is a link going from a node a to a node b, then we say that node a is the *parent* of node b. The graph specifies that the joint distribution factorizes into a product over all nodes of a conditional distribution for the variables at that node conditioned on the states of its parents

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k) \tag{15}$$

where pa_k denotes the set of parents of x_k, and $\mathbf{x} = \{x_1, \ldots, x_K\}$. For the specific case of the graph shown in Figure 7, the factorization takes the form

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5). \tag{16}$$

A specific, and very familiar, example of a directed graph is the hidden Markov model, which is widely used in speech recognition, handwriting recognition, DNA analysis, and other sequential data applications, and is shown in Figure 8. The joint distribution for this model is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n). \tag{17}$$

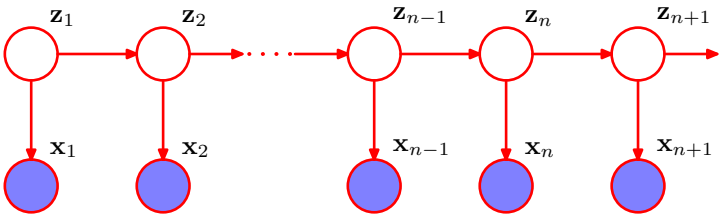

Fig. 8. The directed graph corresponding to a hidden Markov model. It represents the joint distribution over a set of observed variables $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in terms of a Markov chain of hidden variables $\mathbf{z}_1, \ldots, \mathbf{z}_N$. Exactly the same graph also describes the Kalman filter.

Here $\mathbf{x}_1, \ldots, \mathbf{x}_N$ represent the observed variables (i.e. the data). In a graphical model the observed variables are denoted by shading the corresponding notes. The variables $\mathbf{z}_1, \ldots, \mathbf{z}_N$ represent latent (or hidden) variables which are not directly observed but which play a key role in the formulation of the model. In the case of the hidden Markov model the latent variables are discrete, while the observed variables may be discrete or continuous according to the particular application.

We can also consider the graph in Figure 8 for the case in which both the hidden and observed variables are Gaussian, in which case it describes the *Kalman filter*, a model which is widely used for tracking applications [21]. This highlights an important property of graphical models, namely that a particular graph describes a whole family of probability distributions which share the same factorization properties.

One of the powerful aspects of graphical models is the ease with which new models can be constructed, incorporating appropriate domain knowledge in the process. For example, Figure 9 shows an extension of the hidden Markov model which expresses the notion that there are two independent latent processes, and that at each time step the observed variables have distributions which are

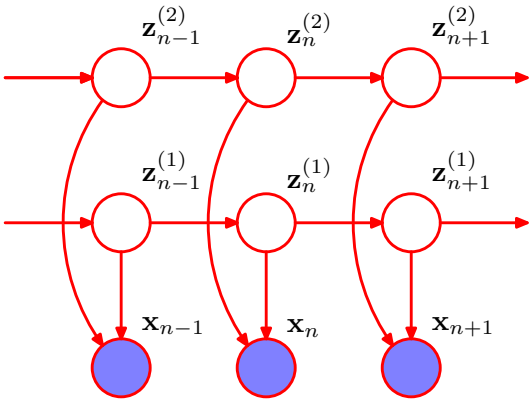

Fig. 9. A factorial hidden Markov model

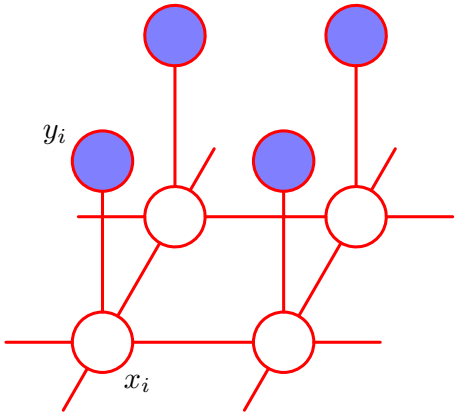

Fig. 10. An undirected graphical model representing a Markov random field for image de-noising, in which x_i is a binary latent (hidden) variable denoting the state of pixel i in the unknown noise-free image, and y_i denotes the corresponding value of pixel i in the observed noisy image

conditioned on the states of both of the corresponding latent variables. This is known as a *factorial hidden Markov model*.

