Application and Optimization of a Two-layer Perceptron for Digit Recognition

Rui Ma

12/4/2017

Outline

- ❑ Introduction
- ❑ Parameter tuning and its results
  - # Hidden units
  - Learning rate
  - Batch size
  - Epoch
- ❑ Dataset filtering and its performance
- ❑ Conclusion

MNIST Dataset

❑ Images of handwritten digits

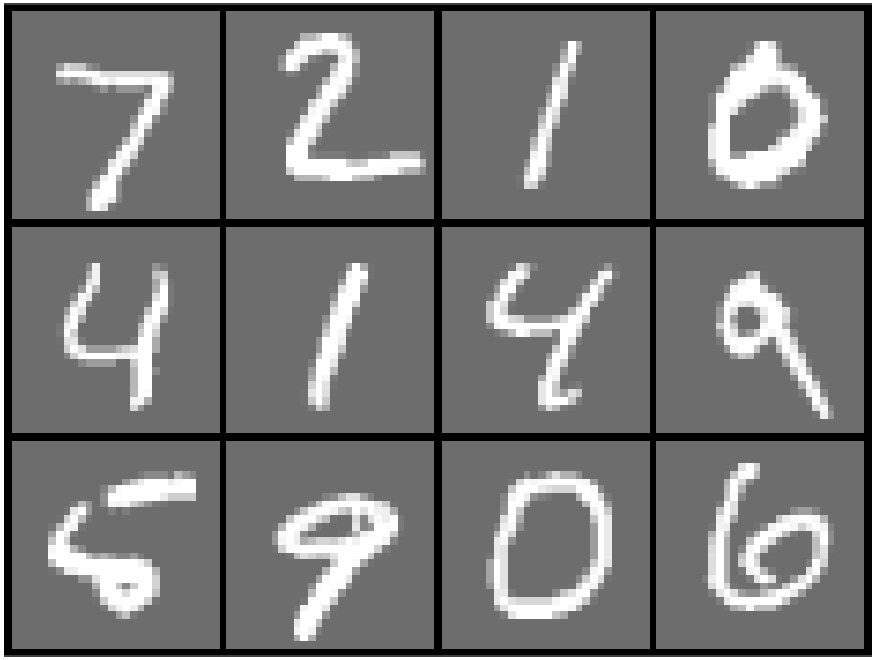


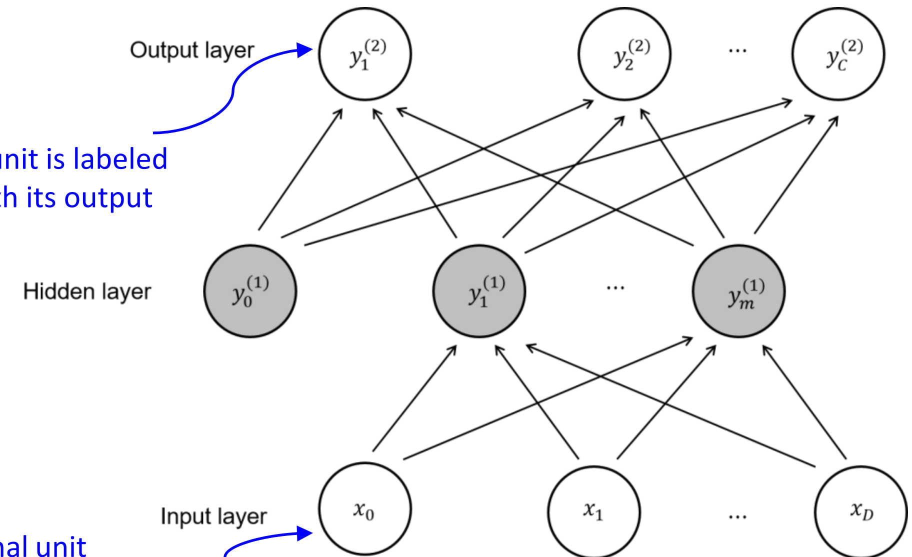
Fig. 1. Typical images from the training set

