732A54/TDDE31 Big Data Analytics

Lecture 8: Machine Learning with MapReduce

Mohammad Seidpisheh
IDA, Linköping University, Sweden

Contents

Why need to apply Machine Learning on MapReduce framework

- Main goal of machine learning is to gain knowledge form raw data
- To gain knowledge, it is needed to apply ML on huge amounts of data to generalize and predict with high accuracy
- For achieving this goal, we have challenges to tackle with huge amount of data (Big data)
- The enlarging volume of data makes machine learning algorithms being complicated and time-consuming
- Handling big data would require high-end computers, currently only available to high-end servers with steep prices.
- MapReduce is a good solution for applying machine learning algorithms on large datasets
- By using MapReduce framework, we can overcome challenges caused by running ML algorithm on big data

MapReduce Framework

- The challenges in facing with big data can be divided into two parts
  - Storage
  - Programming model
- Solutions for storage of big data: Google GFS, Hadoop HDFS
- Hadoop is an open-source software framework for storing data and running applications on cluster of commodity hardware.
- Solutions for Programming model: MapReduce, Spark, Kafka

MapReduce Framework

- Programming framework developed at Google to process large amounts of data by parallelizing computations across a cluster of computers (machines, nodes, systems).
- In fact, It is designed to use many machines instead of single machine.
- Easy to use, since the parallelization happens automatically.
- Easy to speed up by using/adding more computers to the cluster.
- Typical uses at Google:
  - Large-scale machine learning problems, e.g. clustering documents from Google News; to extract valuable information.
  - Extracting properties of web pages, e.g. web access log data; to find meaningful pattern related to number of visitor and visiting time.
  - Large-scale graph computations, e.g. web link graph; to understand dependency of different pages on a website.
  - Statistical machine translation; to generate translation by statistical methods.
  - Processing satellite images; te detect upcoming disaster.
  - Production of the indexing system used for Google's web search engine.

