How to choose algorithms for Azure Machine Learning Studio

12/17/2017 15 minutes to read Contributors all

In this article

The Machine Learning Algorithm Cheat Sheet
Flavors of machine learning
Considerations when choosing an algorithm
Algorithm notes
Next Steps

The answer to the question "What machine learning algorithm should I use?" is always "It depends." It depends on the size, quality, and nature of the data. It depends on what you want to do with the answer. It depends on how the math of the algorithm was translated into instructions for the computer you are using. And it depends on how much time you have. Even the most experienced data scientists can't tell which algorithm will perform best before trying them.

The Machine Learning Algorithm Cheat Sheet

The **Microsoft Azure Machine Learning Algorithm Cheat Sheet** helps you choose the right machine learning algorithm for your predictive analytics solutions from the Azure Machine Learning Studio library of algorithms. This article walks you through how to use it.

> **Note**
>
> To download the cheat sheet and follow along with this article, go to Machine learning algorithm cheat sheet for Microsoft Azure Machine Learning Studio.

This cheat sheet has a very specific audience in mind: a beginning data scientist with undergraduate-level machine learning, trying to choose an algorithm to start with in Azure Machine Learning Studio. That means that it makes some generalizations and oversimplifications, but it points you in a safe direction. It also means that there are lots of algorithms not listed here. As Azure Machine Learning grows to encompass a more complete set of available methods, we'll add them.

These recommendations are compiled feedback and tips from many data scientists and machine learning experts. We didn't agree on everything, but I've tried to harmonize our opinions into a rough consensus. Most of the statements of disagreement begin with "It depends..."

How to use the cheat sheet

Read the path and algorithm labels on the chart as "For *<path label>*, use *<algorithm>*." For example, "For *speed*, use *two class logistic regression*." Sometimes more than one branch applies. Sometimes none of them are a perfect fit. They're intended to be rule-of-thumb recommendations, so don't worry about it being exact. Several data scientists I talked with said that the only sure way to find the very best algorithm is to try all of them.

Here's an example from the Azure AI Gallery of an experiment that tries several algorithms against the same data and compares the results: Compare Multi-class Classifiers: Letter recognition.

> **Tip**
>
> To download and print a diagram that gives an overview of the capabilities of Machine Learning Studio, see **Overview diagram of Azure Machine Learning Studio capabilities**.

Flavors of machine learning

Supervised

Supervised learning algorithms make predictions based on a set of examples. For instance, historical stock prices can be used to hazard guesses at future prices. Each example used for training is labeled with the value of interest—in this case the stock price. A supervised learning algorithm looks for patterns in those value labels. It can use any information that might be relevant—the day of the week, the season, the company's financial data, the type of industry, the presence of disruptive geopolitical events—and each algorithm looks for different types of patterns. After the algorithm has found the best pattern it can, it uses that pattern to make predictions for unlabeled testing data—tomorrow's prices.

Supervised learning is a popular and useful type of machine learning. With one exception, all the modules in Azure Machine Learning are supervised learning algorithms. There are several specific types of supervised learning that are represented within Azure Machine Learning: classification, regression, and anomaly detection.

- **Classification**. When the data are being used to predict a category, supervised learning is also called classification. This is the case when assigning an image as a picture of either a 'cat' or a 'dog'. When there are only two choices, it's called **two-class** or **binomial classification**. When there are more categories, as when predicting the winner of the NCAA March Madness tournament, this problem is known as **multi-class classification**.
- **Regression**. When a value is being predicted, as with stock prices, supervised learning is called regression.

- **Anomaly detection**. Sometimes the goal is to identify data points that are simply unusual. In fraud detection, for example, any highly unusual credit card spending patterns are suspect. The possible variations are so numerous and the training examples so few, that it's not feasible to learn what fraudulent activity looks like. The approach that anomaly detection takes is to simply learn what normal activity looks like (using a history non-fraudulent transactions) and identify anything that is significantly different.

Unsupervised

In unsupervised learning, data points have no labels associated with them. Instead, the goal of an unsupervised learning algorithm is to organize the data in some way or to describe its structure. This can mean grouping it into clusters or finding different ways of looking at complex data so that it appears simpler or more organized.

Reinforcement learning

In reinforcement learning, the algorithm gets to choose an action in response to each data point. The learning algorithm also receives a reward signal a short time later, indicating how good the decision was. Based on this, the algorithm modifies its strategy in order to achieve the highest reward. Currently there are no reinforcement learning algorithm modules in Azure Machine Learning. Reinforcement learning is common in robotics, where the set of sensor readings at one point in time is a data point, and the algorithm must choose the robot's next action. It is also a natural fit for Internet of Things applications.

Considerations when choosing an algorithm

Accuracy

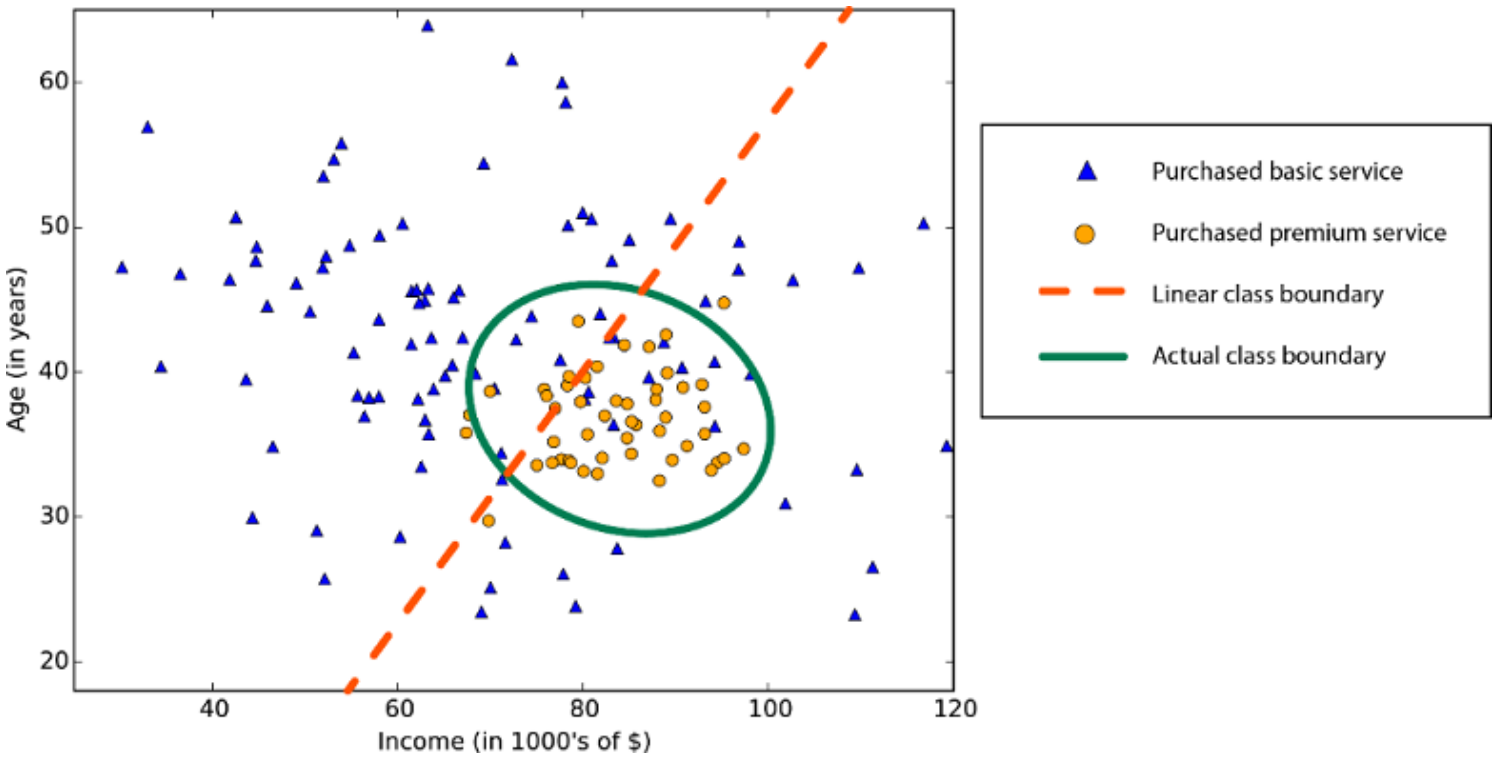
Getting the most accurate answer possible isn't always necessary. Sometimes an approximation is adequate, depending on what you want to use it for. If that's the case, you may be able to cut your processing time dramatically by sticking with more approximate methods. Another advantage of more approximate methods is that they naturally tend to avoid overfitting.

