Machine Learning for Computer Graphics: A Manifesto and Tutorial

Aaron Hertzmann
University of Toronto
www.dgp.toronto.edu/~hertzman

Abstract

I argue that computer graphics can benefit from a deeper use of machine learning techniques. I give an overview of what learning has to offer the graphics community, with an emphasis on Bayesian techniques. I also attempt to address some misconceptions about learning, and to give a very brief tutorial on Bayesian reasoning.

1. Introduction

We live in an age of widespread exploration of art and communication using computer graphics and animation. Filmmakers, scientists, graphic designers, fine artists, and game designers, are finding new ways to communicate and new kinds of media to create. Computers and rendering software are now quite powerful. Arguably, the largest barrier to using digital media is not technological limitations, but the tedious effort required to create digital worlds and digital life. Suppose you wish to simulate life on the streets of New York in the 1930's, or a mysterious alien society. Someone has to actually create all the 3D models and textures for the world, physics for the objects, and behaviors or animations for the characters. Although tools exist for all of these tasks, the sheer scale of even the most prosaic world can require months or years of labor. An alternative approach is to create these models from existing data, either designed by artists or captured from the world. In this paper, I will argue that fitting models from data can be very useful for computer graphics, and that machine learning can provide powerful tools. I will attempt to address some of the common concerns about this approach. I will also provide a brief tutorial on probabilistic reasoning.

Consider the problem of creating motions for a character in a movie. You could create the motions procedurally, i.e. by designing some algorithm that synthesizes motions. However, synthetic motions typically look very artificial and lack the essential style of a character, and pure procedural synthesis is rarely used for animation in practice. More commonly, one animates characters "by hand," or captures motions from an actor in a studio. These "pure data" approaches give the highest quality motions, but at substantial cost in time and the efforts of artists or actors. Moreover, there is little flexibility: if you discover that you did not get just the right motions in the studio, then you have to go back and capture more. The situation is worse for a video game, where one must capture all motions that might conceivably be needed.

Learning techniques promise the best of both worlds: starting from some captured data, we can procedurally synthesize more data in the style of the original. Moreover, we can constrain the synthetic data, for example, according to the requirements of an artist. Of course, we must begin with some data produced by an artist or a capture session. But now, we can do much more with this data than just simple playback. For these problems, machine learning offers an attractive set of tools for modeling the patterns of data.

Data-driven techniques have shown a small but steadily increasing presence in graphics research. Principal components analysis and basic clustering algorithms are becoming almost *de rigueur* at SIGGRAPH. Most significantly, the recent proliferation of papers on texture synthesis and motion texture suggests a growing acceptance of learning techniques. However, the acceptance of these works relies largely on their accessibility — one does not need to know much about machine learning to implement texture synthesis. In this article, I will argue that graphics can benefit more deeply from the learning literature.

Bayesian reasoning. Although I will discuss techniques from machine learning in general, I particularly advocate the use of Bayesian methods. Bayesian statistics is one of the main forms of statistical modeling. Bayesian learning provides three main tools:

1. Principled modeling of uncertainty
2. General purpose models for unstructured data
3. Effective algorithms for data fitting and analysis under uncertainty

I will give simple but detailed examples later on. Of the existing graphics research that uses Bayesian learning, most

of them use existing Bayesian methods as a "black box" within a larger model. I advocate modeling the entire system within a Bayesian framework, which requires more understanding of Bayesian learning, but yields much more powerful and effective algorithms.

There are also many useful non-probabilistic techniques in the learning literature as well. I put more emphasis on probabilistic methods, since I believe these have the most value for graphics.

Intended audience for this paper. This paper is written primarily for computer graphics researchers. There seems to be some resistance to machine learning in the graphics community. For example, one researcher said to me last year:

> Graphics researchers are already starting to grumble about "machine learning doesn't really work, maybe only on the two examples shown in the paper." ... Without commenting on whether this might actually be true or not, I just want to note that I've heard increasing amounts of this sentiment.

On the other hand, the graphics community is very diverse, and cannot be summarized by a single sensibility. A computer vision researcher with an electrical engineering background doubts there is any real controversy:

> I am surprised that you think there's so much resistance to machine learning (even the term)! ... Certainly in EE or other branches of CS (e.g. database query, robotics, etc.), "machine learning" is almost a classical term, and covers an amazing amount of territory. No one would be shocked by someone using "machine learning" in any of the literature I read ...

However, my general impression of the attitude of the graphics community (which could be wrong) is of a mixture of curiosity with deep skepticism. In my opinion, insofar as there is any resistance, this resistance stems from misconceptions about learning and its application. At one time I expected "learning" to be a magical black box that discovers the meaning of some raw data, and I suspect that others expect the same. This promise — unfulfilled and possibly unfulfillable — naturally breeds skepticism. Although this black box does not exist, machine learning research has been very fertile in many domains, even without solving the AI problem. It is a truism that artificial intelligence research can never become successful, because its successes are not viewed as AI. Recent successes include work in bioinformatics, data mining, spam filters, and medical diagnoses. For the reader who is bothered by the term "machine learning," I suggest mentally substituting the phrase "statistical data fitting" instead.

Moreover, data-fitting techniques are widely used in graphics — whether one is fitting a 3D surface to a point cloud obtained from a laser range scanner, or fitting a Mixtures-of-Gaussians (MoG) model to motion capture data, one is fitting a structured model to observed data. It should be noted that the MoG model is a direct generalization of vector quantization, which is already widely used in graphics. Similarly, one may think of a Hidden Markov Model (HMM) as a probabilistic generalization of vector quantization that models temporal coherence.

One may also object to learning techniques because they take away control from the artist — but this is really a complaint about all procedural techniques. In my opinion, the goal of procedural techniques is not to replace the artist, but to provide effective high-level tools. Data-driven methods give the artist the ability to build from captured data, and the ability to design styles by example rather than by setting thousands of parameters manually.

2. What is machine learning?

For the purposes of computer graphics, machine learning should really be viewed as a set of techniques for leveraging data. Given some data, we can model the process that generated the data. Then, we can make more data that is consistent with this process, possibly with new, user-defined constraints.

In learning, we combine our prior knowledge of the problem with the information in the training data; the model that we fit should be carefully chosen. On one hand, trying to model everything about the world — such as the exact shape and dynamics of the muscle tissue in a human actor and the actor's complete mental state — would be hopeless. Instead, we must fit simpler models of observed data, say, of the movements of markers or handles; the parameters of this model will rarely have any direct interpretation in terms of physical parameters. On the other hand, choosing features that are too general may make learning require far too much data. For example, Blanz and Vetter [4] modeled the distribution of possible faces and expressions with a Gaussian probability density function. Such a weak model allowed them to model patterns in the data without requiring explicit *a priori* understanding of them. They can then generate new faces and expressions by sampling from this density, or by estimating the most likely pose that matches some input photograph. However, they did need to represent face data using training data in correspondence; directly learning a model from range data not in correspondence would not be likely to work very well at all. At present, learning algorithms do not perform magic: you must know something about the problem you want to model. As a rule of thumb, the less information you specify in advance, the more training data you will need in order to train a good model. It is

very difficult to get good results without having some high-level understanding of how the model operates. The main benefit is that we can still get good results with fairly high-level models.

3. What does learning have to offer?

The idea of driving graphics from data is hardly new, and one can build some models from data without knowing anything about machine learning. One could argue that the word "learning" should be avoided in computer graphics since it leads to the sort of unrealistic expectations mentioned above. However, I believe that using the tools, terminology, and experience of the machine learning community offer many benefits to computer graphics research and practice. By employing existing ideas and techniques, we get the benefit of the collective experience of the researchers who studied these problems in the past. Otherwise, we will waste substantial effort reinventing the wheel. The literature provides many intellectual tools that can be applied again and again. For example, the authors of the Composable Controllers paper from SIGGRAPH 2001 [13] sought an algorithm to classify data based on some training examples. Rather than attempting to solve the classification problem from scratch, the authors simply used Support Vector Machines (SVMs), a state-of-the-art classification procedure that has consistently outperformed competing methods (including neural networks, and, in this case, nearest-neighbors). In fact, they did not even have to implement an SVM classifier; instead, they downloaded one from the web. This kind of reuse illustrates how accessing existing research can save us from having to reinvent (and reimplement) the wheel. Moreover, it is unlikely that anyone would casually invent a technique as effective as SVMs in the course of conducting a larger project.

