Dictionary Learning on Images

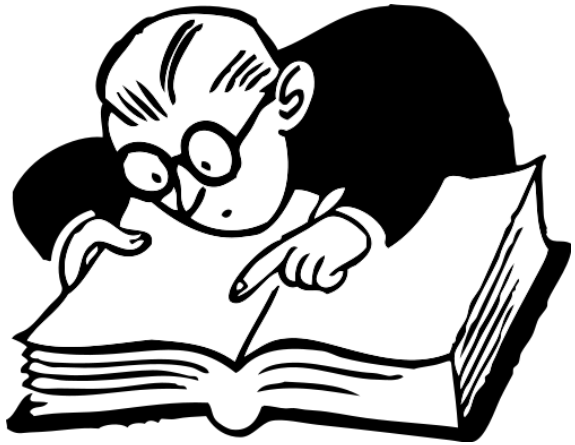
Bachelor's Thesis-II
Presentation
Guide: Prof. Suyash Awate

Indian Institute of Techchnology, Bombay

April 28, 2017

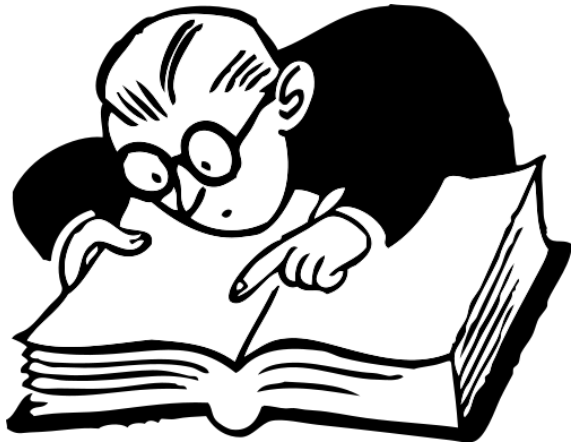
Introduction

Dictionary Learning is a technique used to learn discriminative sparse representations on complex data.
Given a data set, the aim of dictionary learning is to learn a set of basis elements, called atoms, a sparse linear combination of which can represent all the given data points.



Kernel Dictionary Learning

Traditional dictionary learning learns dictionary atoms in the domain of the data, representing data as linear combinations of the atoms.
This may not be discriminative enough so as to learn a linear classifier on the weights, to effectively classify data. In this case, transforming the data to a higher dimensional domain can make learning linear classifiers possible. This is the key idea behind kernel dictionary learning .



Formulating the problem

Consider a set of n d-dimensional data points.

$$\mathbf{X} = [\mathbf{X_1}, \mathbf{X_2} \dots \mathbf{X_n}]_{d \times n}$$

and a transformation function $\mathbf{\Phi} : \mathbb{R}^d \rightarrow \mathcal{V}$, where $\mathcal{V}$ is the kernel feature space. The aim is to learn a set of atoms $\{d_k\}_{k=1}^{m}$ and weights

$$\mathbf{W} = [\mathbf{W_1}, \mathbf{W_2} \dots \mathbf{W_n}]_{m \times n}$$

Formulating the problem

Such that we can represent

$$\mathbf{\Phi}(\mathbf{X_i}) \approx w_{1i}d_1 + w_{2i}d_2 + \cdots + w_{mi}d_m$$

where

$$\mathbf{W_i} = [w_{1i}, w_{2i} \ldots w_{mi}]^T$$

Formulating the problem

For an optimal dictionary, each d_k lies in the span of the transformed input data (in kernel feature space). Hence, it can be represented as a linear combination of these.

$$d_k = \sum_{i=1}^{n} \gamma_{ik} \mathbf{\Phi}(x_i), \text{where } \forall i : \gamma_{ik} \in \mathbb{R}$$

Formulating the problem

The goal can be formulated as

$$\{\gamma_{ij}\}, \mathbf{W} = \arg \min_{\{\gamma_{ij}\}, \mathbf{W}} \sum_{i=1}^{n} \left(\left\| \mathbf{\Phi}(\mathbf{X_i}) - \sum_{k=1}^{n} w_{ki} d_k \right\|_2^2 + \lambda \left\| \mathbf{W_i} \right\|_p^p \right)$$

under the constraints

$$\sum_{j=1}^{m} W_{ij} = 1, \text{for } i = 1, 2, \ldots, n$$

$$\text{and} \quad \|d_i\|_2 = 1, \text{for } i = 1, 2, \ldots, n$$

where $\|\mathbf{W_i}\|_p$ is the l_p norm of the vector $\mathbf{W_i}$ and acts as a sparsity inducing term. This can be reduced to have the final expression in terms of $\{\gamma_{ij}\}$, instead of $\{d_k\}$. Computing the function $\mathbf{\Phi}$ is subverted by simplifying the expression using kernel functions (the kernel trick)

Graph Regularized Sparse Coding (graphSC)

The key idea behind graph regularization is that sparse codes of similar data points should be similar. Hence, the regularization term involved penalises the distance between the sparse codes of 'similar' data elements. Which data points are similar, is determined by constructing a nearest neighbour graph.

Graph Regularized Sparse Coding (graphSC)

Consider a matrix **N** which represents the nearest neighbor graph such that $N_{ij} = 1$ if x_i is among the k nearest neighbours of x_j and 0 otherwise. The regularization term added to the objective function by graphSC is :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \|\mathbf{W_i} - \mathbf{W_j}\|_2^2 \, \mathbf{N_{ij}}$$

To simplify this, let us first define $\Delta_i = \sum_{j=1}^{n} \mathbf{N_{ij}}$, and $\mathbf{D} = \mathrm{diag}(\Delta_1, \Delta_2, \ldots, \Delta_n)$. The above term can then be simplified to $\mathrm{Tr}(\mathbf{WLW}^T)$, where $\mathbf{L} = \mathbf{D} - \mathbf{N}$ is the Laplacian matrix.

Variants of graphSC used

- **kNN regularizer** : The nearest neighbor matrix **N** is constructed using k-nearest neighbours and then normalized to sum 1 along the columns (i.e., divided by k).
- **gaussianNN regularizer** : Instead of a sharp mask of k nearest neighbours, a smooth gaussian neighbourhood measure is used for the matrix **N**. In this case, instead of k, width of the neighborhood mask σ becomes the free parameter. The matrix is computed as

$$N_{ij} = e^{-\frac{\text{dist}(x_i, x_j)^2}{\sigma^2}}$$

and then normalized to sum 1 along the columns.

Experiments

We ran tests to evaluate the performance of kernel dictionary learning with graphSC, against a baseline of simple kernel dictionary learning. The tests were run on MNIST handwritten digits image dataset.