Training time

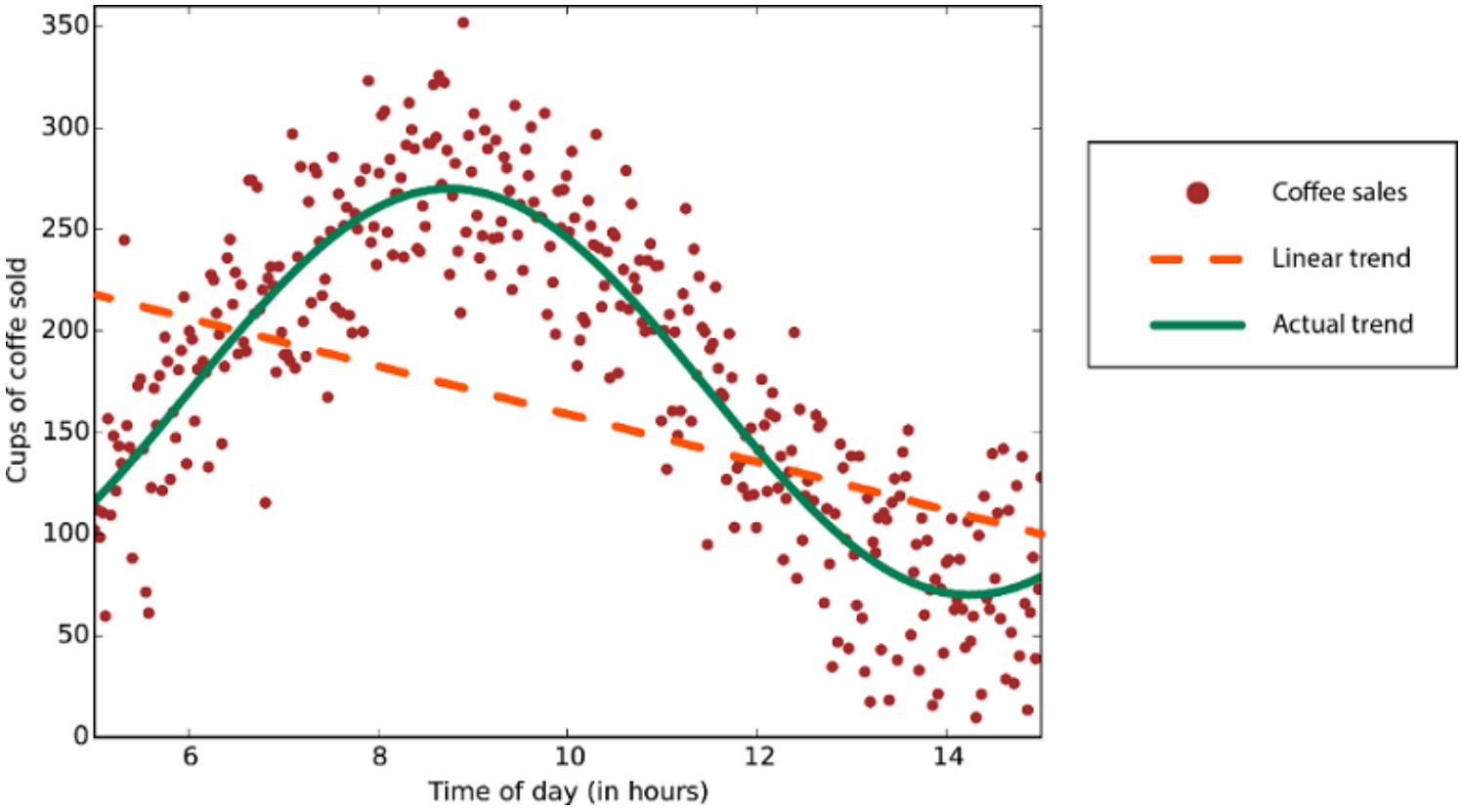
The number of minutes or hours necessary to train a model varies a great deal between algorithms. Training time is often closely tied to accuracy—one typically accompanies the other. In addition, some algorithms are more sensitive to the number of data points than others. When time is limited it can drive the choice of algorithm, especially when the data set is large.

Linearity

Lots of machine learning algorithms make use of linearity. Linear classification algorithms assume that classes can be separated by a straight line (or its higher-dimensional analog). These include logistic regression and support vector machines (as implemented in Azure Machine Learning). Linear regression algorithms assume that data trends follow a straight line. These assumptions aren't bad for some problems, but on others they bring accuracy down.



Non-linear class boundary *- relying on a linear classification algorithm would result in low accuracy*



Data with a nonlinear trend *- using a linear regression method would generate much larger errors than necessary*

Despite their dangers, linear algorithms are very popular as a first line of attack. They tend to be algorithmically simple and fast to train.

Number of parameters

Parameters are the knobs a data scientist gets to turn when setting up an algorithm. They are numbers that affect the algorithm's behavior, such as error tolerance or number of iterations, or options between variants of how the algorithm behaves. The training time and accuracy of the algorithm can sometimes be quite sensitive to getting just the right settings. Typically, algorithms with large numbers parameters require the most trial and error to find a good combination.

Alternatively, there is a parameter sweeping module block in Azure Machine Learning that automatically tries all parameter combinations at whatever granularity you choose. While this is a great way to make sure you've spanned the parameter space, the time required to train a model increases exponentially with the number of parameters.

The upside is that having many parameters typically indicates that an algorithm has greater flexibility. It can often achieve very good accuracy. Provided you can find the right combination of parameter settings.

Number of features

For certain types of data, the number of features can be very large compared to the number of data points. This is often the case with genetics or textual data. The large number of features can bog down some learning algorithms, making training time unfeasibly long. Support Vector Machines are particularly well suited to this case (see below).

Special cases

Some learning algorithms make particular assumptions about the structure of the data or the desired results. If you can find one that fits your needs, it can give you more useful results, more accurate predictions, or faster training times.