In general, the machine learning literature and community have much to offer graphics:

Problem taxonomy. The literature makes a distinction between types of problems, such as density estimation, classification, regression, and reinforcement learning. See Section 4 for more detail. Understanding these distinctions helps one understand a new problem and relate it to existing approaches. For example, the authors of the Video Textures paper of SIGGRAPH 2000 [51] identified their synthesis problem as a reinforcement learning problem, which allowed them to immediately draw on existing solutions rather than to attempt to solve the problem from scratch.

General-purpose models. Machine learning researchers have developed many models for learning structure in arbitrary data. For many fitting problems, it is likely that one of these methods will be useful, either as a complete model or as a starting point for a problem-specific model. Many of these methods are outlined in the next section.

Reasoning with probabilities. One of the major trends in learning research is to reason with probabilities, in order to model the uncertainty present in all of our models and data. Probabilistic modeling provides a very powerful, general-purpose tool for expressing relative certainty in our understanding of the world. Often, one source of information will be more reliable than another, and we must weigh the reliability of data along with the data itself when making estimates or decisions; probability theory provides a principled mechanism for reasoning with uncertainty, learning from data, and generating new data (e.g. by sampling from a learned model). Machine learning researchers have developed (or adapted from other disciplines) many powerful tools for statistical reasoning, such as Expectation-Maximization, Belief Propagation, Markov Chain Monte Carlo methods, and Particle Filtering. Although probabilistic reasoning is not necessary for every problem (and it will always be dependent on some *a priori* assumptions that we make about the world), it has been shown to be a very powerful tool in many situations. Some cognitive science researchers even believe that the human brain can be viewed as performing probabilistic inference, at least in low-level processes [45].

A few papers in graphics have used techniques from learning in interesting ways. Of these few papers, most of them use an existing learning technique as a "black box" subroutine. While these uses are exciting, we have yet to see much work that does not just reuse models but tightly fits them into a graphics system. In contrast, the interaction of learning and computer vision is much more mature. In much computer vision research, there is no "learning subroutine," but a unified system that completely models the process being analyzed. For example, Jojic and Frey's video processing algorithms extract sprites and solve for all relevant parameters in a unified probabilistic framework [25].

Incidentally, probabilistic methods can be useful in graphics entirely separate from data-driven techniques, as argued convincingly by several authors [1, 7, 42, 43]. For digital actors and behaviors, it is important that the animation is not the same every time. Probabilistic models allow multiple solutions to a problem, and can model random subtleties for which an exact model is impractical. (Probabilistic methods have long been used in global illumination, but only as part of numerical integration techniques, not to represent the uncertainty in the scene).

Some probabilistic methods and deterministic methods end up with the same formulations. In the past, this has been a source of contention in the computer vision community. Some people argue that, if one can pose the problem

simply as a least-squares fitting problem (as one does with regression methods such as radial basis functions and neural networks) then there is no need for probabilistic methods. I agree with this, however, if the problem involves weighing between multiple terms, thresholding, and/or estimating terms that have very different meanings, then generally a probabilistic technique will be necessary in order to fit these parameters. For example, linear regression can be posed in a deterministic least-squares setting, and there is no real need to state an explicit noise model. Linear regression where both variables are corrupted by noise, and linear regression that is robust to outliers requires either (a) parameter tuning by the user, or (b) probabilistic methods that can learn the noise and outlier models. Another example is given in Section 5.3.

Note that there is occasionally some confusion in that the probabilistic methods mentioned here are *not* necessarily randomized algorithms. Probabilistic methods model uncertainty, but often involve deterministic optimizations. Randomized algorithms may be used in optimization, and random sampling may be used to draw from a distribution.

Learning all the parameters. Most computer graphics systems (including many current data-driven algorithms) have many parameters to tune. (This fact that is often mentioned in paper reviews; it is a very safe thing to comment on). Bayesian reasoning provides ways to fit energy functions to data, even energy functions that are too complicated to fit by hand. Moreover, I will go on a limb here and say that machine learning systems can learn *all* of the parameters of a model. There are a few caveats: you must choose a model that is suitably powerful for the problem you wish to solve, there must be enough training data, and you must be willing to perform a potentially slow optimization procedure. Probabilistic modeling provides a principled problem formulation for learning all the parameters, although optimizing the resulting objective function may be difficult for certain types of parameters. However, there is flexibility — a good model with few parameters needs less training data and time than a weak model with many parameters. In practice, one will generally specify a few parameters of the model that are difficult to learn (e.g. model dimensionality), and have the algorithm learn the rest (including noise values, outlier thresholds, data labeling, and so on). Of course, if more user control is desired, than one may allow some of the parameters to be specified by hand.

For many graphics applications, the learning process may be viewed as **learning the objective function** for procedural synthesis. Objective functions and energy functions are widely used throughout computer graphics; one typically synthesizes data by optimizing an energy function subject to some constraints. For example, geometric models are often created by optimizing a specific objective function (sometimes implicitly). Instead of designing this objective function by hand, we could use machine learning methods to create the objective function from data — or, more precisely, to fit the parameters of an objective function. Synthesis is then a matter of optimizing with respect to this objective. In a sense, the learned objective function measures the similarity of the synthesized motion to the examples, but in a much more general way. See Section 5.3 for a detailed example.

4. Taxonomy of Problems, with References

In this section, I give an overview of some of the main problem areas in learning, together with some related references. I do not at all promise that this is an authoritative or complete list. I highly recommend the machine learning chapters of MacKay's book [32] as an introduction to Bayesian learning and reasoning. Also, the first two chapters of Bishop's book [3] provide an excellent overview of statistical learning. Forsyth and Ponce's computer vision textbook [14] is a good reference, and contains a significant amount of content on learning related to computer vision.

Learning is typically broken up into three problem areas: Classification, Regression, and Density Estimation. In addition, several other important topics are summarized below.

Classification. Perhaps the most-studied problem in learning is classification: learning to classify data points as belonging to one of two classes. In a typical problem, one is first given a set of training data $\{\mathbf{x}_i, y_i\}$, where each value $\mathbf{x}_i$ is a floating-point vector representing a measurement, and each $y_i \in \{-1, 1\}$ is a binary classification of the training vector $\mathbf{x}_i$. For example, in a classifier that learns cancer diagnoses, the $\mathbf{x}_i$ values might represent various measurements of blood level counts, age, weight, etc., and y_i might indicate whether the diagnosis is positive or negative.

Related reading: Perhaps the most prominent classification algorithm is Support Vector Machines; Burges provides an excellent tutorial [6]. There are several other competitive techniques, such as AdaBoost [50]. One effective application of AdaBoost in vision is real-time face recognition [58].

Regression. In regression, one attempts to learn a mapping $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{x}$ represents some input vector, and $\mathbf{y}$ represents an output vector. A simple example is linear regression, where $f(\mathbf{x}) = \mathbf{A}\mathbf{x}+\mathbf{b}$. Given a set of training data $\{\mathbf{x}_i, \mathbf{y}_i\}$, these mappings are typically learned by minimizing a least-squares error criterion (e.g. $\sum_i ||f(\mathbf{x}_i) - \mathbf{y}_i||^2$), plus some smoothing terms (a.k.a. "regularization"). Neural networks and Radial Basis Functions are both regression techniques, which use two specific forms for the function

$f(\mathbf{x})$; . (RBFs were originally developed for scattered data interpolation and later adopted within the learning community). Radial Basis Functions and neural networks have already found a fair amount of use in graphics. Gaussian Processes, a more recent technique, provide a much more elegant way to formulate regression problems [61].

