10. Support vector machines, boosting and artificial neural networks

Shabana K M
PhD Research Scholar
Computer Science and Engineering

IIT Palakkad

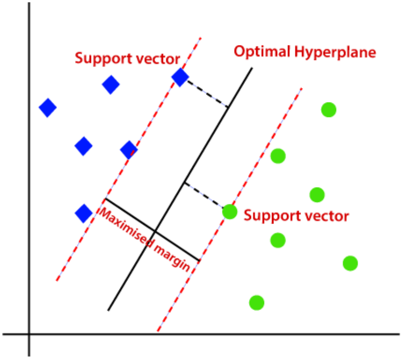
28 November 2021

Recap

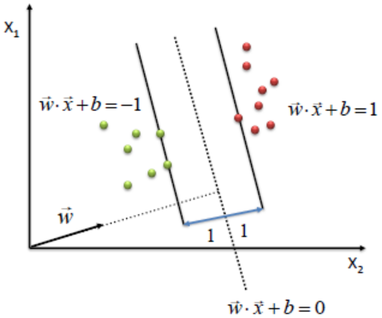
- Random forests
  - bagging, bootstrap sampling
  - learning a random forest model
- Perceptron
  - perceptron learning
  - limitations
- Support vector machine
  - margin, support vector
  - optimization problem

Support vector machine (SVM)

- maximizes the margin around the separating hyperplane
- decision function fully specified by the support vectors
  - data points that lie closest to the decision surface



Hard margin SVM



$$\boxed{\max \frac{2}{\|w\|}}$$

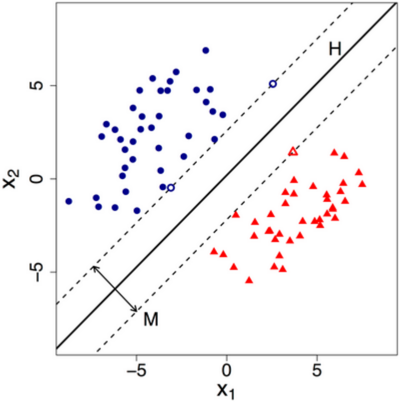
s.t.

$(w\cdot x+b)\geq 1, \forall x \text{ of class } 1$

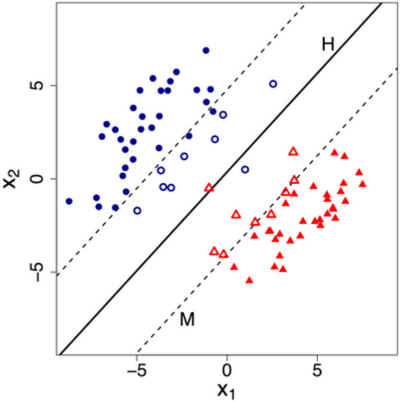
$(w\cdot x+b)\leq -1, \forall x \text{ of class } 2$

- each data point must lie on the correct side of the margin
- doesn't allow any misclassifications
- works only if data is linearly separable

Hard vs Soft SVM

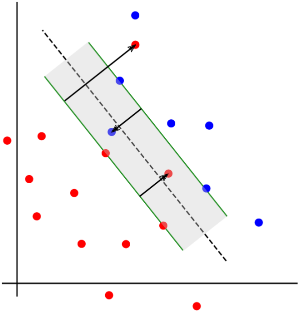


(a) Hard SVM



(b) Soft SVM

Soft margin SVM



- allows some misclassifications by relaxing the hard constraints
- implemented with the help of `Regularization parameter(C)`

Regularization parameter C

The loss function of the SVM is modified as:

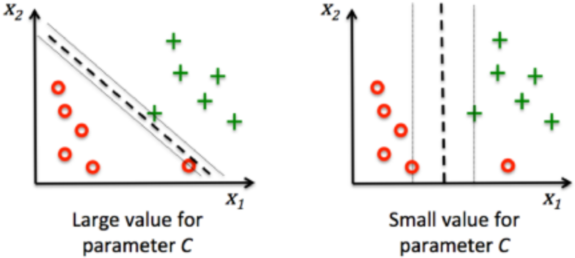
$$\min_{w} \frac{1}{2}||w||^2 + C * (\# \text{ of misclassifications})$$

$\boldsymbol{C}$: represents the trade-off between maximizing the margin and minimizing the mistakes

Small C
: classification mistakes given less importance, more focus on maximizing the margin - Soft SVM!

