# Letter Image Recognition

**Source:** David J. Slate(1991) | UCI ML repository

Group 17 : Jacob & Mrigank

# Data description

→ Character images based on **20** different fonts

→ Each letter was randomly distorted to produce a file of **20,000** unique stimuli

→ Statistical moments and edge counts, scaled to fit into a range of integer values from 0 through 15

→ Attributes: 16 | Target: 26 classes

→ Uniform class distribution among 26 letters.

→ picture borrowed from: Frey, Peter W. & Slate, David J., 1991, Letter Recognition Using Holland-Style Adaptive Classifiers

# Models / Algorithms

→ Logistic Regression using sigmoid activations

→ Neural Networks with a hidden layer of ReLU activations and SoftMax output.

→ K-means / cut-off clustering

→ Ensemble: Logistic with clustering

# Logistic Regression

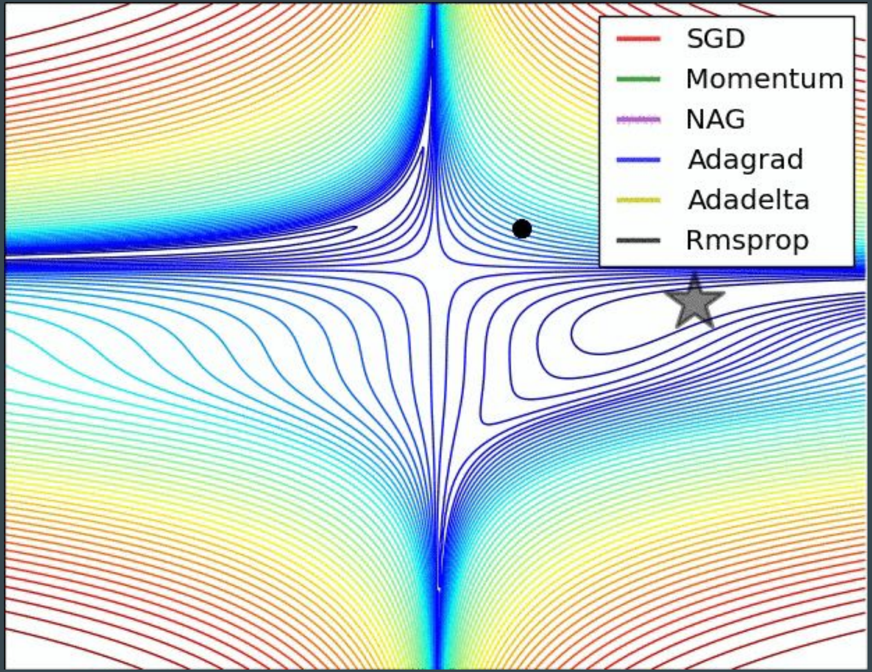
(Multinomial regression)

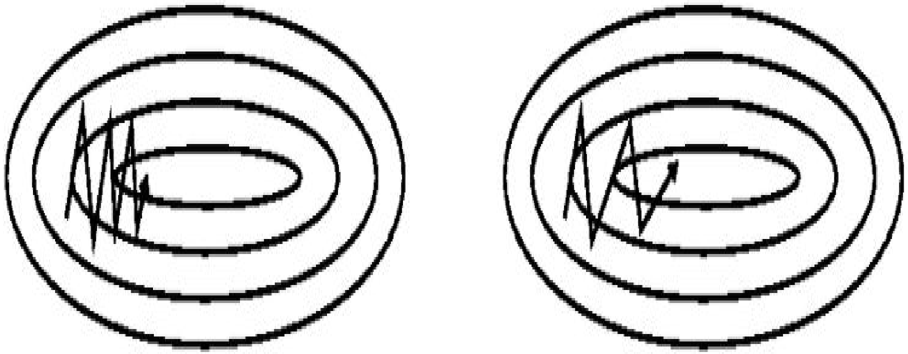
Prediction = maximum likelihood

Problems:

1. Bad performance before normalization
2. Sigmoid does not play well with gradient descent (saturation)
3. Sensitive to small variations

# Gradient Descent with Momentum (Karpathy, Andrej)



How to avoid finding local minima in the error function?

Nesterov momentum.

# Output Encoding

→ Logistic Regression

Actual targets: ‘A’, ‘B’, ...., ‘Z’

Targets encoded to 26 digits initialized to 0s.

Ex. ‘T’ →

```
(array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.]),
```

→ Neural Networks

Each of the 26 letters are assigned a number between 0-25, representing 26 different classes.

Ex. ‘T’ → 19

# Multilayer Perceptron

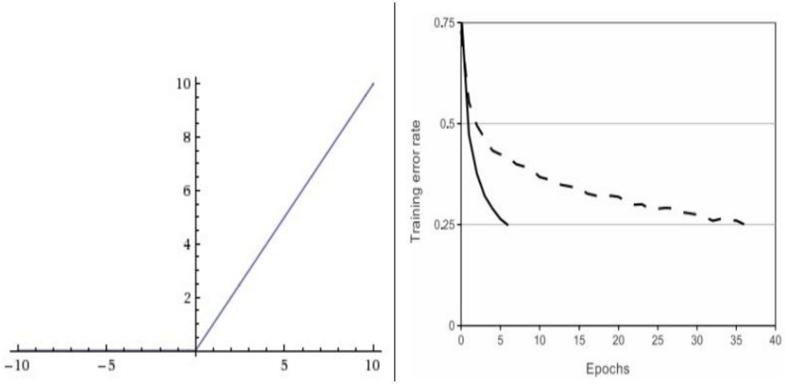
→ MLP Classifier with 1 hidden layer containing ReLU activations and the SoftMax output layer

→ ReLU pros: [Krizhevsky et al., 2012]

1. Does not saturate (in +ve regions)
2. Very computationally efficient
3. Converges much faster than sigmoid/tanh

→ SoftMax (generalization of sigmoid)

Squashes a k-dimensional vector to a k-D vector of real values(0,1) that add up to 1.