Related reading: Bishop's book [3] provides several chapters on neural networks and RBFs. Mackay provides a good tutorial on Gaussian Processes [33].

Density estimation. In density estimation, one attempts to learn a probability distribution from some training data vectors $\{\mathbf{x}_i\}$. One assumes that the data was sampled from some PDF $p(\mathbf{x})$, and attempts to fit parameters of the PDF that best explain the data. This problem is discussed in more detail in Section 5.

Related reading: MacKay's book is a good starting point [32]. Frey and Jojic give a detailed tutorial on learning and graphical models, with a focus on computer vision problems [15].

Relation of classification, regression, and density estimation. The boundaries between these different areas is somewhat fuzzy. For example, one can build a classifier by first learning PDFs $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$ for the two classes, and then classifying new values by testing if $p_0(\mathbf{x}) > p_1(\mathbf{x})$ for the new data. One can learn a classifier by solving a regression problem that maps from input values to positive or negative numbers. And one can learn classifiers and regression in a probabilistic setting. For example, we can learn the parameters of a regression model in a setting of noisy or uncertain data, and also learn the noise model.

That being said, it is often best to use the algorithms designed for a certain task — e.g. a classifier is optimized for classification, whereas density estimation is not. Regression is most appropriate in cases where each input value $\mathbf{x}$ maps to a single output value $\mathbf{y}$ (or is corrupted by Gaussian noise). If there are multiple possible $\mathbf{y}$ values for $\mathbf{x}$, then it may be better to learn the conditional probability distribution $p(\mathbf{y}|\mathbf{x})$ or the joint distribution $p(\mathbf{x}, \mathbf{y})$. Density estimation requires one to select an appropriate functional form for the density — that form might be a regression model plus noise.

Time-series analysis. An important special case of density estimation is density estimation for time-series data or sequential data; these models are also sometimes called **Dynamic Bayesian Networks**. Perhaps the simplest model for continuous data is the Linear Dynamical System (LDS). For example, consider the motion of an object, with coordinates $\mathbf{x}_t$ at time t. In an LDS, we assume that (a) our observations of $\mathbf{x}_t$ are corrupted with Gaussian noise, and (b) that the motion of the object is linear, e.g. velocity is constant plus a Gaussian PDF. In this case, most analysis can be done in closed form. One special case — when we have no knowledge of future events past the current time t — is known as the Kalman filter. However, linear dynamics are an inappropriate model for many real-world cases with complex dynamics. There are many non-linear generalizations of LDS that yield tractable learning algorithms (e.g. [18, 19, 30, 40]). For complex non-linear dynamics, the technique known as particle systems is very widely used, particularly in visual tracking and robotics. Another well-known time-series model is the Hidden Markov Model (HMM), which is widely used in text processing, and a continuous form in speech processing.

Related reading: These topics are widely presented in the signal-processing literature. Rabiner's tutorial is the classic reference for HMMs [44]. Linear Dynamical Systems can be a very deep topic, with relations to analog circuit design, mass-spring systems, and so on, and textbooks in these areas and ordinary differential equations can provide insight into the general properties of LDSs. Some of this material is discussed by Forsyth and Ponce [14] as well as the special case of Kalman filtering. Ghahramani and Hinton describe an algorithm for learning LDSs with the EM algorithm [17]. Welch and Bishop have written a tutorial [60] and webpage focusing on Kalman filters. Several authors have developed tractable non-linear dynamical systems [18, 19, 30, 40]. For more difficult problems, applications in computer vision and robotics use methods such as particle filtering and related methods (e.g. Condensation [21]).

Reinforcement learning. Reinforcement learning is a method for decision-making agents to learn optimal solutions to tasks [27]. Reinforcement learning is sometimes used for creating robot controllers. So far, they have used relatively little in graphics, Video Textures being the only exception [51], and a very non-traditional application at that.

Dimension reduction. In dimension reduction, one attempts to learn a low-dimensional manifold to represent complex data. The idea is that the observed data $\{\mathbf{x}_i\}$ can be described by some natural, continuous parameterization $\mathbf{x} = f(\mathbf{z})$. The goal of dimension reduction (DR) is to learn the parameterization f, even though the manifold structure of the data $\mathbf{z}$ is unknown. DR is often used a preprocess, or as the entire learning process (there is no sharp division between DR and other types of learning), depending on the difficulty of the problem and the sophistication of the DR technique. For example, Matusik et al. [34] apply non-linear dimension reduction (NLDR) to the space of Bidirectional Reflectance Distribution Functions (BRDFs), in order to learn a natural parameterization for capturing,

representing and specifying BRDFs.

Principal components analysis (PCA) is an extremely simple, linear DR method that is primarily used as a preprocess for high-dimensional problems. This can be used as the entirety of the learning process (e.g. as in "eigenfaces" [28, 57]), or as a preprocess before executing a more sophisticated learning procedure. For example, we used PCA as a preprocess in our Style Machines [5] technique. The human body pose data is typically contains 30-40 degrees-of-freedom (DOFs). However, our algorithm would learn very slowly on such a large number of DOFs. Applying linear dimension reduction was very fast, and, the resulting 10-dimensional space maintained the important underlying structure of the motion data; we could then apply our more sophisticated analysis to the linearly-reduced data.

Related reading: Many sophisticated NLDR learning algorithms have been developed; a few of the more interesting ones are IsoMap [55], Locally-Linear Embedding (LLE) [49], Locally Linear Coordination (LLC) [54], Non-Linear PCA [36, 52], and Independent Components Analysis (ICA) [2].

Clustering. Clustering refers to techniques to segmenting data into coherenct "clusters." Perhaps the simplest and most widely-known method is k-means clustering (a.k.a vector quantization) [3, 32, 16]. The Mixtures-of-Gaussians model is a probabilistic generalization of this [3]. Recently, a sophisticated class of techniques has emerged under the name "spectral clustering;" these methods can discover clusters with complex shapes (e.g. see [39, 53, 59]), although a formal understanding of such methods has been harder to come by (although progress is being made in this direction [35, 47]). A method known as mean-shift has also been found to be extremely effective in computer vision problems [8], although again without a strong theoretical foundation (as far as I know).

Model selection. One common question that occurs when building models is: "How big should the model be?" A model with more parameters will fit the data better, but it could also overfit the data. For example, we may not know in advance how many clusters to use in a clustering problem, or how many PCA dimensions to keep. The more general problem is: "Given a choice of two models, which one is more appropriate to the data?" This question is addressed by model selection techniques, most of which implement some formalization of Occam's Razor. Although there is substantial controversy over which is the "right" model selection algorithm (or even if one exists [11]), the use of some technique will generally improve performance.

Related reading: Many model selection principles have been proposed, such as Maximum Entropy [24], and Minimum Description Length [20]. One of the most intriguing techniques, Bayesian Model Selection [31, 32], applies Bayesian reasoning to determine the best model to fit any problem *without* making any additional assumptions about which models are better; many previous methods can be viewed as approximations to this approach, and I am currently very persuaded by the arguments for this approach. However, this approach can be very difficult to work with mathematically.

5. A brief tutorial in Bayesian probabilistic reasoning and learning

"Probability theory is nothing but common sense reduced to calculation." — Marquis de Laplace, 1814 [29]

In this section I will give an overview of some basic concepts of probabilistic reasoning and learning. I will then show a few basic examples to illustrate these concepts. Although none of these examples will be extremely sophisticated, my hope is that they will suffice to make the basic concepts and terminology clear.

The most interesting material from a graphics perspective is in Section 5.3, so you might want to skip directly there if the following material is already famililar.

5.1. Probabilistic reasoning

The classical model for reasoning and decision-making is pure logic, originally proposed by the ancient Greeks. In pure logic, you begin with a set of known facts about the world, and use logical rules to deduce additional facts. Pure logic is a very clean and elegant way to describe reasoning.