Experiments

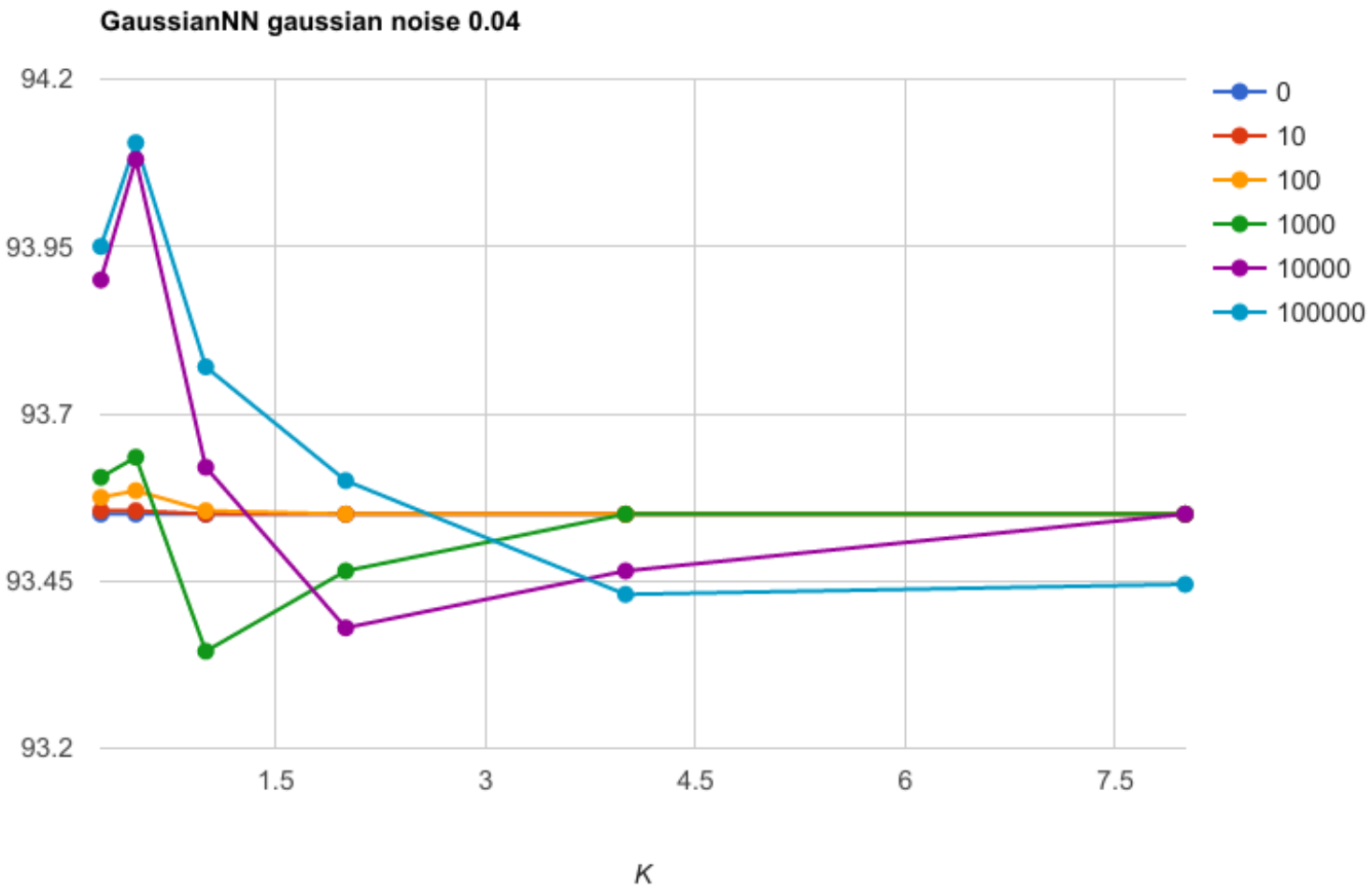
For each of the tests, we learn a dictionary of 20 atoms from 500 images for each digit. Hence, our training image set size is $500 \times 10 = 5000$ images. We then pool together these dictionaries for different digits (class labels) to create a large dictionary for all class atoms. For classification, a linear SVM is first trained on the sparse codes of the training data. Then, in the testing phase, we learn the sparse codes of test data using the pooled dictionary, and classify using the SVM classifier previously learned.

Performance of gaussianNN regularizer

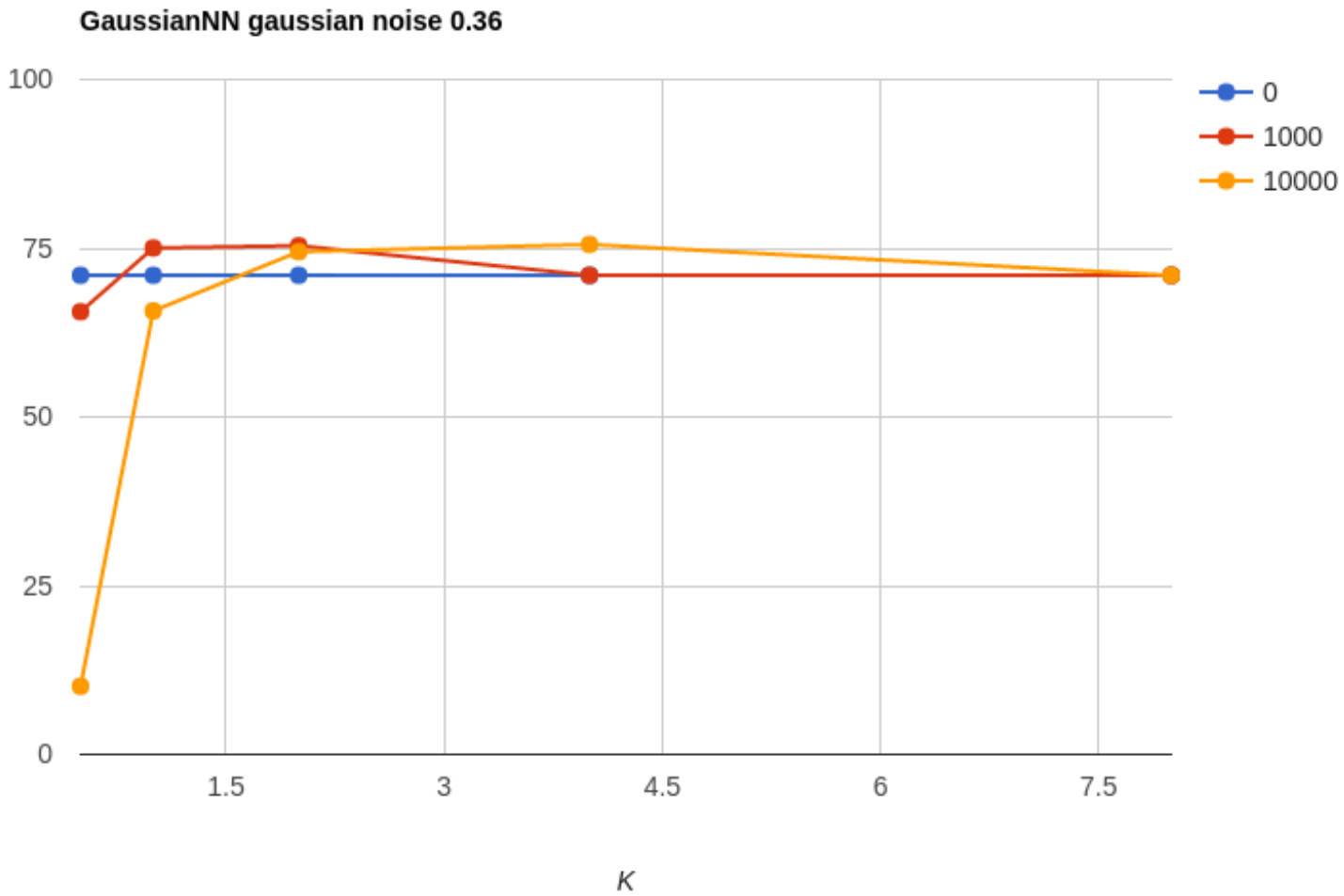
In the following graphs, we have classification accuracy on test data (10000 images) on the y-axis, and gaussian neighbourhood parameter C along the x-axis. C is defined such that it is inversely proportinal to width of gaussian used for calculating neighbourhood measures. In our implementation, we have different σ_i for each column, of the $\mathbf{N}$ matrix, given by

$$\sigma_i^2 = \frac{4 * \mathrm{Var}_{j=1}^{n}(\mathrm{dist}(\mathbf{\Phi}(x_i), \mathbf{\Phi}(x_j)))}{C}$$