Similarly, the Kalman filter can be extended to give a *switching state space model*. This has multiple Markov chains of continuous linear-Gaussian latent variables, each of which is analogous to the latent chain of the standard Kalman filter, together with a Markov chain of discrete variables of the form used in a hidden Markov model. The output at each time step is determined by stochastically choosing one of the continuous latent chains, using the state of the discrete latent variable as a switch, with the distribution of the observation at each step conditioned on the state of the corresponding continuous hidden variable.

Many other models can easily be constructed in this way. The key point is that new models can be formulated simply by drawing the corresponding graphical model, and prior knowledge from the application domain can be expressed through the structure of the graph. In particular, missing links in the graph determine the *conditional independence properties* of the joint distribution [5, Section 8.2].

The second major class of graphical model is based on *undirected graphs*. A well-known example is the *Markov random field*, illustrated in Figure 10. This graphical structure can be used to solve image processing problems such as de-noising and segmentation.

As with directed graphs, an undirected graph specifies the way in which the joint distribution of all variables in the model factorizes into a product of factors each involving only a subset of the variables. To understand this factorization we need to introduce the concept of a *clique*, which is defined as a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset. In other words, the set of nodes in a clique is fully connected.

If we denote a clique by C and the set of variables in that clique by $\mathbf{x}_C$, then the joint distribution is written as a product of *potential functions* $\psi_C(\mathbf{x}_C)$ over the cliques of the graph

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C). \tag{18}$$

Here the quantity Z, sometimes called the *partition function*, is a normalization constant and is given by

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C) \tag{19}$$

which ensures that the distribution $p(\mathbf{x})$ given by (18) is correctly normalized. By considering only potential functions which satisfy $\psi_C(\mathbf{x}_C) \geqslant 0$ we ensure that $p(\mathbf{x}) \geqslant 0$. In (19) we have assumed that $\mathbf{x}$ comprises discrete variables, but the framework is equally applicable to continuous variables, or a combination of the two, in which the summation is replaced by the appropriate combination of summation and integration.

Because we are restricted to potential functions which are strictly positive it is convenient to express them as exponentials, so that

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\} \tag{20}$$

where $E(\mathbf{x}_C)$ is called an *energy function*, and the exponential representation is called the *Boltzmann distribution*. The joint distribution is defined as the product of potentials, and so the total energy is obtained by adding the energies of each of the cliques.

In the case of Figure 10 there are three kinds of cliques, those that involve a single hidden variable, those that involve two adjacent hidden variables connected by a link, and those that involve one hidden and one observed variable, again connected by a link. An example of an energy function for such a model takes the form

$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i \tag{21}$$

which defines a joint distribution over $\mathbf{x}$ and $\mathbf{y}$ given by

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}. \tag{22}$$

Directed and undirected graphs together allow most models of practical interest to be constructed. Which type of graph is more appropriate will depend on the application. Generally speaking, directed graphs are good at expressing causal relationships between variables. For example, if we have set of diseases and a set of symptoms then we can use a directed graph to capture the notion that the symptoms are caused by the diseases, and so there will be arrows (directed edges) going from disease variable nodes to symptom variable nodes. Undirected graphs, however, are better at expressing correlations between variables.

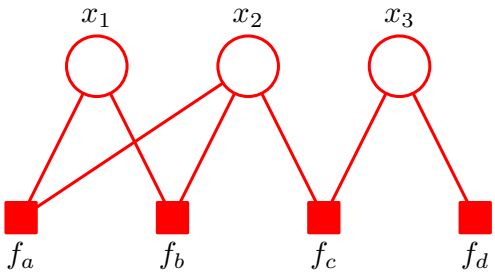

Fig. 11. Example of a factor graph, which corresponds to the factorization (23)

For example, in a model for segmenting an image into foreground and background we know that neighbouring pixels are very likely to share the same label (they are usually either both foreground or both background) and this can be captured using an undirected graph of the form shown in Figure 10.

It is often convenient to work with a third form of graphical representation known as a *factor graph*. We have seen that both directed and undirected graphs allow a global function of several variables to be expressed as a product of factors over subsets of those variables. Factor graphs make this decomposition explicit by introducing additional nodes for the factors themselves in addition to the nodes representing the variables.

Consider, for example, a distribution that is expressed in terms of the factorization

$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3). \tag{23}$$

This can be expressed by the factor graph shown in Figure 11. Factor graphs provide a useful representation for inference algorithms, discussed in the next section, as they allow both directed and undirected graphs to be treated in a unified way.

