A Tour of Machine Learning Algorithms

In this post, we take a tour of the most popular machine learning algorithms.

It is useful to tour the main algorithms in the field to get a feeling of what methods are available.

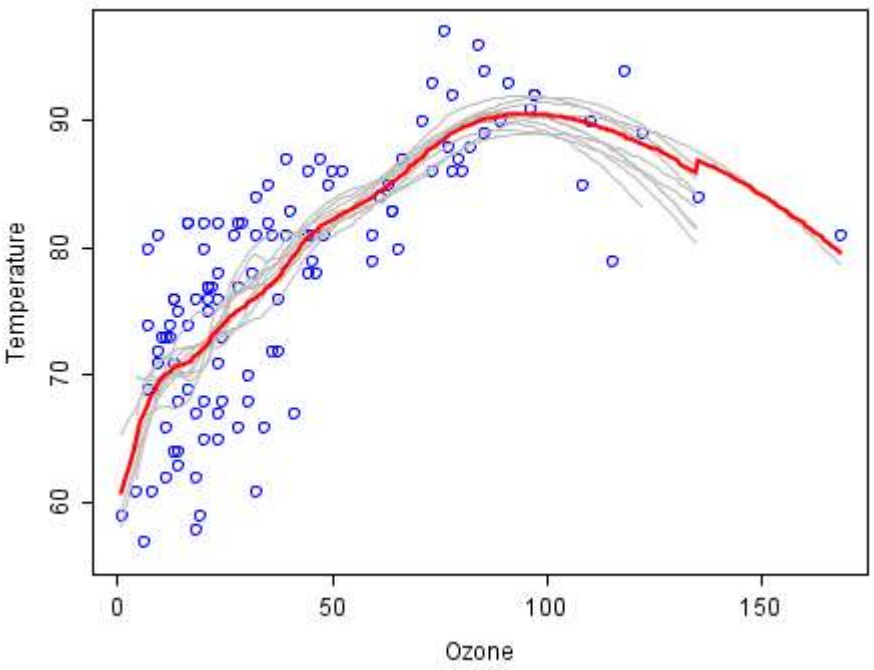
There are so many algorithms available that it can feel overwhelming when algorithm names are thrown around and you are expected to just know what they are and where they fit.

I want to give you two ways to think about and categorize the algorithms you may come across in the field.

- The first is a grouping of algorithms by the **learning style**.
- The second is a grouping of algorithms by **similarity** in form or function (like grouping similar animals together).

Both approaches are useful, but we will focus in on the grouping of algorithms by similarity and go on a tour of a variety of different algorithm types.

After reading this post, you will have a much better understanding of the most popular machine learning algorithms for supervised learning and how they are related.



A cool example of an ensemble of lines of best fit. Weak members are grey, the combined prediction is red.
Plot from Wikipedia, licensed under public domain.

Algorithms Grouped by Learning Style

There are different ways an algorithm can model a problem based on its interaction with the experience or environment or whatever we want to call the input data.

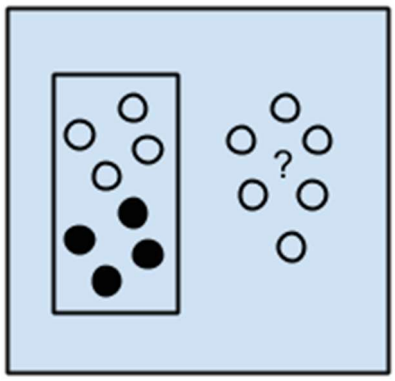
It is popular in machine learning and artificial intelligence textbooks to first consider the learning styles that an algorithm can adopt.

There are only a few main learning styles or learning models that an algorithm can have and we'll go through them here with a few examples of algorithms and problem types that they suit.

This taxonomy or way of organizing machine learning algorithms is useful because it forces you to think about the roles of the input data and the model preparation process and select one that is the most appropriate for your problem in order to get the best result.

Let's take a look at three different learning styles in machine learning algorithms:

1. Supervised Learning



Supervised Learning Algorithms

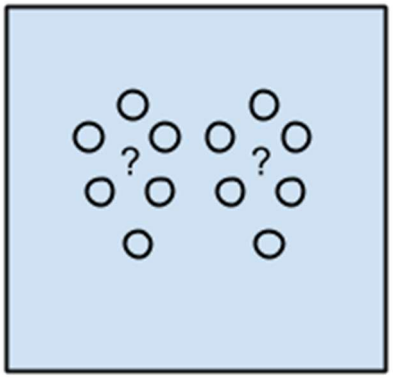
Input data is called training data and has a known label or result such as spam/not-spam or a stock price at a time.

A model is prepared through a training process in which it is required to make predictions and is corrected when those predictions are wrong. The training process continues until the model achieves a desired level of accuracy on the training data.

Example problems are classification and regression.

Example algorithms include Logistic Regression and the Back Propagation Neural Network.

2. Unsupervised Learning



Unsupervised Learning Algorithms

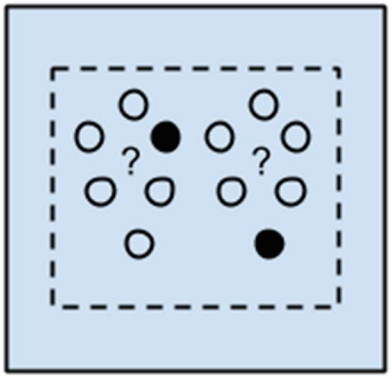
Input data is not labeled and does not have a known result.