Performance of gaussianNN regularizer



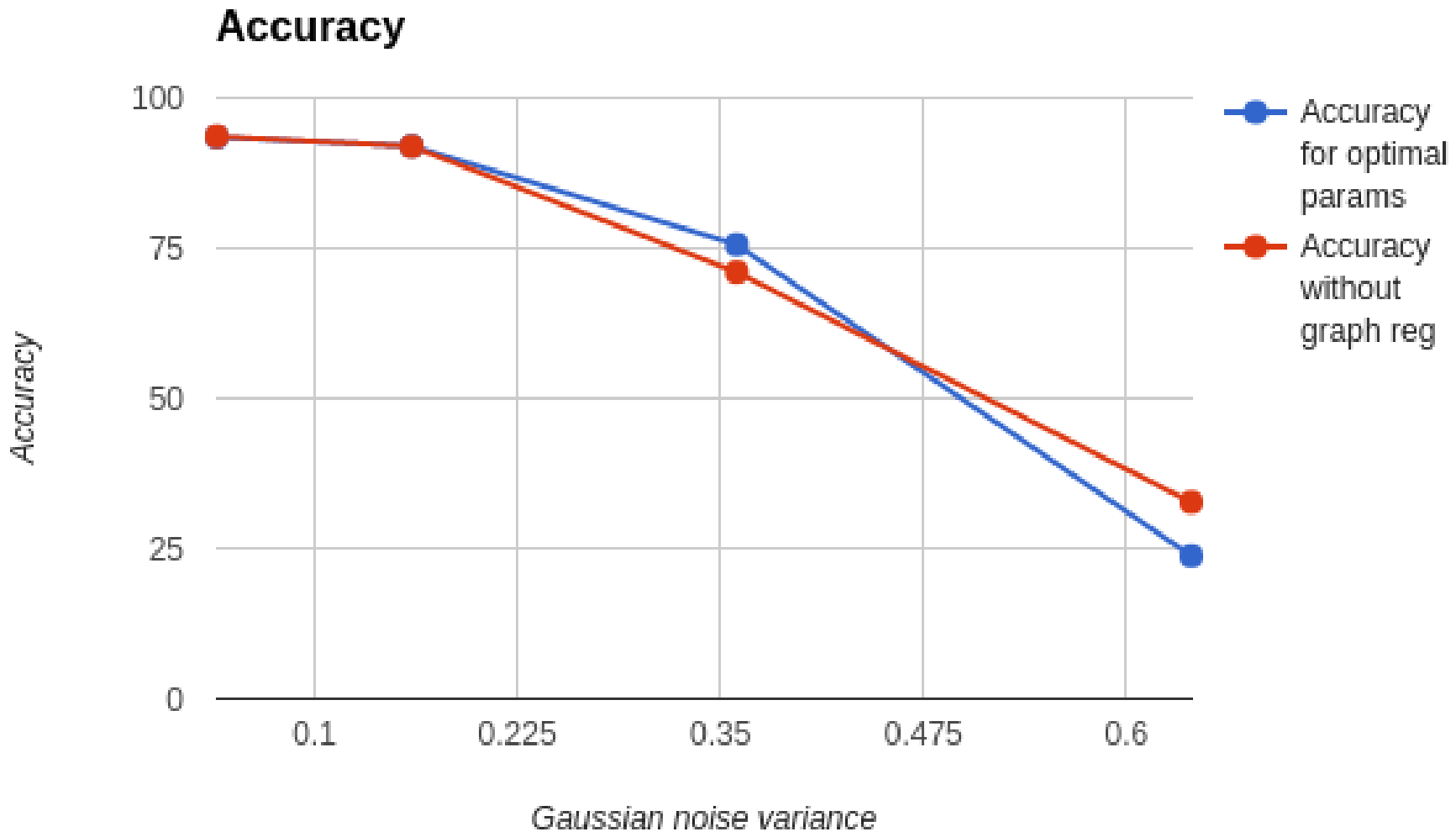
Performance of gaussianNN regularizer



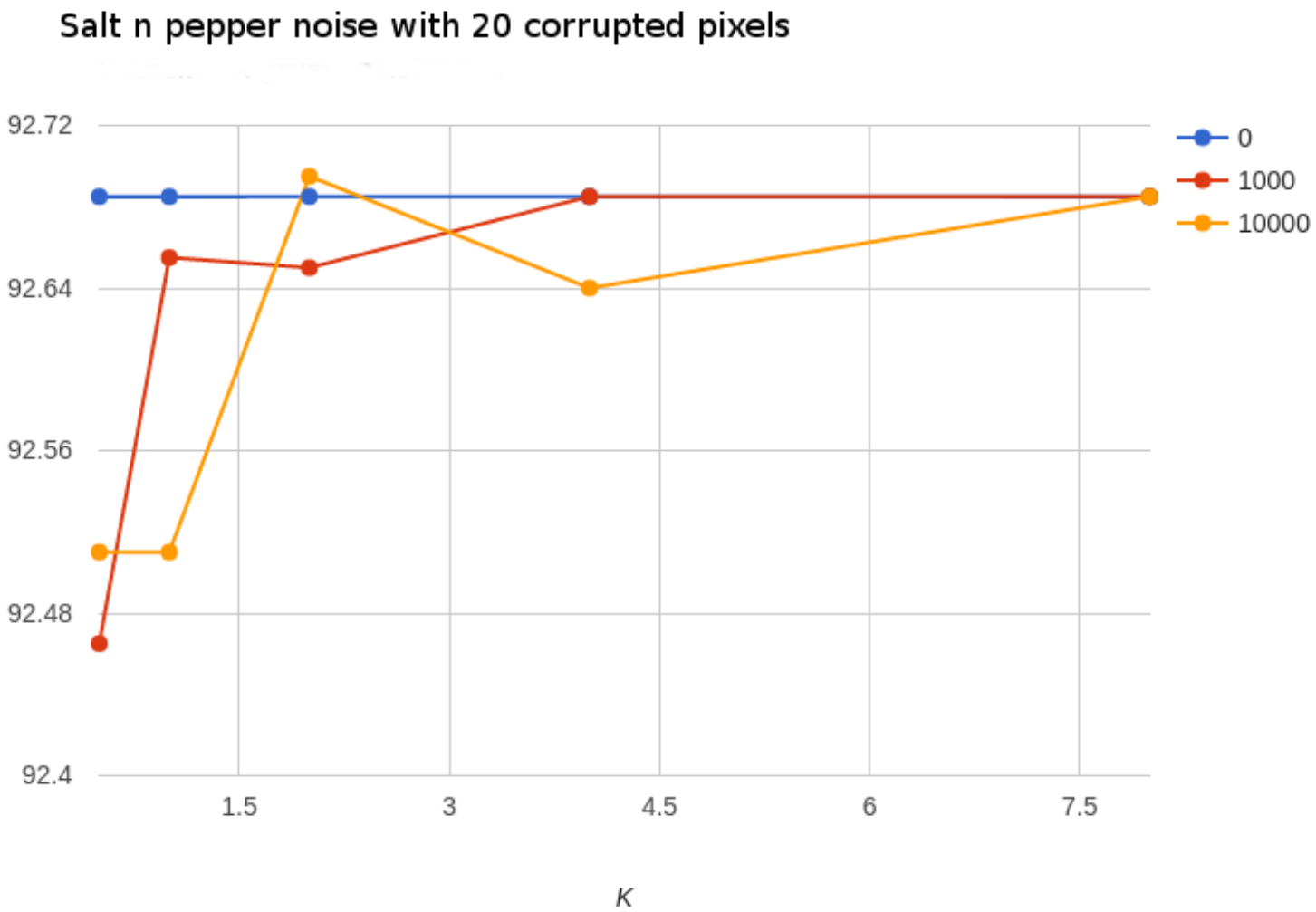
Performance of gaussianNN regularizer

- For low gaussian noise, the general trend is that lower values of K (contribution of large neighborhood) perform better
- Relatively large C is optimal for larger gaussian noise.
- At low noise levels, performance is optimal whenthere is contribution from a larger neighbourhood. While for high noise scenario, a large neighbourhood may spoil classification accuracy because the model is lead to believe that two dissimilar elements should have close by sparse codes, and hence a smaller neighborhood is preferred.
- The accuracy vs noise graph shown is for a given parameter setting ($C = 2$ and $\alpha = 10000$). For the same weight α and for smaller neighbourhood $C = 8$, test accuracy is almost the same as that of baseline.