4 Approximate Inference

Having formulated a model in terms of a probabilistic graph we now need to learn the parameters of the model, and to use the trained model to make predictions. Some of the nodes in the graph correspond to observed variables representing the training data, and we are interested in finding the posterior distribution of other nodes, representing variables whose value we wish to predict, conditioned on the training data. We refer to this as an inference problem. The remaining nodes in the graph represent other latent, or hidden, variables whose values we are not directly interested in.

In a Bayesian setting, the model parameters are also random variables and are therefore represented as nodes in the graph. Computing the posterior distribution of those parameters is therefore just another inference problem!

Fig. 12. A simple undirected graph comprising a chain of nodes, used to illustrate the solution of inference problems

Consider first the problem of performing inference on a simple chain of nodes shown in Figure 12. The cliques of this graph comprise pairs of adjacent nodes connected by links. Thus the joint distribution of all of the variables is given by

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N). \tag{24}$$

Let us consider the inference problem of finding the marginal distribution $p(x_n)$ for a specific node x_n that is part way along the chain. Note that, for the moment, there are no observed nodes. By definition, the required marginal is obtained by summing the joint distribution over all variables except x_n, so that

$$p(x_n) = \sum_{x_1}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\cdots\sum_{x_N} p(\mathbf{x}). \tag{25}$$

In a naive implementation, we would first evaluate the joint distribution and then perform the summations explicitly. The joint distribution can be represented as a set of numbers, one for each possible value for $\mathbf{x}$. Because there are N variables each with K states, there are K^N values for $\mathbf{x}$ and so evaluation and storage of the joint distribution, as well as marginalization to obtain $p(x_n)$, all involve storage and computation that scale exponentially with the length N of the chain.

We can, however, obtain a much more efficient algorithm by exploiting the conditional independence properties of the graphical model. If we substitute the factorized expression (24) for the joint distribution into (25), then we can rearrange the order of the summations and the multiplications to allow the required marginal to be evaluated much more efficiently. Consider for instance the summation over x_N. The potential $\psi_{N-1,N}(x_{N-1}, x_N)$ is the only one that depends on x_N, and so we can perform the summation

$$\sum_{x_N}\psi_{N-1,N}(x_{N-1}, x_N) \tag{26}$$

first to give a function of x_{N-1}. We can then use this to perform the summation over x_{N-1}, which will involve only this new function together with the potential $\psi_{N-2,N-1}(x_{N-2}, x_{N-1})$, because this is the only other place that x_{N-1} appears. Similarly, the summation over x_1 involves only the potential $\psi_{1,2}(x_1, x_2)$ and so can be performed separately to give a function of x_2, and so on.

If we group the potentials and summations together in this way, we can express the desired marginal in the form

$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots\right]}_{\mu_\alpha(x_n)} \underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots\right]}_{\mu_\beta(x_n)}. \tag{27}$$

The key concept that we are exploiting is that multiplication is distributive over addition, so that

$$ab + ac = a(b + c) \tag{28}$$

in which the left-hand side involves three arithmetic operations whereas the right-hand side reduces this to two operations.

Let us work out the computational cost of evaluating the required marginal using this re-ordered expression. We have to perform $N - 1$ summations each of which is over K states and each of which involves a function of two variables. For instance, the summation over x_1 involves only the function $\psi_{1,2}(x_1, x_2)$, which is a table of $K \times K$ numbers. We have to sum this table over x_1 for each value of x_2 and so this has $O(K^2)$ cost. The resulting vector of K numbers is multiplied by the matrix of numbers $\psi_{2,3}(x_2, x_3)$ and so is again $O(K^2)$. Because there are $N - 1$ summations and multiplications of this kind, the total cost of evaluating the marginal $p(x_n)$ is $O(NK^2)$. This is linear in the length of the chain, in contrast to the exponential cost of a naive approach.

We can now give a powerful interpretation of this calculation in terms of the passing of local *messages* around on the graph. From (27) we see that the expression for the marginal $p(x_n)$ decomposes into the product of two factors times the normalization constant

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n). \tag{29}$$

We shall interpret $\mu_\alpha(x_n)$ as a message passed forwards along the chain from node x_{n-1} to node x_n. Similarly, $\mu_\beta(x_n)$ can be viewed as a message passed backwards along the chain to node x_n from node x_{n+1}. Note that each of the messages comprises a set of K values, one for each choice of x_n, and so the product of two messages should be interpreted as the point-wise multiplication of the elements of the two messages to give another set of K values.

