Some Challenges for Machine Learning

Handling complexity: Machine learning applications usually involve many variables, often related in complex ways. How can we handle this complexity without getting into trouble?

Optimization and integration: Most machine learning methods either involve finding the "best" values for some parameters (an optimization problem), or averaging over many plausible values (an integration problem). How can we do this efficiently when there a great many parameters?

Visualization: Understanding what's happening is hard when there are many variables and parameters. 2D plots are easy, 3D not too bad, but 1000D?

All these challenges are greater when there are many variables or parameters — the so-called "curse of dimensionality". But more variables also provide more information — a blessing, not a curse.

Complex Problems

Machine learning applications often involve complex, unknown relationships:

> Many features may be available to help predict the response (but some may be useless or redundant).
>
> Relationships may be highly non-linear.
>
> We don't have a theoretical understanding of the problem, which might have helped limit the forms of possible relationships.

Example: Recognition of hand-written digits.

The dataset referred to in the text (p. 362) uses 16×16 images, giving 256 features, each the intensity of one pixel.

The relationships are very complex and non-linear. Knowing that a particular pixel is black tells you something about what the digit is, but much of the information comes only from looking at more than one pixel simultaneously.

People do very well on this task, but how do they do it? We have some understanding of this, but not enough to easily mimic it.

How Should We Handle Complexity?

Properly dealing with complexity is a crucial issue for machine learning.

Limiting complexity is one approach — use a model that is complex enough to represent the essential aspects of the problem, but that is not so complex that *overfitting* occurs.

Overfitting happens when we choose parameters of a model that fit the data we have very well, but do poorly on new data.

Reducing dimensionality is another possibility. Perhaps the complexity is only apparent — really things are much simpler if we can find out how to reduce the large number of variables to a small number.

Averaging over complexity is the Bayesian approach — use as complex a model as might be needed, but don't choose a *single* set of parameter values. Instead, average the predictions found using *all* the parameter values that fit the data reasonably well, and which are plausible for the problem.

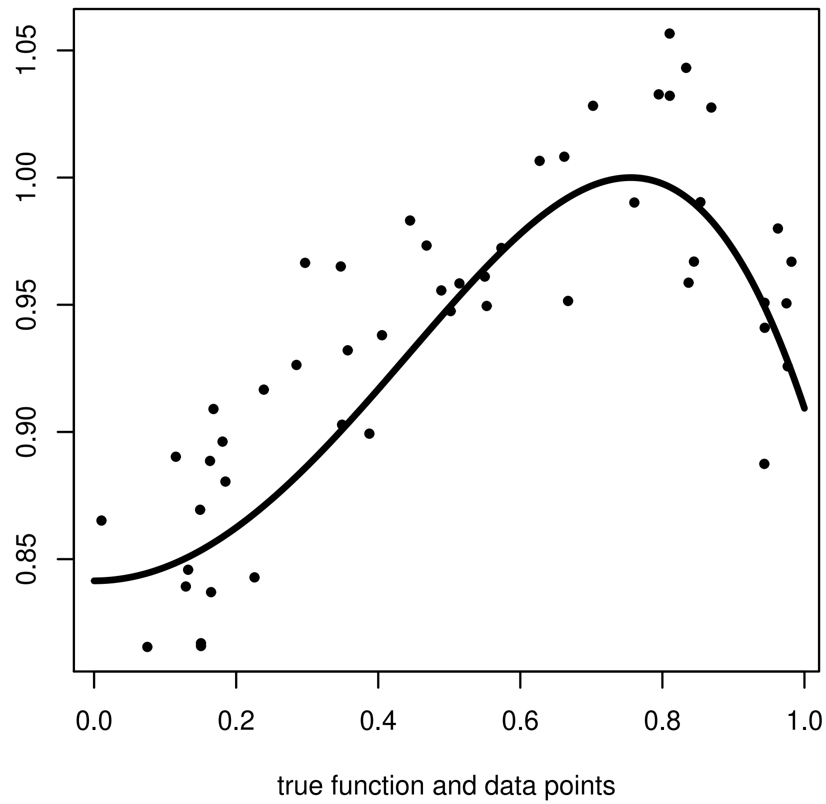
Note that my philosophy (primarily Bayesian) differs from the philosophy in the text by Hastie, Tibshirani, and Friedman.