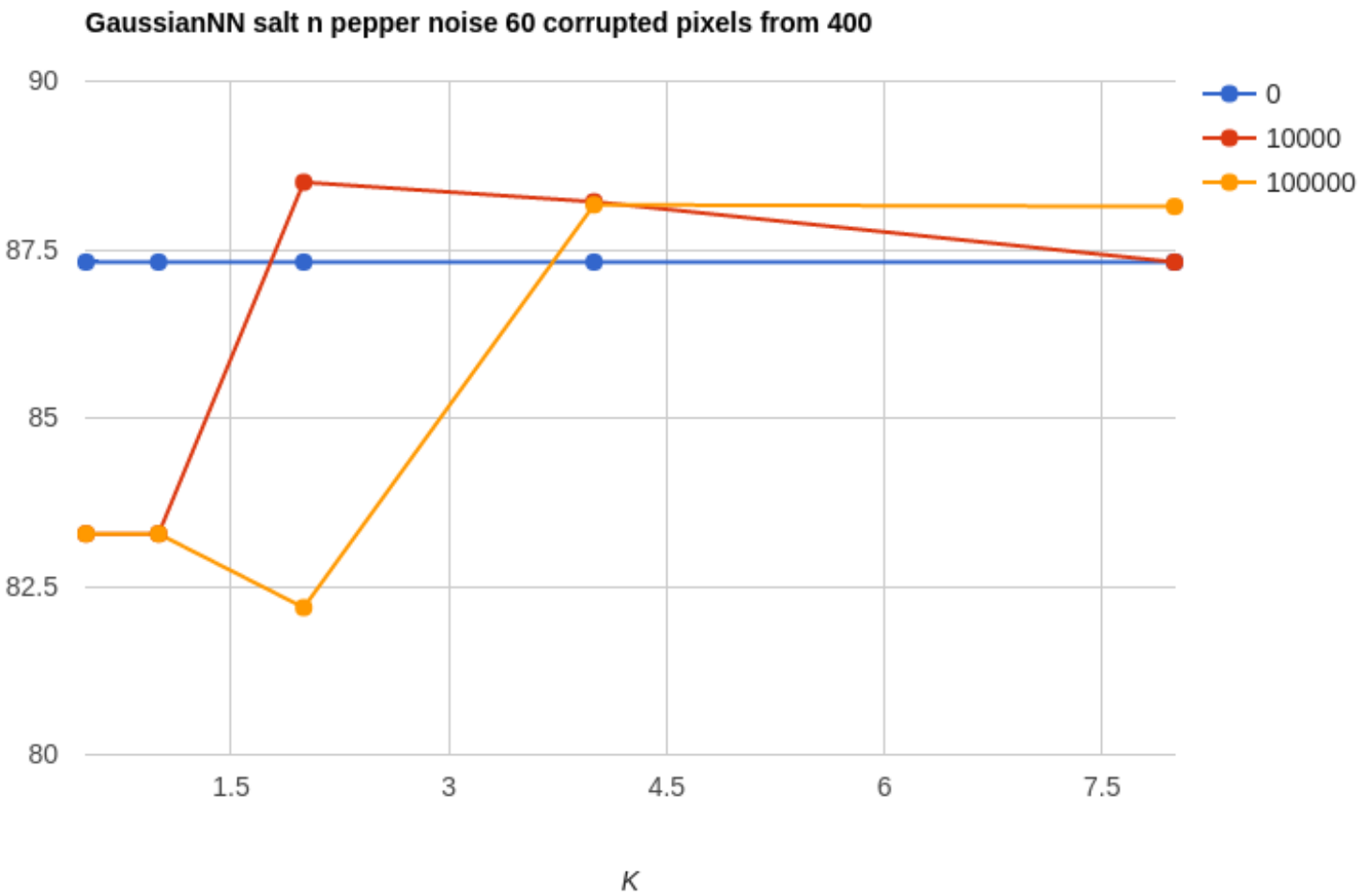
Performance of gaussianNN regularizer



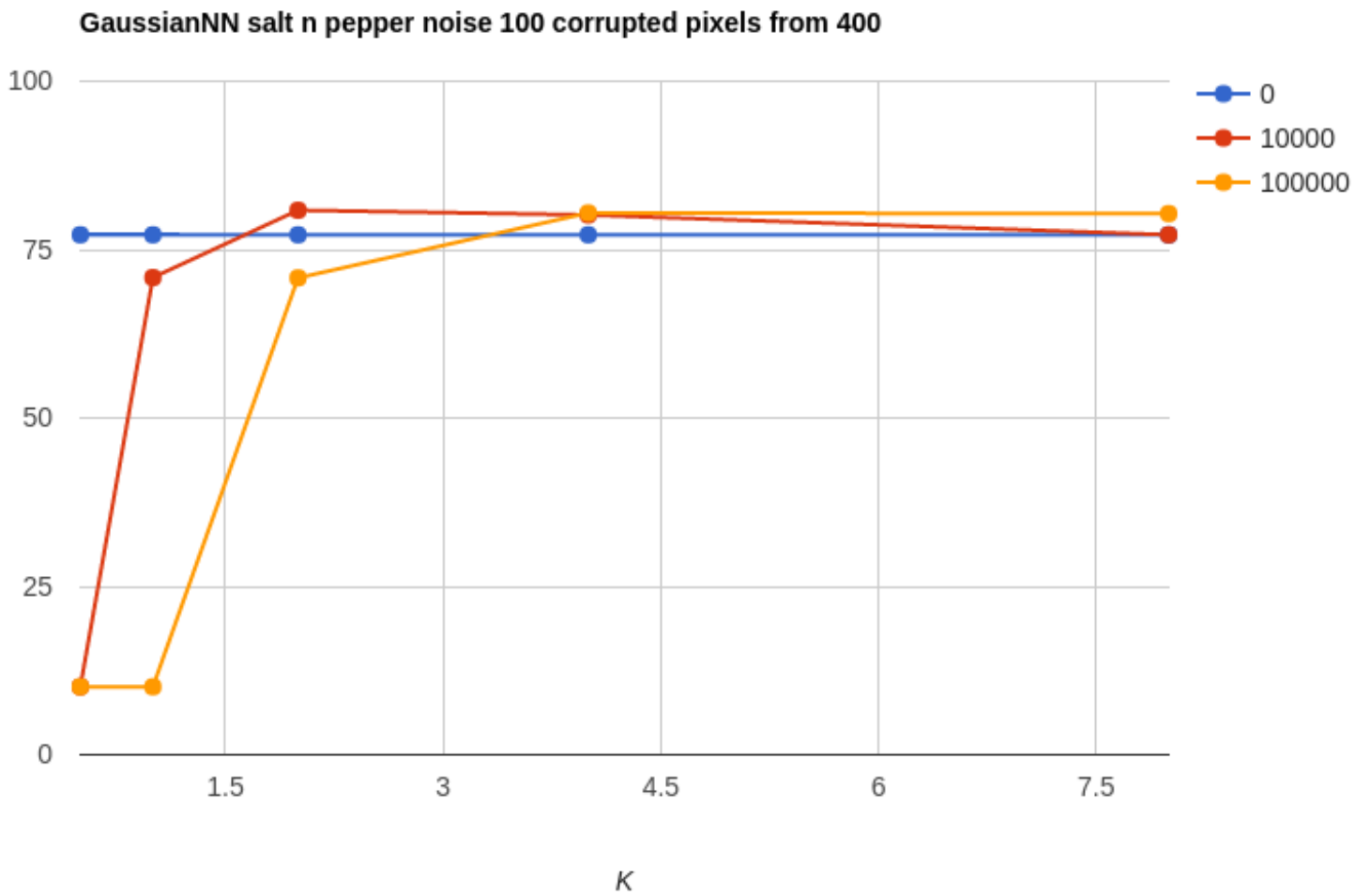
Performance of gaussianNN regularizer



Performance of gaussianNN regularizer



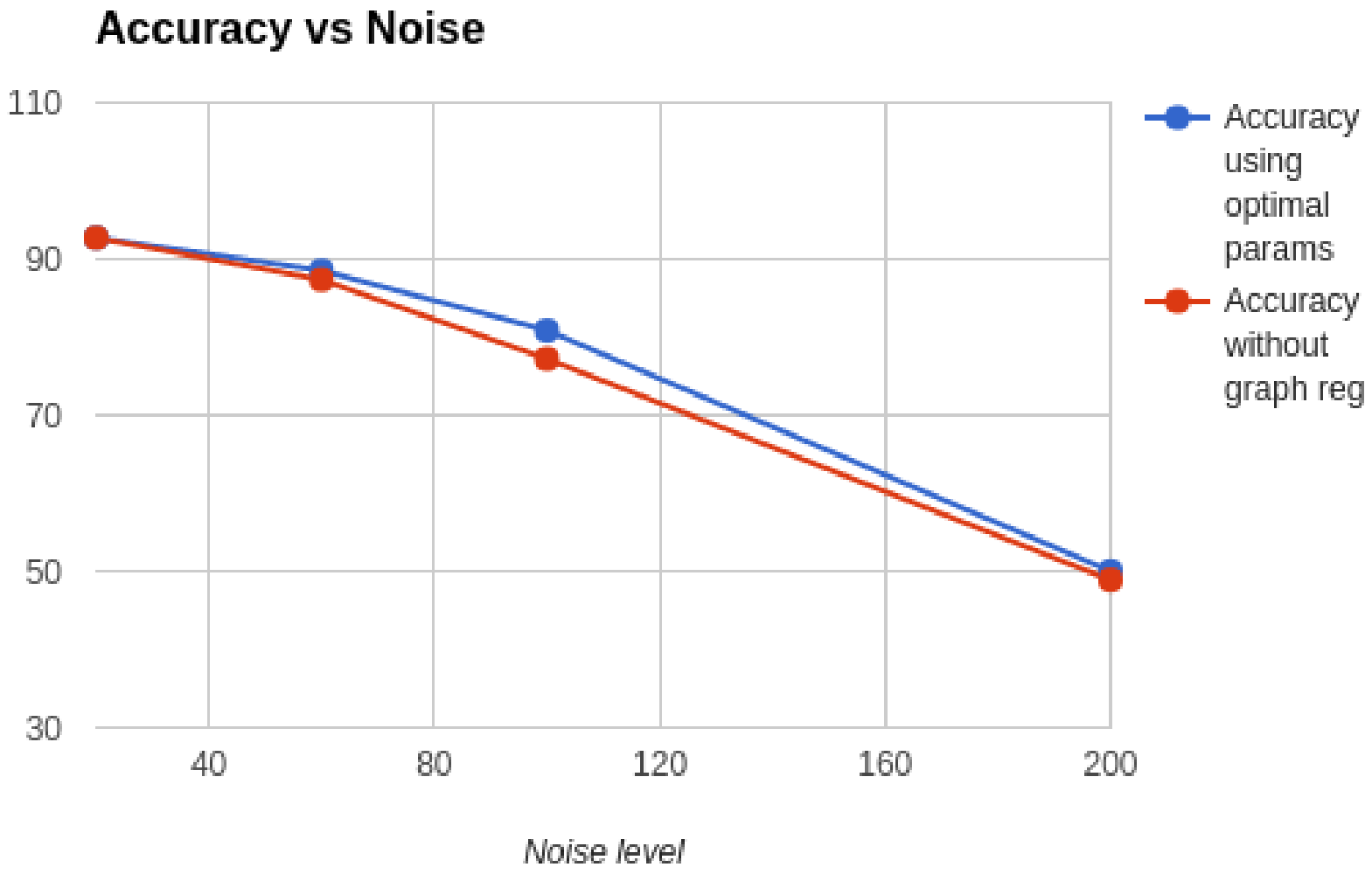
Performance of gaussianNN regularizer



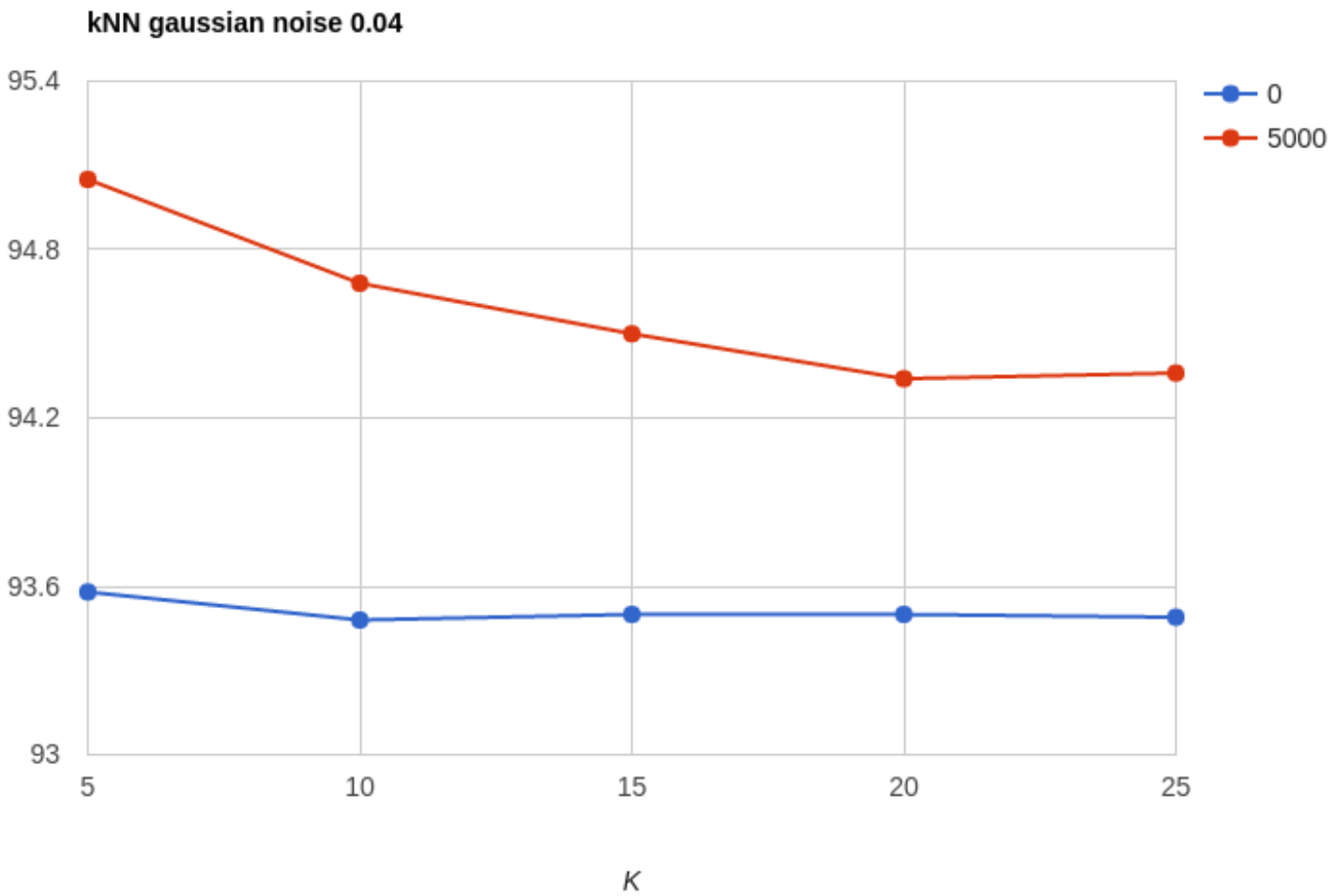
Performance of gaussianNN regularizer

- For low noise, there isn't a big improvement over baseline. For higher C, performance approaches that of the baseline. Seems to suggest that even a few pixels with high intensities can alter distances to the extent that they put different class images close.
- For large noise levels too, the optimal value of C is relatively large.
- For 60 corrupted pixels, test accuracy increase is $\sim 1\%$.
- For 100 corrupted pixels, test accuracy increase is $\sim 3\%$.
- For 200 corrupted pixels, test accuracy increase is $\sim 1.1\%$.