The message $\mu_\alpha(x_n)$ can be evaluated recursively because

$$\begin{aligned}\mu_\alpha(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right] \\ &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}) \end{aligned} \tag{30}$$

with an analogous result for the backward messages.

This example shows how the local factorization implied by the graphical structure allows the cost of exact inference to be reduced from being exponential in the length of the chain to being linear. A similar result holds for more complex graphs provided they have a tree structure (i.e. they do not have any loops). This exact inference technique is known as *belief propagation*.

A well-known special case of this message-passing algorithm is the forward-backward algorithm for inferring the posterior distribution of the hidden variables in a hidden Markov model [1,19]. Similarly, the forward recursions of the Kalman filter [14] and the backward recursions of the Kalman smoother [20] are also special cases of this result.

For most practical applications, however, this efficient exact solution of inference problems is no longer tractable. This lack of tractability arises either because the graph is no longer a tree, or because the individual local marginalizations no longer have an exact closed-form solution.

We therefore need to find approximate inference algorithms which can yield good results with reasonable computational cost. For a long time the only generally applicable method was to use Markov chain Monte Carlo sampling techniques. Unfortunately, this approach tends to be computationally costly and does not scale well to real-world applications involving large data sets. One of the most important advances in machine learning in recent years has therefore been the development of fast, approximate inference algorithms. Like the exact inference algorithms for trees discussed above, these can all be expressed in terms of local message passing on the corresponding graphical model, and this leads naturally to efficient software implementations. They are often called 'deterministic' algorithms because they provide analytical expressions for the posterior distribution, in contrast to Monte Carlo methods which yield their results in the form of a set of samples drawn from the posterior distribution. Here we introduce briefly some of the most prominent deterministic inference techniques. It should be emphasized, however, that there are many other such algorithms and new ones continue to be developed.

One of the simplest such algorithms is called *loopy belief propagation* [9] and simply involves applying the standard belief propagation equations, derived for tree-structured graphs, to more general graphical models. Although this is an ad-hoc procedure, and has no guarantee of convergence, it is often found to yield good results, and indeed gives state-of-the-art results for decoding certain kinds of error-correcting codes [4,8,10,15,16].

A more principled approach to approximate inference is to define a family of approximating distributions (whose members are simpler in some sense than the true posterior distribution) and then to seek the optimal member of that family by minimizing a suitable criterion which measures the dissimilarity between the approximate distribution and the exact posterior distribution. Different algorithms arise according to the simplifying assumptions in the approximate posterior, and according to the choice of criterion.

The *variational Bayes* method defines the dissimilarity between the true posterior distribution $p(\mathbf{Z}|\mathbf{X})$ and the approximating distribution $q(\mathbf{Z})$ to be the Kullback-Leibler divergence given by

$$\mathrm{KL}(q\|p) = -\int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z} \tag{31}$$

where $\mathbf{Z}$ represents the set of all non-observed variables in the problem and $\mathbf{X}$ represents the observed variables.

The approximating distribution can be chosen to have a simple analytical form. For example, it might be a Gaussian, whose mean and covariance are then optimized so as to minimize the KL divergence with respect to the (non-Gaussian) true posterior distribution. A more flexible framework arises, however, if we assume a specific factorization for the approximating distribution $q(\mathbf{Z})$, without any restriction on the functional form of the factors. Suppose we partition the elements of $\mathbf{Z}$ into disjoint groups that we denote by $\mathbf{Z}_i$ where $i = 1, \ldots, M$. We then assume that the q distribution factorizes with respect to these groups, so that

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i). \tag{32}$$

If we substitute (32) into (31) we can then minimize the KL divergence with respect to one of the factors $q_j(\mathbf{Z}_j)$, keeping the remaining factors fixed. This involves a free-form functional optimization performed using the calculus of variations, and gives the result

$$\ln q_j^{\star}(\mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const} \tag{33}$$

where $p(\mathbf{X}, \mathbf{Z})$ is the joint distribution of hidden and observed variables, and the expectation is taken over all groups of variables $\mathbf{Z}_i$ for $i \neq j$. The additive constant corresponds to the normalization coefficient for the distribution. In order to apply this approach in practice, the factors $q_i(\mathbf{Z}_i)$ are first suitably initialized, and then they are updated in turn using (33) until a suitable convergence criterion is satisfied.