- Each image: 28 × 28 (=784) pixels
- Each pixel: [0 (dark), 1 (bright)]
- Training Set: 60,000 images, 784*60,000 matrix
- Validation Set: 10,000 images, 784*10,000 matrix

Two-layer Perceptron

University of Minnesota
Driven to Discover

A unit is labeled with its output



Additional unit $x_0 = 1$ to include the bias as weight

Fig. 2. Typical images from the training set

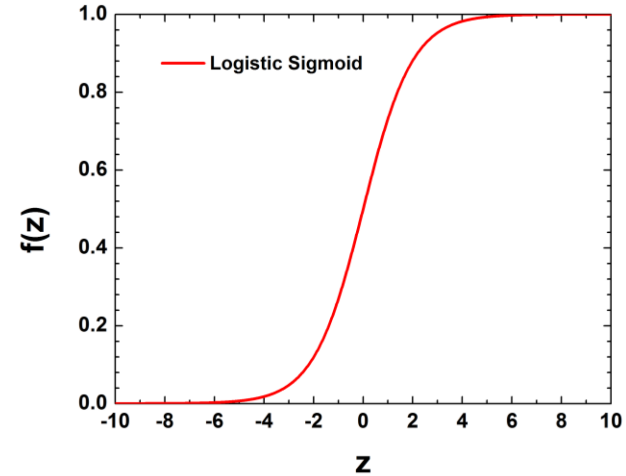


Fig. 3. The logistic sigmoid function

- Activation function for a single unit, i: **sigmoid function**

$$f(z_i) = \frac{1}{1+e^{-z_i}}\text{, where } z_i \text{ is the actual input } z_i = \sum_{k=0}^{D} w_{ik} x_k$$

- Error measure: **Sum-of-squared error function**

k^th entry of the n^th output value

k^th entry of the n^th target value

$$E(w) = \sum_{n=1}^{N} E_n(w) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}(y_k(x_n, w) - t_{nk})^2$$

Two-layer Perceptron

- Training approach: **Stochastic training**
  - Randomly choose an input value and propagate it through the network
  - Update the weights based on the error $E_n(w)$

- Optimization algorithm: **Gradient descent**

$$w[t+1] = w[t] + \Delta w[t], \text{ where t is the } t^{th} \text{ iteration}$$

$$\Delta w[t] = -\gamma \frac{\partial E_n}{\partial w[t]}, \text{ where } \gamma \text{ is called learning rate}$$

- Gradient evaluation: **Error Backpropagation**

$$\frac{\partial E_n}{\partial w_{ij}^{(l+1)}} = \delta_i^{(l+1)} y_j^l, \text{ where error } \delta_i^{(l)} := \frac{\partial E_n}{\partial z_i^{(l)}}$$

$$\delta_i^{(l)} = f'(z_i^{(l)}) \sum_{k=1}^{m^{(l+1)}} w_{ik}^{(l+1)} \delta_k^{(l+1)}$$

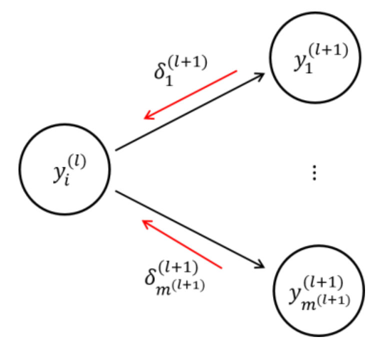


Fig. 4. Error Backpropagation

Errors for output units:

$$\delta_i^{(L+1)} = y_i(x_n, w) - t_{ni}$$

$f(z_i) \rightarrow$ Sigmoid
$E_n \rightarrow$ Sum-of-squared

Parameter Tuning

UNIVERSITY OF MINNESOTA
Driven to Discover

- **Parameters:**
  - **Number of hidden units, m**
  - **Learning rate, γ**
  - **Batch Size**: the number of randomly selected input values for each training iteration
  - **Epoch**: the number of training iterations