Algorithm	Accuracy	Training time	Linearity	Parameters	Notes
Two-class classification					
logistic regression		●	●	5	
decision forest	●	○		6	
decision jungle	●	○		6	Low memory footprint
boosted decision tree	●	○		6	Large memory footprint
neural network	●			9	Additional customization is possible
averaged perceptron	○	○	●	4	
support vector machine		○	●	5	Good for large feature sets
locally deep support vector machine	○			8	Good for large feature sets
Bayes' point machine		○	●	3	
Multi-class classification					
logistic regression		●	●	5	
decision forest	●	○		6	
decision jungle	●	○		6	Low memory footprint

Algorithm	Accuracy	Training time	Linearity	Parameters	Notes
neural network	●			9	Additional customization is possible
one-v-all	-	-	-	-	See properties of the two-class method selected
Regression					
linear		●	●	4	
Bayesian linear		○	●	2	
decision forest	●	○		6	
boosted decision tree	●	○		5	Large memory footprint
fast forest quantile	●	○		9	Distributions rather than point predictions
neural network	●			9	Additional customization is possible
Poisson			●	5	Technically log-linear. For predicting counts
ordinal				0	For predicting rank-ordering
Anomaly detection					
support vector machine	○	○		2	Especially good for large feature sets
PCA-based anomaly detection		○	●	3	
K-means		○	●	4	A clustering algorithm

Algorithm properties:

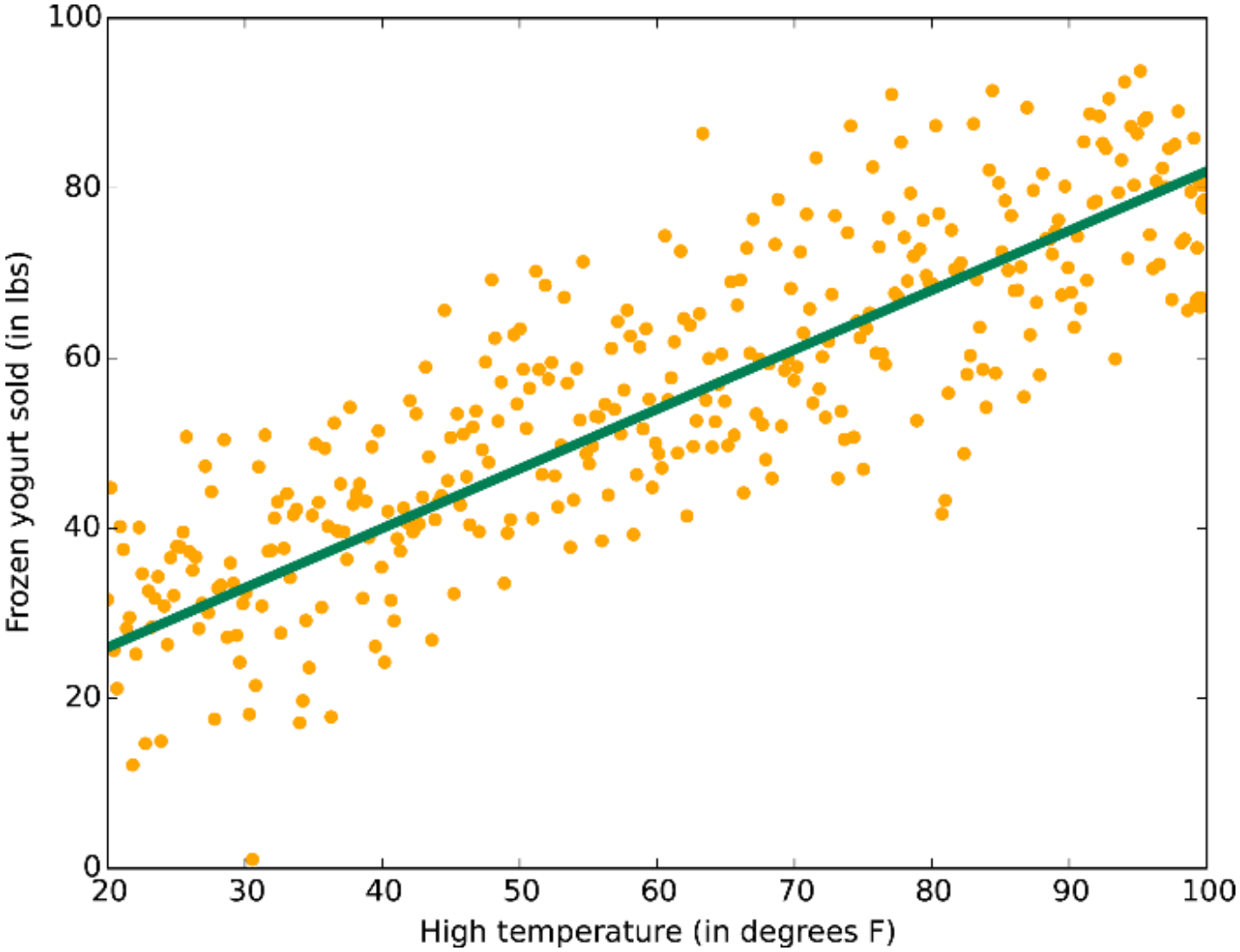
- ● - shows excellent accuracy, fast training times, and the use of linearity

○ - shows good accuracy and moderate training times

Algorithm notes

Linear regression

As mentioned previously, linear regression fits a line (or plane, or hyperplane) to the data set. It's a workhorse, simple and fast, but it may be overly simplistic for some problems. Check here for a linear regression tutorial.