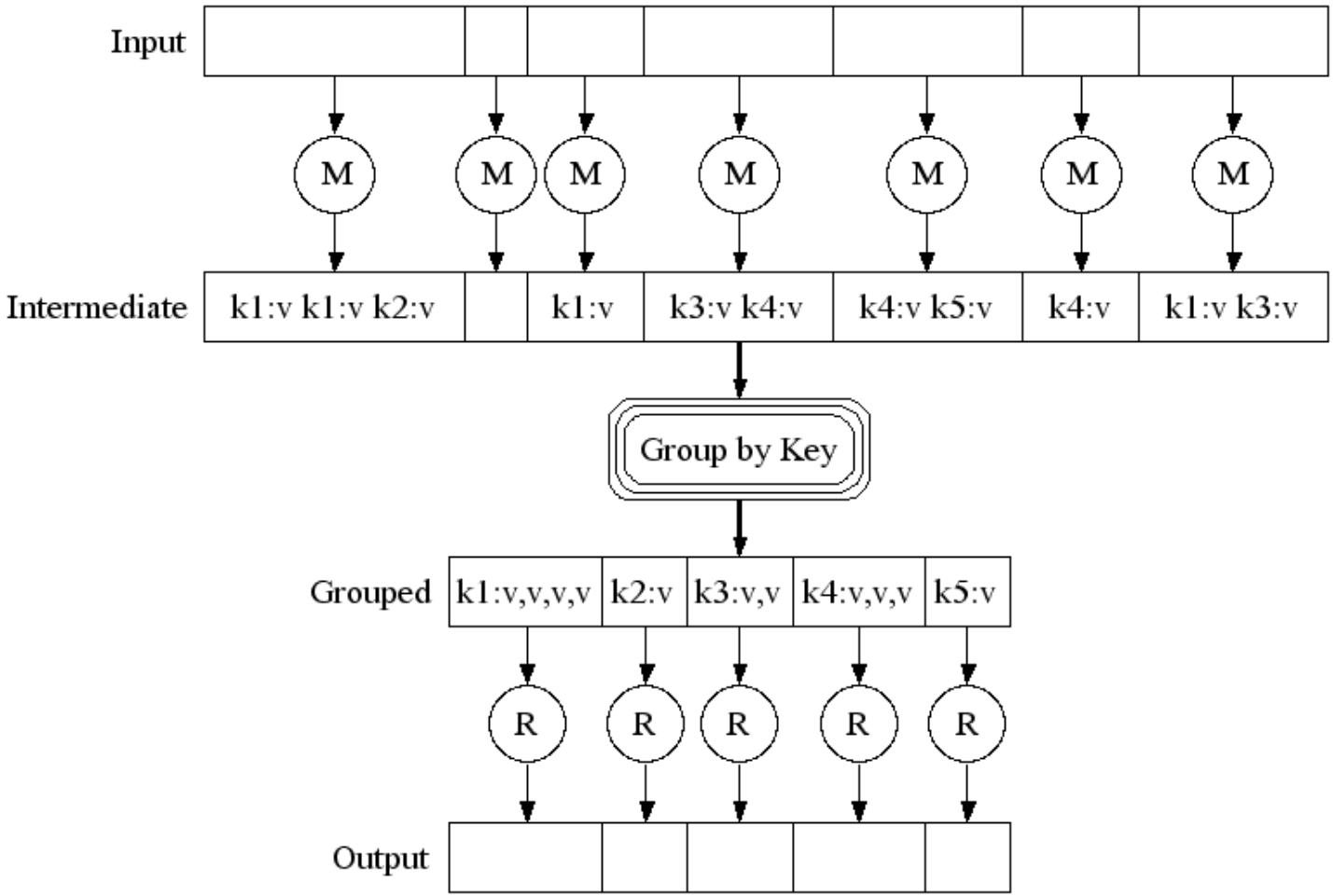
MapReduce Framework

- The user only has to implement the following two functions:
  - Map function:
    - Input: A pair (in_key, in_value).
    - Output: A list $list(intermediate\_key, intermediate\_value)$.
  - Reduce function:
    - Input: A pair $(out\_key, list(intermediate\_value))$.
    - Output: A list $list(out\_value)$.
- The right choices of keys and values depend on needs of application.
- Example for counting word occurrences in a collection of documents:

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```

MapReduce Framework



- We subdivide the map phase into M pieces and the reduce phase into R pieces, as described above. Ideally, M and R should be much larger than the number of worker machines.
- All intermediate values associated with the same intermediate key are grouped together before passing them to the reduce function.

General recommended procedures to adopt machine learning algorithms to a MapReduce framework

We can consider these points to adopt machine learning algorithms to a MapReduce framework

- Algorithms that calculate sufficient statistics or gradients can be adopted to a MapReduce.
- When an algorithm does sum over the data points, we can easily distribute the calculations over multiple machines, so it can be expressed in a MapReduce framework.
- We just divide the data set into as many pieces as there are machines, give each machine its share of the data to compute the equations over (Map task), and aggregate the results at the end (Reduce task).
- This adoption does not change the underlying algorithm and so is not an approximation, but it is instead an exact implementation.
- By taking advantage of the summation form in a mapReduce framework, we can parallelize computation in many machine learning algorithms and achieve speed-up.

Machine Learning with MapReduce: K-Means Algorithm

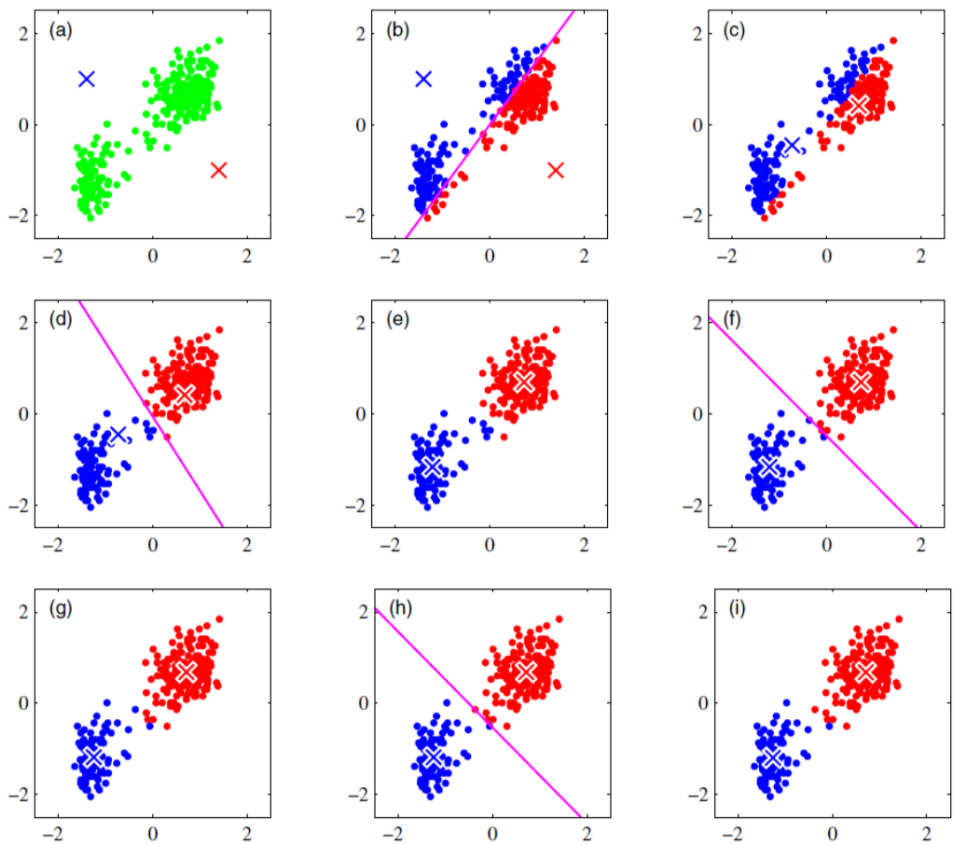
- Clustering is the most well-known unsupervised learning method that aims to divide data points into clusters in such way that each data point in a cluster are similar and different from the data points of other clusters.
- The K-Means algorithm is a well-known and commonly used clustering algorithms; due to its low time cost.
- However, even the use of K-Means in big data can lead to unacceptable time costs, since the time cost of a K-Means tends to grow as the amount of data grows.