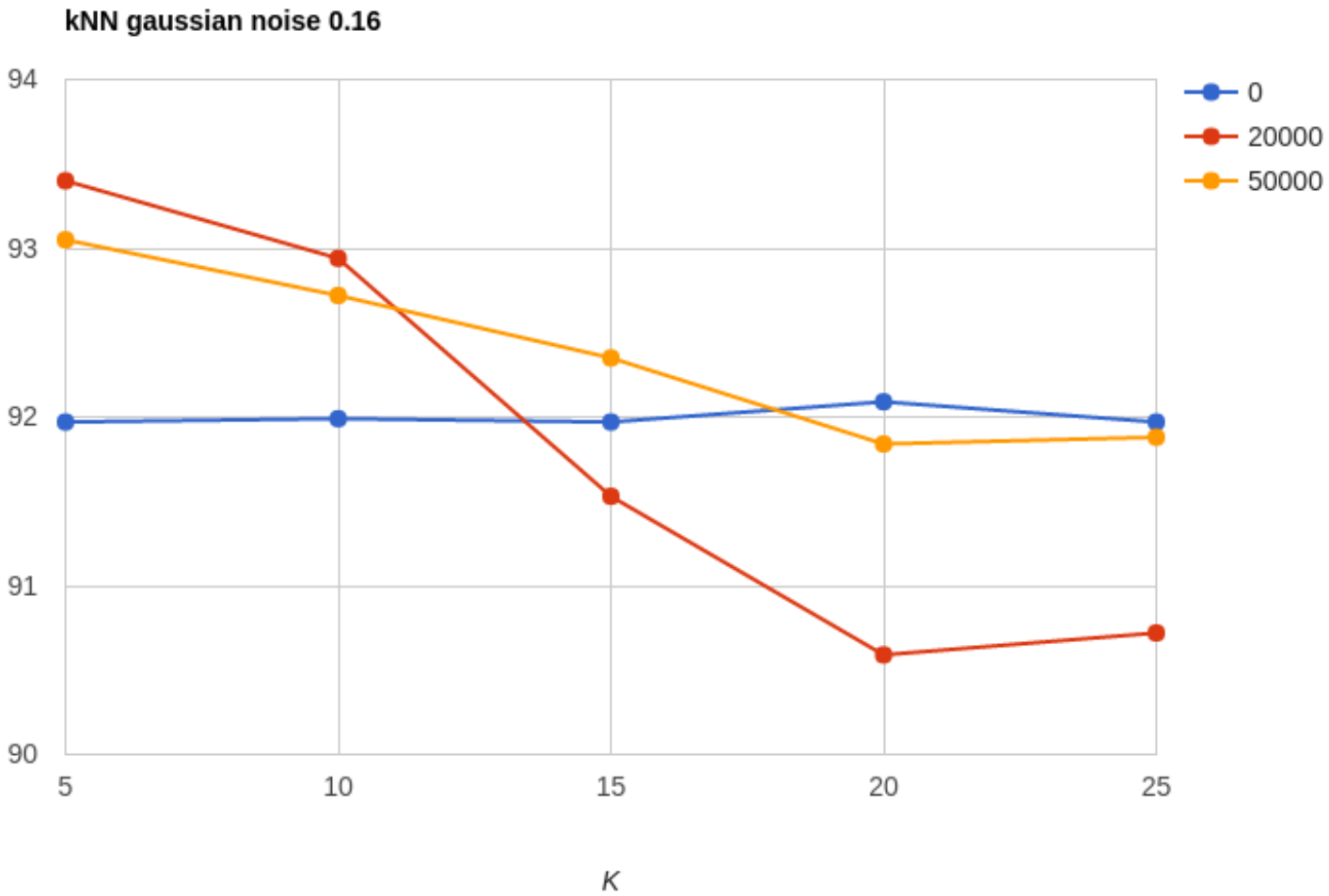
Performance of gaussianNN regularizer



Performance of kNN regularizer



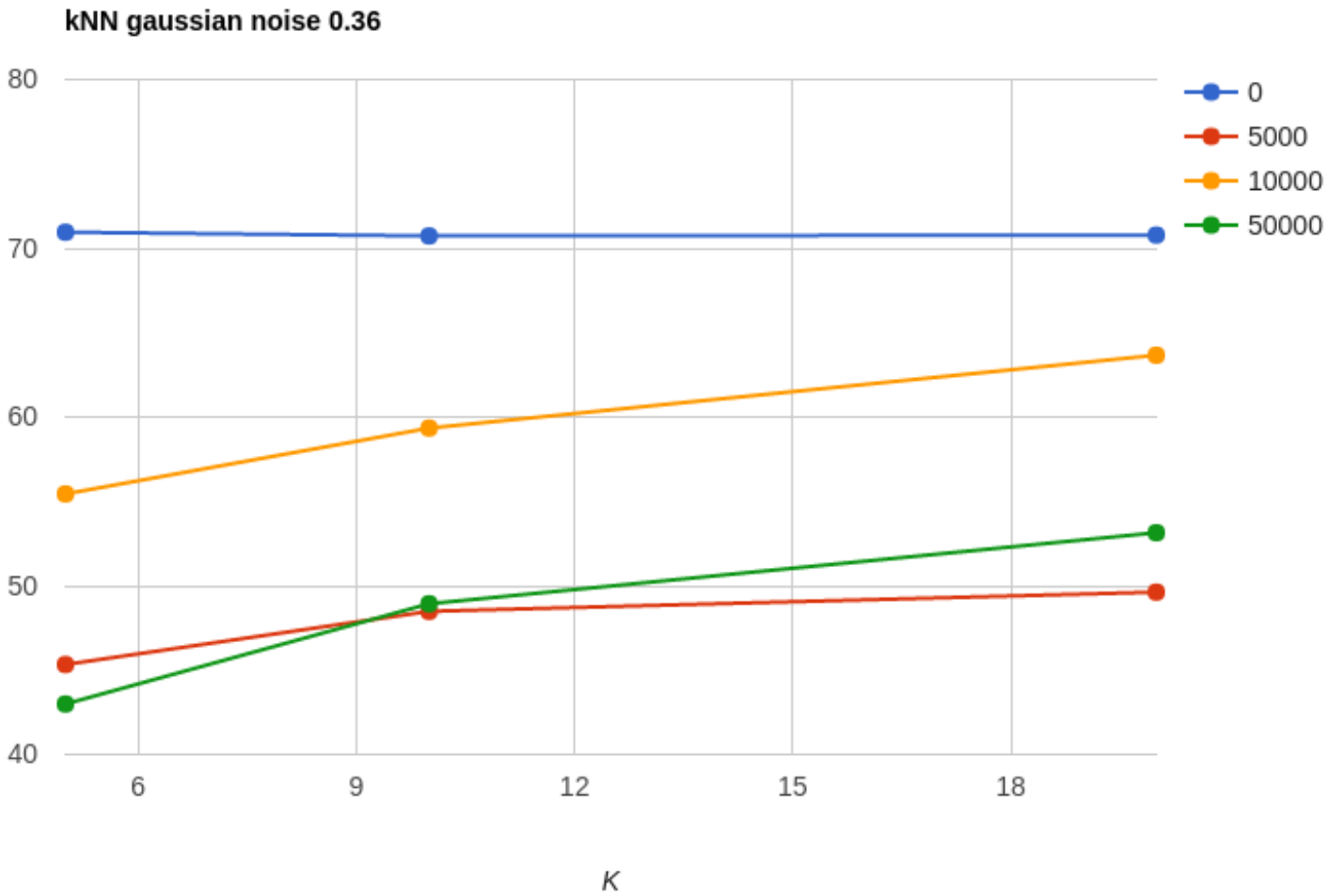
Performance of kNN regularizer



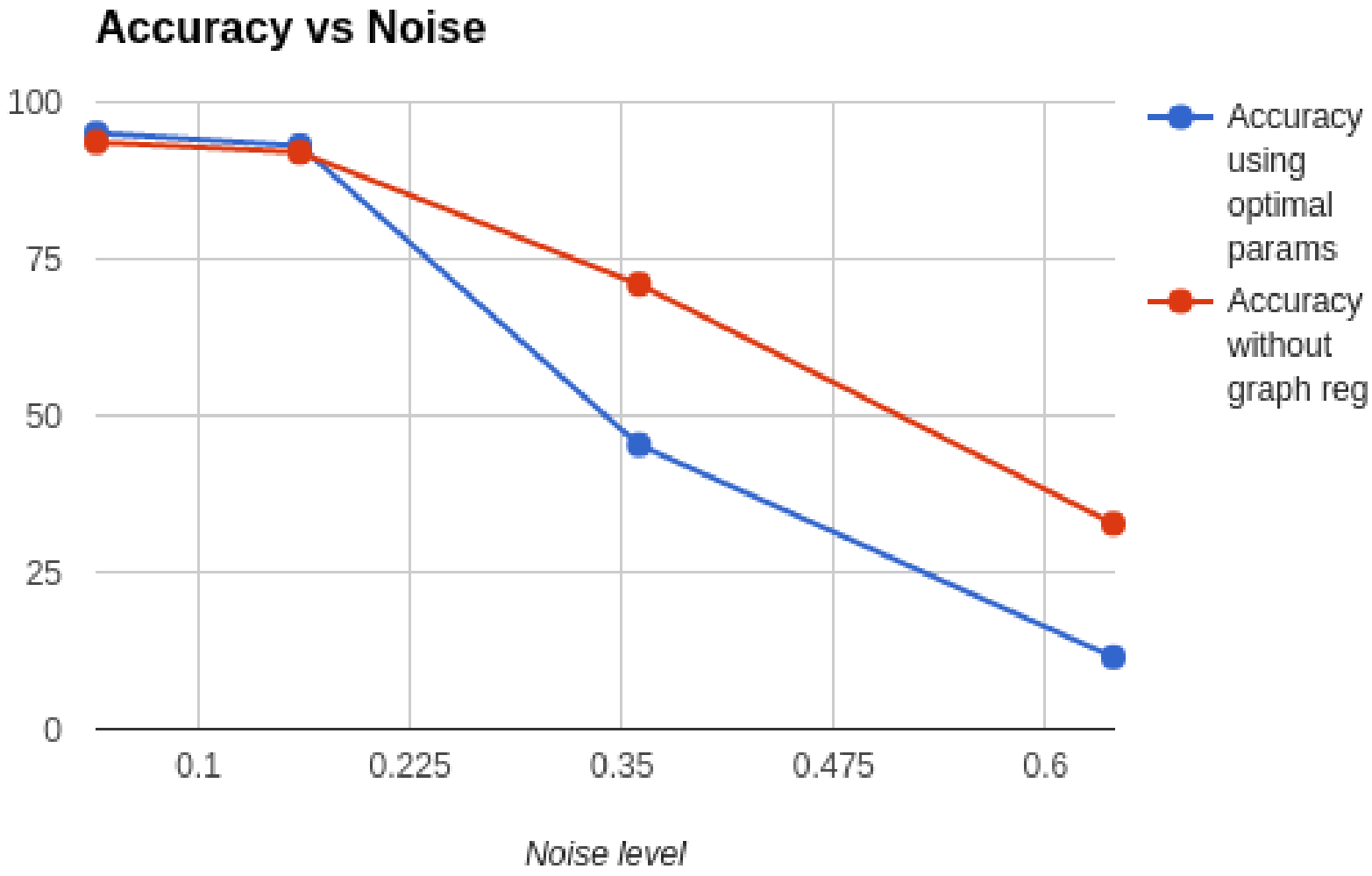
Performance of kNN regularizer

- For both low gaussian noise (std dev = 0.2) and intermediate noise (std dev = 0.4), a low value of K seems optimal.
- kNN is quite a non-smooth regularizer. May make the model prone to noise. Noisy data could make a data element from a different class seem to belong to the same class and make another point of the same class fall out of the neighbourhood, probably because of equal contribution from all neighbourhood data points. This effect seems to get magnified for larger k.
- Above might be the reason why for high noise levels, our model doesn't even put up a competetive performance against baseline.
- For noise levels where it can perform reasonably well, the test accuracy improvement is $\sim 1.5\%$