Unfortunately, pure logic assumes that all of your premises are known with absolute certainty, which makes it useless as a model for how humans actually reason about the real world, or for how computers should reason. For example:

- When deciding whether or not to take your umbrella with you when you go out, you need to know whether or not it will rain. Since it is not possible to predict the weather with absolute certainty, we normally make an estimate of the likelihood that it will rain, by combining various sources of information such as the current weather (is it currently overcast? windy?), the weather forecast, yesterday's weather, and our experience with weather patterns. Pure logic is of no use here — it can only be used to deduce that "it might rain," which is of little help in making a practical decision.
- When you meet someone new, you immediately make hundreds of inferences (most of them unconscious) about who this person is and what their emotions and

goals are. You make these decisions based on the person's appearance, they way they are dressed, their facial expressions, their actions, the context in which you meet, and what you have learned from previous experience with other people. Of course, you have no conclusive basis for making quick opinions (e.g. the panhandler you meet on the street might be a method actor preparing for a role). However, we need to be able to make judgements about other people based on incomplete information; otherwise, normal interpersonal interaction would be impossible (e.g. how do you really *know* that everyone isn't out to get you?).

Probability theory provides the tools necessary for reasoning under uncertainty. The central idea is to model uncertainty with probability distributions. Every variable in our model of the world is treated as not having a known value, but having a range of possible values, some more likely than others. In other words: every variable has a fixed value, but we do not know these values, so we must model their range of possible values. Probabilities are subjective, in that they reflect an individual's level of uncertainty given their knowledge and assumptions, and may change over time as new information is acquired. This mode of reasoning is usually known as *Bayesian statistics*, in contrast to *frequentist* (or "orthodox") statistics which is more commonly taught in statistics classes and in the sciences. More on that distinction in a moment.

A coin-flipping example. To introduce a concrete example, suppose we observe a coin being flipped. Let $\mathbf{H}$ be a variable that indicates the result of the flip: $\mathbf{H} = \mathsf{heads}$ if the coin lands on its head, and $\mathbf{H} = \mathsf{tails}$ otherwise. Although this may seem like familiar territory at first, if your experience is primarily with frequentist statistics, it will soon be quite different from what you are used to.

Suppose you flip a coin, what is the probability that the coin ends up heads? This probability should be some real number h, $0 \leq h \leq 1$. For most coins, we would say $h = .5$. What does this number mean? The number h is a representation of our belief about the possible values of $\mathbf{H}$. Some examples:

$h = 0$	we are absolutely certain the coin will land tails
$h = 1/3$	we believe that tails is twice as likely as heads
$h = 1/2$	we believe heads and tails are equally likely
$h = 1$	we are absolutely certain the coin will land heads

Probabilities can also describe events that have not yet happened, or, more generally, our belief are whether certain statements are true or false. For example, we can use the probability distribution $p(h)$ to describe our beliefs about what biases of the coin are more likely. We can talk about the probability that the stock market will go up next month, that a lone gunner shot JFK, or that you forgot to turn off the stove when leaving your house. A "probability" is nothing more than a mapping from events to real numbers, obeying a few basic rules.

If you are used to the thinking about probabilities as frequencies of repeated events, the above view may seem arbitrary at first. However, Cox [9] has shown that *any* mapping of events to real values that obeys a few intuitive properties *must* lead to the exact same familiar axioms of probability theory. In other words, suppose you are uncomfortable using the word "probability" in this highly-subjective way. But suppose that you would like to describe reasoning under uncertainty with a system based on real numbers anyway. As long as your system obeys a few simple requirements — and these rules are hard to argue against — then Cox showed that you will have, in fact, rederived basic probability theory, and will be using the same Bayesian reasoning as everyone else, whether or not you like to call it that. See the first two chapters of Jaynes' book [23] for a detailed discussion of this point.

Formally, we denote the probability of the coin coming up heads as $P(\mathbf{H} = \mathsf{heads})$, or $P(\mathbf{H})$ for short. In this case, $P(\mathbf{H} = \mathsf{heads}) = h$. In general, we denote the probability of a specific event event as $P(\mathsf{event})$.

Frequentist statistics. In contrast to the above view, frequentist (or "orthodox") statistics defines probabilities in terms of repeated events. For example, suppose we flip a coin many times; what proportion of those trials will land heads? In the frequentist view, the probability of heads is defined as the limit of this ratio as the number of trials goes to infinity; one generally assumes absolute certainty about the other variables in the experiment. This definition of probability has had success in areas where repeated trials are possible, such as in biology and chemistry, where one can perform thousands of repeated tests on chemicals or plants. However, in cases where one cannot repeat trials, the frequentist view is useless for modeling uncertainty. For example, we make judgements based on meeting someone for the first time despite not having thousands of interactions; similarly, in graphics, we would like to synthesize data from small amounts of user input.

There is often confusion in the distinction between Bayesian and Frequentist statistics, since they both yield the same estimates in some very simple cases. However, in most nontrivial examples, frequentist methods provide relatively little value — even the simple coin-flipping reasoning that I will describe in the next section cannot be modeled in the Frequentist view. For an entertaining (though one-sided) history of the debates between Bayesian and Frequentist statistics, see Jaynes [23] (Chapter 16). Minka [37] gives examples of problems with frequentist methods that occur in simple cases.

Although Bayesian methods are very strong in the learning literature, frequentist methods remain useful and widely used in learning — the popular Support Vector Machine (SVM) architecture is frequentist[1]. However, these methods are primarily concered with classification, which, in my opinion, is of much less interest for graphics applications than is density modeling. (In many of the sciences, I am of the impression that frequentist methods remain dominant, as evidenced by the preeminence of significance testing).

Fuzzy logic. Fuzzy logic became fashionable in the 1980's as a way to model a notion of "partial" set membership. In my opinion, this formulation ultimately attempts to express the notion of uncertainty, but does so clumsily; Bayesian methods express it much more elegantly, while also modeling continuously-valued data. Fuzzy logic also appears to be problematic mathematically [12].

5.2. A more detailed discrete example

If we flip a coin and observe the result, then we can be pretty sure that we know the value of $\mathbf{H}$; there is no real need to model the uncertainty in this measurement. However, suppose we do not observe the coin flip, but instead hear about it from a friend, who may be forgetful or untrustworthy. Let $\mathbf{F}$ be a variable indicating how the friend claims the coin landed, e.g. $\mathbf{F} = \mathsf{heads}$ means the friend says that the coin came up heads. How can we make our own estimate of what happened? As we shall see, probabilistic reasoning can obtain quantitative values that correspond directly to our common sense reasoning about likely.

Suppose we know something about our friend's typical behavior. We can represent our beliefs with the following probabilities: $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads})$ represents the likelihood that the friend says "heads" when the the coin landed heads, and so on. Because the friend can only say one thing, we know

$$P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) + P(\mathbf{F} = \mathsf{tails}|\mathbf{H} = \mathsf{heads}) = 1$$
$$P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{tails}) + P(\mathbf{F} = \mathsf{tails}|\mathbf{H} = \mathsf{tails}) = 1$$

If our friend always tells the truth, then we know $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = 1$ and $P(\mathbf{F} = \mathsf{tails}|\mathbf{H} = \mathsf{heads}) = 0$. If our friend *usually* lies, then, for example, we might have $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = .3$

Knowing these probabilities and knowing h allows us to make useful estimates of the coin flip. Supppose our friend says that the coin landed "heads." What is our estimate of the actual state of the coin? In order to make this estimate, we can compute $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads})$, the likelihood that the coin landed heads *given that our friend said it landed heads*. By the axioms of probability theory, we know that $P(a, b) = P(a|b)P(b)$ for any random variables a and b. ($P(a, b)$ denotes the joint probability of a and b; that is, the probability that they both occur). Hence,

$$P(\mathbf{H}, \mathbf{F}) = P(\mathbf{H}|\mathbf{F})P(\mathbf{F}) = P(\mathbf{F}|\mathbf{H})P(\mathbf{H}) \quad (1)$$

Solving for $P(\mathbf{H}|\mathbf{F})$ gives:

$$P(\mathbf{H}|\mathbf{F}) = \frac{P(\mathbf{F}|\mathbf{H})P(\mathbf{H})}{P(\mathbf{F})} \quad (2)$$

($P(\mathbf{H}, \mathbf{F})$ is the joint probability over both events occurring). Using this formula allows us to estimate a distribution over the actual value of $\mathbf{H}$ given all of the evidence that we have (in this case, $\mathbf{F}$). In fact, this is an important formula, known as **Bayes' Rule** — it allows us to *make inferences about the world, based on observations of it*, while incorporating all the uncertainty in the system.