Machine Learning with MapReduce: K-Means Algorithm

- The purpose of K-Means clustering is to minimize the following objective function:

$$J = \sum_{j=1}^{K} \sum_{\boldsymbol{x} \in C_j} d(\boldsymbol{x}, \boldsymbol{\mu}(C_j)),$$

1	Set K points as centroids at random
2	Assign each point to a cluster with the closest centroid
3	recalculate the cluster centroids as the averages of the points assigned to each cluster
	Repeat steps 2 and 3 until the centroids do not change



Adopting K-Means Algorithm to a MapReduce

- The purpose of K-Means clustering is to minimize the following objective function:

$$J = \sum_{j=1}^{K} \sum_{\boldsymbol{x} \in C_j} d(\boldsymbol{x}, \boldsymbol{\mu}(C_j)),$$

- In k-means , the operation of computing the Euclidean distance between the data points and the centroids can be parallelized by (by the mapper)
  - splitting the data into individual subgroups
  - and clustering data points in each subgroup separately.
- In recalculating new centroid (by the reducer),
  - divide the data points into M subgroups;
  - compute the sum of data points in each subgroup in parallel;
  - and finally the reducer will add up the partial sums and compute the new centroids.

Adopting K-Means Algorithm to a MapReduce

1	Set K points as centroids at random
2	Assign each point to a cluster with the closest centroid
3	recalculate the cluster centroids as the averages of the points assigned to each cluster
	Repeat steps 2 and 3 until the centroids do not change

- **Each iteration** of the K-means algorithm can easily be casted into MapReduce terms:
  - Map function: Assign a training point to the cluster with the closest centroid.
    K-meansMap(**x**):
    $$emit(argmin_i ||\boldsymbol{x} - \mu_i||_2^2, (\boldsymbol{x}, 1))$$
  - Reduce function: Recalculate the cluster centroids from the assignments of the map tasks.
    The Reduce phase is just a sum over the values for same cluster(key):
    K-meansReduce((i, (**x**,1)), (i, (**y**,1)),(i, (**z**,1),...),...): i=1,2,..,K
    $$return(i, \boldsymbol{x} + \boldsymbol{y} + \boldsymbol{z} + ...)$$
- Note that $1 \leq M \leq N$, whereas $R = 1$ or $\boldsymbol{R = K}$ depending on whether we decide to use the cluster assignment as intermediate key or not.
- What is the key and what is the value ? What needs to be broadcasted ?

Machine Learning with MapReduce: EM Algorithm

- Model-based clustering aims to assign data points to the subpopulations by applying Bayes theorem as follows:

$$p(k|\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\pi}) = \frac{\pi_k p(\boldsymbol{x}|\boldsymbol{\theta}_k)}{\sum_k \pi_k p(\boldsymbol{x}|\boldsymbol{\theta}_k)}$$

  where $p(\boldsymbol{x}|\boldsymbol{\theta}_k)$ are called mixture components, and $\pi_k = p(k)$ are called mixing coefficients.

- More specifically, for components modeled as multivariate Gaussian distributions, we have that:

$$p(\boldsymbol{x}|\boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) = \frac{1}{2\pi^{D/2}} \frac{1}{|\boldsymbol{\Sigma}_k|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}.$$

- To solve model-based clustering, we have to estimate the model parameters $(\boldsymbol{\theta},\boldsymbol{\pi})$ from data. To this end, we use the EM algorithm.