Example Using a Synthetic Data Set

Here are 50 points generated with x uniform from $(0, 1)$ and y set by the formula:

$$y \quad = \quad \sin(1 + x^2) \ + \ \text{noise}$$

where the noise has $N(0, 0.03^2)$ distribution.

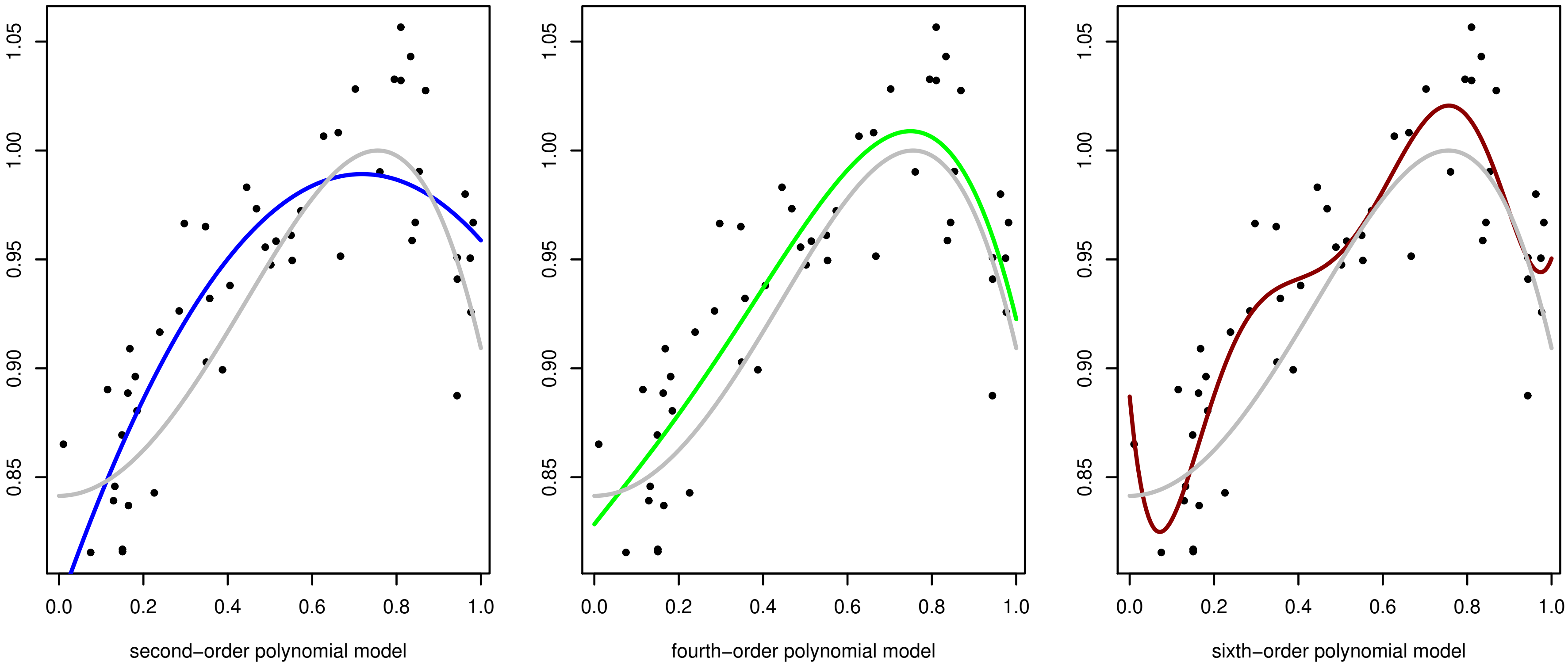


The noise-free function, $\sin(1 + x^2)$, is shown by the line.

Results of Fitting Polynomial Models of Various Orders

Here are the least-squares fits of polynomial models for y having the form (for $p = 2, 4, 6$) of

$$y \;=\; \beta_0 \,+\, \beta_1 x \,+\, \cdots \,+\, \beta_p x^p \,+\, \text{noise}$$



The gray line is the true noise-free function. We see that $p = 2$ is too simple, but $p = 6$ is too complex — if we choose values for β_i that best fit the 50 data points.