As an illustration of the variational Bayes method, consider the toy problem shown in Figure 13. Here the green contours show the true posterior distribution $p(\mu, \tau)$ over the mean μ and precision (inverse variance) τ for a simple inference problem involving a Gaussian distribution [5, Section 10.1.3]. For the sake of illustration, suppose that this distribution is intractable to compute and so we wish to find an approximation using variational inference. The approximating

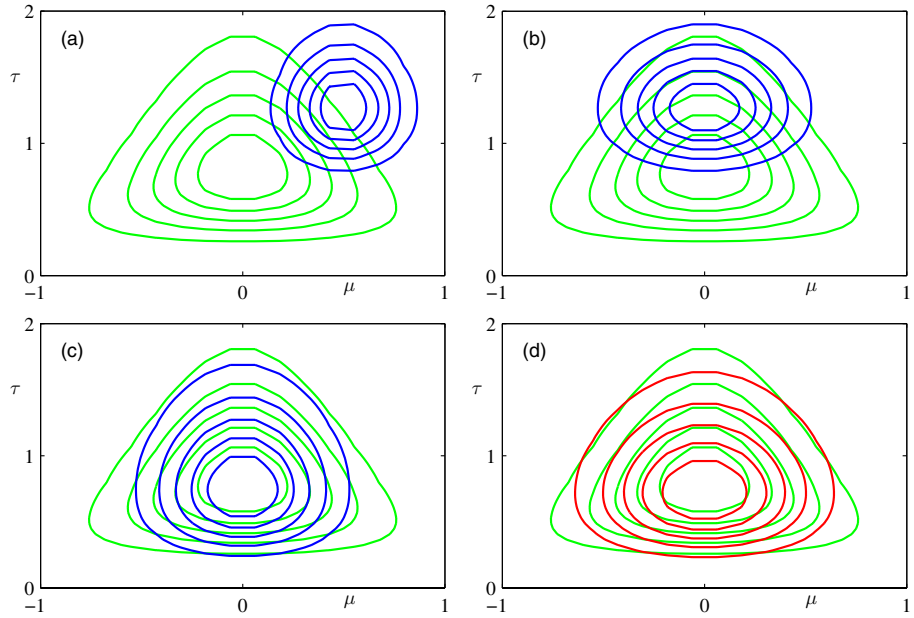

Fig. 13. An illustration of the variational Bayes inference algorithm. See the text for details.

distribution is assumed to factorize so that $q(\mu, \tau) = q_\mu(\mu)q_\tau(\tau)$. The blue contours in Figure 13(a) show the initialization of this factorized distribution. In Figure 13(b) the distribution $q_\mu(\mu)$ has been updated using (33) keeping $q_\tau(\tau)$ fixed. Similarly, in Figure 13(c) the distribution $q_\tau(\tau)$ has been updated keeping $q_\mu(\mu)$ fixed. Finally, the optimal factorized solution, to which the iterative scheme converges, is shown by the red contours in Figure 13(d).

If we consider distributions which are expressed in terms of probabilistic graphical models, then the variational update equations (33) can be cast in the form of a local message-passing algorithm. This makes possible the construction of general purpose software for variational inference in which the form of the model does not need to be specified in advance [6].

Another widely used approximate inference algorithm is called *expectation propagation* or *EP* [17,18]. As with the variational Bayes method discussed so far, this too is based on the minimization of a Kullback-Leibler divergence but now of the reverse form $\mathrm{KL}(p\|q)$, which gives the approximation rather different properties. EP again makes use of the factorization implied by a graphical model, and again the update equations for determining the approximate posterior distribution can be cast as a local message-passing algorithm. Although each step is minimizing a specific KL divergence, the overall algorithm does not optimize a unique quantity globally. However, for approximations which lie within the *exponential family* of distributions, if the iterations do converge, the resulting solution will be a stationary point of a particular energy function [17], although

each iteration of EP does not necessarily decrease the value of this energy function. This is in contrast to variational Bayes, which iteratively maximizes a lower bound on the log marginal likelihood, in which each iteration is guaranteed not to decrease the bound. It is possible to optimize the EP cost function directly, in which case it is guaranteed to converge, although the resulting algorithms can be slower and more complex to implement.

While EP lacks the guaranteed convergence properties of variational Bayes, it can often give better results because the integration in the KL divergence is weighted by the true distribution, rather than by the approximation, which causes the algorithm to take a more global view of approximating the true distribution.