Machine Learning with MapReduce: EM Algorithm

- Given a sample $\{\boldsymbol{x}_n\}$ of size N from a mixture of multivariate Gaussian distributions, the expected log likelihood function is maximized when

$$\begin{aligned}
\pi_k^{ML} &= \frac{\sum_n \gamma_{nk}}{N} \\
\boldsymbol{\mu}_k^{ML} &= \frac{\sum_n \boldsymbol{x}_n \gamma_{nk}}{\sum_n \gamma_{nk}} \\
\boldsymbol{\Sigma}_k^{ML} &= \frac{\sum_n (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})^T \gamma_{nk}}{\sum_n \gamma_{nk}}
\end{aligned}$$

  where $\boldsymbol{z}_n$ is a K-dimensional binary vector indicating component memberships (one-hot encoding):

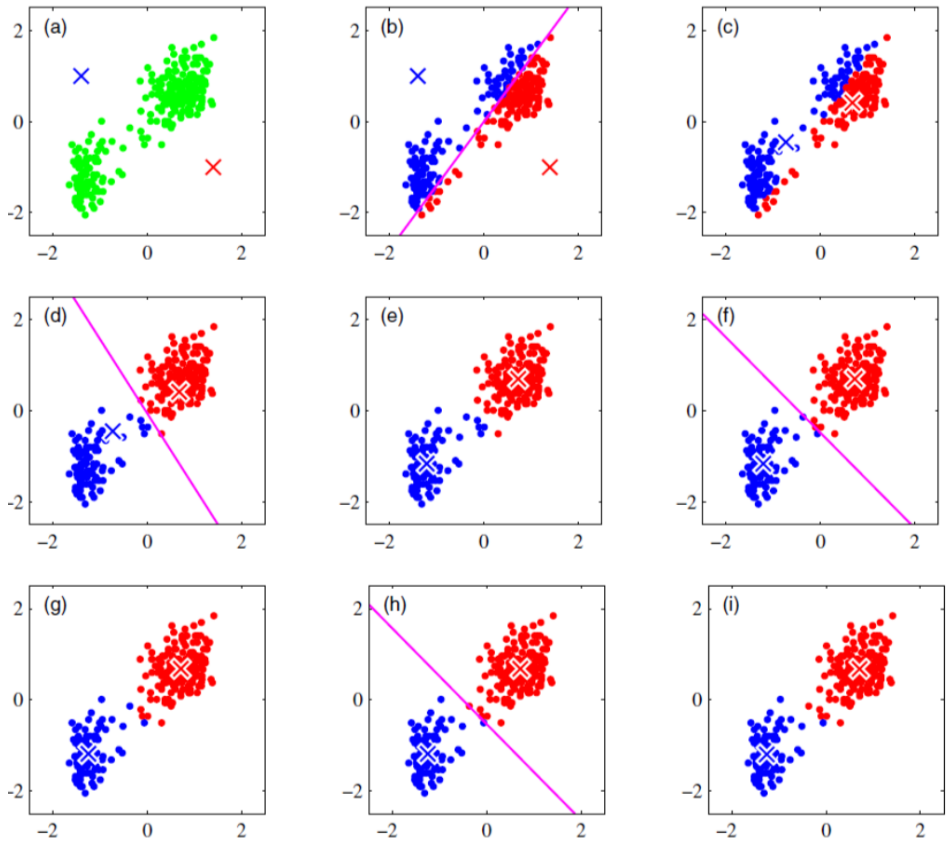
$$\gamma_{nk} = p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{p(\boldsymbol{x}_n|z_{nk}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z_{nk}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\sum_k p(\boldsymbol{x}_n|z_{nk}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z_{nk}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})} = \frac{\pi_k p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_k \pi_k p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- This is not a closed form solution, but it suggests the following algorithm.

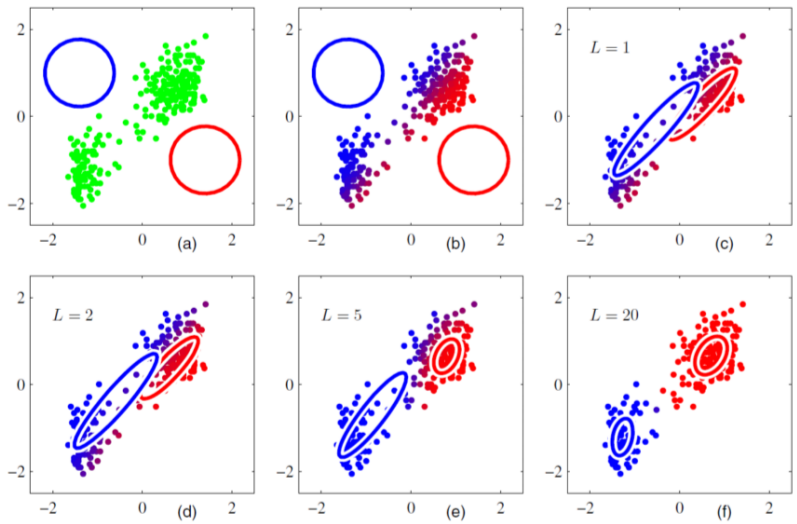
EM algorithm

Set $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to some initial values
Repeat until $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ do not change
    Compute γ_{nk} for all n and k /* E step */
    Set π_k to π_k^{ML}, $\boldsymbol{\mu}_k$ to $\boldsymbol{\mu}_k^{ML}$, and $\boldsymbol{\Sigma}_k$ to $\boldsymbol{\Sigma}_k^{ML}$ for all k /* M step */

Machine Learning with MapReduce: EM Algorithm



K-means algorithm



EM algorithm

Machine Learning with MapReduce: EM Algorithm

Adopting EM Algorithm to a MapReduce, user has to illustrate how EM algorithm can be expressed in summation form.