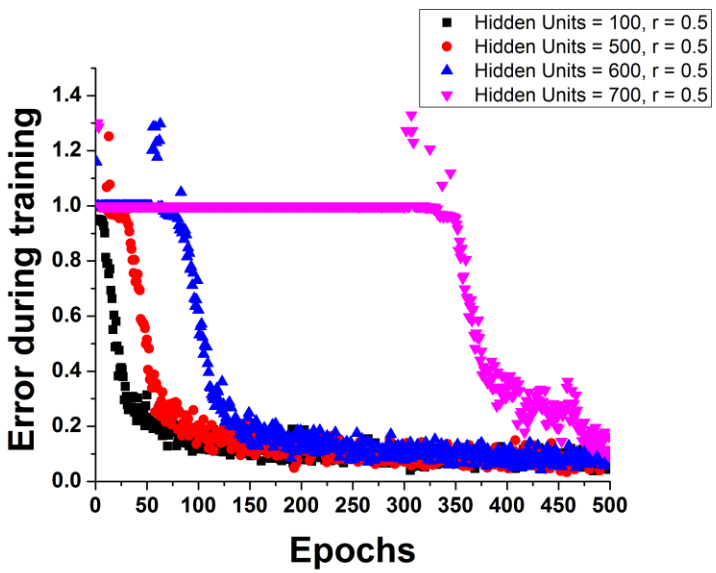


Fig. 5. Error during training for different numbers of hidden units. The batch size is 100.

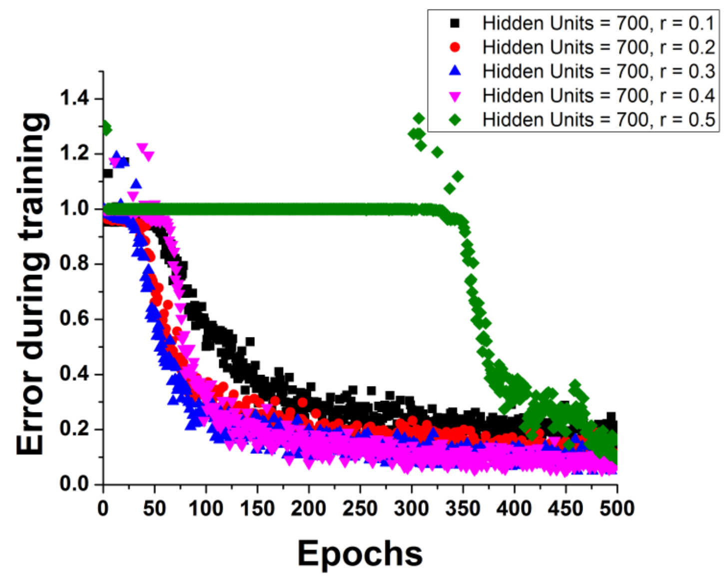


Fig. 6. Error during training for different learning rates. The batch size is 100.

- Want sufficient large network but avoid overfitting
- Want fast learning but avoid oscillation

Tuning Results

- Accuracy on validation set

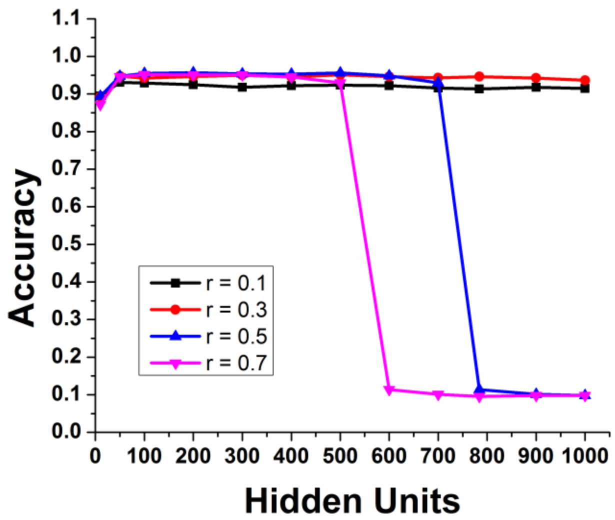


Fig. 7. The dependence of the number of hidden units on the accuracy. The perceptron was trained with a batch size of 100 randomly chosen images iterated 500 times.

