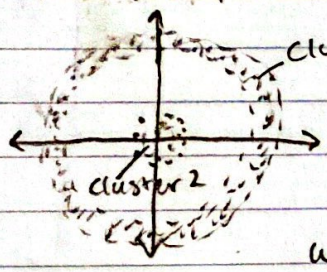
Ryan Blair

# CRE-695 : HW 5

Q1)

1) Higher dimensional data often leads to longer runtimes and overfitting. The KNN algorithm relies on the concept of a "metric" or "distance" between neighbors. If you have N dimensional data, with M data points, you need to calculate M! distances, and when using the Euclidean metric, those calculations each look like: $d(\vec{x},\vec{y}) = \sum_{i}^{N} (x_i - y_i)^2$

2) Basically the problem is that the data may not always arrange itself in data clusters w/ the same variance. It may not even be arranged in a linearly seperable manner. For example:




Would not be clustered well with the Euclidean metric. Therefore using a different metric, something like $d(\vec{x},\vec{y}) = |r_x - r_y| = \left|\sqrt{(x_1^2+x_2^2)} - \sqrt{(y_1^2+y_2^2)}\right|$, which compares the distance each point is from the center.

3)

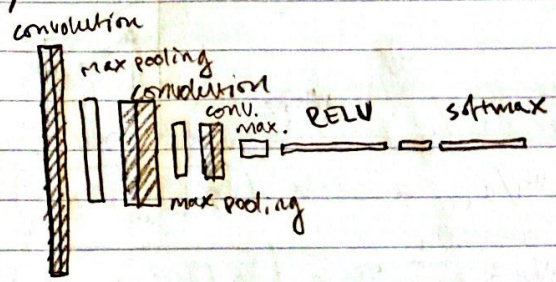
Wyatt Bleier

CPE-695

Q1 (cont.)

3) Given D dimensional data, the GMM's goal is to extract N D-dimensional $\vec{\mu}$ and $\vec{\sigma}$ vectors, which describe N groupings of datapoints. Since KNN does not take non-linear relationships well, GMM allows for the clusters to not be perfect circles, meaning anomolous data points are more readily determined.

4)

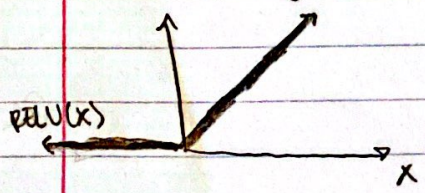



ResNet, a modern CNN architecture uses an extremely deep architecture of 152 layers. It can train such a network because of the clever skip connections in the network.

- a convolution is essentially a weighted mask you can hold up against a section of data to obtain an associated score with that section of the data

- max pooling calculates the maximum values in each patch of a feature map, resulting in downsampled data or pooled feature maps highlighting the most present feature in the patch. Another option is average pooling.

- RELU maps negative values to 0, and returns x if x>0




- Finally softmax is essentially the logistic function in higher dimensions. It converts K real valued vectors to a probability distribution w/ K different outcomes

Wyatt Blair

CPE-695: HW 5

Q1 (cont.))

5)

- Vanishing gradients:
  - Vanishing gradients occur when the derivative gets smaller as one goes backward through back propogation. This means the weights at the beginning of the network are not being adjusted by much
  - Happens when using sigmoid & tan activation functions.

- Exploding Gradients:
  - The exact opposite of vanishing gradients, now derivative is becoming larger as one moves through backpropogation.
  - occurs when the weights are too high

- Remedies:
  - use Gradient Clipping: normalize the error vector before proceeding through backpropogation to prevent very large errors
    - ↳ useful for both vanishing gradients & exploding gradients
  - design architecture such that learning rate is sufficiently low

Wyatt Blair

# CPE-695 : HW 5

Q2)

$$error_D(h) \equiv \underset{x \in D}{P}\left[f(x) \neq h(x)\right] \qquad n = 100$$

$$error_S(h) \equiv \frac{1}{n}\sum_{x \in S} \delta(f(x) \neq h(x)) = \frac{20}{100} = 20\% = 0.2$$

$\rightsquigarrow$ ~95% $error_D(h)$ lies in interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1-error_S(h))}{n}} = 0.2 \pm 1.96\sqrt{\frac{0.2(0.8)}{100}}$$

$$\Rightarrow \quad 0.2 \pm 1.96\sqrt{\frac{0.2(0.8)}{100}} = 0.1216, 0.2784 = 12.16\%, 27.84\%$$