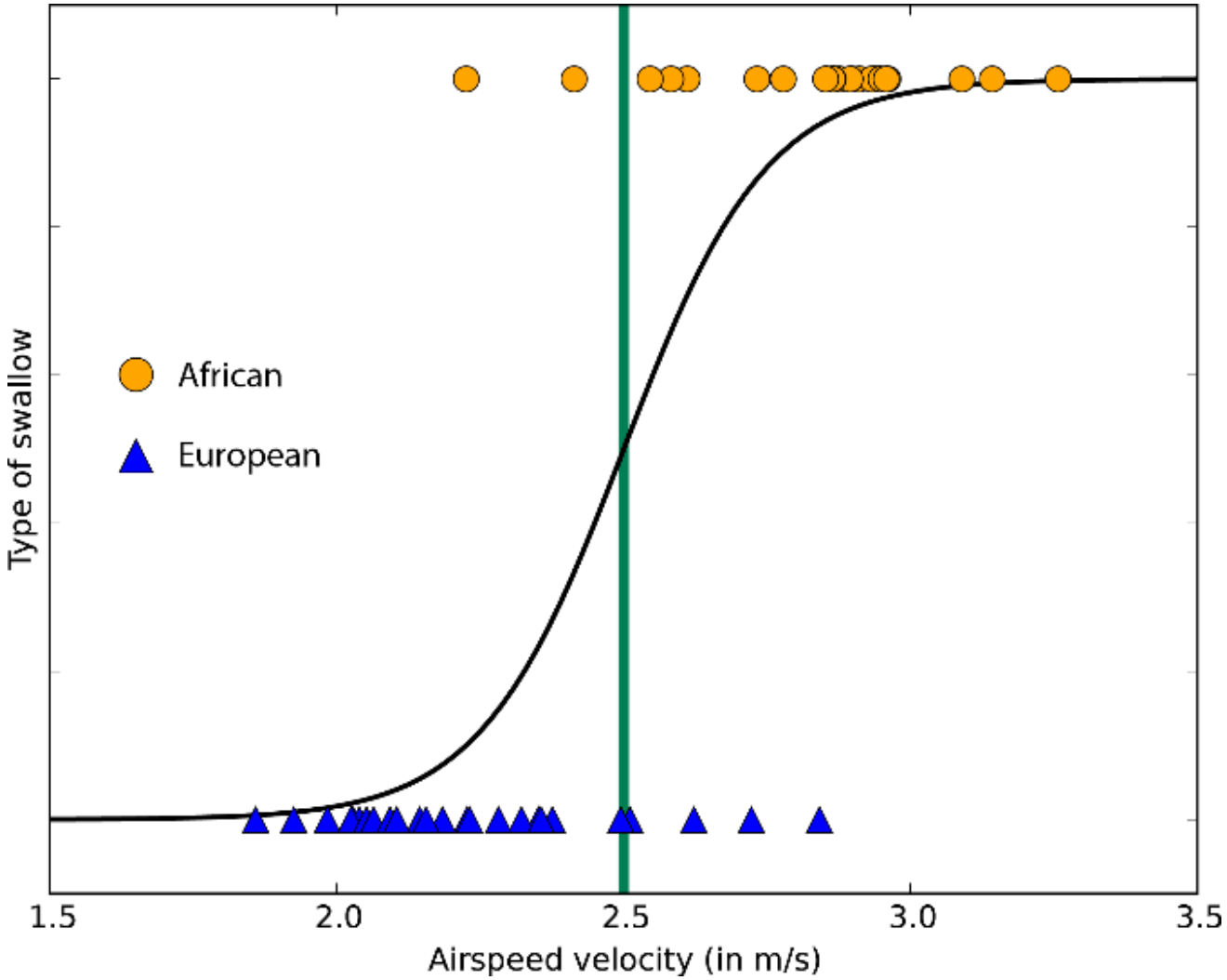


Data with a linear trend

Logistic regression

Although it confusingly includes 'regression' in the name, logistic regression is actually a powerful tool for two-class and multiclass classification. It's fast and simple. The fact that it uses an 'S'-shaped curve instead of

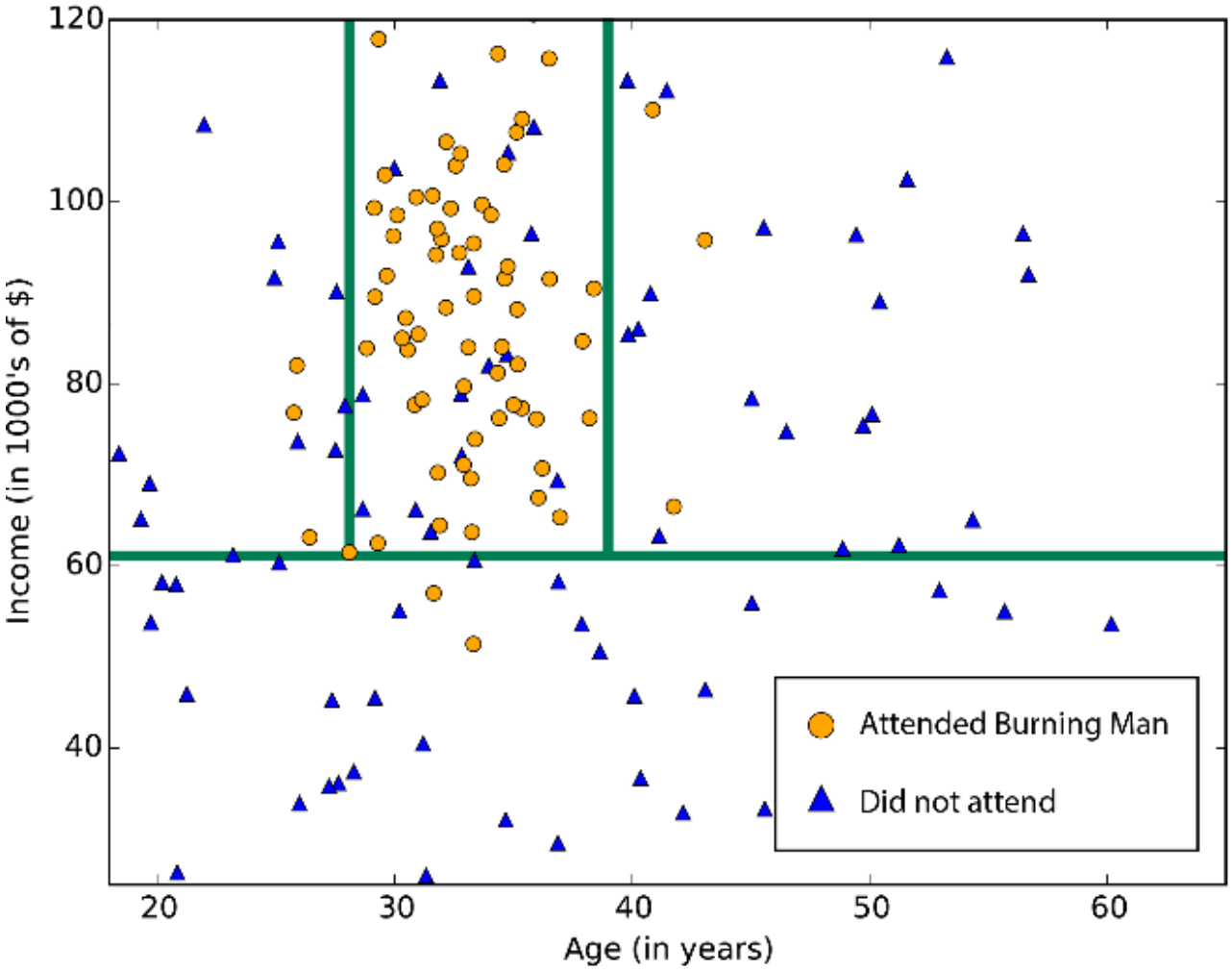
a straight line makes it a natural fit for dividing data into groups. Logistic regression gives linear class boundaries, so when you use it, make sure a linear approximation is something you can live with.