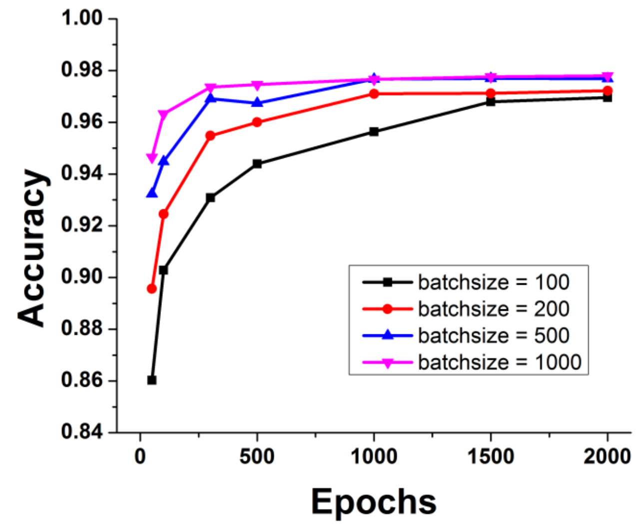


Fig. 8. The impact of increasing the number of epochs. The number of hidden units is 100 and the learning rate is 0.5.

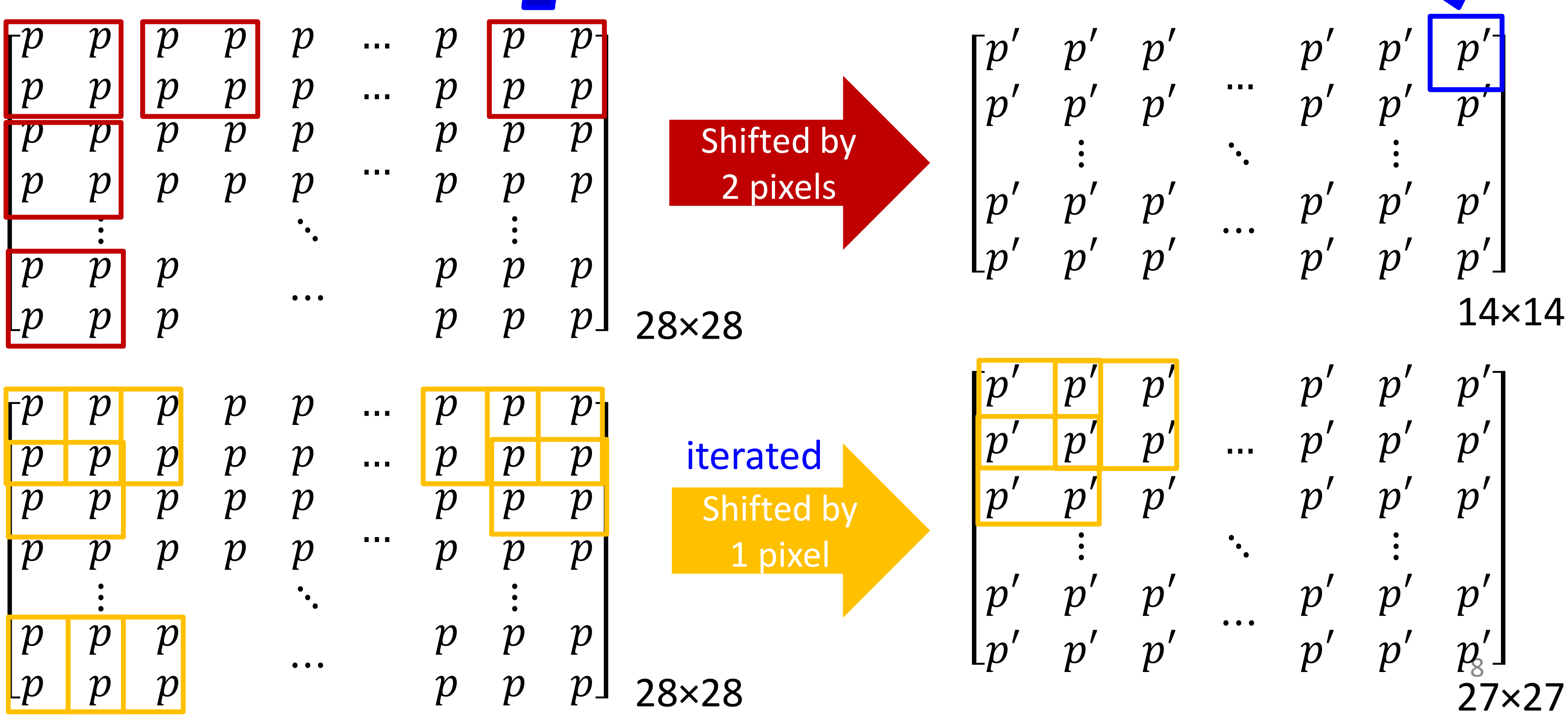
- Smaller learning rate for larger number of hidden units, but worse results for fewer hidden units
- The accuracy increases with rising training set.