A model is prepared by deducing structures present in the input data. This may be to extract general rules. It may be through a mathematical process to systematically reduce redundancy, or it may be to organize data by similarity.

Example problems are clustering, dimensionality reduction and association rule learning.

Example algorithms include: the Apriori algorithm and k-Means.

3. Semi-Supervised Learning



Semi-supervised Learning Algorithms

Input data is a mixture of labeled and unlabelled examples.

There is a desired prediction problem but the model must learn the structures to organize the data as well as make predictions.

Example problems are classification and regression.

Example algorithms are extensions to other flexible methods that make assumptions about how to model the unlabeled data.

Overview

When crunching data to model business decisions, you are most typically using supervised and unsupervised learning methods.

A hot topic at the moment is semi-supervised learning methods in areas such as image classification where there are large datasets with very few labeled examples.

Algorithms Grouped By Similarity

Algorithms are often grouped by similarity in terms of their function (how they work). For example, tree-based methods, and neural network inspired methods.

I think this is the most useful way to group algorithms and it is the approach we will use here.

This is a useful grouping method, but it is not perfect. There are still algorithms that could just as easily fit into multiple categories like Learning Vector Quantization that is both a neural network inspired method and an instance-based method. There are also categories that have the same name that describe the problem and the class of algorithm such as Regression and Clustering.

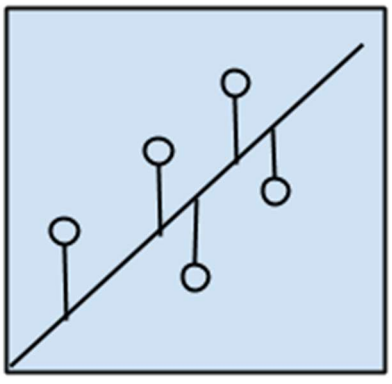
We could handle these cases by listing algorithms twice or by selecting the group that subjectively is the "best" fit. I like this latter approach of not duplicating algorithms to keep things simple.

In this section, I list many of the popular machine learning algorithms grouped the way I think is the most intuitive. The list is not exhaustive in either the groups or the algorithms, but I think it is representative and will be useful to you to get an idea of the lay of the land.

Please Note: There is a strong bias towards algorithms used for classification and regression, the two most prevalent supervised machine learning problems you will encounter.

If you know of an algorithm or a group of algorithms not listed, put it in the comments and share it with us. Let's dive in.

Regression Algorithms



Regression Algorithms

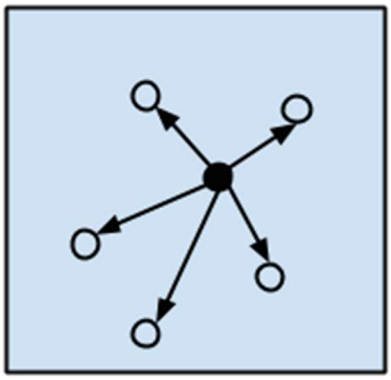
Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model.

Regression methods are a workhorse of statistics and have been co-opted into statistical machine learning. This may be confusing because we can use regression to refer to the class of problem and the class of algorithm. Really, regression is a process.

The most popular regression algorithms are:

- Ordinary Least Squares Regression (OLSR)
- Linear Regression
- Logistic Regression
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)

Instance-based Algorithms



Instance-based Algorithms

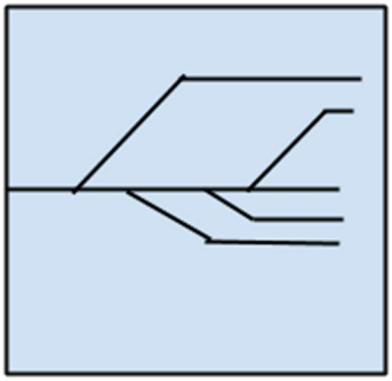
Instance-based learning model is a decision problem with instances or examples of training data that are deemed important or required to the model.

Such methods typically build up a database of example data and compare new data to the database using a similarity measure in order to find the best match and make a prediction. For this reason, instance-based methods are also called winner-take-all methods and memory-based learning. Focus is put on the representation of the stored instances and similarity measures used between instances.

The most popular instance-based algorithms are:

- k-Nearest Neighbor (kNN)
- Learning Vector Quantization (LVQ)
- Self-Organizing Map (SOM)
- Locally Weighted Learning (LWL)

Regularization Algorithms



Regularization Algorithms

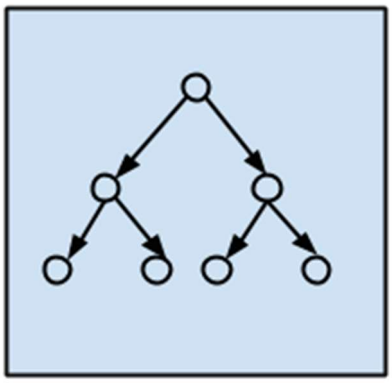
An extension made to another method (typically regression methods) that penalizes models based on their complexity, favoring simpler models that are also better at generalizing.

I have listed regularization algorithms separately here because they are popular, powerful and generally simple modifications made to other methods.

The most popular regularization algorithms are:

- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO)
- Elastic Net
- Least-Angle Regression (LARS)

Decision Tree Algorithms



Decision Tree Algorithms

Decision tree methods construct a model of decisions made based on actual values of attributes in the data.