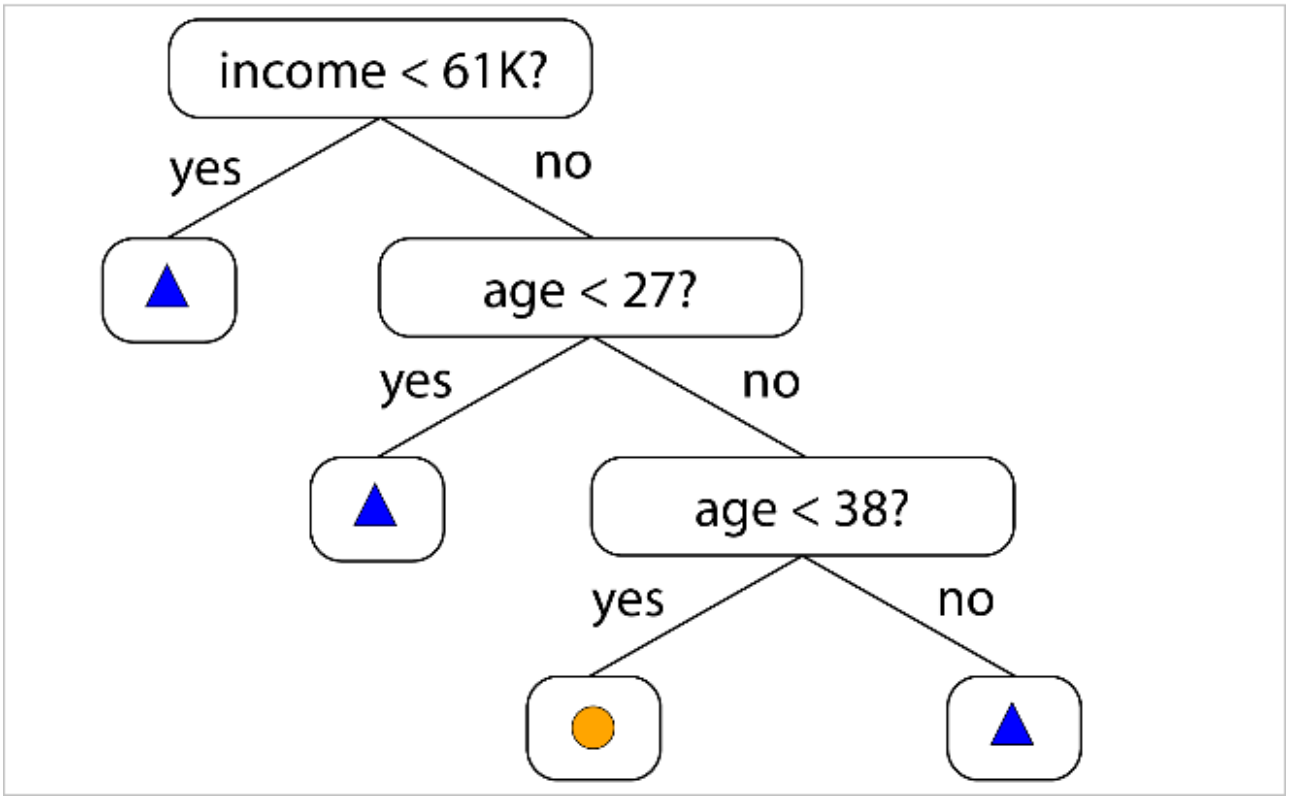


A logistic regression to two-class data with just one feature *- the class boundary is the point at which the logistic curve is just as close to both classes*

Trees, forests, and jungles

Decision forests (regression, two-class, and multiclass), decision jungles (two-class and multiclass), and boosted decision trees (regression and two-class) are all based on decision trees, a foundational machine learning concept. There are many variants of decision trees, but they all do the same thing—subdivide the feature space into regions with mostly the same label. These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.





A decision tree subdivides a feature space into regions of roughly uniform values

Because a feature space can be subdivided into arbitrarily small regions, it's easy to imagine dividing it finely enough to have one data point per region. This is an extreme example of overfitting. In order to avoid this, a large set of trees are constructed with special mathematical care taken that the trees are not correlated. The average of this "decision forest" is a tree that avoids overfitting. Decision forests can use a lot of memory. Decision jungles are a variant that consumes less memory at the expense of a slightly longer training time.

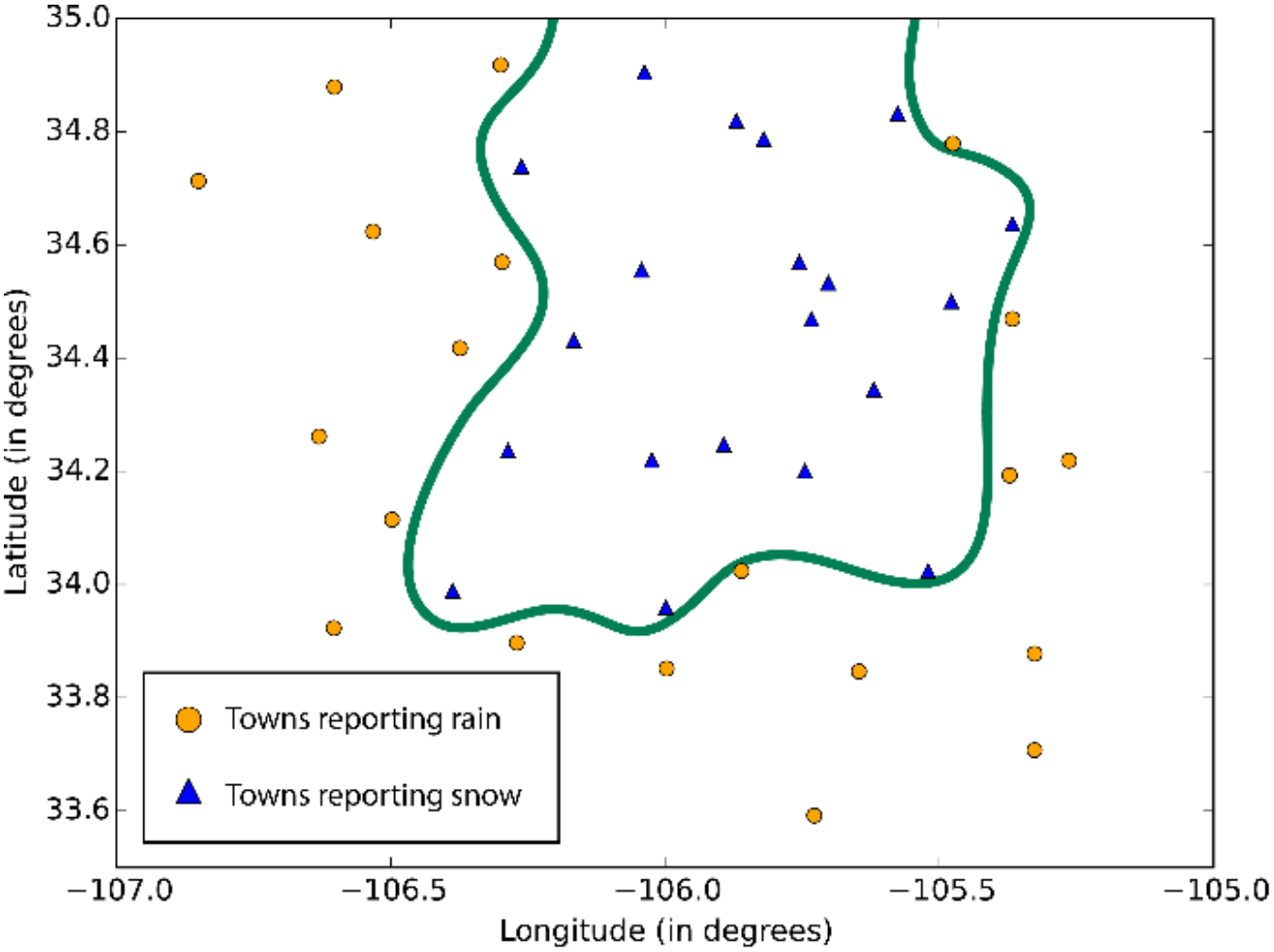
Boosted decision trees avoid overfitting by limiting how many times they can subdivide and how few data points are allowed in each region. The algorithm constructs a sequence of trees, each of which learns to compensate for the error left by the tree before. The result is a very accurate learner that tends to use a lot of memory. For the full technical description, check out Friedman's original paper.

