Typical Machine Learning and Data Mining Problems

Document search:

Given counts of words in a document, determine what its topic is.

Group documents by topic without a pre-specified list of topics.

Many words in a document, many, many documents available on the web.

Cancer diagnosis:

Given data on expression levels of genes, classify the type of a tumor.

Discover categories of tumors having different characteristics.

Expression levels of many genes measured, usually for only a few patients.

More Typical Machine Learning and Data Mining Problems

Marketing:

Given data on age, income, etc., predict how much each customer spends.
Discover how the spending behaviours of different customers are related.

Fair amount of data on each customer, but messy (eg, missing values).
May have data on a very large number of customers (millions).

Game playing:

Create a Backgammon-playing machine that learns by playing against other players (and maybe against itself).

Direct feedback comes only at the end of the game — long after crucial decisions were made.

So What *is* Machine Learning?

The artificial intelligence view: Humans (and animals) learn most of what they know from experience (not books or teachers). Nobody ever told you how to recognize a door knob. We'd like to get computers to do similar things — eg, speech recognition, computer vision, robot navigation.

The software engineering view: Writing programs is hard. Maybe we can get the computer to create programs from examples. This is not always an appropriate view — often writing the program is an essential step in clarifying what we want. Most machine learning applications are for things we wouldn't even think of programming explicitly.

The statistical view: Machine Learning is just statistics as done by computer scientists. But CS people have a different idea of a typical application — many variables, complex interactions — and may have a different perspective than statisticians.

Supervised Learning Problems

A *supervised learning* problem has these characteristics:

We are primarily interested in prediction.

We are interested in predicting only one thing.

The possible values of what we want to predict are specified, and we have some *training* cases for which its value is known.

The thing we want to predict is called the *target* or the *response variable*.

For a *classification* problem, we want to predict the class of an item — the topic of a document, the type of a tumor, whether a customer will purchase a product.

For a *regression* problem, we want to predict a numerical quantity — the amount a customer spends, the blood pressure of a patient, the melting point of an alloy.

To help us make our predictions, we have various *inputs* (also called *features* or *predictors*) — eg, gene expression levels for predicting tumor type, age and income for predicting amount spent. We use these inputs, but don't try to predict them.

In *semi-supervised* learning, targets are known for only some training cases. The "unlabelled" training cases may still help the learning.

Unsupervised Learning Problems

For an *unsupervised learning* problem, we do not focus on prediction of any particular thing, but rather try to find interesting aspects of the data.

One non-statistical formulation: We try to find *clusters* of similar items, or to *reduce the dimensionality* of the data.

Examples: We may find clusters of patients with similar symptoms, which we call "diseases". We may find that an overall "inflation rate" captures most of the information present in the price increases for many commodities.

One statistical formulation: We try to learn the *probability distribution* of all the quantities, often using *latent* (also called *hidden*) variables.

These formulations are related, since the latent variables may identify clusters or correspond to low-dimensional representations of the data.

Reinforcement Learning Problems

The most general formulation is *reinforcement learning*.

> We can sense certain things about the world, but not everything.
>
> We can perform various actions, which may affect the state of the world.
>
> We sometimes get rewards (reinforcement), which can be negative.
>
> We want to take the actions that lead to the most reward over time.

Supervised learning problems are a very special case, in which the actions are predictions for test cases, and the reward is related to how close the prediction is to being right.

More generally, there is a tradeoff between *exploration* — discovering new things about the world — and *exploitation* — using what we know to get rewards.

> Consider a robot navigating in an environment. It knows a slow way to get from A to B, where there is something it needs. Should it take that path, or look for a quicker way, which would be useful in the future?

Assigning credit is also a big problem — what action in the past (maybe the far past) was really responsible for getting a reward?

How Do "Machine Learning" and "Data Mining" Differ?

These terms are often used interchangeably, but...

Data mining is more often used for problems with very large amounts of data, where computational efficiency is more important than statistical sophistication — often business applications.

Machine learning is more often used for problems with a flavour of artificial intelligence — such as recognition of objects in visual scenes, or robot navigation.

The term "data mining" was previously used in a negative sense — to describe the misguided statistical procedure of looking for many, many relationships in the data until you finally find one, but one which is probably just due to chance.

One challenge of data mining is to avoid doing "data mining" in this sense!

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.