Do We Really Need to Limit Complexity?

If we make predictions using the "best fitting" parameters of a model, we have to limit the number of parameters to avoid overfitting.

For this example, with this amount of data, the model with $p = 4$ was about right.

But we know that $\sin(1 + x^2)$ is not a polynomial function — it has an infinite series representation with terms of arbitrarily high order.

How can it be good to use a model that we know is false?

The Bayesian answer: It's not good. We should abandon the idea of using the "best" parameters and instead average over all plausible values for the parameters. Then we can use a model (perhaps a very complex one) that is as close to being correct as we can manage.

Reducing Dimensionality for the Digit Data

For the zip-code digit recognition problem, we have 256 inputs, for each of 7291 training cases. Can we replace these with many fewer inputs, without great loss of information?

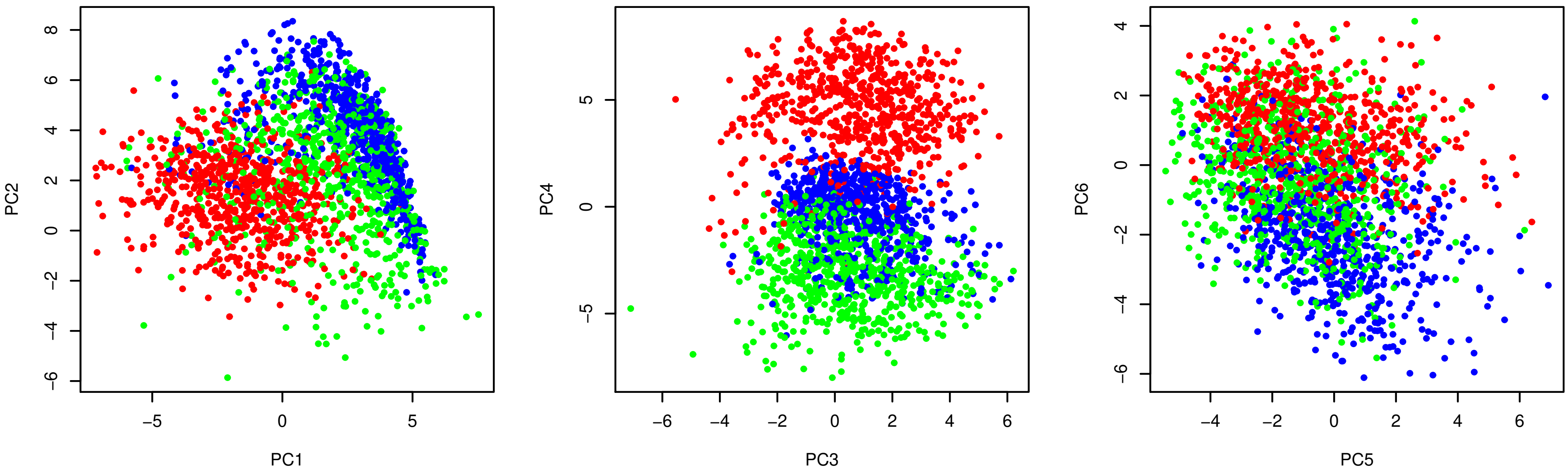
One simple way is by *Principal Components Analysis (PCA)*. We imagine the 7291 training cases as points in the 256 dimensional input space. We then find the direction of *highest variance* for these points, the direction of *second-highest variance* that is orthogonal to the first, etc. (See Figure 3.8 in the textbook.)

We stop before sometime before we find 256 directions — say, after only 20 directions. We the replace each training case by the projections of the inputs on these 20 directions.

In general, this might discard useful information — perhaps the identity of the digit is actually determined by the direction of *least* variance. But often it keeps most of the information we need, with many fewer variables.

Principal Components for the Zip-Code Digits

Here are plots of 1st versus 2nd, 3rd versus 4th, and 5th versus 6th principal components for training cases of digits "3" (red), "4" (green), and "9" (blue):



Clearly, these reduced variables contain a lot of information about the identity of the digit — probably much more than we'd get from any six of the original inputs.

Pictures of What the Principal Components Mean

Directions of principal components in input space are specified by 256-dimensional unit vectors. We can visualize them as 16×16 "images". Here are the first ten: