:: Random Forest

Disadvantages of Random Forest

Random Forest also has some limiataions

- **Large number of decision trees in the random forest can slow down the algorithm**. [1]
  In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

- **Good job at classification but not as good as for regression**
  It's not good at regression problem as it does not give precise continuous nature predictions. In case of regression, it doesn't predict beyond the range in the training data. [2]

- **like a black box approach, random forest is not easily interpretable.**
  Although random forest is intuitively easy to understand, but its difficult to get an insight as to what the algorithm actually does. Analyzing random forest theoretically is difficult.

Picture : Designed by Asierromero / Freepik

1 source: https://www.dezyre.com/article/top-10-machine-learning-algorithms/202

2source: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

Part 3

Miscellaneous topics

:: The last but not the least

- *Advice for designing & building machine learning system*
- *Learning with large dataset*
- *What can we do when learning with limited training data ?*
- *Popular machine learning frameworks*

:: Advice for designing & building machine learning system

Recommended approach

- **Build up your initial system quickly, then iterate**
  - Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data [1]
  - Don’t overthink or make your first system too complicated. Instead, just build a quick and dirty system and use that to help you prioritize how to improve your system

- **Use Cross Validation to assessed how the model will generalize**
  - Divide the total dataset into Training Data, Cross-Validation Data and Test data

- **Use Bias/Variance analysis & Error Analysis to prioritize next step**
  - Plot learning curves to decide if more data, more features, etc. are likely to help [2]
  - Error analysis: manually examine the example (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on [3].

Error Analysis (13 min)

https://www.youtube.com/watch?v=k1JGvqr56Yk

1,2,& 3 Source: Andrew Ng, https://www.youtube.com/watch?v=k1JGvqr56Yk.

:: The last but not the least

Advice for designing & building machine learning system

Learning with large dataset

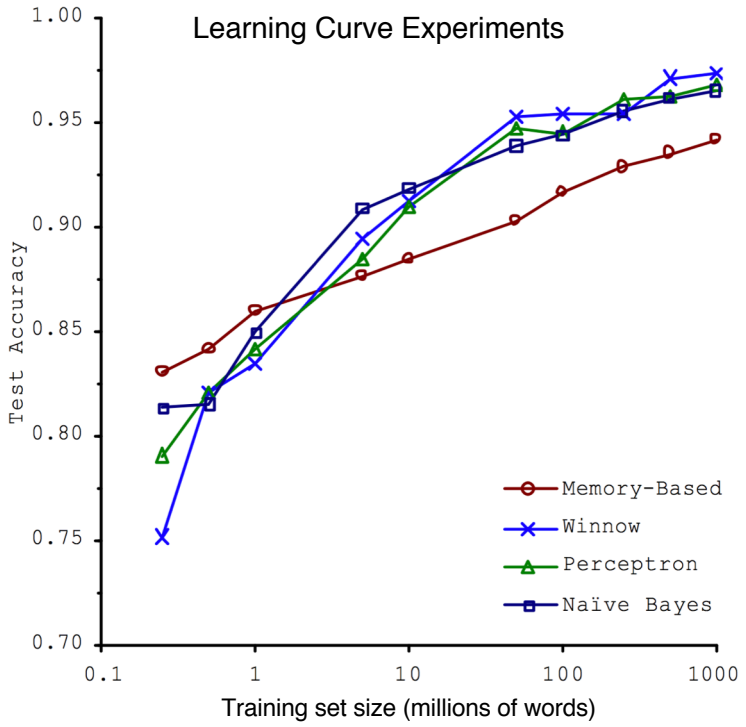
What can we do when learning with limited training data ?

Popular machine learning frameworks

:: Learning with large dataset

Machine learning works best when there is an abundance of data to leverage for training[1]

Today datasets commonly used deep learning is very large. If you look at the amount of traffic that popular websites get, you easily get training sets that are much larger than hundreds of millions of examples[2]. You typically need a lot of data to train a model, especially for deep learning models.



Evaluate the performance of different learning methods on a prototypical natural language disambiguation task. Task example: distinguish pairs of easily-confused words. E.g., {“affect” vs “effect”}, {to, two, too}, {then, than}

“ It’s not who has the best algorithm that wins.

It’s who has the most data [3]”

Figure: Banko and Brill, 2001, Scaling to Very Very Large Corpora for Natural Language Disambiguation
1.,2,3 Source: Andrew Ng, coursera.org/learn/machine-learning/lecture/CipHf/learning-with-large-datasets

:: Challenges in learning with large dataset

Learning with large data sets comes with its own unique problems, specifically, computational problems

Let's say your training set size M is equal to 100,000,000. and this is actually pretty realistic for many modern data sets. And let's say you want to train a linear regression model, or maybe a logistic regression model, in which case you use gradient decent to find the best parameters.

suppose the training set size m = 100,000,000, the **computational expense is too huge**

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

And if you look at what you need to do to compute the gradient, which is this term over here, then when M is a hundred million, you need to carry out a summation over a hundred million terms, in order to compute these derivatives terms and to perform just **one step** of gradient decent.

:: How to deal with the computation problem for large dataset ?

Coming up with a computationally efficient ways is the key in large-scale machine learning

Of course, before we put in the effort into training a model with a hundred million examples, we can plot the learning curve to help judge if large dataset would really improve performance. If the very big data set is really helpful for high performance, you need to figure out how to compute efficiently.

Big Data + Machine Learning

How to scale up your learning algorithms to big data ?

In case that the data is huge, you could

- Use **stochastic gradient descent** to learn iteratively, instead of algorithms that require all data in memory to perform matrix operations
- Use **online learning** techniques [1]
- Split your batch learning work across multiple servers (using the **MapReduce** technique, as we will see later)[2]

Source: 1,2 , Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow

:: How to deal with the computation problem for large dataset ?

Coming up with a computationally efficient ways is the key in large-scale machine learning

Of course, before we put in the effort into training a model with a hundred million examples, we can plot the learning curve to help judge if large dataset would really improve performance. If the very big data set is really helpful for high performance, you need to figure out how to compute efficiently.

Big Data + Machine Learning

How to scale up your learning algorithms to big data ?

In case that the data is huge, you could

- Use **stochastic gradient descent** to learn iteratively, instead of algorithms that require all data in memory to perform matrix operations
- Use **online learning** techniques [1]
- Split your batch learning work across multiple servers (using the **MapReduce** technique, as we will see later)[2]

Source: 1,2 , Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow

:: Use Stochastic gradient descent (SGD) for big datasets

SGD deals with training examples independently, a single training example **in one iteration**

Gradient Descent

- **Batch gradient descent**
  use all m examples in each iteration

- **Mini-batch gradient descent**
  It is something between Batch gradient descent and Stochastic gradient descent). It use b examples in each iteration.
  (b =mini-batch size, e.g. b =10)

- **Stochastic gradient descent**
  use 1 example in each iteration

1. Randomly shuffle (reorder) training examples
2. Update the parameters:

Repeat {

for $i := 1, \ldots, m$ {

$$\theta_j := \theta_j - \alpha\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

(for every $j = 0, \ldots, n$)

}

}

More details

:: How to deal with the computation problem for large dataset ?

Coming up with a computationally efficient ways is the key in large-scale machine learning

Of course, before we put in the effort into training a model with a hundred million examples, we can plot the learning curve to help judge if large dataset would really improve performance. If the very big data set is really helpful for high performance, you need to figure out how to compute efficiently.

Big Data + Machine Learning

How to scale up your learning algorithms to big data ?

In case that the data is huge, you could

- Use **stochastic gradient descent** to learn iteratively, instead of algorithms that require all data in memory to perform matrix operations
- **Use online learning techniques** [1]
- Split your batch learning work across multiple servers (using the **MapReduce** technique, as we will see later)[2]

Source: 1,2 , Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow

:: Online learning can learn incrementally from a stream of incoming data

If you really have a continuous stream of data, then an online learning algorithm can be very effective

Online learning is great for systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously. It is also a good option. if you have limited computing resources: once an online learning system has learned about new data instances, it does not need them anymore, so you can discard them (unless you want to be able to roll back to a previous state and “replay” the data). This can save a huge amount of space.[1]

Stochastic gradient descent

Vs

Online learning

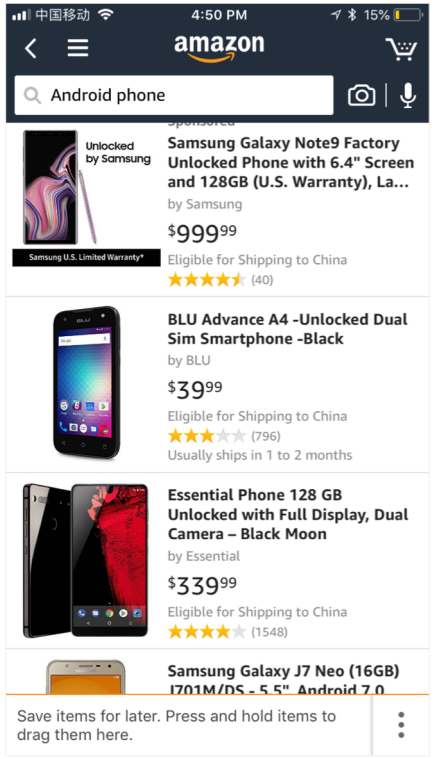
Online learning can deal with very large quantities of data.
Online learning it is really very similar to this stochastic gradient descent algorithm, only instead of scanning through a fixed training set, we're instead getting one example from a user, learning from that example, then discarding it and moving on. We never use it again. [2]

1 Source: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow
2 source: Andew Ng, https://www.coursera.org/lecture/machine-learning/online-learning-ABO2q

:: Online learning example : Product search

Use online learning algorithm to learn user preferences from the stream of data and optimize the strategy

Let's say you run an online store that sells many mobile phones. If user searches for something, a learning algorithm can be used to help us figure out what are ten phones we should return to the user in response to a user-search query.



- User search for " Android phone"
- Suppose we Have 1000 phones in store. Will return 10 results

Input: x

- Different features/properties of phone
- How many words in user query match the name of phone
- How many words in query match description of phone, etc.

y

- $y = 1$ If user clicks on link
- $y = 0$ if user doesn't click on link

We want to show the user phones that they are likely to want to buy, want to show the user phones that they have high probability of clicking in web browser /tapping in mobile app. [1]

So we **learn** $p(y = 1 | x; \theta)$ to estimate the probability that a user will click on the link for a specific phone. Then we can use it to show user the 10 phones they're most likely to click on. This will be a pretty reasonable way to decide what ten results to show to the user.[2]

Online learning algorithm can also adapt to changing user preferences. [3]

1,2,3 Source: Andrew Ng, https://www.coursera.org/lecture/machine-learning/online-learning-ABO2q

:: How to deal with the computation problem for large dataset ?

Coming up with a computationally efficient ways is the key in large-scale machine learning

Of course, before we put in the effort into training a model with a hundred million examples, we can plot the learning curve to help judge if large dataset would really improve performance. If the very big data set is really helpful for high performance, you need to figure out how to compute efficiently.

Big Data + Machine Learning

How to scale up your learning algorithms to big data ?

In case that the data is huge, you could

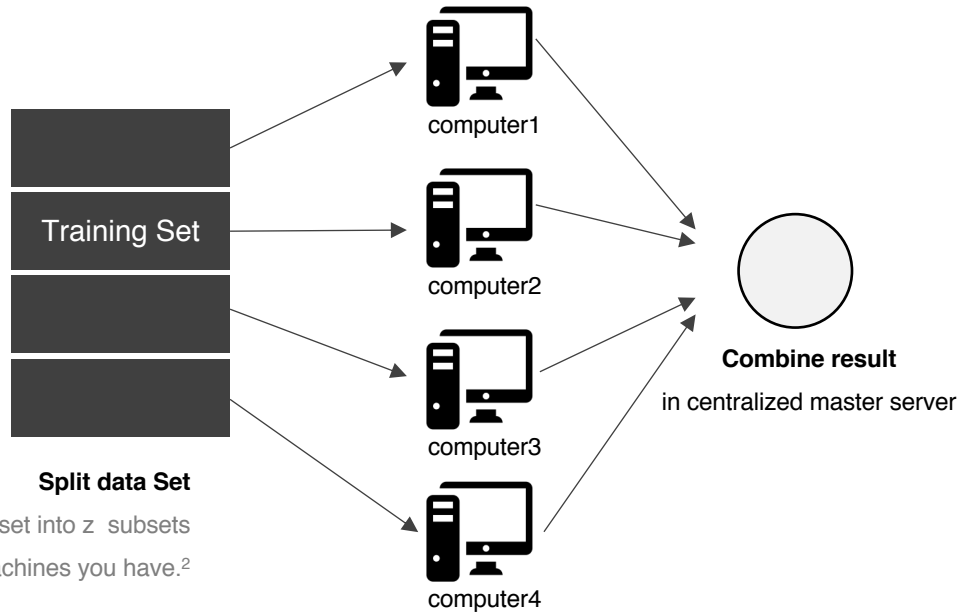
- Use **stochastic gradient descent** to learn iteratively, instead of algorithms that require all data in memory to perform matrix operations
- **Use online learning techniques** [1]
- Split your batch learning work across multiple servers (using the **MapReduce** technique, as we will see later)[2]

Source: 1,2 , Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow

:: Use MapReduce for large scale machine learning problem

Some machine learning problems are just too big to run on one machine

Sometimes maybe you just have so much data , you just don‘t ever want to run all that data through a single computer, no matter what algorithm you would use on that computer[1]. In such case you can MapReduce to address the problem. In other words, you can split the large training set in to different subsets, and then send them into different machines for computing in parallel



Split data Set

You can split your training set into z subsets corresponding to the number of machines you have.[2]

Divide the computation workload into multiple machines

instead of needing to sum over all four hundred training examples on just one machine, we can instead divide up the work load on four machines..

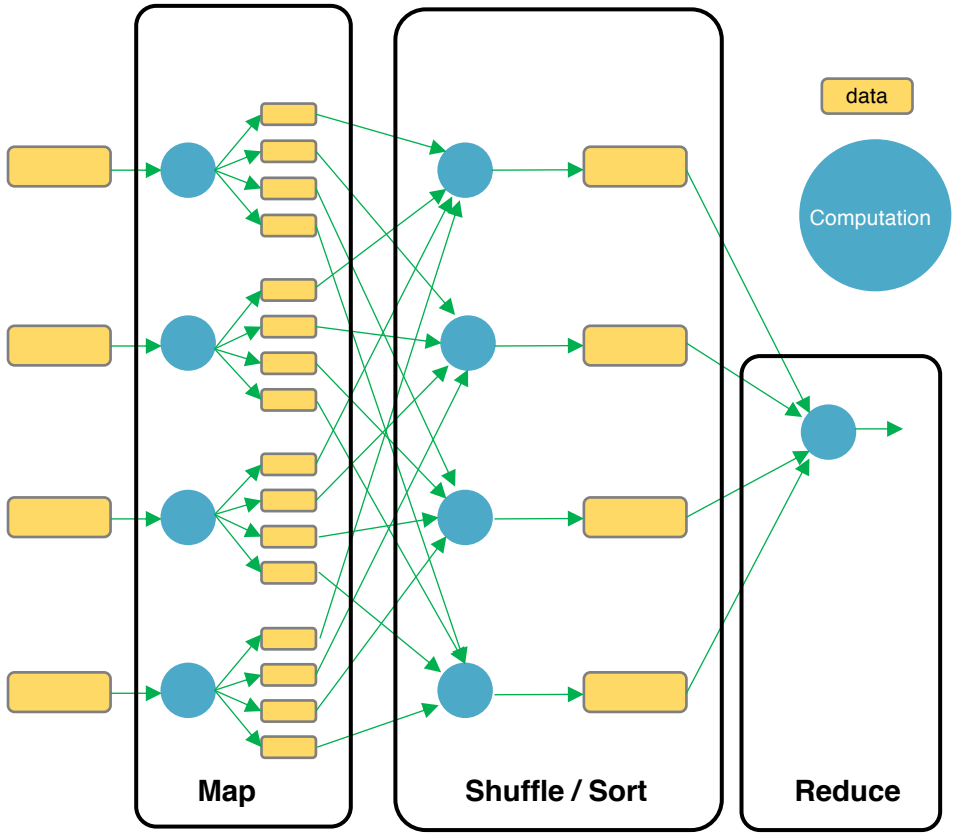
We can divide up batch gradient descent and dispatch the cost function for a subset of the data to many different machines so that we can train our algorithm in parallel.[3]

1,2, 3,& source of diagram :https://www.coursera.org/lecture/machine-learning/map-reduce-and-data-parallelism-

:: About MapReduce

MapReduce is a framework for processing parallelizable problems across large datasets using cluster

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. [1]



MapReduce Overview

A MapReduce framework (or system) is usually composed of three operations (or steps):[2]

- **Map:** (Send each input record to a key)
  - Apply a user-written map function to each input elements
  - The output of map function is set of key-value pairs
- **Shuffle/Sort :** (put all of one key in the same place)
- **Reduce:** (operate on each key and its set of values)
  - User-written Reduce function is applied to each key-(list of values)

1,2 https://en.wikipedia.org/wiki/MapReduce
Source of Diagram: http://www.cs.cmu.edu/~wcohen/10-605/2016/map-reduce.pdf

:: More about using MapReduce for computing gradient descent

Use MapReduce to parallelize the learning algorithm and scale it up to very large training dataset

Whenever your learning algorithm can be expressed as a sum of the training set and whenever the bulk of the work of the learning algorithm can be expressed as the sum of the training set, then map-Reduce might a good candidate[1]

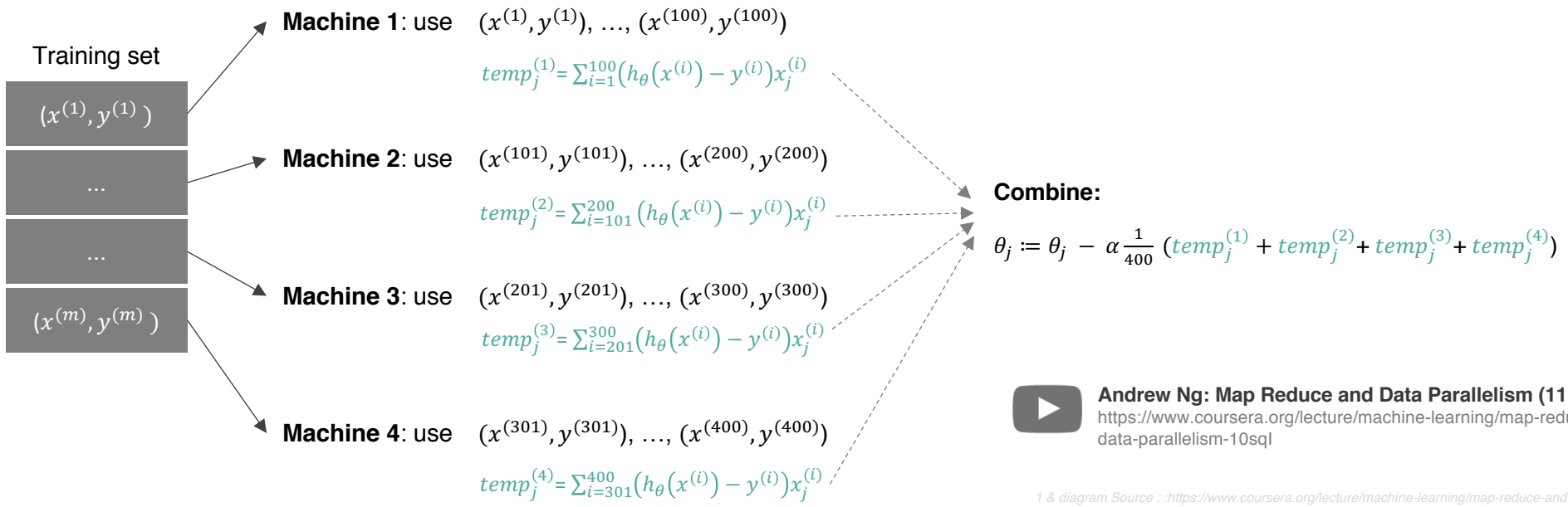
Batch gradient descent: $\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

In real case, the training set could have more than 400,000,000 examples.

Just for sake of writing simplicity, pretend we have 400 examples here

If the dataset is very large, the computation will be very expensive.

We can use **MapReduce** to address the challenge.



Andrew Ng: Map Reduce and Data Parallelism (11 mins)
https://www.coursera.org/lecture/machine-learning/map-reduce-and-data-parallelism-10sql

1 & diagram Source : :https://www.coursera.org/lecture/machine-learning/map-reduce-and-data-parallelism-10sql

:: The last but not the least

- *Advice for designing & building machine learning system*
- *Learning with large dataset*
- ***What can we do when learning with limited training data ?***
  - *Artificial data synthesis*
  - *Transfer learning*
- *Popular machine learning frameworks*

:: What to do if there isn't enough data for the model ?

Deep learning is data-hungry. How to use Deep Learning when you have Limited Data ?

Huge amounts of labelled data is crucial for high performance of model. But in many case gathering the labelled data is very expensive, time-consuming, or even dangerous. What can we do ? Where did you get so much training data from ? We can use artificial data synthesis to artificially growing the training set.

artificial data synthesis

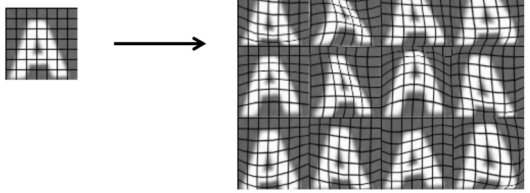
- **Create new data from scratch**
  for example, you can generate brand new images from scratch

example

- **Data Augmentation**

  Data augmentation, consists of generating new training instances from existing ones, artificially boosting the size of the training set.[1]

example



1 Source: Hands-On Machine Learning with scikit-Learn and TensorFlow

source of image: Andrew Ng, https://www.coursera.org/lecture/machine-learning/getting-lots-of-data-and-artificial-data-K0XQT

:: Create new data from scratch

Example: artificial data synthesis for photo OCR

Let’s use character recognition to explain how to create new training instances. The following two pictures look very similar. One is the real data. Another one is just artificially synthesized image.

Real Data

Synthetic Data

How these synthesized images are created ?

One thing you can do is just take characters from different fonts, and paste these characters against different random backgrounds. And then apply maybe a little blurring operators, distortions, scaling , and little rotation operations, etc.

:: Data Augmentation

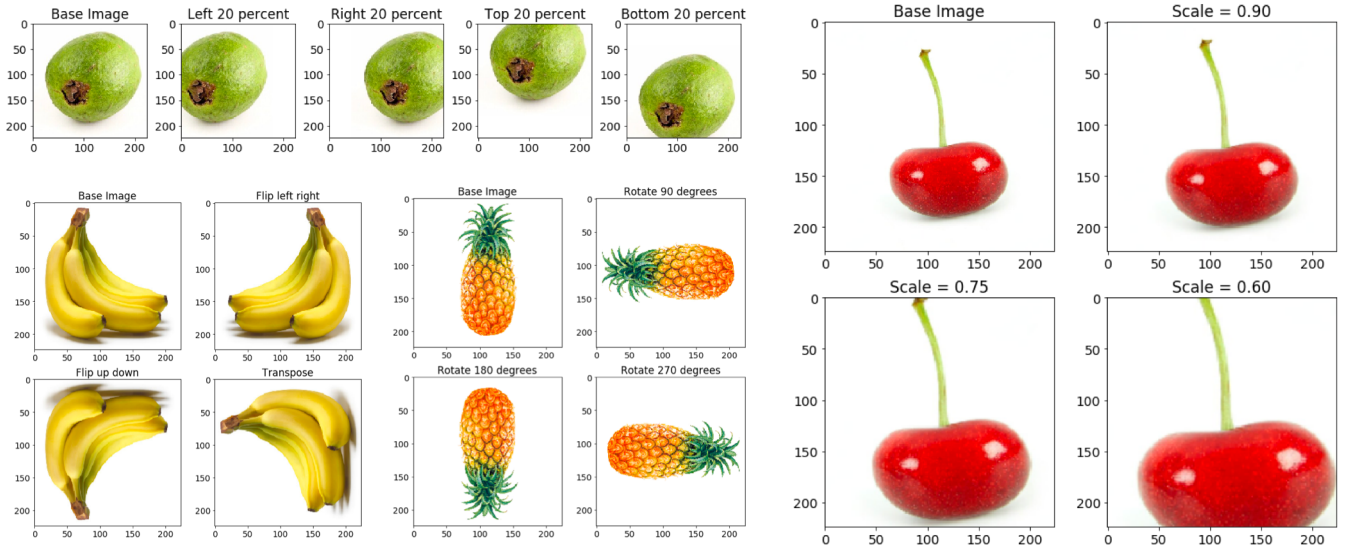
Amplify your training set by creating new training instances based on existing ones

To address the problem of limited quantity and limited diversity of data, we can generate new data with the existing data. For example, you can generate new audio clip by adding different background noise.

Synthesizing data by introducing distortions

For example, you can use following simple techniques to generate new images: [1]

- Flipping (both vertically and horizontally)
- Rotating
- Zooming and scaling
- Cropping
- Translating (moving along the x or y axis)
- Adding Gaussian noise (distortion of high frequency features)



Advanced techniques:

- Neural Style Transfer
- GANs

1 source: https://blog.algorithmia.com/introduction-to-dataset-augmentation-and-expansion/
Source: https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

:: The last but not the least

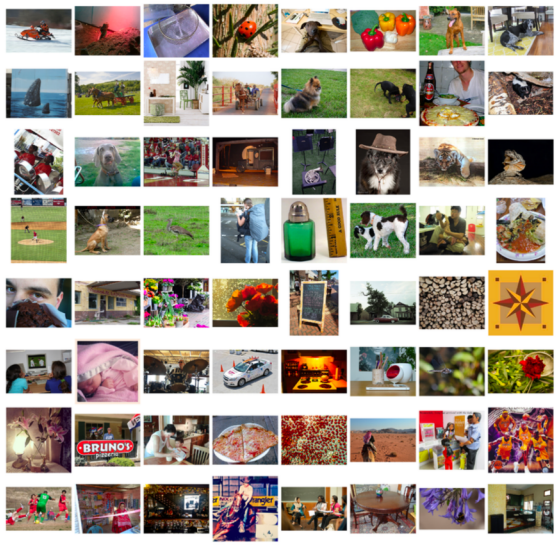
- *Advice for designing & building machine learning system*
- *Learning with large dataset*
- ***What can we do when learning with limited training data ?***
  - *Artificial data synthesis*
  - *Transfer learning*
- *Popular machine learning frameworks*

:: Transfer Learning can also help when working with a limited dataset

Transfer Learning not only speeds up training considerably, but also requires much less training data

Deep learning models for image recognition or NLP usually have millions of parameters. Training them from scratch requires a lot of labeled training data and a lot of computing power (hundreds of GPU-hours or more)[1]. Transfer learning is a technique that shortcuts much of this by taking a piece of a model that has already been trained on a related task and reusing it in a new model[3].

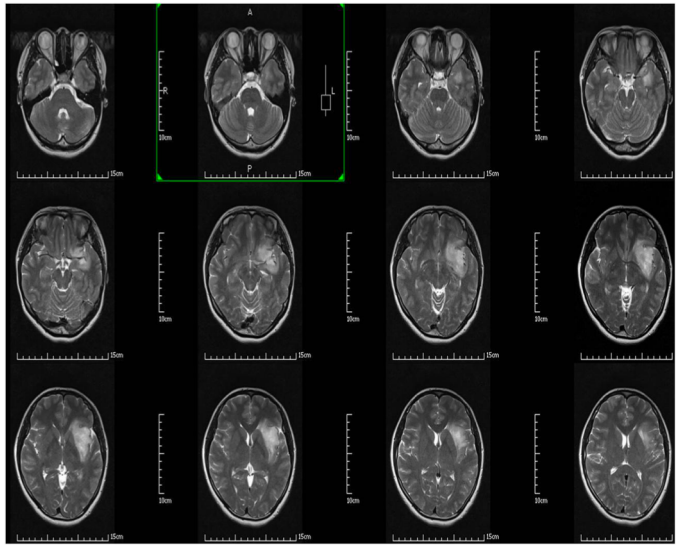
Sample of Images from the ImageNet Dataset



Over millions of images with annotations, e.g. images and their descriptions are available in ImageNet. It's easy to get such kind of data.

Transfer learning ?

Medical Image sample



Training Deep Neural Network needs lots of data but many real-world problems typically do not have millions of labeled data points to train such complex models. For example, it's hard to get enough labelled medical image for training a deep neural network for diagnosis.

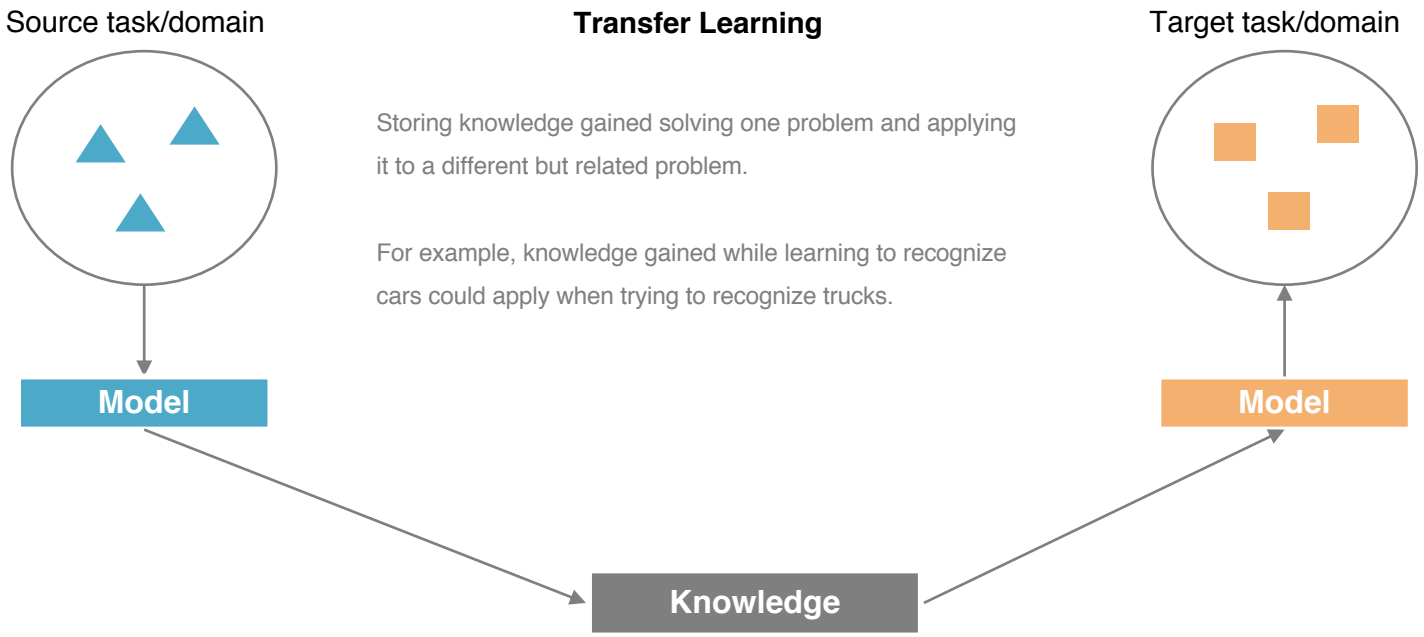
1 souree: https://www.tensorflow.org/hub/tutorials/image_retraining
2 source: https://towardsdatascience.com/transfer-learning-in-tensorflow-9e4f7eae3bb4

:: What's Transfer Learning ?

Transfer Learning can reuse a pre-trained model on a different but related problem

Transfer learning and domain adaptation refer to the situation where what has been learned in one setting … is exploited to improve generalization in another setting[1].

In other words, Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task[2].



It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks. For example, maybe you could have the neural network learn to recognize objects like cats and then use that knowledge or use part of that knowledge to help you do a better job in reading x-ray images.

Source of figure : http://ruder.io/transfer-learning/
1. Ian Goodfellow , et al, Deep Learning
2. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

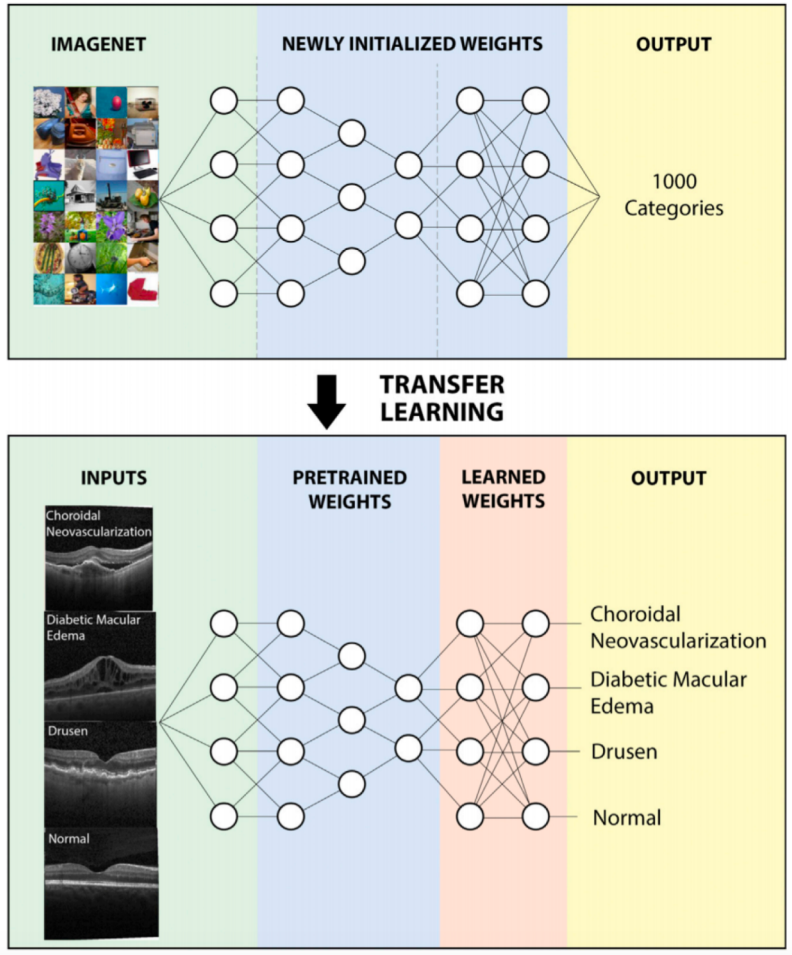
:: Application example of Transfer Learning

Example: apply transfer learning to medical image analysis

In the papers by Zhang et. al. and Burlina et. al . researchers demonstrated the application of transfer learning to detect eye diseases in medical images, with accuracy comparable to human experts.[1]

Source task/domain:
learning from ImageNet, which is the largest image dataset (14 million images over 22,000 categories of everyday objects)

Target task:
To detect specific eye diseases in optical coherence tomography (OCT) images of the retina.



Transfer Learning

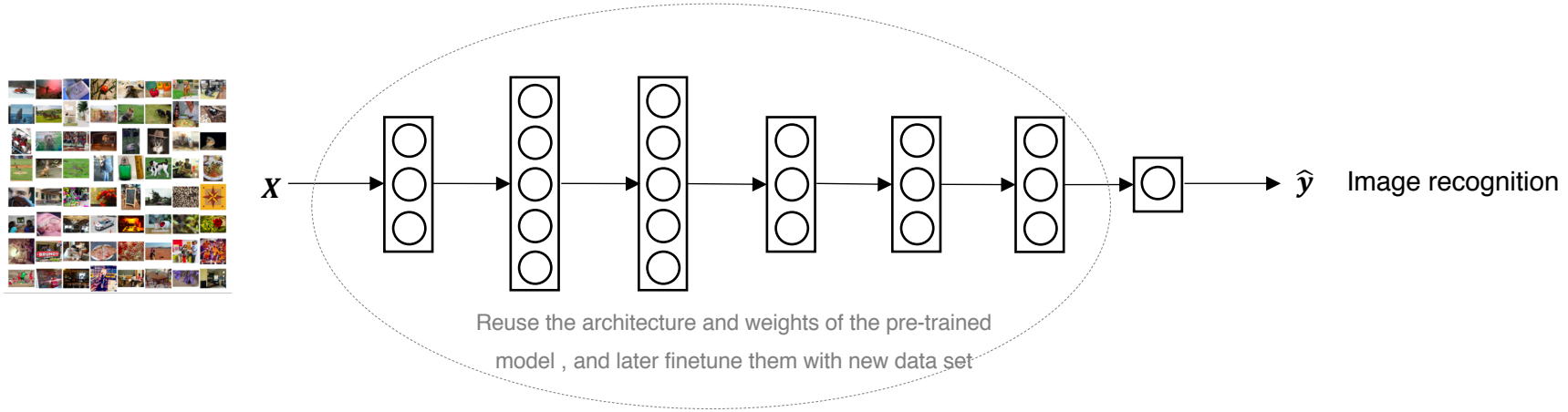
- Using a pre-trained net as a feature extractor [2]
- Fine-tune the “New” net by continuing the training on the new data [3]

1 source: https://software.intel.com/en-us/articles/use-transfer-learning-for-efficient-deep-learning-training-on-intel-xeon-processors
2,3 and figure: , Fabian Brand, dentifying Medical Diagnoses and Treatabe Diseases by Image-Based Deep Learning

:: How to use transfer learning ?

Select a pre-trained model

Suppose you take a neural network and train it on (x, y) pairs, where X is an image and Y is some object. An image is a cat or a dog or a bird or something else in large dataset like ImageNet. Later you can reuse that model and fine-tune it to analyze the medical images for diagnosing disease. Although you can train such model from scratch by yourself, it's common to reuse the pre-trained models from many research institutions.



Pre-trained models

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size.[1] You often make much faster progress if you download ways that someone else has already trained on. the network architecture and use that as pre-training. and transfer that to a new task that you might be interested in.[2]

There are a some famous **pre-trained models** such as Microsoft ResNet Model, Google Inception Model, Google's word2vec Model, Caffe Model Zoo, Oxford VGG Model, etc. They're trained on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories) and the trainings takes weeks/months.

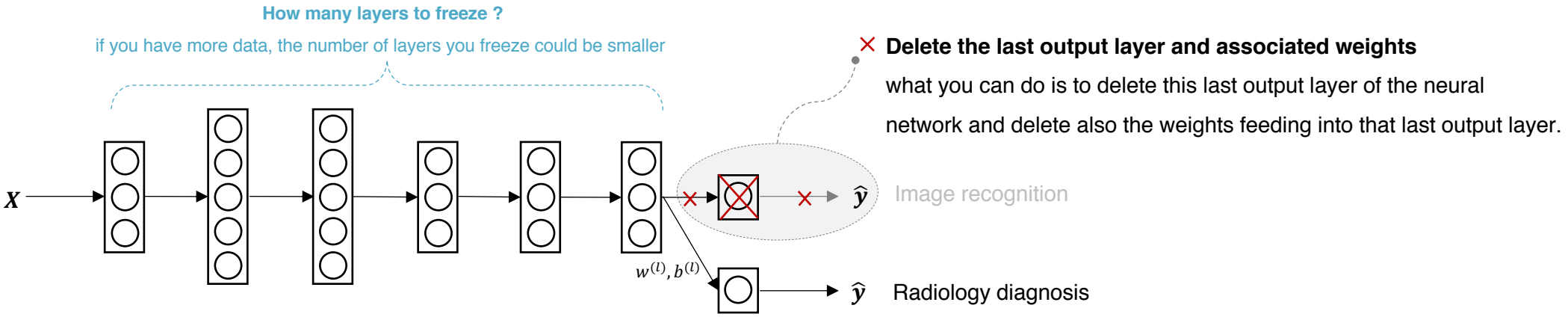
You can use that model as good starting point, and use transfer learning to transfer knowledge from some of these very large public data sets to your own problem.