Large C
: more focus on avoiding misclassifications at the expense of keeping the margin small - Hard SVM!

Regularization parameter and SVM



- `Large C`: margin decreases - Hard SVM
- `Small C`: margin increases - Soft SVM

The Kernel trick

- used to tackle the problem of linear inseparability
- **idea:** map data from the original space into a higher dimensional feature space where they are linearly separable
- fit a decision boundary in this higher dimensional space to separate the classes

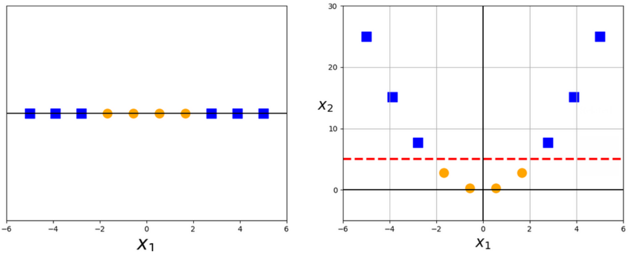
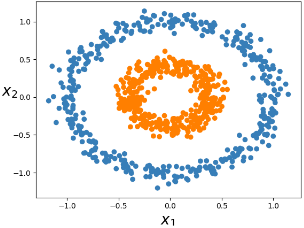


Figure: After applying the transformation $\phi(x) = x^2$

Non-linear transformations: Example



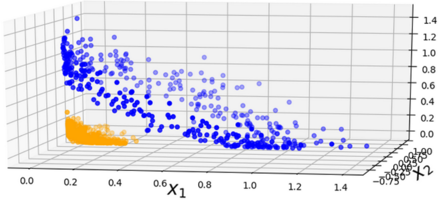


Figure: After applying the transformation $\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = \begin{bmatrix} {x_1}^2 \\ \sqrt{2}x_1x_2 \\ {x_2}^2 \end{bmatrix}$

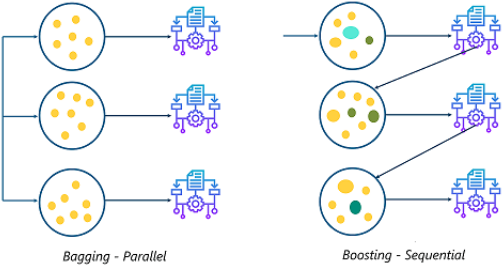
Kernel functions

- accepts inputs in the original lower dimensional space and returns the dot product of the transformed vectors in the higher dimensional space
- doesn't explicitly compute the coordinates in the higher dimensional space
- formally, if we have $x, z \in X$ and a map $\phi : X \to \mathcal{R}^N$ then $k(x, z) = < \phi(x), \phi(z) >$ is a kernel function

The Kernel trick

- the objective function of SVM depends on the dot product of input vector pairs $(x_i.x_j)$
- use kernel functions in place of dot product
  - has the capability of measuring similarity in higher dimensions without increasing the computational costs
- enables classification of linearly inseparable data
- some of the popular kernels are: Polynomial, Radial Basis Function (RBF), etc.

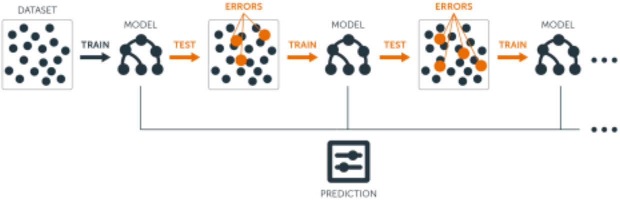
Bagging vs Boosting



Bagging weak learners produced parallelly by training on bootstrapped data sets

Boosting weak learners sequentially produced by assigning a higher weightage to the previous, incorrectly classified samples

Boosting

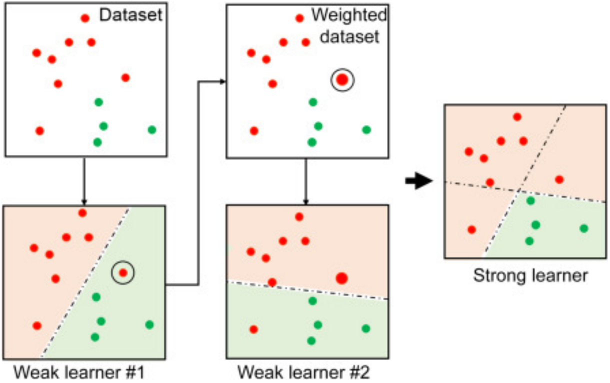


- ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors
  - weak learner: slight correlation with the true classification
  - strong learner: arbitrarily well-correlated with the true classification
- a series of models constructed such that each model tries to compensate for the weaknesses of its predecessor
- weak learners finally combined to form a strong learner