In our case, we are interested in determining:

$$P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads}) = \frac{P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads})P(\mathbf{H} = \mathsf{heads})}{P(\mathbf{F} = \mathsf{heads})} \quad (3)$$

We can also expand the denominator using the axioms of probability:

$$P(\mathbf{F} = \mathsf{heads}) = P(\mathbf{F} = \mathsf{heads}, \mathbf{H} = \mathsf{heads}) + P(\mathbf{F} = \mathsf{heads}, \mathbf{H} = \mathsf{tails}) \quad (4)$$

which can then be expanded using Equation 1. All of the terms of $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads})$ are known; now, observe what happens mathematically in different situations when our friend says "heads":

- If our friend is totally trustworthy ($P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = 1$), then we know with certainty that the coin landed heads, because $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads}) = (1 + 0)/(1 + 0) = 1$.
- If our friend always lies ($P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = 0$), then we know that the coin landed tails, by similar reasoning.
- If our friend behaves completely randomly ($P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = .5$), then $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads}) = .5h/(.5h + .5(1 - h)) = h = P(\mathbf{H} = \mathsf{heads})$. In this case, what our friend says gives no information about how the coin landed, and our uncertainty about its state is the same as if our friend had said nothing.
- If our friend sometimes tells the truth, but we know with certainty that the coin lands heads $h = 1$, then $P(\mathbf{H} = \mathsf{heads}) = 1$; in this case, since we know that the coin can only land heads, what the friend says is irrelevant.

[1] Although there is also a Bayesian derivation of SVMs [22].

- Suppose our friend usually tells the truth, e.g. $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = .7$ and $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{tails}) = .3$. In this case, $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads}) = .7h/(.7h + .3(1-h)) = .7h/(.4h + .3)$. So if $h = .5$, then $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads}) = .7$. If $h = .8$, then $P(\mathbf{H} = \mathsf{heads}|\mathbf{F} = \mathsf{heads}) \approx .91$.

In each of these cases, we combine the information provided by our friend, our prior assumptions about the friend's trustworthiness, and the coin's bias to obtain a probability estimate of whether the coin actually landed heads. The resulting distribution *qualitatively matches our common sense reasoning.* We can estimate whether the coin landed heads simply by evaluating whether $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) > .5$; if it is, then the coin is more likely to be heads. However, $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads})$ gives us more than just this estimate, it also expresses our uncertainty in the estimate. If $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = .99$ then we will guess heads with great certainty; if $P(\mathbf{F} = \mathsf{heads}|\mathbf{H} = \mathsf{heads}) = .55$, then, even though we would say that the coin probably landed heads, we are not very sure about this. These uncertainties can be used in decision-making (should you place a bet based on the coin toss?) and in further inferences (for other unknown events that depend on the coin toss).

The basic elements of Bayes Rule are used so often, that they have all been given names:

$$\underbrace{P(\mathbf{H}|\mathbf{F})}_{\text{posterior}} = \frac{\overbrace{P(\mathbf{F}|\mathbf{H})}^{\text{likelihood}}\overbrace{P(\mathbf{H})}^{\text{prior}}}{\underbrace{P(\mathbf{F})}_{\text{evidence}}} \quad (5)$$

In the general case, $\mathbf{H}$ is some variable we wish to estimate, given data $\mathbf{F}$. The **prior distribution** describes our assumptions about $\mathbf{H}$ before observing the data $\mathbf{F}$. The **likelihood distribution** describes the likelihood of $\mathbf{F}$ given $\mathbf{H}$ — it reflects our assumptions about how the data $\mathbf{H}$ was generated. The **posterior distribution** describes our knowledge of $\mathbf{H}$, incorporating both the data and the prior. In general, when people discuss their "priors," they are discussing prior assumptions made about the problem that is separate from the data at hand. The meaning of the **evidence** is somewhat more esoteric (it can be used for model selection [31, 32]); this variable is often ignored, since it is constant with respect to the unknown variable.

Learning. We can use these tools to learn a model of the world. Suppose we wish to learn h by flipping a coin 100 times and observing the results $\mathbf{H}_i$ for $i \in [1..100]$. Suppose k of the coin flips landed heads, and $100-k$ were tails. We now will treat h as the random variable to be estimated, and assume that we have a **uniform prior**:[2] $p(h) = 1$.

$$p(h|\mathbf{H}_1, ..., \mathbf{H}_{100}) = \frac{P(\mathbf{H}_1, ..., \mathbf{H}_{100}|h)p(h)}{P(\mathbf{H}_1, ..., \mathbf{H}_{100})} \quad (6)$$

$$= \frac{\prod_i P(\mathbf{H}_i|h)p(h)}{P(\mathbf{H}_1, ..., \mathbf{H}_{100})} \quad (7)$$

$$= \frac{.5h^k(1-h)^{100-k}}{P(\mathbf{H}_1, ..., \mathbf{H}_{100})} \quad (8)$$

This form gives a probabilty distribution over h that expresses our uncertainty over what h might be, and can be used we making predictions about later coin-tosses. Since maintaining this distribution is often unwieldly, and it is easier simply to compute a single estimate of h by taking its most-likely value: $\hat{h} = \arg\max p(h|\mathbf{H}_1, ..., \mathbf{H}_{100})$. This is called the **Maximum a Posteriori (MAP)** estimate of h (since it maximizes the posterior). We compute this estimate by setting $\partial p(h|\mathbf{H}_1, ..., \mathbf{H}_{100})/\partial h = 0$ and solving for h, yielding the intuitive estimate $\hat{h} = k/100$. (Note that we made use of the fact that $P(\mathbf{H}_1, ..., \mathbf{H}_{100})$ is constant with respect to h and can be ignored).

The general form of Bayes' Rule for learning is:

$$\underbrace{P(\mathsf{model}|\mathsf{data})}_{\text{posterior}} = \frac{\overbrace{P(\mathsf{data}|\mathsf{model})}^{\text{likelihood}}\overbrace{P(\mathsf{model})}^{\text{prior}}}{\underbrace{P(\mathsf{data})}_{\text{evidence}}} \quad (9)$$

where data is the data that we observed, and model denotes parameters of a model we wish to learn. In other words, MAP estimation entails choosing the model that assigns highest probability to the data, biased by our assumptions (if any) about which model is more likely.

Very often, we assume "uniform priors," in which $P(\mathsf{model})$ is a uniform distribution. In this case, $P(\mathsf{model}|\mathsf{data}) \propto P(\mathsf{data}|\mathsf{model})$, and estimating the model amounts to maximizing the likelihood function. This estimate of the model is called the **Maximum Likelihood (ML)** estimate; it is simply the MAP estimate under uniform priors. In this case, no preference is given to any specific models during estimation.[3]

So far, we have obtained a conventional result for estimating h. However, the power of the approach presented here is that we can easily generalize to more difficult situations, when our observations our noisy, or where there are multiple sources of information:

[2] Note that $p(h) = 1$ is a uniform probability distribution; it states that all values of $h \in [0, 1]$ are equally likely. It should *not* be read as "the probability of h." For example, the probability that h lies in the interval 0 to .3 is $\int_0^{.3} p(h)dh = .3$. The lowercase p is used in $p(h)$ because h is a real-valued variable rather than a discrete variable.

[3] However, it should be noted that uniform priors are not invariant to reparameterizing the model.

- Suppose we wish to estimate h based only on what our friend tells us about 100 coin flips, rather than observing them directly. In this case, we do not observe $\mathbf{H}_i$ directly, but rather indirectly through the friend. Again, we can solve for the optimal h by plugging in the elements of the model to $p(h|\mathbf{H}_1, ..., \mathbf{H}_{100})$ and optimizing. In this case, our final uncertainty about h will be increased if our friend is not always reliable, and may be skewed towards $h = 1$ or $h = 0$, if we believe the friend has a preference for lying one way or another.

- Suppose we get two different sources of information about each coin flip; perhaps our friend tells us something about every coin flip, and another, more reliable friend tells us something about just a few of those coin flips. We can merge these two sources of information — and will have less uncertainty for data where we get the more reliable information — to estimate h.

5.3. Learning Gaussians

The above examples were all given in the discrete case; in graphics, we are generally interested in continuous distributions. In this case, we wish to learn from a collection of training data vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ — these could represent body poses, face shapes and expressions, radiance transfer coefficients, or any other continuously-valued data. Perhaps the simplest and most general model of continuous data is the Gaussian probability distribution function (PDF):

$$p(\mathbf{x}|\mu, \phi) \quad = \quad \frac{1}{\sqrt{(2\pi)^d |\phi|}} e^{-(\mathbf{x}-\mu)^T \phi^{-1} (\mathbf{x}-\mu)/2} \quad (10)$$

where d is the dimensionality of a data vector. In this case, we model the data by assuming that it was randomly sampled from a Gaussian PDF with some mean vector μ and $d \times d$ covariance matrix ϕ. There are many good reasons to use Gaussians in situations where we cannot or do not want to make any further assumptions (see [3], Section 2.1.2).

As an example, consider the head-shape modeling described by Blanz and Vetter [4]. In this case, a single person's head is represented by a parameter vector $\mathbf{x}$, containing the 3D positions of a set of facial features and the colors of a texture map. Blanz and Vetter assumed that human head shapes and textures are "generated" by random sampling from a Gaussian PDF[4] . We are given a set of N head shapes, and would like to learn the parameters of the Gaussian (μ, ϕ). As described in the previous section, learning according the **Maximum A Posteriori** principle requires computing the values of μ and ϕ that maximize $p(\mu, \phi|\mathbf{X})$ with respect to these variables. In other words, given the data $\mathbf{X}$, we would like to compute the parameters of the Gaussian that is *most likely* to have generated the data. We will further assume uniform priors. In this case, the MAP estimate is equivalent to maximizing the likelihood:

$$p(\mu, \phi|\mathbf{X}) \quad \propto \quad p(\mathbf{X}|\mu, \phi) \quad (11)$$

which follows from Bayes Rule. Since we assume that the head shapes are independently sampled:

$$p(\mathbf{X}|\mu, \phi) \quad = \quad \prod_i p(\mathbf{x}_i|\mu, \phi) \quad (12)$$

A very common trick for optimizing this function is to instead optimize

$$L(\mu, \phi) \quad \equiv \quad -\ln p(\mathbf{X}|\mu, \phi) \quad (13)$$

The function $L(\mu, \phi)$ is usually known as the "negative log-likelihood," for reasons which should be obvious. Since $\ln$ is a monotonically-increasing function, minimizing $L(\mu, \phi)$ is equivalent to maximizing $p(\mu, \phi|\mathbf{x})$. We could use either one, but the negative log-likelihood is usually much easier to work with. Simplifying $L(\mu, \phi)$ gives

$$\begin{aligned} L(\mu, \phi) &= -\sum \ln p(\mathbf{x}_i|\mu, \phi) && (14) \\ &= \sum_i (\mathbf{x}_i - \mu)^T \phi^{-1} (\mathbf{x}_i - \mu)/2 + && (15) \\ &\quad \frac{N}{2} \ln |\phi| + \frac{Nd}{2} \ln(2\pi) && (16) \end{aligned}$$

Solving for μ and ϕ by setting $\partial L(\mu, \phi)/\partial \mu = 0$ and $\partial L(\mu, \phi)/\partial \phi = 0$ gives the maximum likelihood estimates:

$$\begin{aligned} \hat{\mu} &= \frac{1}{N} \sum_i \mathbf{x}_i && (17) \\ \hat{\phi} &= \frac{1}{N} \sum_i (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T && (18) \end{aligned}$$

The ML estimates make intuitive sense: we estimate the Gaussian's mean to be the mean of the data, and the Gaussian's covariance to be the covariance of the data. Maximum likelihood estimates usually make sense intuitively. This is very helpful when debugging your math — I have found bugs in derivations on a few occasions simply because the ML estimates did not look right.

The $L(\mu, \phi)$ can be viewed as an energy function to be optimized for μ and ϕ. Inspecting the terms of $L(\mu, \phi)$ can be enlightening. The first term measures the fit of the data to the model. Note that the first and second terms must be balanced to optimize ϕ: the first term prefers large covariances ($\phi \to \infty$), whereas the second term penalizes increasing ϕ. The second term can be thought of as a penalty for

[4] Due to the large size of the data vectors, Blanz and Vetter use PCA to represent the Gaussian PDF. Note that there is a close connection between PCA and Gaussian distributions, e.g. see [48].

learning too "vague" a model — such a penalty is built-in to Bayesian learning methods in general [32]. We did not have to manually specify this penalty term — it is a consequence of the fact that the likelihood function is required to be a normalized PDF.

Once we have estimates of μ and ϕ, we have a description of how "likely" any given face model is. For example, suppose we wish to estimate a face shape from a given image of someone's face. We assume that the face was created by selecting some viewing and lighting parameters $\mathbf{V}$, rendering an image $\mathbf{I}$ of the face $\mathbf{x}$, and adding zero-mean Gaussian noise with variance σ^2. In other words,

$$p(\mathbf{I}|\mathbf{x},\mathbf{V},\sigma^2) = \prod_{(x,y)} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}(\mathbf{I}(x,y)-\mathbf{I}_{rendered}(x,y,\mathbf{x}))^2}$$

where $\mathbf{I}_{rendered}(x,y,\mathbf{x},\mathbf{V})$ represents a rendering of the face $\mathbf{x}$ at pixel (x,y) with pose and lighting parameters $\mathbf{V}$.

To solve for the head shape and pose, we wish to estimate unknowns $\mathbf{V}$ and $\mathbf{x}$ by maximizing $p(\mathbf{x},\mathbf{V}|I,\mu,\phi)$. Assuming uniform priors on $\mathbf{V}$, and assuming that $\mathbf{x}$ and $\mathbf{V}$ are independent, we have

$$p(\mathbf{x},\mathbf{V}|\mathbf{I},\hat{\mu},\hat{\phi},\sigma^2) = \frac{p(\mathbf{I}|\mathbf{x},\mathbf{V},\sigma^2)p(\mathbf{x}|\hat{\mu},\hat{\phi})p(\mathbf{V})}{p(\mathbf{I})} \quad (19)$$

Again, maximizing this is equivalent to minimizing the negative log-likelihood :

$$L(\mathbf{x},\mathbf{V}) = -\ln p(\mathbf{x},\mathbf{V}|\mathbf{I},\hat{\mu},\hat{\phi},\sigma^2) \quad (20)$$

$$= (\mathbf{x}-\hat{\mu})^T\hat{\phi}^{-1}(\mathbf{x}-\hat{\mu})/2 + \frac{1}{2\sigma^2}\sum_{(x,y)}(\mathbf{I}(x,y)-\mathbf{I}_{rendered}(x,y,\mathbf{x},\mathbf{V}))^2 \quad (21)$$

(Terms that are constant with respect to $\mathbf{x}$ and $\mathbf{V}$ have been dropped.) Observe that this energy function over head shapes includes two terms: a facial "prior" that measures how "face-like" our 3D reconstruction is, and an image-fitting term. Although we could have arrived at this energy function without thinking in probabilistic terms, the advantage of the Bayesian approach is that we learned the μ and ϕ parameters of the energy function from data — we did not have to tune those parameters. Optimizing this objective form is more difficult than the previous one, and requires an iterative solver [4, 46].