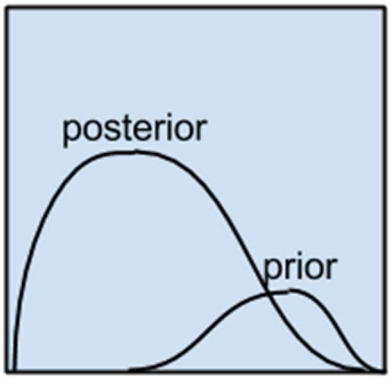
Decisions fork in tree structures until a prediction decision is made for a given record. Decision trees are trained on data for classification and regression problems. Decision trees are often fast and accurate and a big favorite in machine learning.

The most popular decision tree algorithms are:

- Classification and Regression Tree (CART)

- Iterative Dichotomiser 3 (ID3)
- C4.5 and C5.0 (different versions of a powerful approach)
- Chi-squared Automatic Interaction Detection (CHAID)
- Decision Stump
- M5
- Conditional Decision Trees

Bayesian Algorithms



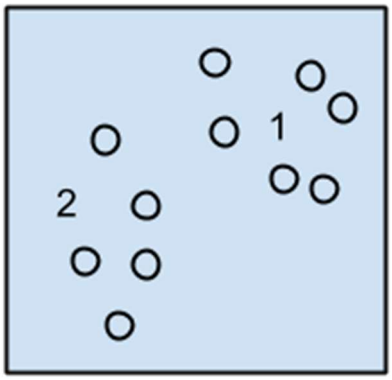
Bayesian Algorithms

Bayesian methods are those that explicitly apply Bayes' Theorem for problems such as classification and regression.

The most popular Bayesian algorithms are:

- Naive Bayes
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Averaged One-Dependence Estimators (AODE)
- Bayesian Belief Network (BBN)
- Bayesian Network (BN)

Clustering Algorithms



Clustering Algorithms

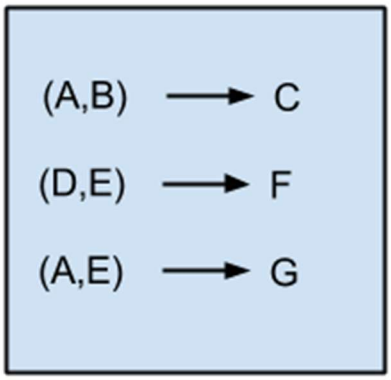
Clustering, like regression, describes the class of problem and the class of methods.

Clustering methods are typically organized by the modeling approaches such as centroid-based and hierarchal. All methods are concerned with using the inherent structures in the data to best organize the data into groups of maximum commonality.

The most popular clustering algorithms are:

- k-Means
- k-Medians
- Expectation Maximisation (EM)
- Hierarchical Clustering

Association Rule Learning Algorithms



Association Rule Learning Algorithms

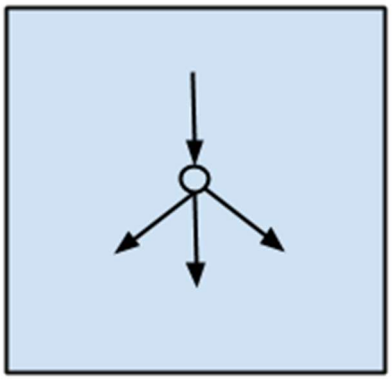
Association rule learning methods extract rules that best explain observed relationships between variables in data.

These rules can discover important and commercially useful associations in large multidimensional datasets that can be exploited by an organization.

The most popular association rule learning algorithms are:

- Apriori algorithm
- Eclat algorithm

Artificial Neural Network Algorithms



Artificial Neural Network Algorithms

Artificial Neural Networks are models that are inspired by the structure and/or function of biological neural networks.

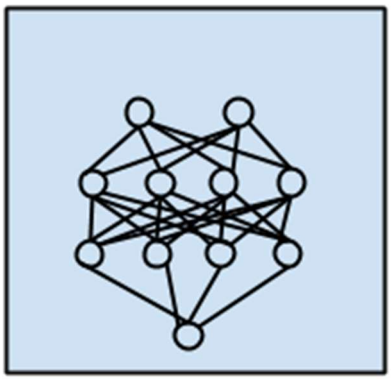
They are a class of pattern matching that are commonly used for regression and classification problems but are really an enormous subfield comprised of hundreds of algorithms and variations for all manner of problem types.

Note that I have separated out Deep Learning from neural networks because of the massive growth and popularity in the field. Here we are concerned with the more classical methods.

The most popular artificial neural network algorithms are:

- Perceptron
- Back-Propagation
- Hopfield Network
- Radial Basis Function Network (RBFN)

Deep Learning Algorithms



Deep Learning Algorithms

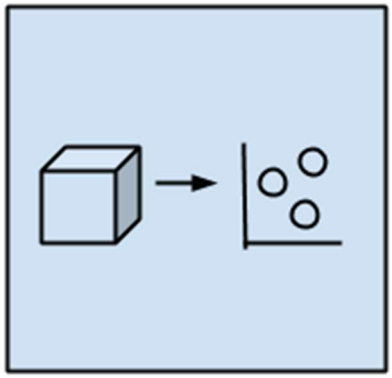
Deep Learning methods are a modern update to Artificial Neural Networks that exploit abundant cheap computation.

They are concerned with building much larger and more complex neural networks and, as commented on above, many methods are concerned with semi-supervised learning problems where large datasets contain very little labeled data.