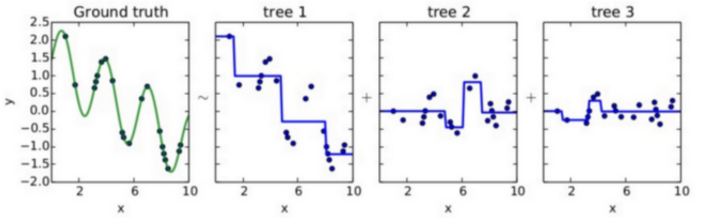
AdaBoost

- combines multiple weak learners into a single strong learner
- weak learners are decision trees with a single split, called decision stumps
- all observations weighted equally while creating the first decision stump
- observations incorrectly classified assigned higher weights
- a new decision stump is drawn by considering the observations with higher weights
- this process continues until a specified limit is reached in terms of the number of models or accuracy
- can be used for classification as well as regression

AdaBoost

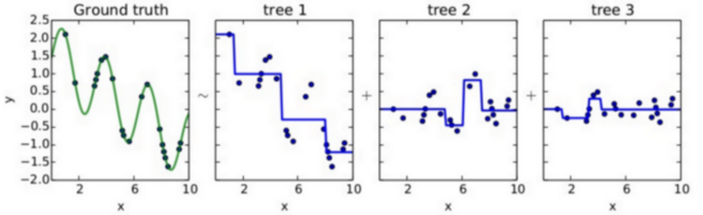

Dataset
Weighted dataset
Weak learner #1
Weak learner #2
Strong learner

Gradient boosting



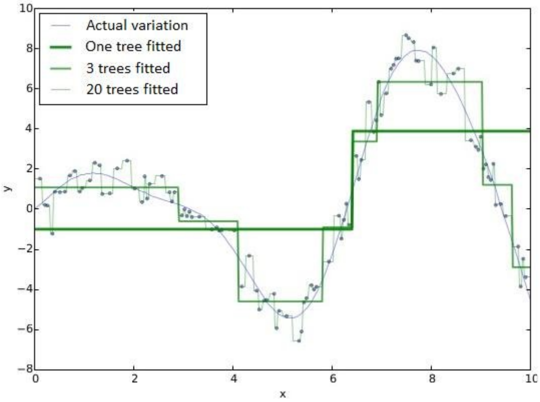
- base learners generated sequentially such that the present base learner is always more effective than the previous one
- tries to fit the new predictor to the residual errors made by the previous predictor
- can be used for classification as well as regression

Gradient boosting - Regression

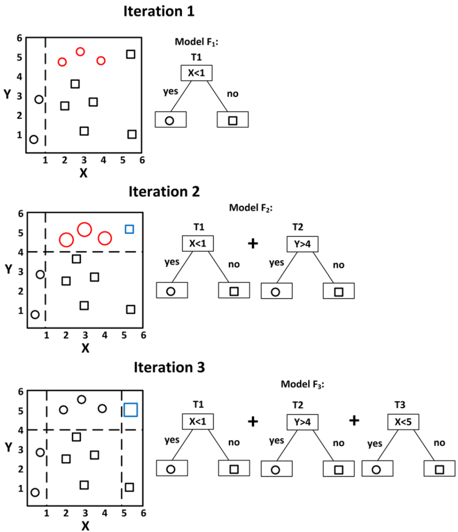


1. Fit a model to the data $f_1(x) = y$
2. Fit a model to the residuals $h1(x) = y - f_1(x)$
3. Create a new model $f_2(x) = f_1(x) + h_1(x)$
4. In general, $f_{m+1}(x) = f_m(x) + h_m(x)$, where $h_m(x) = y - f_m(x)$

