1. UNDERSTANDING KERNEL DESITY ESTIMATION

* Non parametric, unsupervised technique to estimate the unknown probability distribution of a random variable, based on a sample of points taken from that distribution. We are estimating the probability density function of the variable, and we use kernels to do this, hence the name. This work sample required to use Gaussian kernel.
* A Gaussian bell curve is typically used because smoother the curve (for the Gaussian this means higher variance), the more we spread out the evidence: if we observed several points close together, we are going to assume **all** points in that neighborhood are very likely. A region where we didn't observe any points in our sample will have a very low density.
* How does this work? The simple technique is used: to make a prediction for any point x, just sum, for all points xi in our sample, all the kernel functions of the difference between xi and x:

∑K(x−xi), for n , where n is the total sample size. Often we include a parameter in the kernel function signifying the *width* h of the kernel. If a kernel has smaller width, the neighborhoods over which we average will be taken will be much smaller, leading to less generalization. When the width is very large, it will be much harder to detect any small differences in density.

* It’s a good idea just to try to analyse the formula, since it is pretty straightforward:

https://qph.fs.quoracdn.net/main-qimg-3612293e16e11100877cec60f9a8e7ca

This formula approximates a probability for some value x, given you have some data sample vector **X**. **K** is a kernel function, and it takes in a distance between your value **x** and a value within the sample vector xi. As far as I know all Kernel’s PDFs are centered on 0, so 0 distance will get you the highest probabilities.

1. EXPERIMENT RESULTS

|  |  |  |
| --- | --- | --- |
| Sigma | MNIST train | CIFAR-100 train |
| 0.05 | -3153.4699360009135 | -13621.238039998123 |
| 0.08 | -613.728085029534 | -2882.5006067734294 |
| 0.1 | -117.54234061115153 | -756.6303841687113 |
| 0.2 | 233.88546553963494 | 862.5173175190464 |
| 0.5 | -233.7958576755206 | -902.695997261545 |
| 1.0 | -740.8756232239083 | -2881.8820127645317 |
| 1.5 | -1051.1173717742595 | -4099.253182779342 |
| 2.0 | -1272.9603036975886 | -4972.590699399304 |

With optimal value of sigma = 0.2 using grid seach. Follwing results are observed for test sets:

|  |  |
| --- | --- |
| Run time | MNIST test |
| 1994.9850001335144 seconds | 231.66964857150407 |

|  |  |
| --- | --- |
| Run Time | CFIR-100 test |
| 84445.99079346657 seconds | 850.4698164628228 |

1. PROS AND CONS OF USING KDE

* Typically don’t make any distributional assumptions
* As we have more data, we should be able to learn more complex models.
* Let number of parameters scale with number of training data.
* need all points to compute estimate/

1. CONCLUSIONS

The running time for CIFAR-100 is more than MNIST data even after applying same pre-processing to scale the values because CIFAR-100 are 3 channel images whereas MNIST are grayscale images.

REFERENCES:

[1.] <https://www.homeworkhelponline.net/blog/math/tutorial-kde>

[2.] <https://en.wikipedia.org/wiki/Kernel_density_estimation>