The most popular deep learning algorithms are:

- Deep Boltzmann Machine (DBM)
- Deep Belief Networks (DBN)
- Convolutional Neural Network (CNN)
- Stacked Auto-Encoders

Dimensionality Reduction Algorithms



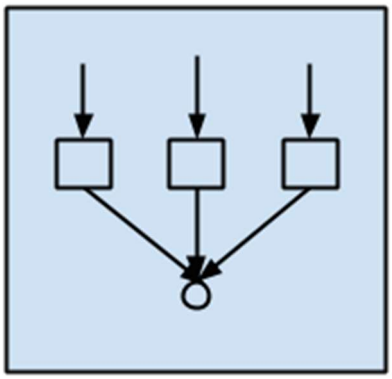
Dimensional Reduction Algorithms

Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, but in this case in an unsupervised manner or order to summarize or describe data using less information.

This can be useful to visualize dimensional data or to simplify data which can then be used in a supervised learning method. Many of these methods can be adapted for use in classification and regression.

- Principal Component Analysis (PCA)
- Principal Component Regression (PCR)
- Partial Least Squares Regression (PLSR)
- Sammon Mapping
- Multidimensional Scaling (MDS)
- Projection Pursuit
- Linear Discriminant Analysis (LDA)
- Mixture Discriminant Analysis (MDA)
- Quadratic Discriminant Analysis (QDA)
- Flexible Discriminant Analysis (FDA)

Ensemble Algorithms



Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction.

Much effort is put into what types of weak learners to combine and the ways in which to combine them. This is a very powerful class of techniques and as such is very popular.

- Boosting
- Bootstrapped Aggregation (Bagging)
- AdaBoost
- Stacked Generalization (blending)
- Gradient Boosting Machines (GBM)
- Gradient Boosted Regression Trees (GBRT)
- Random Forest

Other Algorithms

Many algorithms were not covered.

For example, what group would Support Vector Machines go into? Its own?

I did not cover algorithms from specialty tasks in the process of machine learning, such as:

- Feature selection algorithms
- Algorithm accuracy evaluation
- Performance measures

I also did not cover algorithms from specialty subfields of machine learning, such as:

- Computational intelligence (evolutionary algorithms, etc.)
- Computer Vision (CV)
- Natural Language Processing (NLP)
- Recommender Systems
- Reinforcement Learning
- Graphical Models
- And more…

These may feature in future posts.

Further Reading

This tour of machine learning algorithms was intended to give you an overview of what is out there and some ideas on how to relate algorithms to each other.

I've collected together some resources for you to continue your reading on algorithms. If you have a specific question, please leave a comment.

Other Lists of Algorithms

There are other great lists of algorithms out there if you're interested. Below are few hand selected examples.

- List of Machine Learning Algorithms: On Wikipedia. Although extensive, I do not find this list or the organization of the algorithms particularly useful.
- Machine Learning Algorithms Category: Also on Wikipedia, slightly more useful than Wikipedias great list above. It organizes algorithms alphabetically.
- CRAN Task View: Machine Learning & Statistical Learning: A list of all the packages and all the algorithms supported by each machine learning package in R. Gives you a grounded feeling of what's out there and what people are using for analysis day-to-day.
- Top 10 Algorithms in Data Mining: Published article and now a book (Affiliate Link) on the most popular algorithms for data mining. Another grounded and less overwhelming take on methods that you could go off and learn deeply.

How to Study Machine Learning Algorithms

Algorithms are a big part of machine learning. It's a topic I am passionate about and write about a lot on this blog. Below are few hand selected posts that might interest you for further reading.

- How to Learn Any Machine Learning Algorithm: A systematic approach that you can use to study and understand any machine learning algorithm using "algorithm description templates" (I used this approach to write my first book).
- How to Create Targeted Lists of Machine Learning Algorithms: How you can create your own systematic lists of machine learning algorithms to jump start work on your next machine learning problem.
- How to Research a Machine Learning Algorithm: A systematic approach that you can use to research machine learning algorithms (works great in collaboration with the template approach listed above).
- How to Investigate Machine Learning Algorithm Behavior: A methodology you can use to understand how machine learning algorithms work by creating and executing very small studies into their behavior. Research is not just for academics!
- How to Implement a Machine Learning Algorithm: A process and tips and tricks for implementing machine learning algorithms from scratch.

How to Run Machine Learning Algorithms

Sometimes you just want to dive into code. Below are some links you can use to run machine learning algorithms, code them up using standard libraries or implement them from scratch.

- How To Get Started With Machine Learning Algorithms in R: Links to a large number of code examples on this site demonstrating machine learning algorithms in R.

- Machine Learning Algorithm Recipes in scikit-learn: A collection of Python code examples demonstrating how to create predictive models using scikit-learn.
- How to Run Your First Classifier in Weka: A tutorial for running your very first classifier in Weka (**no code required!**).

Take Control By Creating Targeted Lists of Machine Learning Algorithms

Machine Learning AlgorithmsAny book on machine learning will list and describe dozens of machine learning algorithms.

Once you start using tools and libraries you will discover dozens more. This can really wear you down, if you think you need to know about every possible algorithm out there.

A simple trick to tackle this feeling and take some control back is to make lists of machine learning algorithms.

This ridiculously simple tactic can give you a lot of power. You can use it to give you a list of methods to try when tackling a whole new class of problem. It can also give you a list of ideas when you get stuck on a dataset or your favorite method does not give you good results.

In this post you will discover the benefits of creating lists of machine learning algorithms, how to do it, how to do it well and why you should start creating your first list of algorithms today.

Create a List of Machine Learning Algorithms
Photo by Joel Montes de Oca, some rights reserved

Dealing with So Many Algorithms

There are hundreds of machine learning algorithms.

I see this leading to two problems:

1. Overwhelm

The fact that there are so many algorithms to choose from and try on a given machine learning problem causes some people to freeze up and do nothing.