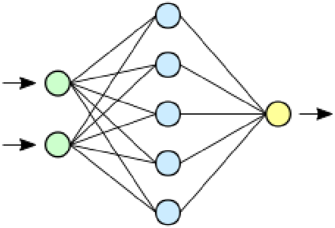
Gradient boosting - Regression



Gradient boosting - Classification

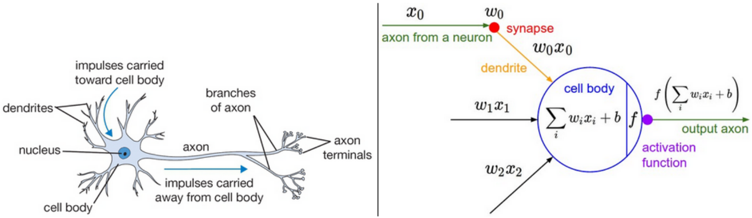


Neural networks



- computing systems inspired by biological neural systems
- a collection of connected units or nodes called artificial neurons
- artificial neuron receives a signal, processes it and signal neurons connected to it
  - signal at a connection is a real number
  - output computed by some non-linear function of its inputs
  - connections referred to as edges
- a neural network is a **function!!**

Biological neuron and its mathematical model

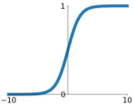


- edges typically have a weight that adjusts as learning proceeds
- weight increases or decreases the strength of the signal at a connection
- neuron computes weighted sum of its inputs - referred as **activation**
- weighted sum passed through a nonlinear activation function to produce the output

Activation functions

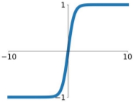
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



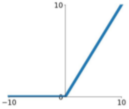
tanh

$\tanh(x)$



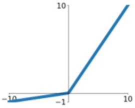
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

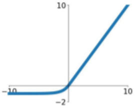


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

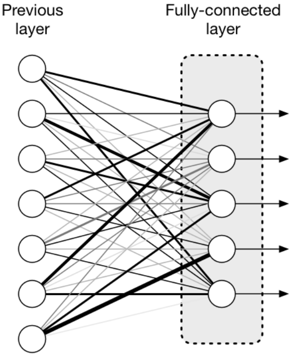
ELU

$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Fully connected layer

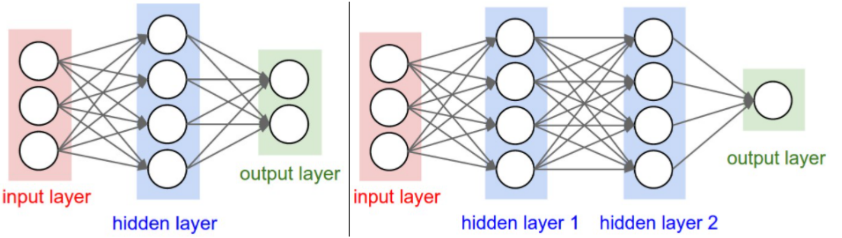
- neural network models often organized into distinct layers of neurons
- one of the most common layer type is `fully-connected layer`
  - neurons between two adjacent layers are fully pairwise connected
  - neurons within a single layer share no connections



Naming convention

N-layer neural network

- N-1 layers of hidden units
- 1 output layer
- input layer not counted

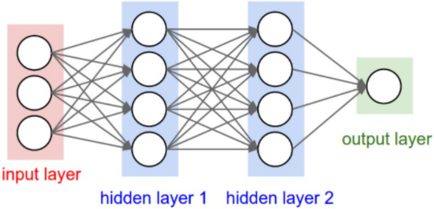


Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs.
Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer.

Sizing neural networks

Two common metrics used:

- number of neurons
- number of parameters (more commonly used)



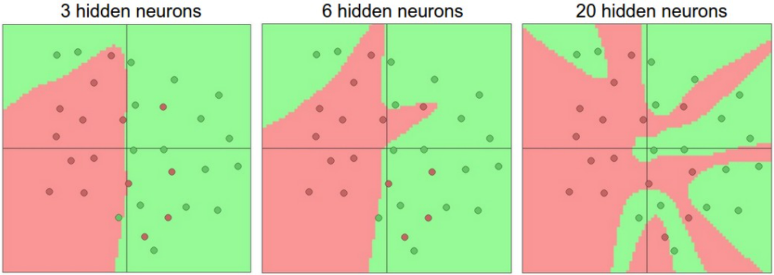
No: of neurons $4 + 4 + 1 = 9$ (not counting inputs)

No: of parameters

- `weights:` $[3 \times 4] + [4 \times 4] + [4 \times 1] = 32$
- `biases:` $4 + 4 + 1 = 9$
- `total:` $32 + 9 = 41$

Representation power of neural networks

- larger neural networks can represent more complicated functions
- prone to overfitting



References

1. https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/
2. https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f
3. https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5
4. https://cs231n.github.io/neural-networks-1/
5. http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Thanks Google for the pictures!