- **Each iteration** of the EM algorithm can easily be casted into **two chained** MapReduce jobs:
  - Map function I: For the n-th training point, compute

$$\gamma_{nk} \tag{1}$$

    and

$$\boldsymbol{x}_n \gamma_{nk}. \tag{2}$$

  - Reduce function I: Sum up the results (1) of the map tasks and divide it by N. Sum up the results (2) of the map tasks and divide it by the sum of the results (1). This gives $\boldsymbol{\pi}_k^{ML}$ and $\boldsymbol{\mu}_k^{ML}$.
  - Map function II: For the n-th training point, compute

$$\gamma_{nk} \tag{3}$$

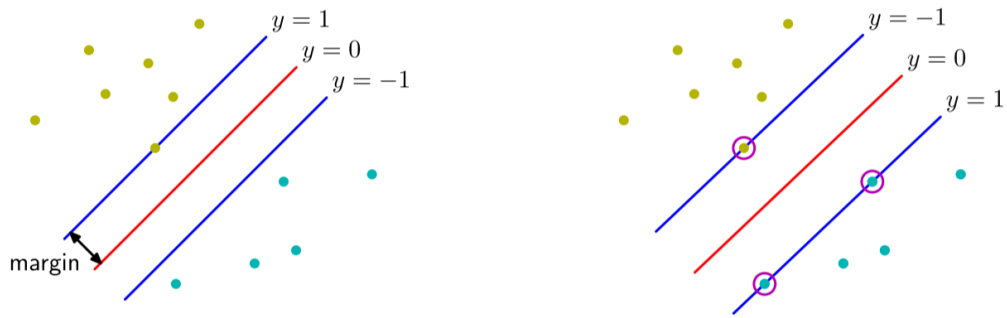
    and

$$(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})^T \gamma_{nk}. \tag{4}$$

  - Reduce function II: Sum up the results (4) of the map tasks and divide it by the sum of the results (3). This gives $\boldsymbol{\Sigma}_k^{ML}$.
- Note that $1 \leq M \leq N$, whereas $R = 1$ or $\boldsymbol{R = K}$ in both jobs, depending on whether we decide to use the component index as intermediate key or not. What is the key and what is the value ? What needs to be broadcasted ?

Machine Learning with MapReduce: Linear Support Vector Machines

- The objective of the support vector machine algorithm is to find a hyperplane that distinctly classifies the data points.



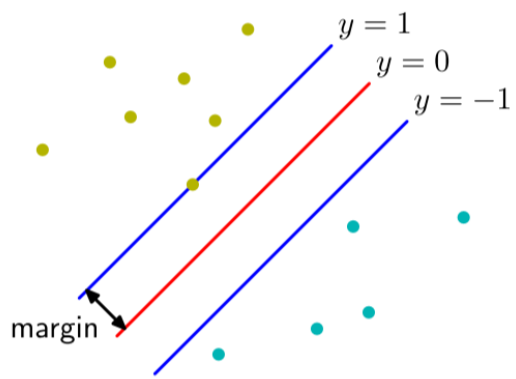
- Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.
- To split data points, there are many possible hyperplanes that could be selected. The objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of classes.
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.
- The motivation is that the larger the margin, the smaller the generalization error.

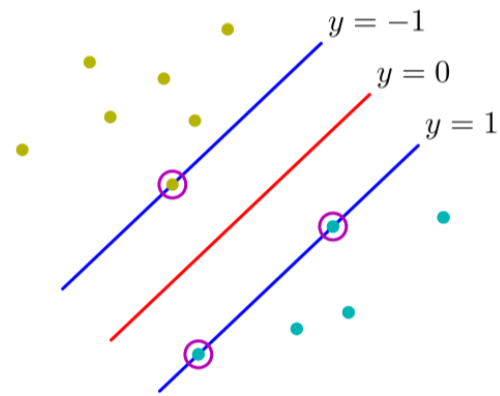
Machine Learning with MapReduce: Linear Support Vector Machines

- Consider binary classification with input space $\mathbb{R}^D$. Consider a training set $\{(\boldsymbol{x}_n, t_n)\}$ where $t_n \in \{-1, +1\}$. Consider using the linear model

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

so that a new point $\boldsymbol{x}$ is classified according to the sign of $y(\boldsymbol{x})$.