The fact is, you don't need to get the best result, you only need a result – a beachhead on the problem – and you can get there by spot checking a few algorithms.

2. Favorites

Because there are so many algorithms, some people select one or two favorite algorithms and only use them. This limits the results they can achieve and the problems that they can address.

Favorites are dangerous. Some algorithms are more powerful than others, but that power comes at a cost of complexity and parsimony. They are tools, leave your emotional attachment at the door.

Take Control of the Algorithms

You need focus, a starting point to address the problem of dealing with so many machine learning algorithms.

This involves finding the edges and pushing back the fog on what is out there and what you can use when. This will give you a sense of control over the algorithms and help you to wield them rather than make you feel overwhelmed.

The great thing is, you don't need to become an expert in each algorithm to make progress. You don't need to know very much about each algorithm at all.

Collecting simple information such as algorithm names and the general problems to which they are applicable can help you quickly and confidently build familiarization and confidence with the extent of machine learning algorithms available.

How to Build and Maintain a List of Algorithms

The answer is to build your own personal list of machine learning algorithms.

I'm a list maker, and this method really lights up my brain.

Open a text file, word document or spreadsheet and start listing down the names of algorithms. It's that simple. You can also list the general class to which the algorithm belongs and the general types of problems that it can address.

Define your own categories. This list is a tool to help you understand and navigate the machine learning tools at your disposal. Customize the list to include the algorithm details that you care about.

Examples of Algorithm Lists To Create

Below are 10 examples of machine learning algorithm lists that you could create.

- Regression algorithms
- SVM algorithms
- Data projection algorithms
- Deep learning algorithms
- Time series forecasting algorithms
- Rating system algorithms
- Recommender system algorithms
- Feature selection algorithms
- Class imbalance algorithms
- Decision tree algorithms

Tips for Great Algorithm Lists

Creating a list of algorithms is relatively easy. The hard part is knowing why you want the list. The "*why*" will help you define the type of list you want to create and the algorithm properties you want to describe in your list.

Start with the current project you are working on or your current interests. For example, if you are working on a time series or image classification problem, list all the algorithms that you could apply

to that problem. If you are deeply interested in Support Vector Machines, list all the variations of SVM that you can find.

Don't try to create the perfect list in one sitting. Create it and keep adding to it over days and weeks. It is a useful resource that you can turn back to again and again and add to as your knowledge and experience grows.

In summary, 5 tips for creating great algorithm lists are:

- Start with why you want the list and use that to define the type of list to create.
- Only capture the algorithm properties you actually need, keep it as simple as possible.
- Start with a current project or interest and create a list of related algorithms.
- Don't aim for abstract perfection, the list is for you and your needs alone.
- Add to your list over time and expand it as your skills and experience grow.

When To Use An Algorithm List

Algorithm lists are more valuable than you think.

For example, you can use it as a technique when tackling a problem type that you have never worked on before, such as recommender systems, face detection or rating systems. A simple algorithm list gives you a list of things to try.

When working on a familiar problem, your prior biases often limit the results that you can achieve. A list of algorithms relevant to a problem domain can get you unstuck and even push you to achieve new and better results. That does not mean you should try all algorithms you can find, you still require reasoned and systematic application. Nevertheless, a list can provide a useful starting point.

Algorithm lists are a tool, but you can take them further. To make effective use of machine learning algorithms, you need to study them, research them, and even describe them. This is a natural extension for the algorithm list method and your lists can provide the basis for your self-study curriculum.

You could start by collecting additional properties about each algorithm and expand your list into a mini-encyclopaedia of algorithms, with one page per algorithm. I use an algorithm description template and focus on template elements that I will find useful as I refer back to the descriptions in the future, such as pseudo code and usage heuristics.

In summary, 3 examples of when you can use an algorithm list are:

- When you start working on a new class of problem.
- When you are stuck or looking for algorithms to try on an existing problem.
- When you are looking for algorithms to describe in more detail or research.

Anyone Can Create Machine Learning Algorithm Lists

You don't need to deep dive into machine learning textbooks or open source libraries. A simple google search or browse of wikipedia will uncover many algorithm names to start-off your list.

If you are stuck on what to create for your first list, pick one of the examples above or browse a site like DataTau and pick a class of algorithm to list mentioned in an article or article title.

Again, you do not have to list every algorithm that you could list, narrow your scope to those algorithms in the libraries and tools you prefer. You don't need to list every permutation of every algorithm, for example, you could focus on one aspect of an algorithm, such as the kernel functions for an SVM or the transfer functions for a neural network.

Don't list all possible features of each algorithm. Stick to just the name and maybe the general class of algorithm and general types of problems for which it can be used. If you want to go deeper into an algorithm, consider the algorithm description method and template described previously.

You don't need to understand the algorithms yet and you don't need to be an academic. This is a beachhead that you are taking to expand your idea of what is out there, to overcome overwhelm and to finally to provide a point of departure on your journey deeper into applied machine learning.

Action Steps

In this post you learned about the simple tactic of creating lists of machine learning algorithms.

You discovered that this simple tactic can help you to overcome algorithm overwhelm and to help get you unstuck from the dangers of having favorite algorithms.

Your action step for this post is to create your first algorithm list. Pick something small, like a subclass of an algorithm. Pick something fun, like an algorithm that is hot right now.

Share your list if you like (or what your list is about), it would help to motivate others.

How to Learn a Machine Learning Algorithm

Machine Learning AlgorithmsThe question of how to learn a machine learning algorithm has come up a few times on the email list.