5 Example Application: Bayesian Ranking

We now describe a real-world application of the machine learning framework discusses in this paper. It is based on a Bayesian formulation, which is expressed as a probabilistic graphical model, and where predictions are obtained using expectation propagation formulated as local message-passing on the graph. The application is known as *TrueSkill*™ [12], and is a Bayesian system for rating player skill in competitive games. It can be viewed as a generalization of the well known Elo system, which is used for example in Chess, and which was adopted by the World Chess Federation in 1970 as an international standard. With the advent of online gaming, the importance of skill rating systems has increased significantly because the quality of the online experience of millions of players each day is at stake.

Elo assigns each player i a skill rating s_i, and the probability of the possible game outcomes is modelled as a function of the two players skills s_1 and s_2. In a particular game each player exhibits a performance

$$p_i \sim \mathcal{N}(p_i|s_i, \beta^2) \tag{34}$$

which is normally distributed around their skill value with fixed variance β^2. The probability that player 1 wins is given by the probability that their performance exceeds that of player 2, so that

$$P(p_1 > p_2|s_1, s_2) = \Phi\left(\frac{s_1 - s_2}{\sqrt{2}\beta}\right) \tag{35}$$

where Φ is the cumulative density of a zero-mean, unit-variance Gaussian. The Elo system then provides an update equation for the skill ratings which causes the observed game outcome to become more likely, while preserving the constraint $s_1 + s_2 = \text{const}$. There is a variant of the Elo system which replaces the cumulative Gaussian with a logistic sigmoid. In Elo, a player's rating is regarded as provisional as long as it is based on less than a fixed number of, say, 20 games. This problem was addressed previously by adopting a Bayesian approach, known as *Glicko*, which models the belief about a player's rating using a Gaussian with mean μ and variance σ^2 [11].

An important new application of skill rating systems are multiplayer online games, which present the following challenges:

1. Game outcomes often refer to teams of players, and yet a skill rating for individual players is needed for future matchmaking.
2. More than two players or teams compete, such that the game outcome is a permutation of teams or players rather than just a winner and a loser.

TrueSkill addresses both of these challenges in the context of a principled Bayesian approach.

Each player has a skill distribution which is Gaussian $s_i \sim \mathcal{N}(s_i|\mu_i, \sigma_i^2)$, and the performance of a player is again a noisy version of their skill given by (34). The performance t_j of team j is modelled as the sum of the performances of the players comprising that team, and the ordering of the team performances gives the ordering of the match results. This model can be expressed as the factor graph shown in Figure 14. Draws can be incorporated into the model by requiring a non-zero margin, governed by a parameter ϵ, between the performance of two teams in order to achieve a victory.

Skill estimates need to be reported after each game, and so an online learning scheme is used known as *Gaussian density filtering*, which can be viewed as a special case of expectation propagation. The posterior distribution is approximated by a Gaussian, and forms the prior distribution for the next game.

Extensive testing of *TrueSkill* demonstrates significant improvements over Elo [12]. In particular, the number of games which need to be played in order to determine accurate values for player skills can be substantially less (up to an order of magnitude) compared to Elo. This is illustrated in Figure 15 which shows the evolution of the skill ratings for two players over several hundred games, based on data collected during beta testing of the Xbox title 'Halo 2'. We see that *TrueSkill* exhibits good convergence within the first 10–20 games, whereas the Elo estimates are continuing to change significantly even after 100 games.

Intuitively, the reason for the faster convergence is that knowledge of the uncertainty in the skill estimates modulates the magnitude of the updates in an optimal way. For instance, informally, if a player with a skill rating of 120 ± 20 beats a player of rating 130 ± 2 then the system can make a substantial increase in the rating of the winning player. Elo by contrast does not have access to uncertainty estimates and so makes numerous small corrections over many games in order to increase the skill estimate for a particular player.

Xbox 360 Live is Microsoft's online console gaming service, allowing players to play together across the world on hundreds of different game titles. *TrueSkill* has been globally deployed as the skill rating system for *Xbox 360 Live*, analyzing millions of game outcomes resulting from billions of hours of online play. It processes hundreds of thousands of new game outcomes per day, making it one of the largest applications of Bayesian inference to date. *TrueSkill* provides ratings and league table information to the players, and is used to perform real-time matchmaking.