Dataset Filtering

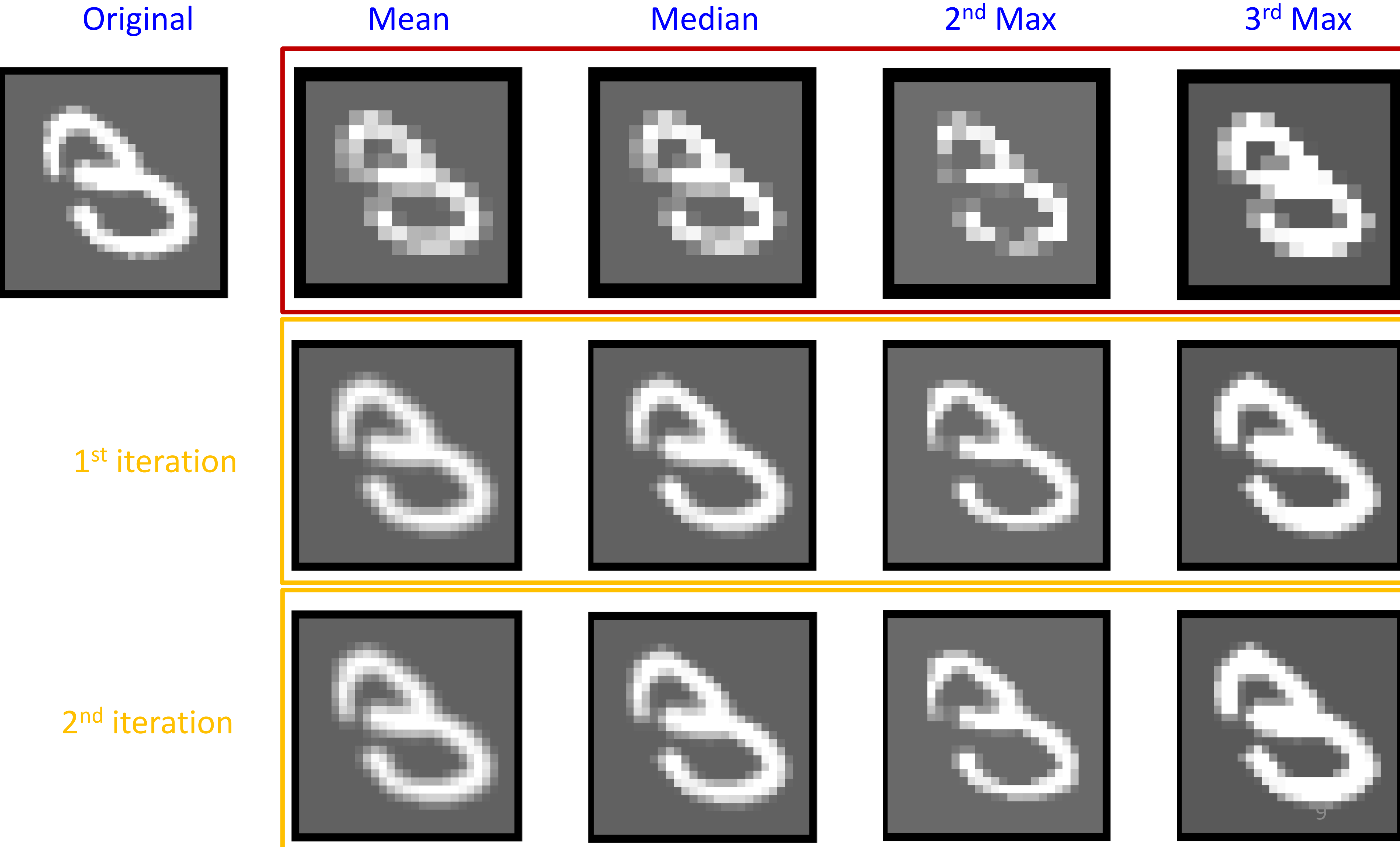
- **Motivations for filtering the input dataset**
  - Achieve more efficient training
  - Avoid the curse of dimensionality
  - Avoid overfitting