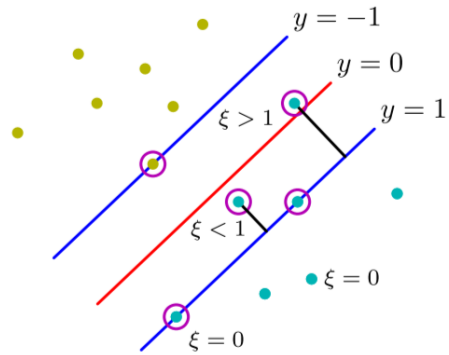




Machine Learning with MapReduce: Linear Support Vector Machines

- Without the assumption of linearly separability and with a quadratic penalty for (almost-)misclassified points, the optimal separating hyperplane is given by

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n\in E}(\underbrace{\boldsymbol{w}^T\boldsymbol{x}_n - t_n}_{\xi_n})^2$$

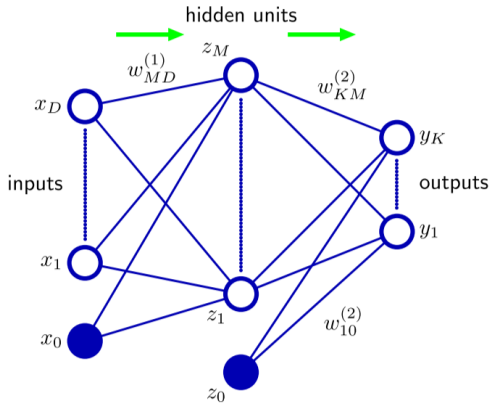


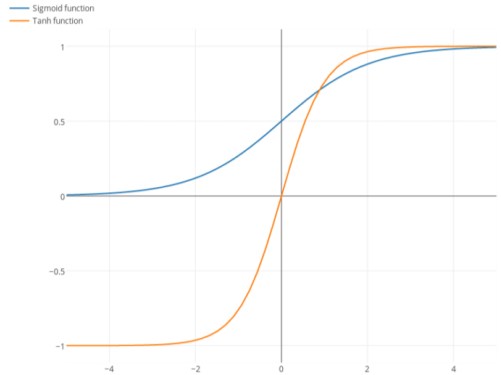
  where C is a user-defined parameter.
- Note that the previous expression is a quadratic function and, thus, it is concave (up) and, thus, "easy" to minimize. For instance, we can use again batch gradient descent.
- The gradient is given by

$$\boldsymbol{w} + 2C\sum_{n\in E}(\boldsymbol{w}^T\boldsymbol{x}_n - t_n)\boldsymbol{x}_n$$

- **Each iteration** of batch gradient descent can easily be casted into MapReduce terms:
  - Map function: Compute the gradient for a training point.
  - Reduce function: Sum the partial gradients and update $\boldsymbol{w}$ accordingly.
- Note that $1 \leq M \leq N$, whereas $R = 1$. What is the key and what is the value ? What needs to be broadcasted ?

Machine Learning with MapReduce: Neural Networks





- Activations: $a_j = \sum_i w^{(1)}_{ji} x_i + w^{(1)}_{j0}$
- Hidden units and activation function: $z_j = h(a_j)$
- Output activations: $a_k = \sum_j w^{(2)}_{kj} z_j + w^{(2)}_{k0}$
- Output activation function for regression: $y_k(\mathrm{x}) = a_k$
- Output activation function for classification: $y_k(\mathrm{x}) = \sigma(a_k)$
- Sigmoid function: $\sigma(a) = \frac{1}{1+\exp(-a)}$
- Two-layer NN:

$$y_k(\mathrm{x}) = \sigma\left(\sum_j w^{(2)}_{kj} h\left(\sum_i w^{(1)}_{ji} x_i + w^{(1)}_{j0}\right) + w^{(2)}_{k0}\right)$$

- Evaluating the previous expression is known as forward propagation. The NN is said to have a feed-forward architecture.
- All the previous is, of course, generalizable to more layers.

Machine Learning with MapReduce: Neural Networks

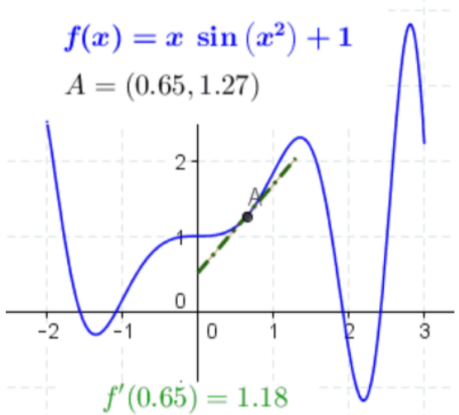
- Consider regressing an K-dimensional continuous random variable on a D-dimensional continuous random variable.
- Consider a training set $\{(\boldsymbol{x}_n, \boldsymbol{t}_n)\}$ of size N. Consider minimizing the error function