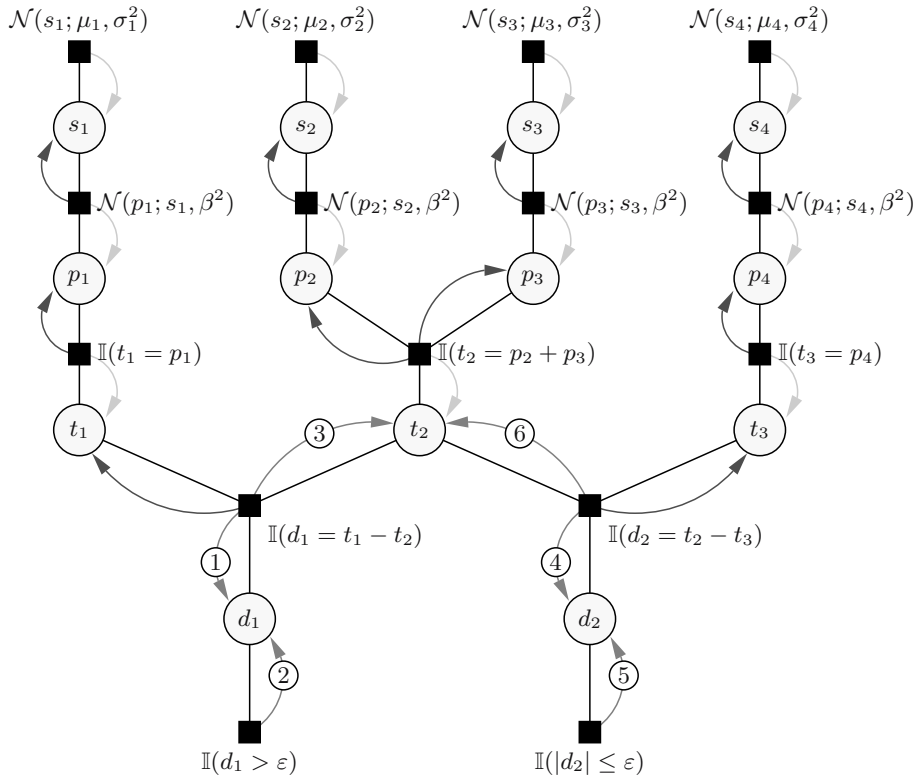

Fig. 14. An example TrueSkill factor graph, for two matches involving three teams and four players. The arrows indicate the optimal message passing schedule.

The graphical model formulation of *TrueSkill* makes it particularly straightforward to extend the model in interesting ways. For example, we can allow the skills to evolve in time by adding additional links to the graph corresponding to Gaussian temporal dynamics [12]. Similarly, the use of full expectation-propagation updates allows information to be propagated backwards in time (smoothing) as well as forwards in time (filtering). This permits a full analysis of historical data, for example a comparison of the strength of different chess players over a period of 150 years [7].

6 Discussion

In this paper we have highlighted the emergence of a new framework for the formulation and solution of problems in machine learning. The three main ingredients, namely a Bayesian approach, the use of graphical models, and use of approximate deterministic inference algorithms, fit very naturally together. In a fully Bayesian setting every unknown variable is given a probability distribution and hence corresponds to a node in a graphical model, and deterministic

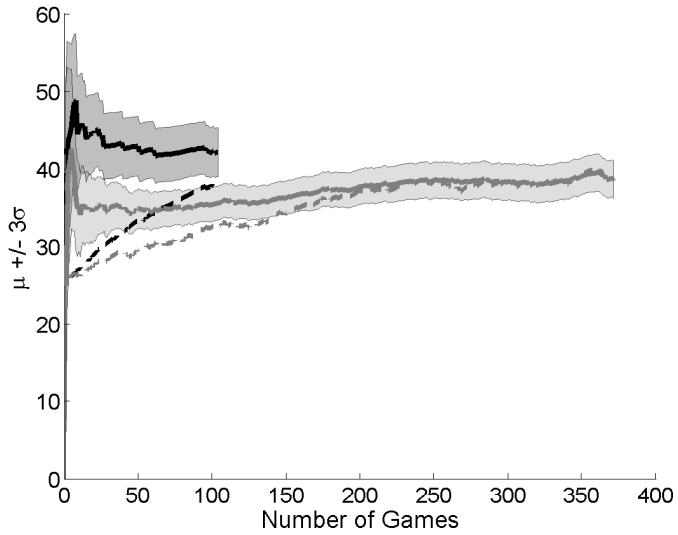

Fig. 15. Convergence trajectories for two players comparing Elo (dashed lines) with *TrueSkill* (solid lines). Note that the latter includes uncertainty estimates, shown by the shaded regions.

approximation algorithms, which provide efficient solutions to inference problems, can be cast in terms of messages passed locally between nodes of the graph.

