# ML Algorithms Explained

Chenshu Liu

May 31, 2022

## 1 KNN

KNN is the abbreviation of K Nearest Neighbors.
The process of determining the classification of the sample is find k (defined by user) nearest neighbors of the sample point and calculate the euclidean distance of the neighbor points from the sample points.
Since we already know the categories that the neighbor points belong to, we can use the majority of the neighbor category as the classification for the sample point.

## 2 Linear Regression

Linear regression is used to predict continuous values using the approximation formula $\hat{y} = wx + b$
To find the optimal values of the weight w and intercept b, we introduce MSE to evaluate the error of prediction for some w and b. MSE is calculated as follows:

$$MSE = J(w, b) = \frac{1}{N}\Sigma i = 1^n (yi - (wxi + b))^2$$

Then, we use gradient descent to find the optimal set of $(w, b)$ that minimizes the partial derivates of MSE:

$$J'(w, b) = \begin{bmatrix} \frac{\partial f}{\partial w} \\ \frac{\partial f}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\Sigma - 2xi(yi - (wxi + b)) \\ \frac{1}{N}\Sigma - 2(yi - (wxi + b)) \end{bmatrix}$$

Gradient descent is an iterative process that iterates the calculation until the local minimum is reached. In this case, since we have two parameters $(w, b)$, we are conducting gradient descent in a two-dimensional space.
After every iteration, we will update the parameters in the following way:

$$w = w - \alpha \cdot dw$$

$$b = b - \alpha \cdot db$$

# 3 Logistic Regression

Logistic regression is a binary classification algorithm.
In linear regression, we model the behavior of the data using linear function to predict continuous values. In logistic regression, we want to predict binary outcomes, thus we will use a sigmoid function $s(x) = \frac{1}{1+e^{-x}}$. And, for the prediction, we have $\hat{y} = h\theta(x) = \frac{1}{1+e^{-wx+b}}$. Similar to the process of finding optimal parameters for linear regression function, we also use gradient descent to find the best set of values for $(w, b)$ that provides the best prediction performance. The cost function (cross entropy) is:

$$J(w,b) = J(\theta) = \frac{1}{N}\Sigma i = 1^n[y^i log(h\theta(x^i) + (1-y^i)log(1-h\theta(x^i))]$$

The gradient descent procedure is the same as those from the linear regression section.

# 4 Naive Bayes

Naive Bayes is a classification algorithm that is achieved through the Bayes Theorem.
Bayes Theorem:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

In our case, using X to predict for y:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Though in real life cases, all features are rarely independent, but the assumption of all features being mutually independent still proves to be effective:

$$P(y|X) = \frac{P(x1|y) \cdot P(x2|y) \cdot \cdots \cdot P(xn|y) \cdot P(y)}{P(X)}$$

Where:

1. Prior probability $P(y)$ is just the frequency
2. Class conditional probability follows the gausian distribution: $P(xi|y) = \frac{1}{\sqrt{2\pi\sigma y^2}} \cdot exp(-\frac{(xi-\mu y)^2}{2\sigma y^2})$

Then, the classification is performed by selecting the class with the highest probability:

$$y = argmaxyP(y|X) = argmaxy\frac{P(x1|y) \cdot P(x2|y) \cdot \cdots \cdot P(xn|y) \cdot P(y)}{P(X)}$$

$$y = argmaxyP(x1|y) \cdot P(x2|y) \cdot \cdots \cdot P(xn|y) \cdot P(y)$$

Because all the probabilities are values between 0 and 1, and multiplying them may lead to the product being a very small number. Thus, in order to prevent overflow , we can apply logarithm to the product:

$$y = argmaxylog(P(x1|y)) + log(P(x2|y)) + \cdots + log(P(xn|y)) + log(P(y))$$

# 5 Perceptron

Perceptron is a simplified version of a biological neuron.
The neuron in the biological system receives inputs through dendrites and generates one response ( i.e. output ) through the axon. In the perceptron model, we can express the pipeline of information processing using a linear function $y = w^T x + b$ where $w$ is the weight assigned to each input node and $b$ is the bias. In the image above, we can see that after the input and weights are manipulated, an activation function is called. In the case of perceptron, it is simple the unit step function that becomes significant only when the value reaches certain threshold.
Thus, we can summarize the pipeline as

$$\hat{y} = g(f(w, b)) = g(w^T x + b)$$

and through the training process, we can update the parameters using the perceptron update rule, such that for each training sample $xi$:

1. $w := w + \Delta w$
2. $\Delta w := \alpha \cdot (yi - \hat{yi}) \cdot xi$
3. learning rate $\alpha$ is between [0, 1]

During the dynamic update of the weights, the weights are adjusted to better match the prediction with the actual value. So, the reason why there is a $(yi - \hat{yi})$ term is that:

1. When the predicted $\hat{yi}$ is greater than the actual $yi$, the weight update $\Delta w$ has a negative value, reducing the weight in the new iteration
2. When the predicted $\hat{yi}$ is less than the actual $yi$, the weight update $\Delta w$ has a positive value, increasing the weight in the new iteration
3. In sum, the weights are pushed towards positive or negative target class in cases of misclassification

NOTE: perceptron only works for linearly separable classes ( i.e. classes that can be separated using a linear function ). For further improvements, the activation function can be changed accordingly ( i.e. instead of using the unit step function, we can use functions like the sigmoid function )