In this post I'll share with you the strategy I have been using for years to learn and build up a structured description of an algorithm in a step-by-step manner that I can add to, refine and refer back to again and again. I even used it to write a book.

This was just a strategy I used personally and I've been really surprised by the positive feedback.

Algorithm Descriptions are Broken

Learning a machine learning algorithm can be overwhelming. There are so many papers, books and websites describing how the algorithm works mathematically and textually. If you are really lucky you might find a pseudocode description of the algorithm.

If you are really really lucky you might find some suggested ways to configure the method for different situations. These descriptions are rare and typically buried deep in the original publication or in technical notes by the original authors.

A fact you learn quickly when you want to implement a method from research papers is that algorithms are almost never described in sufficient detail for you to reproduce them. The reasons vary, from the micro-decisions that are left out of the paper, to whole procedures that are summarized ambiguously in text, to symbols that are used inconsistently.

Piece it Together

To understand an algorithm you have to piece together an understanding yourself from disparate descriptions. This is the only tactic that I have found to be successful.

Desperate descriptions means resources such as the original descriptions of the method in the primary sources as well as authoritative secondary interpretations made of original descriptions in review papers and books.

It is common for there to be prototype implementations of a method released with the primary sources and reading this code (typically C, FORTRAN, R or Matlab) can be very enlightening for the details you need to reproduce an algorithm.

Algorithm Descriptions

An algorithm is an island of research and in all reality it can be difficult to pin down the canonical definition. For example, is it the version described in the primary source or is it the version that includes all the fixes and enhancements that are “best practice”.

A solution is to consider a given algorithm from multiple perspectives, each of which can serve a different purpose. For example, the abstract information processing description of the algorithm could be realized by a variety of different specific computational implementations.

I like this approach because it defends the need to telescope in on a specific case of the algorithm from many possible cases at each step of the description while also leaving the option open for the description of variations.

There are many descriptions you could use of varying specificity depending on your needs. Some that I like to use include: inspiration for the algorithm, metaphor or analogy for the strategy, information processing objectives, pseudocode and code.

Algorithm Prescriptions

When I started my own research projects, I thought the answer to this problem was to read everything on an algorithm and create the definitive implementation in code. Nice idea perhaps, but code is just one way to communicate an algorithm, and it is limited.

There is more to an algorithm description than the computation. There is meta information around an algorithm that can be invaluable for certain use cases.

For example, usage heuristics for an algorithm are embedded in papers. Having a summary of usage heuristics collected together in one place can mean the difference of getting a good enough result quickly and running sensitivity analysis on the algorithm for days or weeks.

Other examples include the standard experimental datasets used to test the algorithm, the general classes of problem to which the algorithm is suited, and known limitations that have been identified and described for the algorithm.

Design an Algorithm Description Template

An algorithm description template provides a structured way for you to learn about a machine learning algorithm.

You can start with a blank document and list out the section headings for the types of descriptions you need of the algorithm, for example applied, implementation, or your own personal reference cheat sheet.

To figure out what sections to include in your template, list out questions you would like to answer about the algorithm, or algorithms if you are looking to build up a reference. Some questions you could use include:

- What is the standard and abbreviations used for the algorithm?
- What is the information processing strategy of the algorithm?
- What is the objective or goal for the algorithm?
- What metaphors or analogies are commonly used to describe the behavior of the algorithm?
- What is the pseudocode or flowchart description of the algorithm?
- What are the heuristics or rules of thumb for using the algorithm?
- What classes of problem is the algorithm well suited?
- What are common benchmark or example datasets used to demonstrate the algorithm?
- What are useful resources for learning more about the algorithm?
- What are the primary references or resources in which the algorithm was first described?

Once you have settled on some questions, turn them into section headings.

For each section heading clearly define the requirements of the section and the form that the description in that section will take. I like to include motivating questions for each section that once answered will satisfy the section at the minimum level of detail.

Start Small and Build it Up

The beauty of this approach is that you don't need to be an expert in the algorithm or in research. As long as you can find some resources that mention the algorithm, you can start to capture notes about an algorithm in the template.

You can start really simply and collect high-level descriptions of the algorithm, its names and abbreviations and the resources you have found and what they talk about. From here you can decide to expand the description further, or not.

You will end up with a one-to-two page description of the algorithm very quickly.

I Use It

I've been using algorithm templates for a long time. Some examples where I have found this strategy practically useful include:

- Implementing machine learning algorithms using a descriptive-focused template.
- Applying a machine learning algorithm using an applied-focused template.
- Building a catalog of algorithms to use and refer to using a general purpose template.

In this last case, I turned my catalog into a book of 45 nature inspired algorithms which I published in early 2011. The book is called Clever Algorithms: Nature-Inspired Programming Recipes (affiliate link).

Summary

In this post you learned about using an algorithm description template as a strategy for learning a machine learning algorithm.

You learned that algorithm descriptions are broken and the answer to learning an algorithm effectively is to design an algorithm template that meets your needs and to fill in the template as you read and learn about the algorithm.

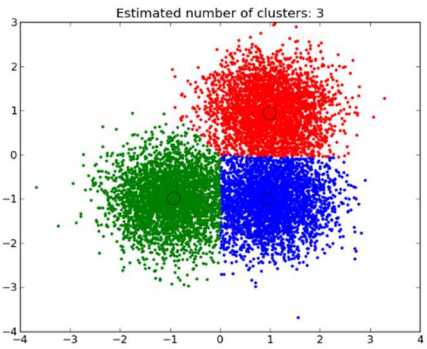
You learned that the template is an efficient and structured way to tackle an overwhelming problem.