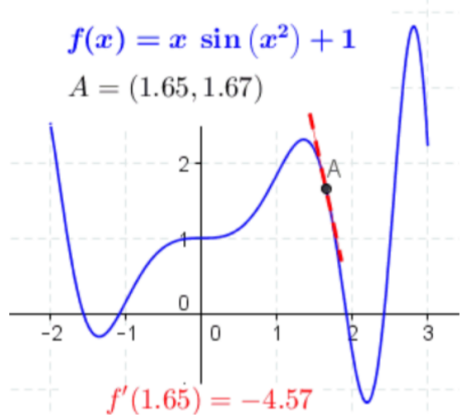
$$E(\boldsymbol{w}) = \sum_n E_n(\boldsymbol{w}) = \sum_n \frac{1}{2}(\boldsymbol{y}(\boldsymbol{x}_n) - \boldsymbol{t}_n)^2 = \sum_n \sum_k \frac{1}{2}(y_k(\boldsymbol{x}_n) - t_{nk})^2$$

- The weight space is highly multimodal and, thus, we have to resort to approximate iterative methods to minimize the previous expression.
- Batch gradient descent

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta \nabla E(\boldsymbol{w}^t) = \boldsymbol{w}^t - \eta \sum_n \nabla E_n(\boldsymbol{w}^t)$$

where $\eta > 0$ is the learning rate, and $\nabla E_n(\boldsymbol{w}^t)$ can be computed efficiently thanks to the backpropagation algorithm.





Machine Learning with MapReduce: Neural Networks

- **Each iteration** of batch gradient descent can easily be casted into MapReduce terms:
  - Map function: Compute the gradient for a training point. Note that this implies forward and backward propagation.
  - Reduce function: Sum the partial gradients and update $\boldsymbol{w}$ accordingly.
- Note that $1 \leq M \leq N$, whereas $R = 1$.
- What is the key and what is the value ? What needs to be broadcasted ?

Machine Learning with MapReduce

- In the experiments to compare the speed up on data sets, each algorithm had two different versions: One applying ML with MapReduce, and the other without it.
- Value more than one in the table indicates the speedup of machine learning algorithms running on Map reduce and one shows speed of both version are similar.
- The Table shows the speedup on over all the algorithms on all the data sets.

	lwlr	gda	nb	logistic	pca	ica	svm	nn	kmeans	em
Adult	1.922	1.801	1.844	1.962	1.809	1.857	1.643	1.825	1.947	1.854
Helicopter	1.93	2.155	1.924	1.92	1.791	1.856	1.744	1.847	1.857	1.86
Corel Image	1.96	1.876	2.002	1.929	1.97	1.936	1.754	2.018	1.921	1.832
IPUMS	1.963	2.23	1.965	1.938	1.965	2.025	1.799	1.974	1.957	1.984
Synthetic	1.909	1.964	1.972	1.92	1.842	1.907	1.76	1.902	1.888	1.804
Census Income	1.975	2.179	1.967	1.941	2.019	1.941	1.88	1.896	1.961	1.99
Sensor	1.927	1.853	2.01	1.913	1.955	1.893	1.803	1.914	1.953	1.949
KDD	1.969	2.216	1.848	1.927	2.012	1.998	1.946	1.899	1.973	1.979
Cover Type	1.961	2.232	1.951	1.935	2.007	2.029	1.906	1.887	1.963	1.991
Census	2.327	2.292	2.008	1.906	1.997	2.001	1.959	1.883	1.946	1.977
avg.	1.985	2.080	1.950	1.930	1.937	1.944	1.819	1.905	1.937	1.922

Table 3: Speedups achieved on a dual core processor, without load time. Numbers reported are dual-core time / single-core time. Super linear speedup sometimes occurs due to a reduction in processor idle time with multiple threads.