Kernels:
(1) Mean
(2) Median
(3) Second largest value
(4) Third largest value

- **Filtering approaches**



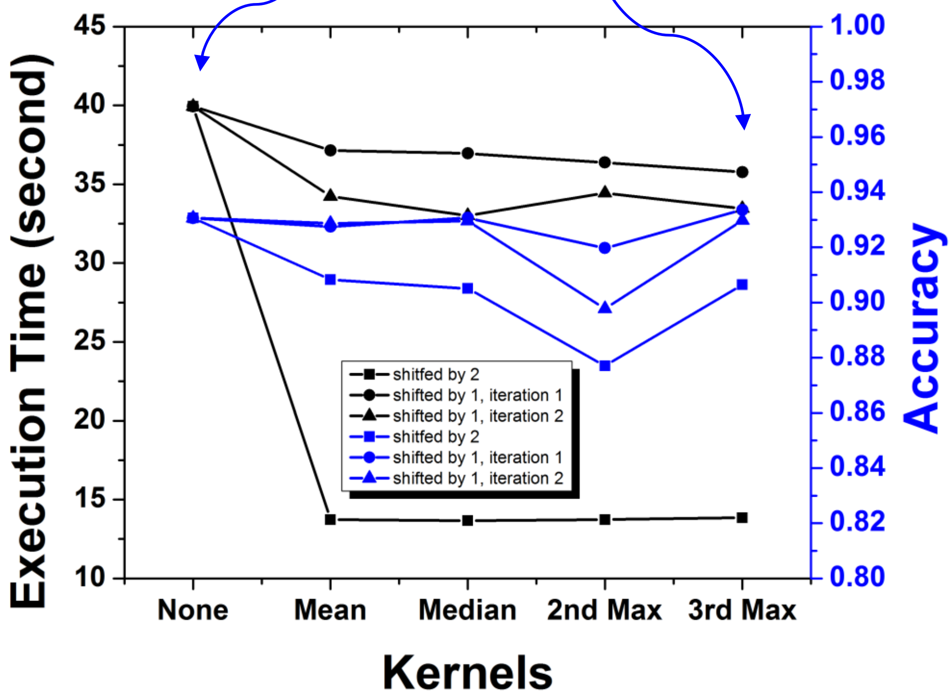
Filtering Results



Filtering Results

❑ Performance Comparison

Each point is the average value of 5 executions



Only included training and testing time

Fig. 9. The effects of different filter methods and Kernels on execution time and accuracy. The perceptron was trained with 100 hidden layers, 0.1 learning rate, 100 batch size and 500 epochs.

Conclusion

- The two-layer perceptron can be optimized by tuning the number of hidden units, learning rate, batch size and epochs.
- Efficiency and accuracy of the perceptron can be further improved by applying data filtering with proper kernels
- Future work includes trying other data filtering techniques and kernels

References

- David Stutz, “Introduction to Neural Networks”, *Selected Topics in Human Language Technology and Pattern Recognition WS 13/14*, 2014.
- Christopher M. Bishop, “Neural Networks for Pattern Recognition”, *Clarendon Press*, Oxford, 1995.
- Richard O. Duda, Peter E. Hart, and David G. Stork, “Pattern Classification”, *Wiley Interscience Publication*, New York, 2001.
- Simon Haykin, “Neural Networks A Comprehensive Foundation”, *Pearson Education*, New Delhi, 2005.

Questions

Thank you!