Machine Learning Algorithm Recipes in scikit-learn

You have to get your hands dirty.

You can read all of the blog posts and watch all the videos in the world, but you're not actually going to start really get machine learning until you start practicing.

The scikit-learn Python library is very easy to get up and running. Nevertheless I see a lot of hesitation from beginners looking get started. In this blog post I want to give a few very simple examples of using scikit-learn for some supervised classification algorithms.



Scikit-Learn Recipes

You don't need to know about and use all of the algorithms in scikit-learn, at least initially, pick one or two (or a handful) and practice with only those.

In this post you will see 5 recipes of supervised classification algorithms applied to small standard datasets that are provided with the scikit-learn library.

The recipes are principled. Each example is:

- **Standalone**: Each code example is a self-contained, complete and executable recipe.
- **Just Code**: The focus of each recipe is on the code with minimal exposition on machine learning theory.

- **Simple**: Recipes present the common use case, which is probably what you are looking to do.
- **Consistent**: All code example are presented consistently and follow the same code pattern and style conventions.

The recipes do not explore the parameters of a given algorithm. They provide a skeleton that you can copy and paste into your file, project or python REPL and start to play with immediately.

These recipes show you that you can get started practicing with scikit-learn right now. Stop putting it off.

Logistic Regression

Logistic regression fits a logistic model to data and makes predictions about the probability of an event (between 0 and 1).

This recipe shows the fitting of a logistic regression model to the iris dataset. Because this is a mutli-class classification problem and logistic regression makes predictions between 0 and 1, a one-vs-all scheme is used (one model per class).

For more information see the API reference for Logistic Regression for details on configuring the algorithm parameters. Also see the Logistic Regression section of the user guide.

Naive Bayes

Naive Bayes uses Bayes Theorem to model the conditional relationship of each attribute to the class variable.

This recipe shows the fitting of an Naive Bayes model to the iris dataset.

For more information see the API reference for the Gaussian Naive Bayes for details on configuring the algorithm parameters. Also see the Naive Bayes section of the user guide.

k-Nearest Neighbor

The k-Nearest Neighbor (kNN) method makes predictions by locating similar cases to a given data instance (using a similarity function) and returning the average or majority of the most similar data instances. The kNN algorithm can be used for classification or regression.

This recipe shows use of the kNN model to make predictions for the iris dataset.

For more information see the API reference for the k-Nearest Neighbor for details on configuring the algorithm parameters. Also see the k-Nearest Neighbor section of the user guide.

Classification and Regression Trees(CART)

Classification and Regression Trees (CART) are constructed from a dataset by making splits that best separate the data for the classes or predictions being made. The CART algorithm can be used for classification or regression.

This recipe shows use of the CART model to make predictions for the iris dataset.

For more information see the API reference for CART for details on configuring the algorithm parameters. Also see the Decision Tree section of the user guide.

Support Vector Machines(codice sul link)

Support Vector Machines (SVM) are a method that uses points in a transformed problem space that best separate classes into two groups. Classification for multiple classes is supported by a one-vs-all method. SVM also supports regression by modeling the function with a minimum amount of allowable error.

This recipe shows use of the SVM model to make predictions for the iris dataset.

Summary

In this post you have seen 5 self-contained recipes demonstrating some of the most popular and powerful supervised classification problems.

Each example is less than 20 lines that you can copy and paste and start using scikit-learn, right now. Stop reading and start practicing. Pick one recipe and run it, then start to play with the parameters and see what effect that has on the results.

How To Investigate Machine Learning Algorithm Behavior

Machine learning algorithms are complex systems that require study to understand.

Static descriptions of machine learning algorithms are a good starting point, but are insufficient to get a feeling for how the algorithm behaves. You need to see the algorithm in action.

Experimenting on a running machine learning algorithms will allow you to build an intuition for the cause and effect relationship of the algorithm parameters with the results you can achieve on different classes of problem.

In this post you will discover how to investigate a machine learning algorithm. You will learn about a simple 5-step process that you can use today to design and complete your first machine learning algorithm experiment.

You will discover that machine learning experiments are not just for academics, that you can do them to, and that experimentation is required on the path to mastery as the empirical cause-and-effect knowledge that you will gain is simply not available anywhere else.

What is Investigating Machine Learning Algorithms

Your objective when investigating a machine learning algorithm is to find behaviors that lead to good results that are generalizable across problems and classes of problems.

You investigate machine learning algorithms by performing systematic research into the algorithms behavior. This is done by designing and executing controlled experiments.

Once you have completed an experiment, you can interpret and present the results. The results give you glimpses into the cause and effect between changes to the algorithm, it's behaviors and the results you can achieve.

How to Investigate Machine Learning Algorithms

In this section we will look at a simple 5-step procedure that you can use to investigate a machine learning algorithm.

1. Select an Algorithm

Select an algorithm that you have questions about.

This may be an algorithm that you are using in ernest on a problem, or an algorithm that you see doing well in other contexts that you may want to use in the future.

For the purposes of experimentation, it is useful to take an off-the-shelf implementation of the algorithm. This gives you a baseline that most likely has few if any bugs.

Implementing the algorithm yourself can be a great way to learn about the algorithm procedure, but can also introduce additional variables into the experiment such as bugs and the myriad of micro-decisions that must be made for each algorithm implementation.

2. Identify a Question

You must have a research question that you are seeking to answer. The more specific the question, the more useful the answer.

Some example questions include:

- What is the effect of increasing k in kNN as a fraction of the training dataset size?
- What is the effect of selecting different kernels in SVM on binary classification problems?
- What are the effects of different attribute scaling on logistic regression on binary classification problems?
- What is the effect of adding random attributes to the training dataset on classification accuracy in random forest?

