March 6, 2018

Problem Set 4

Sanchit Singhal

INF 385T – Introduction to Machine Learning with Danna Gurari

Spring 2018

School of Information

1. Artificial Intelligence

a. Computer Vision and Machine Learning

Computer vision is a field that deal with how machines can gain an understanding of pictures. It seeks to automate tasks that the human visual system can do such as the automatic extraction, analysis, and understanding of useful information from a single or a sequence of images. Computer vision involves transforming images to high-dimensional data to produce an equivalent numerical array that the machine can interpret. Computer vision is essential to machine learning in that a machine needs to be able to "see" an image before it is able to absorb from it – that is to learn to interpret images, figure out what it is seeing and analyze it before it is able to develop algorithms that can be used for other future tasks which may require analysis of similar pictures or videos.

b. Low-level, mid-level, and high-level vision features

After extracting the raw pixel values from an image – the computer must be able to identify features from the picture. These features of an image can be classified into three different levels: low, mid, and high based on the amount of detail of useful information that can be deduced from it. Low-level features are details that are relatively easy to deduce but do not convey much information on their own such as dots, edges, corners, lines, and curves. On the other hand, high-level features are harder to interpret but provide a lot of information such as objects, scenes, and emotions. Mid-level features fall somewhere in between the two extremes – slightly difficult to deduce and convey only some important details such as forms, colors, and shapes. Usually, high-level features are built upon lower level features and algorithms.

2. Artificial Neurons

a. Perceptron vs Adaline

Perceptron and Adaline are both single-layer neural network models that are classifiers for binary classifications. Both models, based on the human brain, include input features, model coefficients, and outputs that are the weighted sum of inputs through an activation function that has a certain threshold. Both models have a linear decision boundary line, and learn iteratively – sample by sample. The key difference between the two is that while Perceptron makes update to the weights every sample, Adaline updates the weights with the accumulated values. Hence, the two models use different gradient descent methods– Perceptron uses a Stochastic gradient descent while Adaline uses a Batch Gradient descent. Further, Adaline updates are based on continuous valued predictions rather than integer (discrete) predictions as in the case with Perceptron and therefore, Perceptron tends to jump around a lot more when finding the optimum solution.

b)

Sample	X_1	X_2	X_3	Y
1	0	0	0	-1
2	1	1	0	1
3	1	0	1	1
4	0	1	0	-1

$\phi(w^T x)$ { 1 if $\phi(w^T x) \geq 0$; -1 otherwise

learning rate = 0.2

⇒ initialize $w_0 = w_1 = w_2 = w_3 = 0$

epoch 1

	w_0	w_1	w_2	w_3	Predicted	
1	-0.4	0	0	0	1	wrong!
2	0	0.4	0.4	0	-1	wrong!
3	0	0.4	0.4	0	1	right!
4	-0.4	0.4	0	0	1	wrong!

① $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = 0 + 0(0) + 0(0) + 0(0) = 0 \therefore 1$

$\Delta w_0 = 0.2(-1-1) = -0.4$ $\Delta w_1 = (-0.4)(0)$ $\Delta w_2 = (-.4)(0)$ $\Delta w_3 = .4 \cdot 0$

$w_0 = 0 - 0.4 = -0.4$ $w_1 = 0$ $w_2 = 0$ $w_3 = 0$

② $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = -0.4 + 0(1) + 0(1) + 0(0) = -0.4 \therefore -1$

$\Delta w_0 = 0.2(1-(-1)) = 0.4$ $\Delta w_1 = 0.4(1) = 0.4$ $\Delta w_2 = 0.4(1) = 0.4$ $\Delta w_3 = 0.4(0) = 0$

$w_0 = -0.4 + 0.4 = 0$ $w_1 = 0 + 0.4 = 0.4$ $w_2 = 0 + 0.4 = 0.4$ $w_3 = 0 + 0 = 0$

③ $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = 0 + 0.4(1) + 0.4(0) + 0(1) = 0.4 \therefore 1$

$\Delta w_0 = 0.2(1-1) = 0$ $\Delta w_1 = 0(1) = 0$ $\Delta w_2 = 0(0) = 0$ $\Delta w_3 = 0(1) = 0$

$w_0 = 0 + 0 = 0$ $w_1 = 0.4 + 0 = 0.4$ $w_2 = 0.4 + 0 = 0.4$ $w_3 = 0 + 0 = 0$

④ $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = 0 + 0.4(0) + 0.4(1) + 0(0) = 0.4 \therefore 1$