It would also be straightforward to learn the σ^2 parameter from one or more images by optimizing $p(\mathbf{x},\mathbf{V},\sigma^2|\mathbf{I})$. Writing out the terms of the negative log-likelihood in this case, we get an optimization similar to the above fitting, but, with σ^2 as a free parameter, and a penalty term proportional to $\ln\sigma^2$ that penalizes large image noise estimates. If we do this, there are now *no parameters to tune in the energy function; all parameters are learned from the input data*. This is one of the many benefits of the Bayesian approach — without this approach, you might come up with an optimization similar to the above, but you would have the tedious and difficult task of manually tuning the parameters μ, ϕ and σ^2. (Again, the caveats regarding the need for adequate initialization in the training, and the need for adequate training data apply.)

Not only do they require less user effort, automatically-learned parameters often perform better in vision tasks than hand-tuned parameters, since setting such parameters by hand can be very difficult. The quadratic error function $L(\mathbf{x},\mathbf{V})$ for faces may have thousands of parameters, which would be impractical to set by hand. For example, in a recent project on non-rigid modeling from video, we used maximum likelihood to estimate 3D shapes, non-rigid motion, image noise, outlier likelihoods, and visibility from an image sequence [56]. We found that learning these parameters always gave better results than the manually-specified initial values.

Note that the probabilistic model is more expressive than an energy function, since it can be used to randomly generate data as well. For example, we can randomly sample heads by sampling from $p(\mathbf{x}|\hat{\mu},\hat{\phi})$. Given some user-specified constaints on a shape model, we can sample heads that satisfy that constraint.

5.4. Marginalization

One subtle but interesting technical point is that the MAP and ML learning principles are heuristics that discard knowledge. An "ideal" Bayesian learner would never make these approximations, but would always keep around all uncertainty. For example, consider the facial modeling example in Section 5.3. Given the face shape data $\mathbf{X}$, we have a distribution $p(\mu,\phi|\mathbf{X})$ over the possible parameters μ and ϕ. We still don't know for sure what these values are, and to pick specific ML estimates of them is to discard the uncertainty that we have in these values. Instead, the ideal option would be to keep around this uncertainty, and use it when making future decisions. For example, when estimating a new face shape from an image, the posterior distribution is

$$p(\mathbf{x},\mathbf{V}|\mathbf{I},\mathbf{X},\sigma^2) = \int p(\mathbf{x},\mathbf{V},\mu,\phi|\mathbf{I},\sigma^2)d\mu d\phi \quad (22)$$

$$= \int p(\mathbf{x},\mathbf{V}|\mu,\phi,\mathbf{I},\sigma^2)p(\mu,\phi|\mathbf{X})d\mu d\phi$$

In other words, we marginalize out the model parameters[5]. We could then use this posterior distribution for any further tasks regarding this new face. If we need to pick a single

[5]One subtlety is that it is necessary to define the priors $p(\mu,\phi)$ in such a way that is it normalizable, in order for the integral to be finite [32]. Normally, we would use a prior that is uniform over an extremely large extent. This is detail that usually be ignored when we are not marginalizing.

estimate of the face (for example, for rendering from a new view), we could use ML estimates of $\mathbf{x}$ and $\mathbf{V}$.

In most cases, people use MAP or ML estimates because the full integration problem above is often quite complicated, difficult, or slow. Quite often, the MAP/ML estimates are good enough. The full Bayesian solution can often give better results, however, since it uses more of the information present in the training data.

One informative case to consider is learning a Gaussian distribution from only a single data point [37]. In this case, the posterior distribution $p(\phi, \mu|\mathbf{x})$ is uniform with respect to the variance, which makes sense — a single data point gives no information about the variance of the Gaussian. Computing the ML estimate in this case makes no sense, since it makes no sense to attempt to compute the "best" estimate from a uniform distribution. However, the posterior in Equation 22 is still meaningful in this case; marginalization handles this degenerate case without requiring special treatment. Of course, this model is most useful when we have multiple training face shapes.

In general, we can integrate out some unknown parameters when estimating others [32]. Integrating out parameters allows us to get more reliable estimates of the other parameters that we are most interested in. In many cases, these integrals cannot be optimized in closed-form, and variational optimization — usually, the Expectation-Maximization (EM) algorithm — is used for optimization [10, 15, 38].

At this point, it is worth noting that some authors in learning and vision use the word "Bayesian" in their paper titles and algorithms in different ways. If you see a paper title "Bayesian X," it may mean that some parameters are being marginalized out (which is considered better than solving for them), but it could also just mean that the paper is introducing a probabilistic formulation to the problem.

5.5. Generative models and graphical models

The approach of learning a model by maximizing the probability of the model given the data is not limited to Gaussians. As discussed in more detail, a general strategy for probabilistic reasoning is to write a probabilistic model that describes how the data is generated, and then solve for the parameters of this model from some data by maximum likelihood.

In this section, I'll briefly describe two ways of viewing probabilistic modeling that may sound trivial at first, but are extremely useful in thinking about and describing such problems.

Up to now, most graphics research that uses learning methods at all uses them as a black box, e.g. a classification subroutine, a clustering subroutine, a dimension reduction subroutine, etc. This makes the most sense when a problem may be broken into subtasks that can be solved separately. However, this is not possible in some cases, for example, when a problem involves many interdependent unknown variables that cannot easily be separated. In this case, it is natural to formulate a single energy function or probability model that can be used to optimize all parameters simultaneously.

When formulating such problems, it is useful to view them in the framework of **generative models**. In a generative model, one describes the variables in a problem as being generated by a random sampling process. For example, in the facial modeling example in Section 5.3, suppose we wanted to generate a random facial photograph according to a learned model: first, the parameters of a person's face are generated by randomly sampling from "face space" (i.e. the Gaussian distribution over face shape and texture); next, pose and lighting parameters are uniformly sampled from the space of possible poses and lightings; the image is finally generated by projecting and lighting the face, and adding random image noise. In the generative view of this problem, we imagine that the original data was created by exactly this process, i.e. the data arose by random sampling from $p(\mathsf{data}|\mathsf{model})$. When we wish to describe a model for a new problem, we write down all of the variables, and the probability models that describe how they are generated/sampled, and which variables are generated as a function of which others. Although this may not seem like it has changed the problem much, this formalism provides a useful way of thinking and talking about probability models. It is sometimes tempting to avoid a generative formulation, since one often doesn't *really* believe that the generative model accurately describes how the data is generated — it is a simplified model of the world. However, without a generative model, there can be substantial confusion between the elements of the model, the problem statement, and the algorithm actually used. Stating a generative model clarifies thinking about the problem and helps avoid some of the confusion. In my experience, once you start thinking in terms of generative models, it is difficult to stop.

An important special class of generative models are **graphical models** [15, 26, 32, 41]. In graphical models, one represents the variables in a model visually with a graph. Graphical models also go under a few other names, including **Bayesian Networks**, and **Belief Networks**, and, confusingly, are occasionally referred to as neural networks.

The graphical model formalism may be viewed as the marriage of probability theory with graph theory. Although it does not change the fundamental structure of a problem, it is extremely useful for visualizing the structure, and for communicating it. It provides a sort of flowchart for the variables in the generative model. Additionally, many algorithms are easiest to describe and understand in terms of graph operations [26, 41, 62].