In many conventional machine learning applications, the formulation of the model, and the algorithm used to perform learning and make predictions, are intertwined. One feature of the new framework is that there is a beneficial separation between the formulation of the model in terms of its graphical structure, and the solution of inference problems using local message passing. Research into new inference algorithms can proceed largely independently of the particular application, while domain experts can focus their efforts on the formulation of new application-specific models. Indeed, general purpose software can be developed[1] which implements a range of alternative inference algorithms for broad classes of graphical structures. Another major benefit of the new framework is that it allows fully Bayesian methods to be applied to large scale applications, something which was previously not feasible.

We live in an increasingly data-rich world, with ever greater requirements to extract useful information from that data. The framework for machine learning reviewed in this paper, which scales well to large data sets, offers the opportunity to develop many new and exciting applications for machine learning in the years ahead.

[1] One example is *Infer.Net* which is described at:
http://research.microsoft.com/mlp/ml/Infer/Infer.htm

References

1. Baum, L.E.: An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. Inequalities 3, 1–8 (1972)
2. Berger, J.O.: Statistical Decision Theory and Bayesian Analysis, 2nd edn. Springer, Heidelberg (1985)
3. Bernardo, J.M., Smith, A.F.M.: Bayesian Theory. Wiley, Chichester (1994)
4. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon limit error-correcting coding and decoding: Turbo-codes (1). In: Proceedings ICC 1993, pp. 1064–1070 (1993)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
6. Bishop, C.M., Spiegelhalter, D., Winn, J.: VIBES: A variational inference engine for Bayesian networks. In: Becker, S., Thrun, S., Obermeyer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15, pp. 793–800. MIT Press, Cambridge (2003)
7. Dangauthier, P., Herbrich, R., Minka, T., Graepel, T.: Trueskill through time: Revisiting the history of chess. In: Advances in Neural Information Processing Systems, vol. 20 (2007), `http://books.nips.cc/nips20.html`
8. Frey, B.J.: Graphical Models for Machine Learning and Digital Communication. MIT Press, Cambridge (1998)
9. Frey, B.J., MacKay, D.J.C.: A revolution: Belief propagation in graphs with cycles. In: Jordan, M.I., Kearns, M.J., Solla, S.A. (eds.) Advances in Neural Information Processing Systems, vol. 10. MIT Press, Cambridge (1998)
10. Gallager, R.G.: Low-Density Parity-Check Codes. MIT Press, Cambridge (1963)
11. Glickman, M.E.: Parameter estimation in large dynmaical paird comparison experiments. Applied Statistics 48, 377–394 (1999)
12. Herbrich, R., Minka, T., Graepel, T.: Trueskill$^{\mathrm{tm}}$: A Bayesian skill rating system. In: Advances in Neural Information Processing Systems, vol. 19, pp. 569–576. MIT Press, Cambridge (2007)
13. Jaynes, E.T.: Probability Theory: The Logic of Science. Cambridge University Press, Cambridge (2003)
14. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the American Society for Mechanical Engineering, Series D, Journal of Basic Engineering 82, 35–45 (1960)
15. MacKay, D.J.C., Neal, R.M.: Good error-correcting codes based on very sparse matrices. IEEE Transactions on Information Theory 45, 399–431 (1999)
16. McEliece, R.J., MacKay, D.J.C., Cheng, J.F.: Turbo decoding as an instance of Pearl's 'Belief Ppropagation' algorithm. IEEE Journal on Selected Areas in Communications 16, 140–152 (1998)
17. Minka, T.: Expectation propagation for approximate Bayesian inference. In: Breese, J., Koller, D. (eds.) Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, pp. 362–369. Morgan Kaufmann, San Francisco (2001)
18. Minka, T.: A family of approximate algorithms for Bayesian inference. Ph.D. thesis, MIT (2001)
19. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–285 (1989)
20. Rauch, H.E., Tung, F., Striebel, C.T.: Maximum likelihood estimates of linear dynamical systems. AIAA Journal 3, 1445–1450 (1965)
21. Zarchan, P., Musoff, H.: Fundamentals of Kalman Filtering: A Practical Approach, 2nd edn. AIAA (2005)