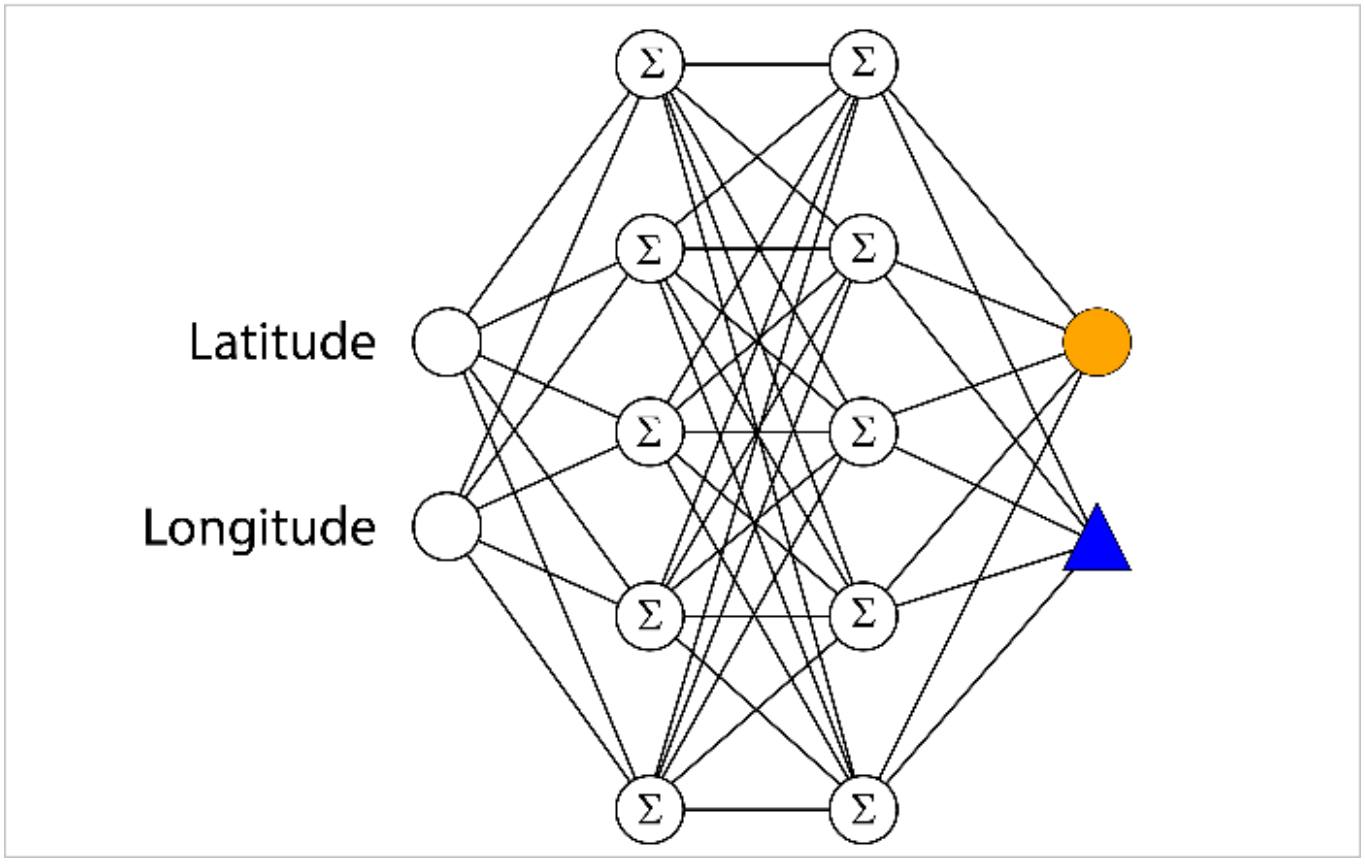
Fast forest quantile regression is a variation of decision trees for the special case where you want to know not only the typical (median) value of the data within a region, but also its distribution in the form of quantiles.

Neural networks and perceptrons

Neural networks are brain-inspired learning algorithms covering multiclass, two-class, and regression problems. They come in an infinite variety, but the neural networks within Azure Machine Learning are all of the form of directed acyclic graphs. That means that input features are passed forward (never backward) through a sequence of layers before being turned into outputs. In each layer, inputs are weighted in various combinations, summed, and passed on to the next layer. This combination of simple calculations results in the ability to learn sophisticated class boundaries and data trends, seemingly by magic. Many-layered networks of this sort perform the "deep learning" that fuels so much tech reporting and science fiction.

This high performance doesn't come for free, though. Neural networks can take a long time to train, particularly for large data sets with lots of features. They also have more parameters than most algorithms, which means that parameter sweeping expands the training time a great deal. And for those overachievers who wish to specify their own network structure, the possibilities are inexhaustible.



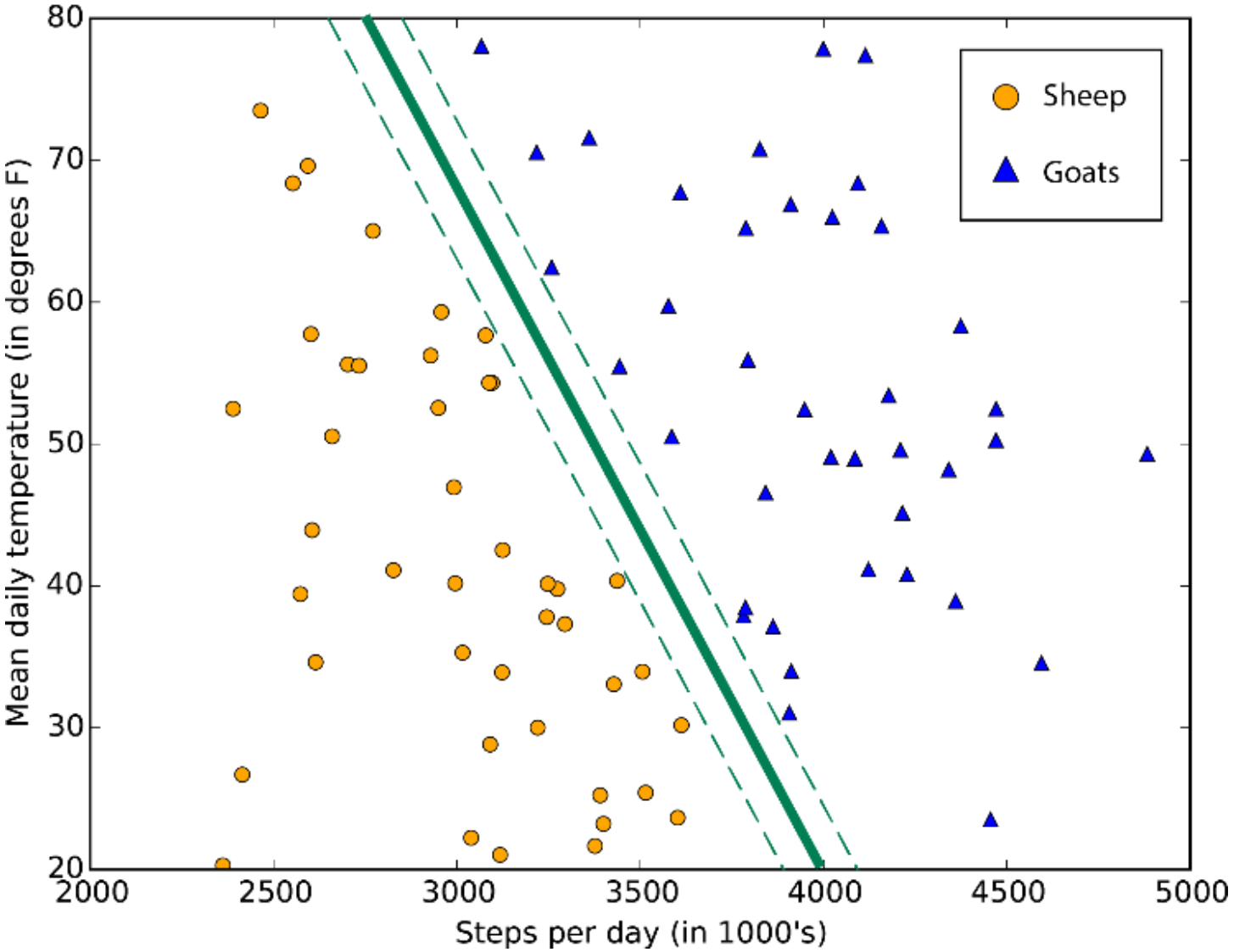


The boundaries learned by neural networks can be complex and irregular

The two-class averaged perceptron is neural networks' answer to skyrocketing training times. It uses a network structure that gives linear class boundaries. It is almost primitive by today's standards, but it has a long history of working robustly and is small enough to learn quickly.