Machine Learning with MapReduce

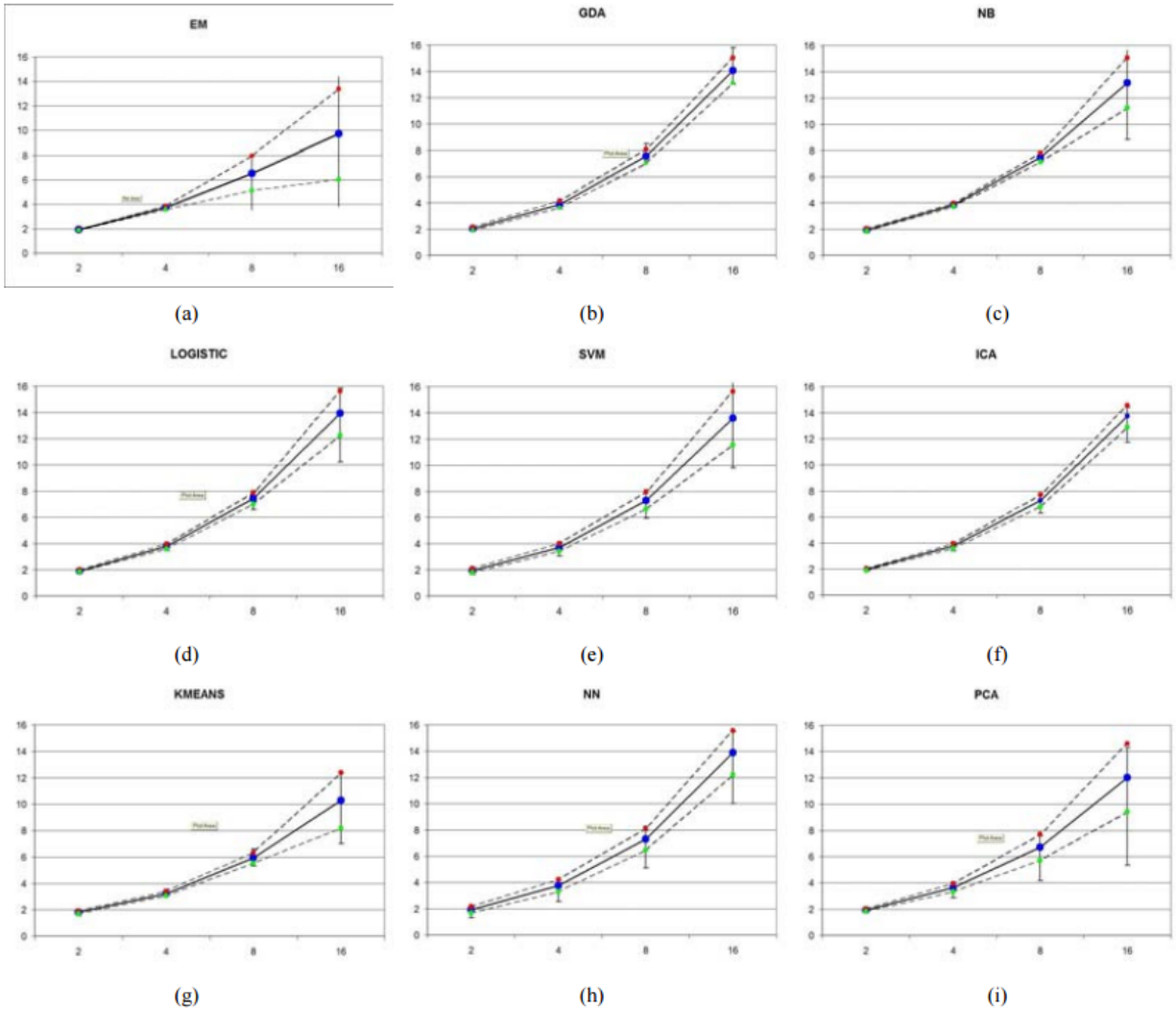


Figure 2: (a)-(i) show the speedup from 1 to 16 processors of all the algorithms over all the data sets. The Bold line is the average, error bars are the max and min speedups and the dashed lines are the variance.

Summary

- MapReduce is a framework to process large datasets by parallelizing computations.
- The user only has to specify the map and reduce functions, and parallelization happens automatically.
- Many machine learning algorithms (e.g. SVMs, NNs, MMs, K-means and EM algorithms) can easily be reformulated in terms of such functions.
- Google replaced it with Cloud Dataflow/Dataproc/Platform, since it could not process the amount of data they produced.
- However, it is still the processing core of Apache Hadoop, another framework for distributed storage and distributed processing of large datasets on computer clusters.
- Moreover, it is a straightforward way to adapt some machine learning algorithms to cope with big data.

Summary

- Apache Mahout is a project to produce distributed implementations of machine learning algorithms. It builds on Hadoop's MapReduce. However, these implementations are now deprecated, in favor of Apache Spark.
- This does not apply for algorithms based on stochastic gradient descent.
- Moreover, MapReduce is **inefficient** for iterative tasks on the same dataset: Each iteration is a MapReduce call that loads the data anew from disk.
- Such iterative tasks are common in many machine learning algorithms, e.g. gradient descent, K-means and EM algorithms.
- Solution: Spark framework, in the next lecture.

Literature

- Main sources
  - Dean, J. and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107-113, 2008.
  - Chu, C.-T. et al. Map-Reduce for Machine Learning on Multicore. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, 281-288, 2006.
  - Bodoia, Max. MapReduce Algorithms for k-means Clustering. Computer Science, (2016).
- Additional sources
  - Dean, J. and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, 2004.
  - Gillick, D., Faria, A. and DeNero, J. MapReduce: Distributed Computing for Machine Learning. Technical Report, Berkley, 2006.
  - Slides for 732A99/TDDE01 Machine Learning.

Thank Jose M Pena for sharing his valuable experience and materials.