1 source: http://cs231n.github.io/transfer-learning/
2 soure: : https://www.coursera.org/lecture/convolutional-neural-networks/transfer-learning-4THzO

:: How transfer learning works *-continued*

Retain the neural network on new data set like x-ray images

You can take out the last output layer of pre-trained network, freeze some of layers (keep the parameters fixed), and finetune the weights of the unfrozen layers with new data set.



You can have different approaches to retrain the neural network, depending on the size of new dataset. It is possible to fine-tune all the layers of the network, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network.

- **If you have a small radiology dataset:**
  Randomly initialize weights just for the last layer , and then retain the network on new radiology dataset. In other words, just retrain the weights of the last layer, $w^{(l)}$, $b^{(l)}$, and freeze the parameters of rest layers (the weights are not updated)

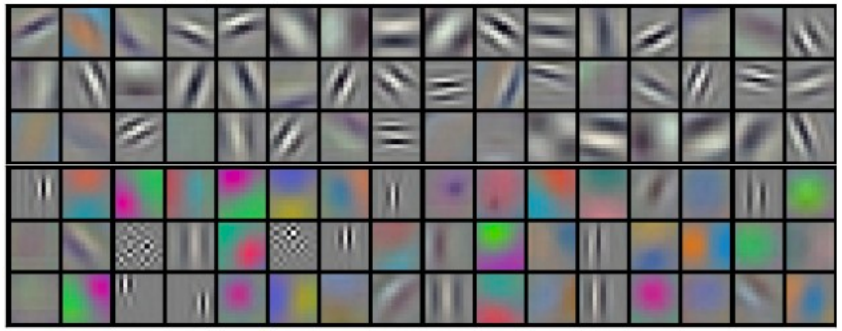
- **If you have enough radiology data:**
  You can just use the open source pre-trained network's weights as initialization, then retrain all the layers of the whole neural network. In other words, You can retrain all the parameters in the network.

:: The magic of transfer learning's success

Transfer learning is more successful in computer vision than other areas

The magic of transfer learning is that lower convolutional layers contain more generic features (e.g. edge detectors or color blob detectors) that can be reused for many recognition tasks , but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset[1].

When you look at what these Deep Learning networks learn, they try to detect edges in the earlier layers, Shapes in the middle layer and some high level data specific features in the later layers. These trained networks are generally helpful in solving other computer vision problems.[2]



lower convolutional layers capture low-level image features, e.g. edges (see Figure on the left), while higher convolutional layers capture more and more complex details, such as body parts, faces, and other compositional features.[3]

1 source: http://cs231n.github.io/transfer-learning/
2 source: https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8
3soure: http://ruder.io/transfer-learning/index.html#usingpretrainedcnnfeatures

:: When you should use transfer learning

When does transfer learning make sense ?

So to summarize, transfer learning has been most useful if you're trying to do well on some Task B, usually a problem where you have relatively little data.

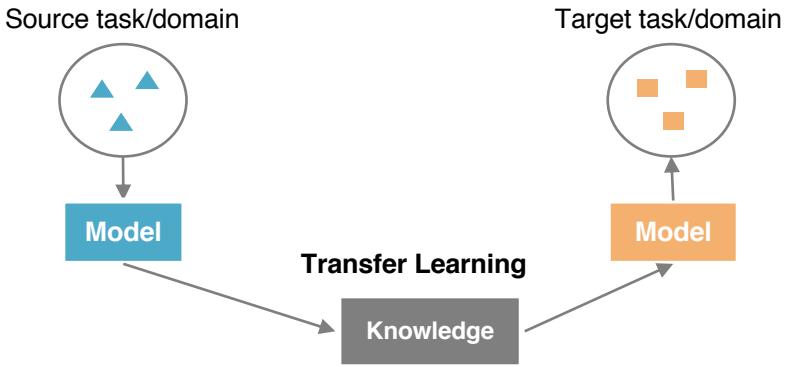
- **Task A and B have the same input x .**
    - For example, A and B both have images as input in image recognition.
    - For example, In speech recognition case, both A and B have audio clips as input.

- **You have a lot more data for Task A than Task B**

  for example, let's say you have a million examples for image recognition task. But for the radiology task, maybe you have only a hundred examples. So a lot of knowledge you learn from image recognition can be transferred and can really help you get going with radiology recognition even if you don't have all the data for radiology.

- **Low level features form A could be helpful for learning B**

  For example, in radiology, you know it's difficult to get that many x-ray scans to build a good radiology diagnosis system. So in that case, you might find a related but different task, such as image recognition, where you can get maybe a million images and learn a lot of low level features from that, so that you can then try to do well on Task B on your radiology task despite not having that much data for it.



:: The last but not the least

- *Advice for designing & building machine learning system*
- *Learning with large dataset*
- *What can we do when learning with limited training data ?*
- ***Popular machine learning frameworks***

:: Popular machine learning frameworks

TensorFlow might be the most popular framework

There are many machine learning/deep learning frameworks. The introduction of them is beyond the scope of this tutorial.

CNTK

:: Other Resources

Machine Learning Data Sets

- Links to many ML data repositories
- UCI Machine Learning Repository - Univ of California Irvine
- Kaggle: Machine Learning and data mining activities
- COCO-Text: Dataset for Text Detection and Recognition

Done !

Table of contents

Picture : Designed by Asier_Relampagoestudio / Freepik

:: More notes will be available in the future

More content would probably be added in the future

- Anomaly Detection
- Recommender System
- EM
- SVM & Kernels
- CNN
- RNN
- LSTM
- …

Any feedbacks are appreciated. Please reach me by dozing_runner@msn.com

:: About me

Jim Liang // 梁 劲

Product Manager & UX Designer

dozing_runner@msn.com

:: 感谢

感谢素不相识的网友们的赞赏，真的感谢你们 ！

截止到8月1日，收到以下网友的赞赏：

刘小龙、李云峰、Lilian、Leo、Vincent、王彬、马越、苏江文、段英杰、王永志、Blueve、Felix、J、Lei、何春瑜、蒲军、Thomas 肖、闻兴、Levimin、Irene、张萌萌、章炎、师傅快给我找钱啊、曹胜东、智帆、风信子、南辰、高明、黄野、Avon、sklearn、施晓林、酱、mrgord、冯祥卫、DraDroDo、薛毅华Kenny、Marcus Chen、东东、马勇（Daniel)、Hong、蔚蓝、Steve、Felix、赵忠印、黄哲Chris、温雨金、袁露露、李晟