SVMs

Support vector machines (SVMs) find the boundary that separates classes by as wide a margin as possible. When the two classes can't be clearly separated, the algorithms find the best boundary they can. As written in Azure Machine Learning, the two-class SVM does this with a straight line only. (In SVM-speak, it uses a linear kernel.) Because it makes this linear approximation, it is able to run fairly quickly. Where it really shines is with feature-intense data, like text or genomic. In these cases SVMs are able to separate classes more quickly and with less overfitting than most other algorithms, in addition to requiring only a modest amount of memory.



A typical support vector machine class boundary maximizes the margin separating two classes

Another product of Microsoft Research, the two-class locally deep SVM is a non-linear variant of SVM that retains most of the speed and memory efficiency of the linear version. It is ideal for cases where the linear approach doesn't give accurate enough answers. The developers kept it fast by breaking down the problem into a bunch of small linear SVM problems. Read the full description for the details on how they pulled off this trick.

Using a clever extension of nonlinear SVMs, the one-class SVM draws a boundary that tightly outlines the entire data set. It is useful for anomaly detection. Any new data points that fall far outside that boundary are unusual enough to be noteworthy.

Bayesian methods

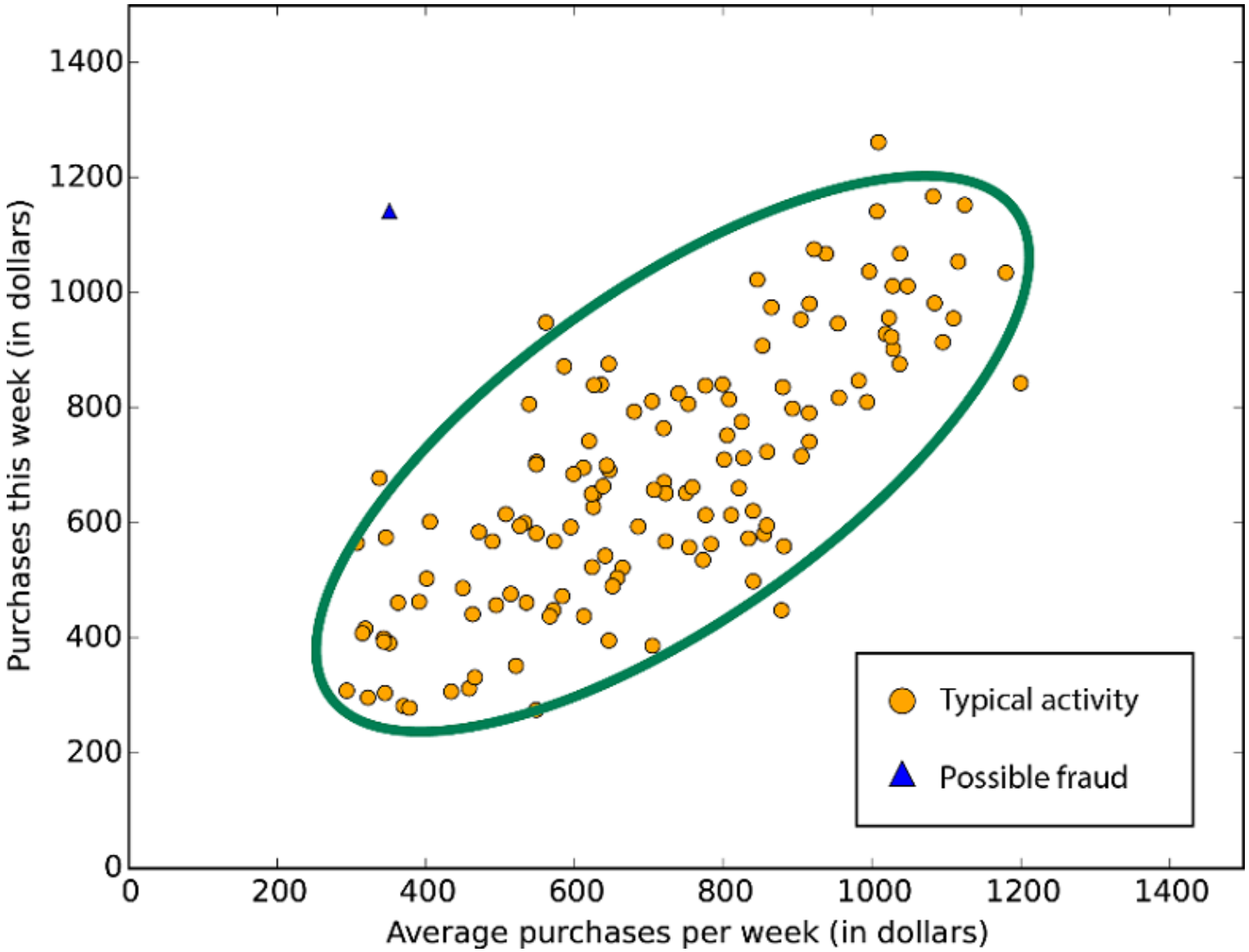
Bayesian methods have a highly desirable quality: they avoid overfitting. They do this by making some assumptions beforehand about the likely distribution of the answer. Another byproduct of this approach is that they have very few parameters. Azure Machine Learning has both Bayesian algorithms for both classification (Two-class Bayes' point machine) and regression (Bayesian linear regression). Note that these assume that the data can be split or fit with a straight line.

On a historical note, Bayes' point machines were developed at Microsoft Research. They have some exceptionally beautiful theoretical work behind them. The interested student is directed to the original article in JMLR and an insightful blog by Chris Bishop.