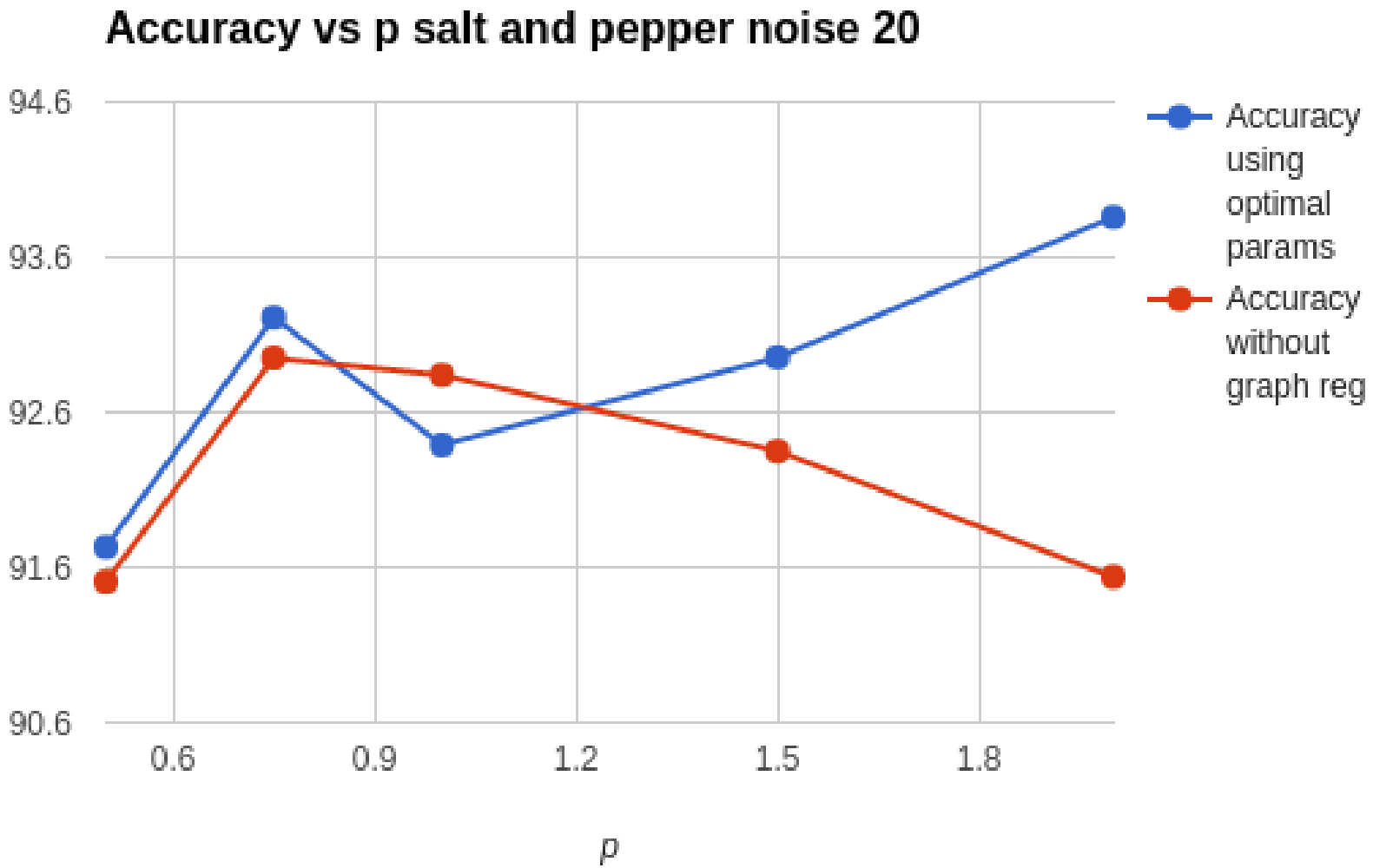
Performance of kNN regularizer



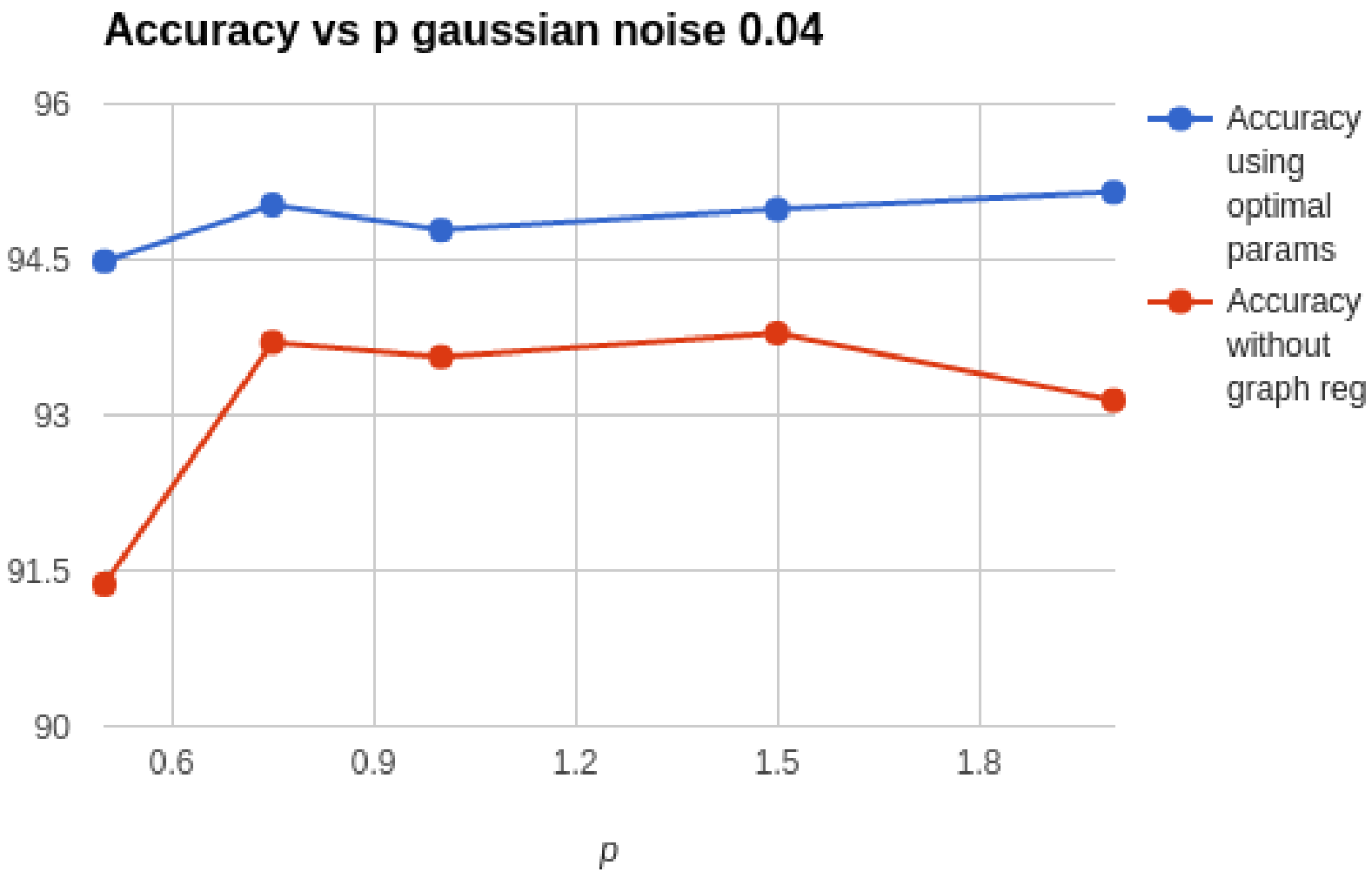
Performance of kNN regularizer



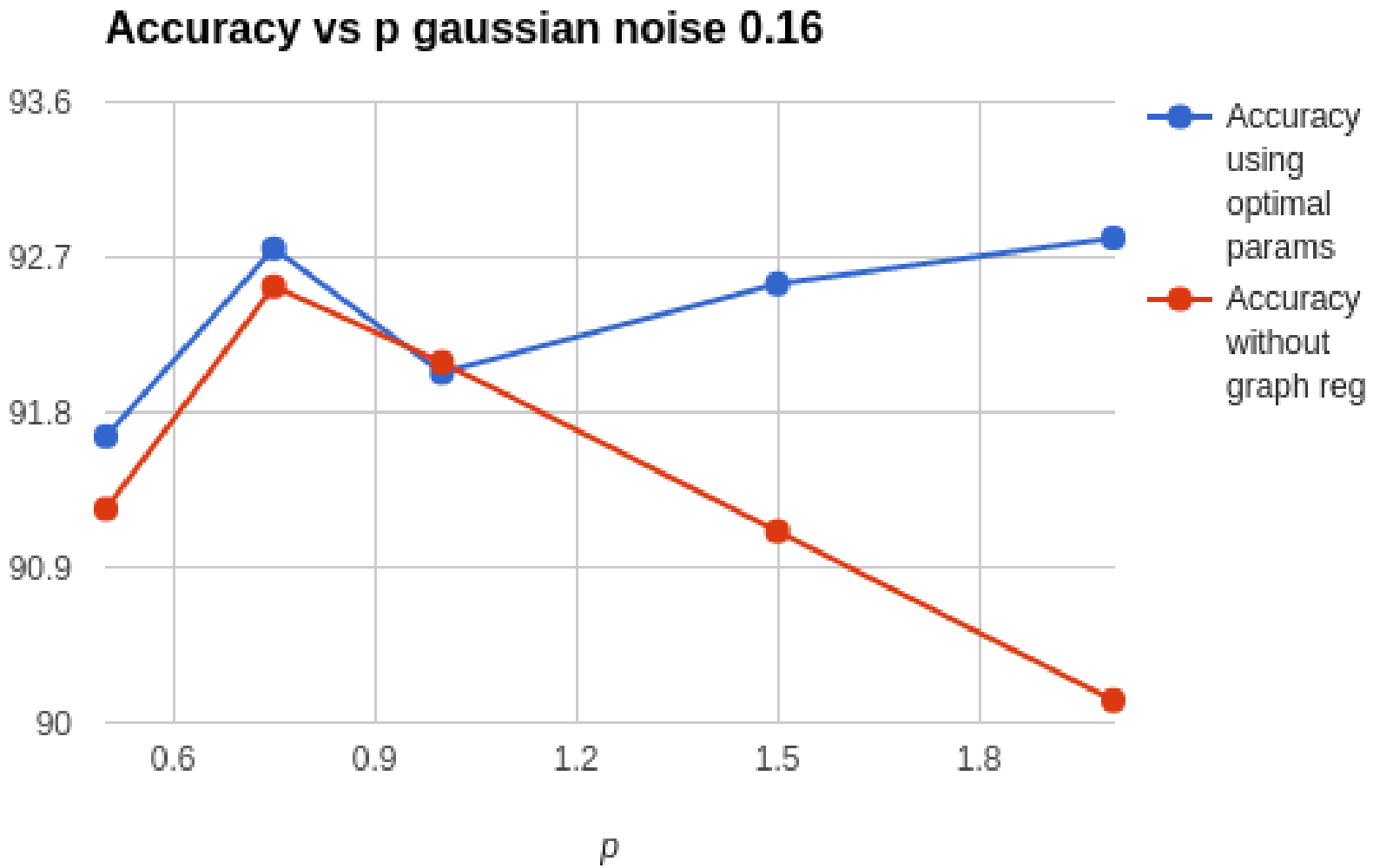
Performance with varying p



Performance with varying p



Performance with varying p



Performance with varying p

- At low and intermediate noise levels (where kNN was able to perform reasonably well), it performed quite better for $p = 2$ than it did for $p = 1$ or lower.
- At $p = 2$, test accuracy for our model is even better than baseline at $p = 1$, from which it might be inferred that kNN graph regularization can act as a robustness measure on its own (upto a certain extent of noise).
- This however, is not seen in gaussianNN where, though the performance improved for $p = 1$ (as compared to $p = 1$ for baseline), $p = 2$ was usually just a little better or no better than baseline with $p = 2$.

Discussion and Future Work

- We tried different variants of graph regularization, and found improvements over the baseline, for the variants in different noise scenarios. However, we need to run more tests to better understand the effect of the parameters involved. We could thus come up with a model that can perform significantly better than the baseline at all noise levels.
- Further, we aim to study the improvement that graph regularization can deliver in case of kernel dictionary learning on the sphere (i.e., explicitly using the manifold structure on which data lies in the kernel feature space)