还有一些没有留名的朋友

赞赏的朋友，请记得留下名字或昵称

如果觉得这份文档写的优秀，可以微信扫一扫赞赏支持一下

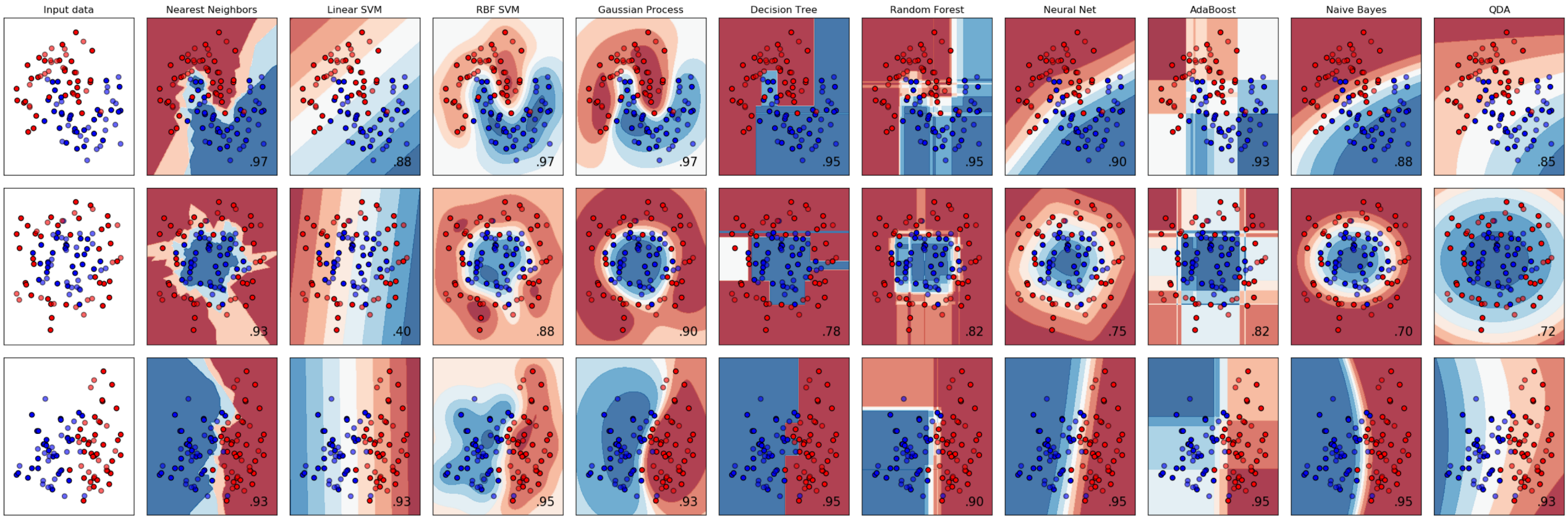
本文档不定期更新，最新版本可在百度云下载
https://pan.baidu.com/s/1tNXYQNadAsDGfPvuuj7_Tw

Appendix

:: Classifier comparison

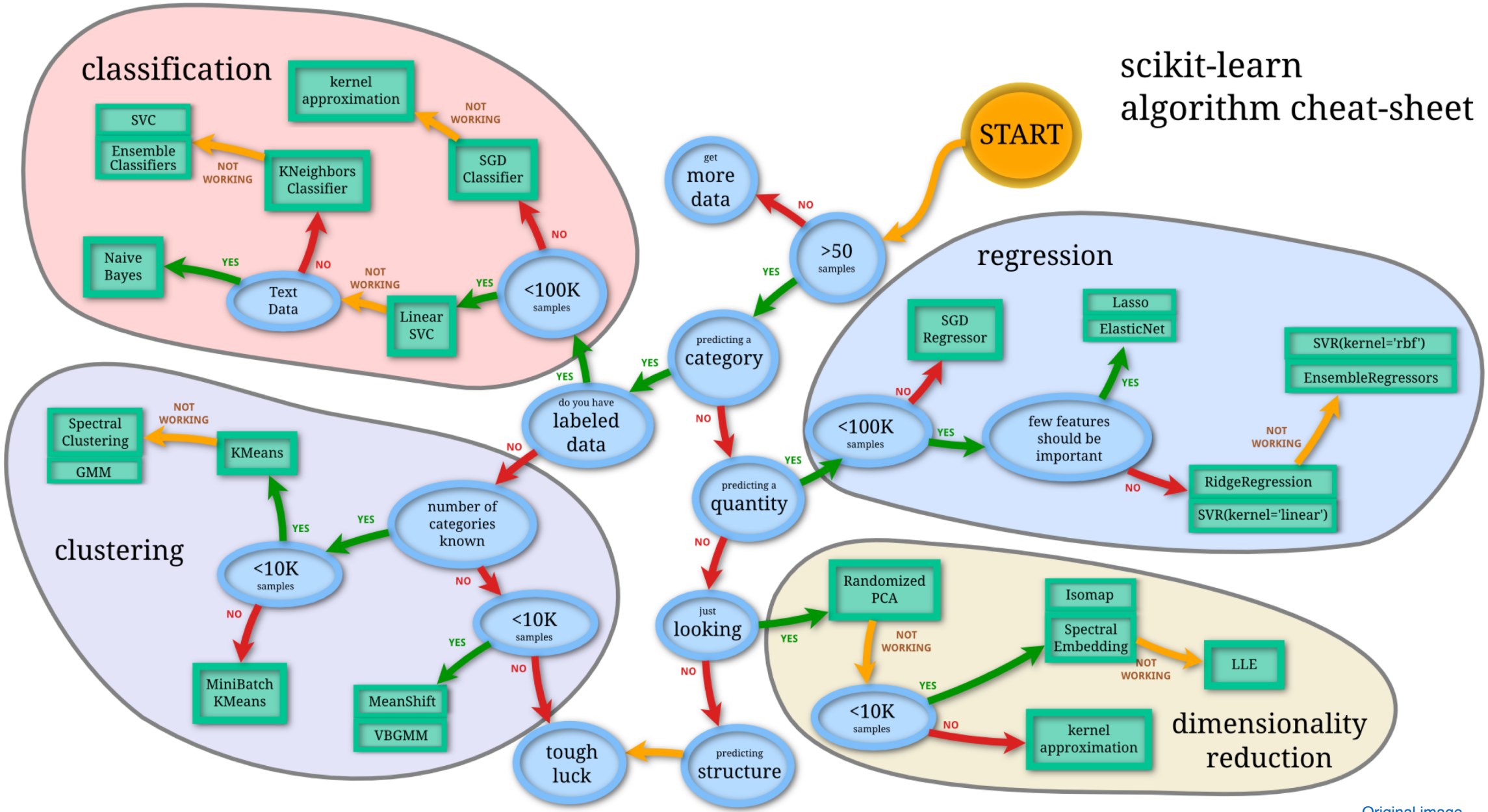
A comparison of a several classifiers in scikit-learn on synthetic datasets

The plots show training points in solid colors and testing points semi-transparent. The lower right shows the classification accuracy on the test set.



The point of this example is to illustrate the nature of decision boundaries of different classifiers. This should be taken with a grain of salt, as the intuition conveyed by these examples does not necessarily carry over to real datasets.

Source: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py



Original image

Often the hardest part of solving a machine learning problem can be finding the right estimator for the job. Different estimators are better suited for different types of data and different problems. The flowchart below is designed to give users a bit of a rough guide on how to approach problems with regard to which estimators to try on your data.

Source : http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Microsoft Azure Machine Learning: Algorithm Cheat Sheet

This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.

START

Finding unusual data points

Discovering structure

Predicting categories

Predicting values

Three or more

Two

ANOMALY DETECTION

- **One-class SVM** — >100 features, aggressive boundary
- **PCA-based anomaly detection** — Fast training

CLUSTERING

- **K-means**

MULTI-CLASS CLASSIFICATION

- Fast training, linear model — **Multiclass logistic regression**
- Accuracy, long training times — **Multiclass neural network**
- Accuracy, fast training — **Multiclass decision forest**
- Accuracy, small memory footprint — **Multiclass decision jungle**
- Depends on the two-class classifier, see notes below — **One-v-all multiclass**

REGRESSION

- **Ordinal regression** — Data in rank ordered categories
- **Poisson regression** — Predicting event counts
- **Fast forest quantile regression** — Predicting a distribution
- **Linear regression** — Fast training, linear model
- **Bayesian linear regression** — Linear model, small data sets
- **Neural network regression** — Accuracy, long training time
- **Decision forest regression** — Accuracy, fast training
- **Boosted decision tree regression** — Accuracy, fast training, large memory footprint

TWO-CLASS CLASSIFICATION

- **Two-class SVM** — >100 features, linear model
- **Two-class averaged perceptron** — Fast training, linear model
- **Two-class logistic regression** — Fast training, linear model
- **Two-class Bayes point machine** — Fast training, linear model

- Accuracy, fast training — **Two-class decision forest**
- Accuracy, fast training, large memory footprint — **Two-class boosted decision tree**
- Accuracy, small memory footprint — **Two-class decision jungle**
- >100 features — **Two-class locally deep SVM**
- Accuracy, long training times — **Two-class neural network**