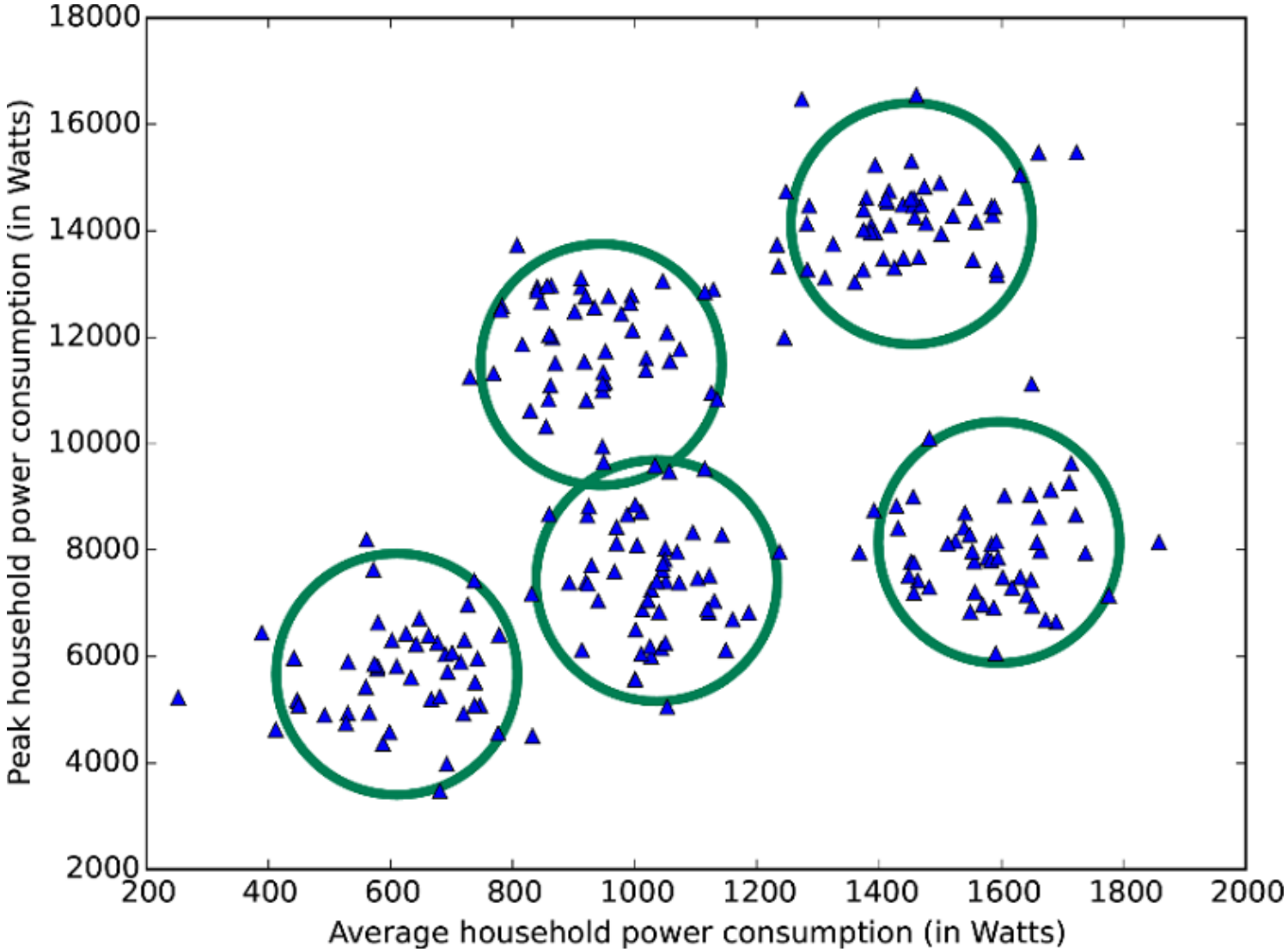
Specialized algorithms

If you have a very specific goal you may be in luck. Within the Azure Machine Learning collection, there are algorithms that specialize in:

- rank prediction (ordinal regression),
- count prediction (Poisson regression),
- anomaly detection (one based on principal components analysis and one based on support vector machines)
- clustering (K-means)

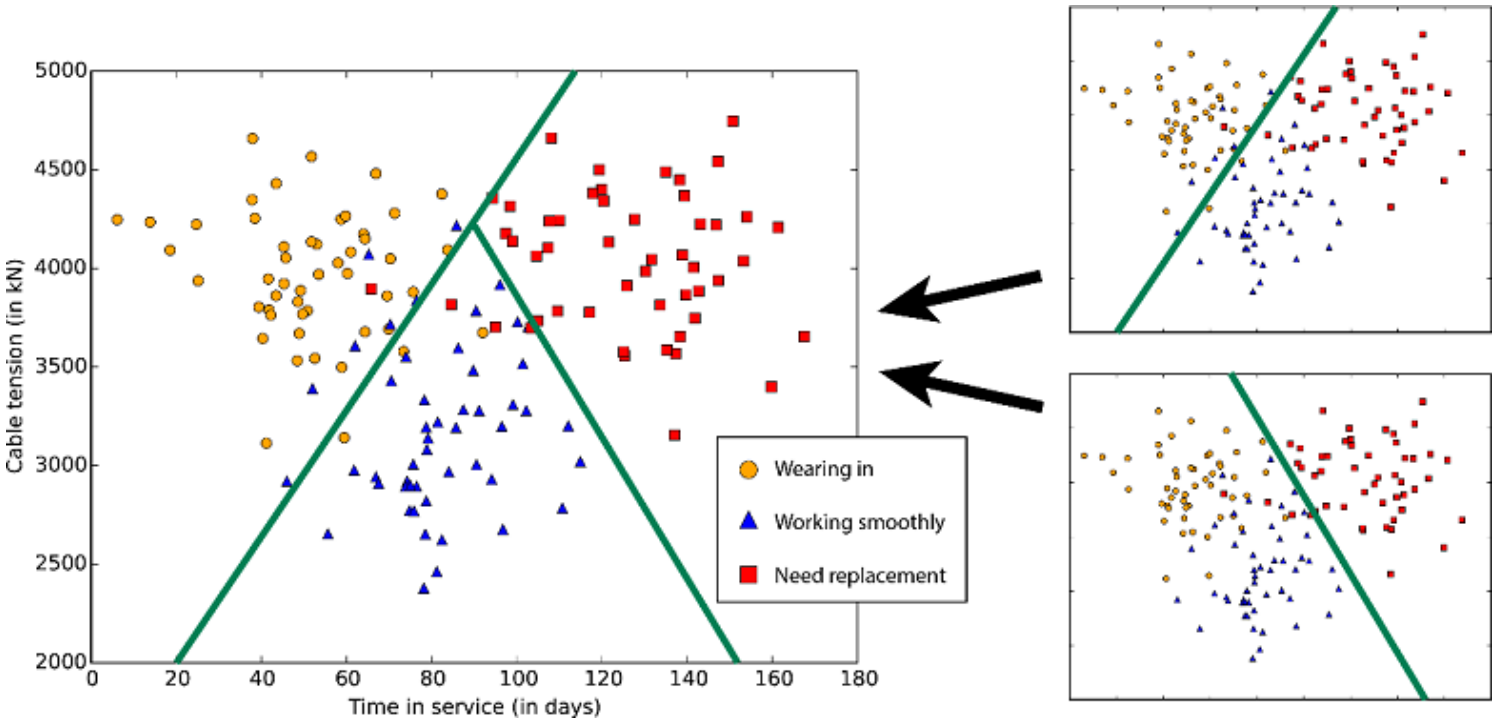


PCA-based anomaly detection *- the vast majority of the data falls into a stereotypical distribution; points deviating dramatically from that distribution are suspect*



A data set is grouped into five clusters using K-means

There is also an ensemble [one-v-all multiclass classifier](), which breaks the N-class classification problem into N-1 two-class classification problems. The accuracy, training time, and linearity properties are determined by the two-class classifiers used.



A pair of two-class classifiers combine to form a three-class classifier

Azure Machine Learning also includes access to a powerful machine learning framework under the title of Vowpal Wabbit. VW defies categorization here, since it can learn both classification and regression problems and can even learn from partially unlabeled data. You can configure it to use any one of a number of learning algorithms, loss functions, and optimization algorithms. It was designed from the ground up to be efficient, parallel, and extremely fast. It handles ridiculously large feature sets with little apparent effort. Started and led by Microsoft Research's own John Langford, VW is a Formula One entry in a field of stock car algorithms. Not every problem fits VW, but if yours does, it may be worth your while to climb the learning curve on its interface. It's also available as stand-alone open source code in several languages.

Next Steps

- For a downloadable infographic that describes algorithms and provides examples, see Downloadable Infographic: Machine learning basics with algorithm examples.
- For a list by category of all the machine learning algorithms available in Machine Learning Studio, see Initialize Model in the Machine Learning Studio Algorithm and Module Help.
- For a complete alphabetical list of algorithms and modules in Machine Learning Studio, see A-Z list of Machine Learning Studio modules in Machine Learning Studio Algorithm and Module Help.
- To download and print a diagram that gives an overview of the capabilities of Machine Learning Studio, see Overview diagram of Azure Machine Learning Studio capabilities.