$\Delta w_0 = 0.2(-1-1) = -0.4$ $\Delta w_1 = (-0.4)(0) = 0$ $\Delta w_2 = (-0.4)(1) = -0.4$ $\Delta w_3 = (-0.4)(0) = 0$

$w_0 = 0 - 0.4 = -0.4$ $w_1 = 0.4 + 0 = 0.4$ $w_2 = 0.4 - 0.4 = 0$ $w_3 = 0 + 0 = 0$

epoch 2

	w_0	w_1	w_2	w_3	predicted	
1	-0.4	0.4	0	0	-1	right!
2	-0.4	0.4	0	0	1	right!
3	-0.4	0.4	0	0	1	right!
4	-0.4	0.4	0	0	-1	right!

① $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = -0.4 + 0.4(0) + 0(0) + 0(0) = -0.4 \therefore -1$

$\Delta w_0 = 0.2(-1-(-1)) = 0$ $\Delta w_1 = 0(0) = 0$ $\Delta w_2 = 0(0) = 0$ $\Delta w_3 = 0(0) = 0$

$w_0 = -0.4+0 = -0.4$ $w_1 = 0.4+0 = 0.4$ $w_2 = 0+0 = 0$ $w_3 = 0+0 = 0$

② $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = -0.4 + 0.4(1) + 0(1) + 0(0) = 0 \therefore 1$

$\Delta w_0 = 0.2(1-1) = 0$ $\Delta w_1 = 0(1) = 0$ $\Delta w_2 = 0(1) = 0$ $\Delta w_3 = 0(0) = 0$

$w_0 = -0.4+0 = -0.4$ $w_1 = 0.4+0 = 0.4$ $w_2 = 0+0 = 0$ $w_3 = 0+0 = 0$

③ $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = -0.4 + 0.4(1) + 0(0) + 0(1) = 0 \therefore 1$

$\Delta w_0 = 0.2(1-1) = 0$ $\Delta w_1 = 0(1) = 0$ $\Delta w_2 = 0(0) = 0$ $\Delta w_3 = 0(1) = 0$

$w_0 = -0.4+0 = -0.4$ $w_1 = 0.4+0 = 0.4$ $w_2 = 0+0 = 0$ $w_3 = 0+0 = 0$

④ $w_0 + w_1(x_1) + w_2(x_2) + w_3(x_3) = -0.4 + 0.4(0) + 0(1) + 0(0) = 0 - 0.4 \therefore -1$

$\Delta w_0 = 0.2(-1+1) = 0$ $\Delta w_1 = 0(0) = 0$ $\Delta w_2 = 0(1) = 0$ $\Delta w_3 = 0(0) = 0$

$w_0 = -0.4+0 = -0.4$ $w_1 = 0.4+0 = 0.4$ $w_2 = 0+0 = 0$ $w_3 = 0+0 = 0$

It's noteworthy to see that once the Perceptron is able to correctly predict the observed value, it stops updating the weights and therefore all four rows in epoch 2 are identical. This is how we know that we have found a good solution and we do not need to continue running epochs.

3. Gradient Descent

a. Stochastic, batch, and mini-batch gradient descent

The idea behind gradient descent is that you iteratively adjust the model parameters to try and make the error smaller until an optimum solution is found. Stochastic, batch, and mini-batch gradient descent are all methods that do this – but through different approaches. In Stochastic Gradient Descent (SGD), each update is using calculations from one training example, whereas in Batch Gradient Descent (BGD), the update uses calculations from all the training examples. Mini-batch gradient descent is a balance between the two – using a subset of training examples for the calculations at every step. Below is table comparing some strengths and weaknesses between the three types of gradient descent:

	Stochastic Gradient Descent (SGD)	Batch Gradient Descent (BGD)	Mini-batch Gradient Descent
Advantage	Each iteration is fast to compute Work well with large datasets as it only has to store 1 instance in memory at each iteration	Does not bounce too much	Bounces less erratically when finding model parameters than SGD Can train using large datasets as only some instances have to be in memory
Weakness	Updates bounce around a lot Can sometimes diverge and never find a good solution	Very slow Infeasible with a large dataset to store all examples in memory	Can sometimes still be slow when working with huge datasets

b. Gradient Descent for Perceptron

Perceptron uses Stochastic Gradient Descent (SGD).

c. Gradient Descent for Adaline

Adaline uses Batch Gradient Descent (BGD).