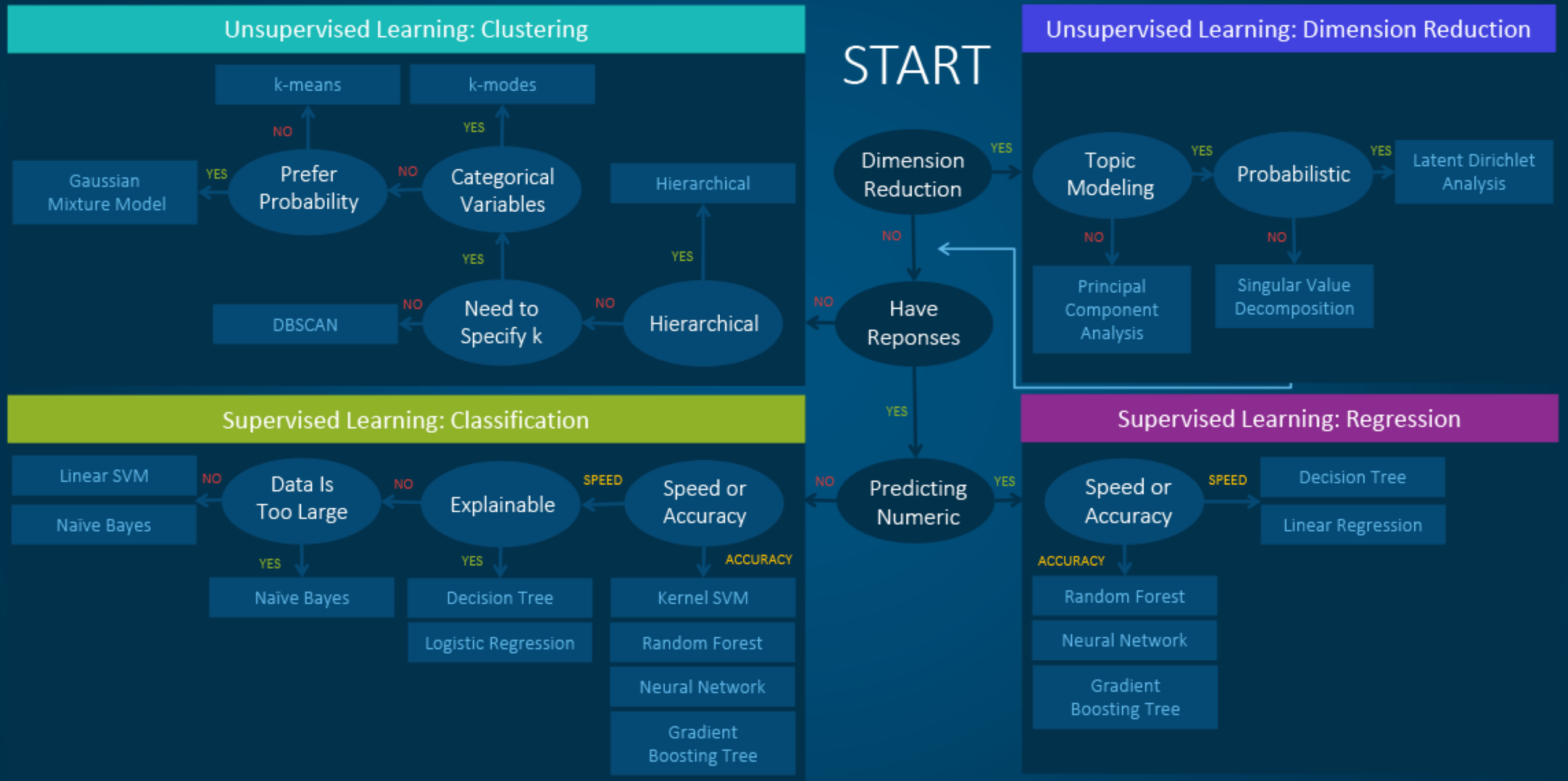
Created by the Azure Machine Learning Team

Email: AzurePoster@microsoft.com

Download this poster: http://aka.ms/MLCheatSheet

Microsoft

Machine Learning Algorithms Cheat Sheet



Azure Machine Learning Studio

Machine Learning in ML Studio

Anomaly Detection
- One-class Support Vector Machine
- Principal Component Analysis-based Anomaly Detection
- Time Series Anomaly Detection*

Classification

Two-class Classification
- Averaged Perceptron
- Bayes Point Machine
- Boosted Decision Tree
- Decision Forest
- Decision Jungle
- Logistic Regression
- Neural Network
- Support Vector Machine

Multi-class Classification
- Decision Forest
- Decision Jungle
- Logistic Regression
- Neural Network
- One-vs-all

Clustering
- K-means Clustering

Recommendation
- Matchbox Recommender

Regression
- Bayesian Linear Regression
- Boosted Decision Tree
- Decision Forest
- Fast Forest Quantile Regression
- Linear Regression
- Neural Network Regression
- Ordinal Regression
- Poisson Regression

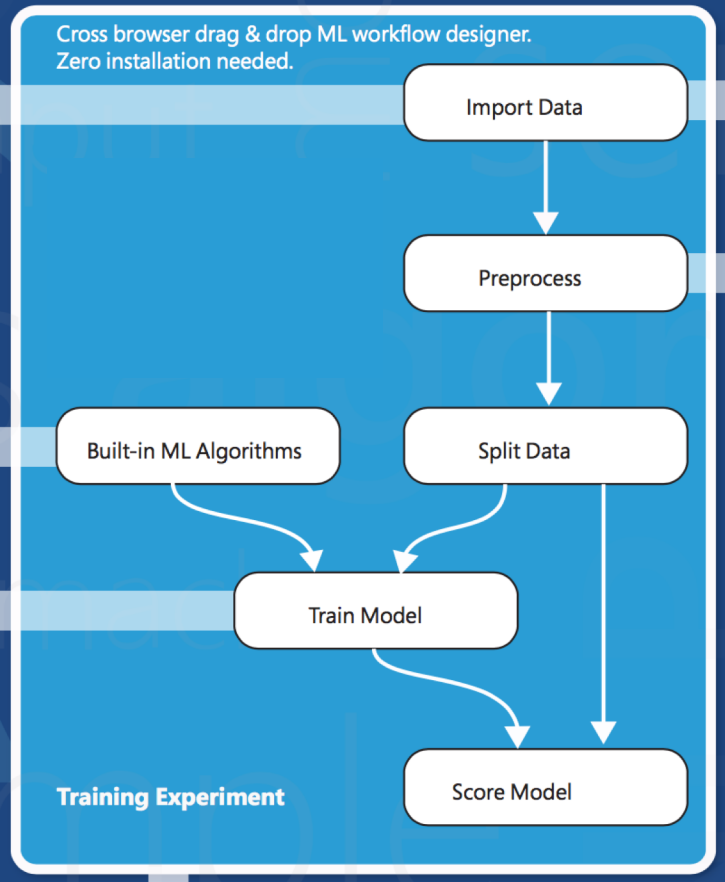
Statistical Functions
- Descriptive Statistics
- Hypothesis Testing T-Test
- Linear Correlation
- Probability Function Evaluation

Text Analytics
- Feature Hashing
- Named Entity Recognition
- Vowpal Wabbit

Computer Vision
- OpenCV Library

Data/Model Visualization
- Scatterplots
- Bar Charts
- Box plots
- Histogram
- R and Python Plotting Libraries
- REPL with Jupyter Notebook
- ROC, Precision/Recall, Lift
- Confusion Matrix
- Decision Tree*

Training
- Cross Validation
- Retraining
- Parameter Sweep



Data Source
- Azure Blob Storage
- Azure SQL DB
- Azure SQL DW*
- Azure Table
- Desktop Direct Upload
- Hadoop Hive Query
- Manual Data Entry
- OData Feed
- On-prem SQL Server*
- Web URL (HTTP)

Data Format
- ARFF
- CSV
- SVMLight
- TSV
- Excel
- ZIP

Data Preparation
- Clean Missing Data
- Clip Outliers
- Edit Metadata
- Feature Selection
- Filter
- Learning with Counts
- Normalize Data
- Partition and Sample
- Principal Component Analysis
- Quantize Data
- SQLite Transformation
- Synthetic Minority Oversampling Technique

One-click Operationalization

Predictive Experiment

Make Prediction with Elastic APIs
- Request-Response Service (RRS)
- Batch Execution Service (BES)
- Retraining API