Week 4: Formalize into Mahalanobis distance

Now that we’ve begun to investigate the differences in activation patterns for benign vs perturbed adversarial images, let’s quantify it a little bit more with Mahalanobis distances some visualization.

Assignment:

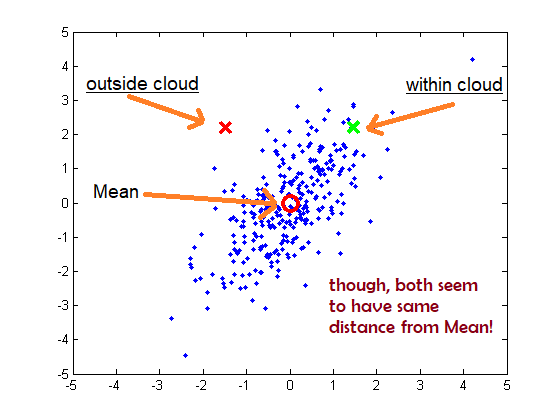
1. First, let’s visualize the activation differences for each sample based on the mean and stdev of classes we calculated last week. We will visualize using heatmaps. If you can hack keract’s heatmap visualization, that’s fine, or you can use matplotlib’s function (<https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/image_annotated_heatmap.html>). Each pixel will represent a neuron, and its value how many standard deviations away the sample’s activation is from the class mean.
   * 1. Created heatmaps for the successful adversarial examples out of the 100 generated using the DeepFool attack with an epsilon value of 0.1. These have been uploaded to GitHub under Week 4.
2. Let’s formalize this metric a little bit more, using better statistics. Figuring out individual stdevs is great and all, but what if some pixels are highly correlated? (Not surprising in an image—pixels are usually similar to those around them.) So if one pixel is outside the norm, others nearby will likely be as well. Should these be counted separately as independent events? How to adjust for correlations in the features? To address these issues, let’s explore Mahalanobis distances:
   1. <https://www.machinelearningplus.com/statistics/mahalanobis-distance/>
   2. <https://www.youtube.com/watch?v=spNpfmWZBmg>
   3. Calculate Mahalanobis distances for your adversarial examples against the class distributions, i.e instead of calculating how far away adversarial activations are from the mean in terms of the stdev, use Mahalanobis to calculate distance.
   4. Hint: to calculate the covariance matrix of the training samples in each class, you can use numpy’s cov() function. However, to prevent long computation times and numerical underflow errors, you may want to transform the data first into an uncorrelated bases (brush off that linear algebra) through PCA. You don’t necessarily need to the full bases; truncating the number of features essentially reduces the dimensionality of the problem space. Use sklearn’s PCA function and its explained variance parameter to pick a reasonable truncation point, e.g. if the first 10 axes explain 97% of the variance, try using just the first 10 axes.
   5. If you use PCA to transform the data, then by definition the bases are uncorrelated and the covariance matrix reduces down to a diagonal matrix of its eigenvalues, which is much easier to work with (and calculate!)
      * 1. Used PCA to reduce the dimensionality down from 128 to 40 components which accounts for more than 97% of the variance for each class. Recorded the activations for 100 benign examples of each class and used pca.fit to fit the data and then transform the data and the adversarial examples.
   6. If you’re rusty on your linear algebra/PCA, feel free to learn more about it! <https://www.youtube.com/watch?v=FgakZw6K1QQ>
      * 1. Followed each tutorial thoroughly and watched all videos.
3. Compare your Mahalanobis distances to the simpler mean/stdev work from last week. Do the Mahalanobis distances say anything different? Do you get more nuance? One big benefit is that Mahalanobis distances are what are known in statistics as being chi-square distributed (<https://www.khanacademy.org/math/statistics-probability/inference-categorical-data-chi-square-tests/chi-square-goodness-of-fit-tests/v/chi-square-distribution-introduction>). These distributions are great, because we can actually assign probabilities to the numbers now, much like normal distributions have a 3% chance of values falling outside of 2 stdevs of the mean. Use scipy.stats.chi2.ppf to find critical values for our degrees of freedom (dense=128? Not sure if the function goes this high, but let’s try.) We often use p values < 0.05 for significance (see tutorial in 2.a above), so try that value of p. Can we detect adversarial examples as significantly out of range of the distribution?
   * + 1. After calculating the Mahalanobis distance for each successful adversarial example, for which there were 253 out of the 1000 examples using DeepFool with an epsilon value of 0.1, the distance was compared to the critical value of 54.572 (p-value of 0.05 and 39 degrees of freedom). If the Mahalanobis distance exceeded the critical value, or the calculated p-value was less than 0.05, than the example could be labeled ‘extreme,’ or adversarial. 199 of the 253 successful adversarial examples had Mahalanobis distances greater than the critical value. This is a better metric for detecting adversarial examples because it takes into account the distance from the distribution, not just the mean as shown below.
       2. 

Photo from StackOverflow

An interesting question that this poses is the fact that since the adversarial examples were transformed into the space fit on the benign examples, some components that were truncated (since they did not account for much variance) in the benign examples could actually have accounted for a greater amount of variance in the adversarial examples. This is just one possible explanation for why there isn’t even a high number of successful adversarial examples with mahalanobis distances greater than the critical value. Although it would be more computationally extensive, it would be interesting to see how the Mahalanobis distance results compare without doing PCA first in this case. I am interested in discovering why exactly more of the adversarial examples do not exceed the critical value. While there are some successful adversarial examples do not exceed the critical value, the Mahalanobis distance is still a reasonably effective strategy for detecting adversarial examples.

1. Finally, as this is our last week, make sure you document, document, document. Put all these code and answers on your github. Bonus: create a short 5-10 slide presentation on what you’ve done this past month; you never know when you’ll be asked to give a short technical presentation, and having one ready to go is a huge plus. Prepare it now while it’s fresh in your memory!
   * + 1. Prepared a short presentation on the work I have accomplished, also posted on GitHub!