Design the question that you want answered about your algorithm. Consider listing five variations of the question and hone in on the one that is the most specific.

3. Design the Experiment

Pick the elements out of the question that will make-up your experiment.

For example, take the following question from above: "*What are the effects of different attribute scaling on logistic regression on binary classification problems?*"

The elements you can pick out of this question for the design of your experiment are:

- **Attribute Scaling Methods**. You could include methods like normalization, standardization, raising an attribute to a power, taking the logarithm, etc.
- **Logistic Regression**. Which implementation of logistic regression you want to use.
- **Binary Classification Problems**. Different standard binary classification problems that have numeric attributes. Multiple problems will be required, some with attributes all the

same scale (like ionosphere) and others that have attributes with a variety of scales (like diabetes).
- **Performance**. A model performance score is required such as classification accuracy.

Take the time to carefully select the elements of your question to best answer your question.

4. Execute the Experiment and Report Results

Complete your experiment.

If the algorithm is stochastic, you may need to repeat experimental runs multiple times and take a mean and standard deviation.

If you are looking for differences in results between experimental runs (such as different parameters), you may want to use a statistical tool to indicate whether the differences are statistically significant (such as the student t-test).

Some tools like R and scikit-learn/SciPy have the tools available to complete these types of experiments, but you will need to bring them together and script the experiment. Other tools like Weka have the tools built into a graphical user interface (see this tutorial on running your first experiment in Weka). The tools you use matter less than the question and the rigor of your experimental design.

Summarize the results of your experiment. You may want to use tables and graphs. Presenting results alone is insufficient. They are just numbers. You must tie the numbers back to your question and filter their meaning through the design of your experiment.

What do the results indicate about your research question?

Put on your skeptical hat. What holes or limitations can you place on the results. Do not shy away from this part. Knowing the limitations is just as important as knowing the outcomes of an experiment.

5. Repeat

Repeat the process.

Continue to investigate your selected algorithm. You may even want to repeat the same experiment with different parameters or different test datasets. You may want to address the limitations in your experiment.

Don’t stop with one experiment, start building up a knowledge base and an intuition for the algorithm.

With some simple tools, some good questions and a good splash of rigor and skepticism, you can very quickly start coming up with world-class understandings into the behavior of an algorithm.

Investigating Algorithms is Not Just for Academics

You can investigate the behaviors of machine learning algorithms.

You do not need a higher degree, you do not need to be trained in research methods, you do not need to be an academic.

Careful systematic investigation of machine learning algorithms is open to anyone with a computer and a deep interest. In fact, if you want to master machine learning, you must get comfortable with systematic investigations of machine learning algorithms. The knowledge is simply not out there, you must go out and collect it yourself, empirically.

You do need to be skeptical and to be careful when talking about the applicability of your findings.

You do not need to have unique questions. You will gain a lot by investigating the standard questions, such as the effect of one parameter generalized across a few standard datasets. You may very well find limitations or counter points to common best practice heuristics.

Action Steps

In this post you discovered the importance of investigating the behaviors of machine learning algorithms through controlled experimentation. You discovered a simple 5-step process that you can use you design and execute your first experiment on a machine learning algorithm.

Take action. Use the process you learned in this blog post and complete you first machine learning experiment. Once you have completed one, even a very small one, you will have the confidence, tools, and ability to complete a second and many more.

I would love to hear about your first experiment. Leave a comment and share your results or what you learned.

Category:Machine learning algorithms

The following 58 pages are in this category, out of 58 total. This list may not reflect recent changes (learn more).

A

- Algorithms of Oppression
- Almeida–Pineda recurrent backpropagation

B

- Backpropagation
- Bootstrap aggregating

C

- CN2 algorithm
- Constructing skill trees

D

- Dehaene–Changeux model

- Diffusion map
- Dominance-based rough set approach
- Dynamic time warping

E

- Error-driven learning
- Evolutionary multimodal optimization
- Expectation–maximization algorithm

F

- FastICA
- Forward–backward algorithm

G

- GeneRec
- Genetic Algorithm for Rule Set Production
- Growing self-organizing map

H

- HEXQ
- Hyper basis function network

I

- IDistance

K

- K-nearest neighbors algorithm
- Kernel methods for vector output
- Kernel principal component analysis

L

- Leabra
- Linde–Buzo–Gray algorithm
- Local outlier factor
- Logic learning machine
- LogitBoost
- Loss functions for classification

M

- Manifold alignment
- Minimum redundancy feature selection
- Mixture of experts

- Multiple kernel learning

N

- Non-negative matrix factorization

O

- Online machine learning
- Out-of-bag error

P

- Prefrontal cortex basal ganglia working memory
- PVLV

Q

- Q-learning
- Quadratic unconstrained binary optimization
- Query-level feature
- Quickprop

R

- Radial basis function network
- Randomized weighted majority algorithm
- Reinforcement learning
- Repeated incremental pruning to produce error reduction (RIPPER)
- Rprop
- Rule-based machine learning

S

- Skill chaining
- Sparse PCA
- State–action–reward–state–action
- Stochastic gradient descent
- Structured kNN

T

- T-distributed stochastic neighbor embedding
- Temporal difference learning

W

- Wake-sleep algorithm
- Weighted majority algorithm (machine learning)


Machine Learning Algorithms
Deep Learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Bayesian
Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR)
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
Clustering
k-Means
k-Medians
Expectation Maximization
Hierarchical Clustering