6. Caveats

A few words of caution:

- You can't learn something from nothing. Your algorithm only "knows" what (a) you tell it explicitly, and (b) its models can deduce from the data. If you learn a Gaussian PDF over XYZ marker data, you will not get a very useful model of human motion. You will get a much better model working from joint angles.
- As tempting as it is to use a learning algorithm as a "black box," the more you understand about the inner workings of your model, the better. If you understand your formalisms well, you can predict when it will and won't work, and can debug it much more effectively. Of course, if you find that some method works as a "black box" for a given problem, then it is still useful.
- On a related note, it always pays to understand your assumptions and to make them explicit. For example, human faces are not *really* generated by random sampling from a Gaussian distribution — this model is an approximation to the real process, and should be understood as such. (Making your assumptions explicit and discussing the advantages and disadvantages is also very important in communicating your results to other people.)
- There are times when trying to define formal problem statements and to properly optimize objective functions or posterior likelihoods can impede creativity. The graphics community has a long history of being driven by clever, powerful hacks. Sometimes, clever hacks end up being more useful than their formal counterparts. On the other hand, clever hacks can sometimes lead to deeper understanding of a problem, and more formal and principled generalizations that would not be possible with hacks alone.

7. Research problems

The future is bright for applying learning to graphics problems, both in research and applications to industry and art. In the future, I expect that we will see more examples of directly modeling in a Bayesian setting in graphics. I expect that some of the major themes of research will be:

- Designing good models and algorithms for various concepts used in graphics
- Novel synthesis and sampling algorithms for learned models
- Discovering which models work will for which problems
- Integrating learned models and learning algorithms with user interfaces
- Providing artistic control in a system that uses learning in some components
- Interactive and real-time learning and synthesis

Acknowledgements

I am grateful to James Davis, Rich Radke, Adrian Secord, and Steve Seitz for many useful discussions and for commenting on drafts of this paper. I am grateful to everyone that I have collaborated with on projects related to learning and graphics.

References

[1] Ronen Barzel, John F. Hughes, and Daniel Wood. Plausible motion simulation for computer animation. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation*, 1996.

[2] Anthony J. Bell and Terry J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 187–194, August 1999.

[5] Matthew Brand and Aaron Hertzmann. Style machines. *Proceedings of SIGGRAPH 2000*, pages 183–192, July 2000.

[6] Christopher J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[7] Stephen Chenney and D. A. Forsyth. Sampling plausible solutions to multi-body constraint problems. In *Proceedings of ACM SIGGRAPH 2000*, pages 219–228, July 2000.

[8] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE. Trans. Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[9] Richard T. Cox. Probability, frequency, and reasonable expectation. *American J. of Physics*, 14(1):1–13, 1946.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society series B*, 39:1–38, 1977.

[11] Pedro Domingos. The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.

[12] Charles Elkan. The Paradoxical Success of Fuzzy Logic. *IEEE Expert*, pages 3–8, August 1994.

[13] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable Controllers for Physics-Based Character Animation. In *Proceedings of SIGGRAPH 2001*, August 2001.

[14] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[15] Brendan J. Frey and Nebojsa Jojic. Advances in Algorithms for Inference and Learning in Complex Probability Models, 2003. http://www.research.microsoft.com/∼jojic/tut.pdf.

[16] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[17] Zoubin Ghahramani and Geoff E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto, 1996.

[18] Zoubin Ghahramani and Geoff E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, 1998.

[19] Zoubin Ghahramani and Sam Roweis. Learning nonlinear dynamical systems using an EM algorithm. In *Proc. NIPS 11*, pages 431–437. MIT Press, 1998.

[20] Peter Grünwald. *The Minimum Description Length Principle and Reasoning under Uncertainty*. PhD thesis, University of Amsterdam, 1998. ILLC Dissertation series 1998-03.

[21] Michael Isard and Andrew Blake. CONDENSATION – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

[22] Tommi Jaakkola, Marina Meilă, and Tony Jebara. Maximum entropy discrimination. *Proc. NIPS 12*, 1991.

[23] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003. In press; http://omega.math.albany.edu:8008/JaynesBook.html.

[24] Edwin T. Jaynes. On the Rationale of Maximum-Entropy Methods. *Proc. IEEE*, 70(9):939–952, 1982.

[25] Nebojsa Jojic and Brendan Frey. Learning Flexible Sprites in Video Layers. In *Proc. CVPR 2001*, 2001.

[26] Michael I. Jordan, editor. *Learning in Graphical Models*. MIT Press, 1998.

[27] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *J. of Artificial Intelligence Research*, 4:237–285, 1996.

[28] M. Kirby and L. Sirovich. Application of the Karhunen-Loève Procedure for the Characterization of Human Faces. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 12(1):103–108, January 1990.

[29] Pierre-Simon Laplace. *A Philosophical Essay on Probabilities*. Dover Publications, 1814.

[30] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion Texture: A Two-Level Statistical Model for Character Motion Synthesis. *ACM Transactions on Graphics*, 21(3):465–472, July 2002.

[31] D. J. C. MacKay. Probable Networks and Plausible Predictions — A Review of Practical Bayesian Methods for Supervised Neural Networks. *Network: Computation in Neural Systems*, 6:469–505, 1995.

[32] David MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. In press; http://www.inference.phy.cam.ac.uk/mackay/itila.

[33] David J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–166. Kluwer Academic Press, 1998.

[34] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A Data-Driven Reflectance Model. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3), July 2003.

[35] Marina Meilă. The Multicut Lemma. Technical Report TR417, University of Washington Statistics Dept, 2002.

[36] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel PCA and de-noising in feature spaces. In *Proc. NIPS*. MIT Press, 1999.

[37] Thomas Minka. Pathologies of Orthodox Statistics, 2001. Unpublished note. http://www.stat.cmu.edu/∼minka/papers/pathologies.html.

[38] Radford M. Neal and Geoff E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[39] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Proc. NIPS 14*. MIT Press, 2002.

[40] Vladimir Pavlović, James M. Rehg, and John MacCormick. Learning Switching Linear Models of Human Motion. In *Proc. NIPS 13*. MIT Press, 2001.

[41] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 2nd edition, 1997.

[42] Ken Perlin. An Image Synthesizer. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):287–296, July 1985.

[43] Ken Perlin and Athomas Goldberg. IMPROV: A System for Scripting Interactive Actors in Virtual Worlds. In *Proceedings of SIGGRAPH 96*, pages 205–216, August 1996.

[44] L. R. Rabiner. A tutorial on hidden Markov models and its application to speech recognition. *IEEE*, 17:257–286, February 1989.

[45] Rajesh P. N. Rao, Bruno A. Olshausen, and Michael S. Lewicki, editors. *Probabilistic Models of the Brain: Perception and Neural Function*. MIT Press, 2002.

[46] Sami Romdhani, Volker Blanz, and Thomas Vetter. Face Identification by Fitting a 3D Morphable Model using Linear Shape and Texture Error Functions. In *Proc. ECCV 2002*, pages 3–19, May 2002.

[47] Rómer Rosales and Brendan Frey. Learning Generative Models of Affinity Matrices. In *Proc. UAI 2003*, 2003. To appear.

[48] Sam T. Roweis. EM algorithms for PCA and SPCA. In *Proc. NIPS 10*, pages 626–632, 1998.

[49] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, December 2000.

[50] Robert E. Schapire. The boosting approach to machine learning: An overview. In *Proc. MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[51] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video Textures. *Proceedings of SIGGRAPH 2000*, pages 489–498, July 2000.

[52] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[53] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[54] Yee Whye Teh and Sam T. Roweis. Automatic Alignment of Hidden Representations. In *Proc. NIPS 15*. MIT Press, 2003.

[55] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 5500(290):2319–2323, December 2000.

[56] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Robust Model-Free Tracking of Non-Rigid Shape. Technical Report TR2003-840, New York University, June 2003.

[57] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. of Cognitive Neuroscience*, 3(1):71–86, 1991.

[58] Paul Viola and Michael Jones. Robust object detection using a boosted cascade of simple features. In *Proc. CVPR 2001*, 2001.

[59] Yair Weiss. Segmentation using eigenvectors: a unifying view. In *Proc. ICCV 99*, pages 975–982, 1999.

[60] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina, 1995.

[61] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian Processes for Regression. In *Proc. NIPS 8*. MIT Press, 1996.

[62] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding Belief Propagation and Its Generalizations. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufman, 2003.