*Left*: ReLU activations | *Right*: the significant improvement in convergence with the ReLU unit compared to the tanh unit (reference)

$$p_k = \frac{e^{f_k}}{\sum_j e^{f_j}}$$

$$L_i = -\log(p_{y_i})$$

# Network Learning

→ Tried training networks with different activations

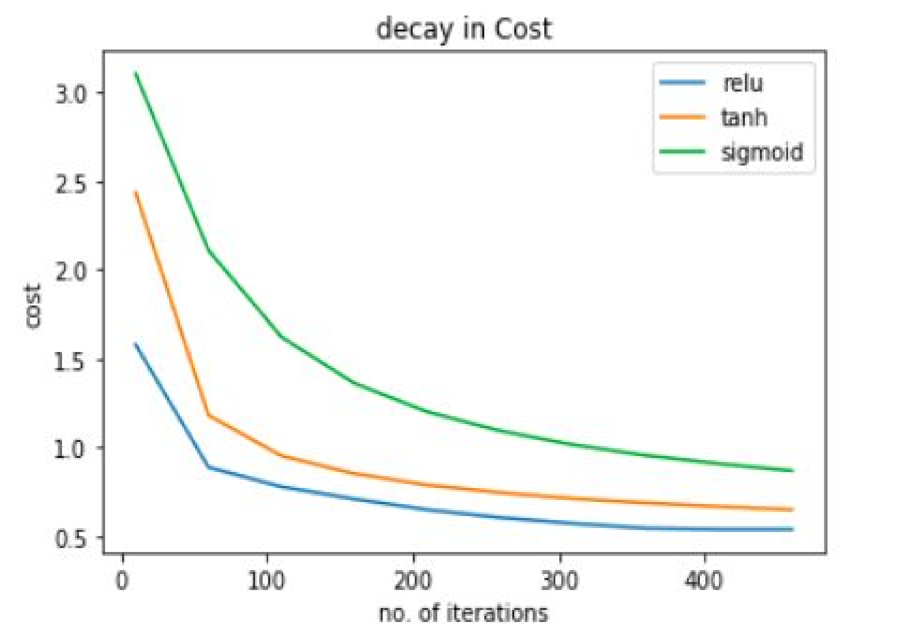
## Hyperparameters

```
clf1.get_params()
```

```
{'activation': 'relu',
 'alpha': 1e-05,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100,),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_iter': 200,
 'momentum': 0.9,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': 1,
 'shuffle': True,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
```

# Network Learning

→ Tried training networks with different activations




## Hyperparameters

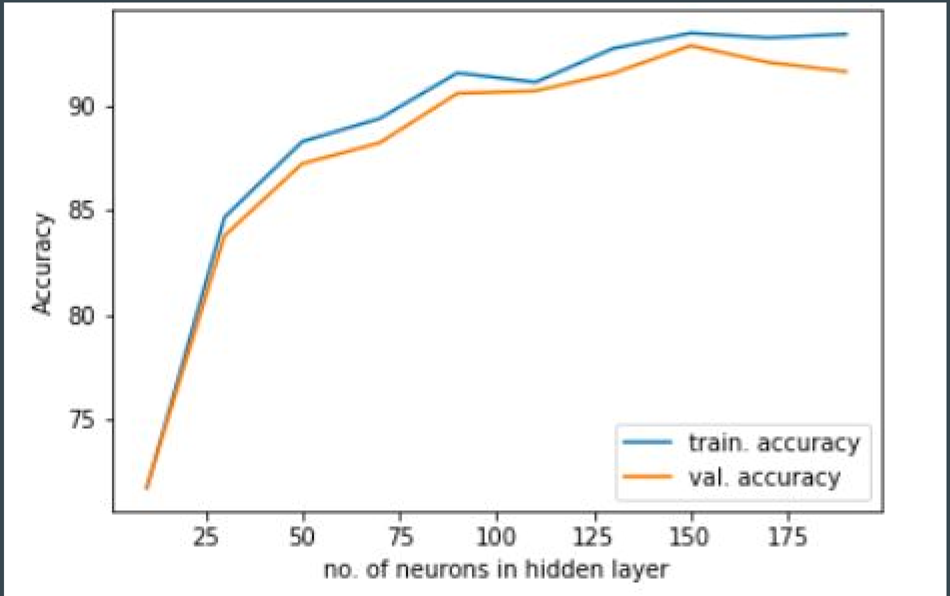
```
clf1.get_params()
```

```
{'activation': 'relu',
 'alpha': 1e-05,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100,),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_iter': 200,
 'momentum': 0.9,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': 1,
 'shuffle': True,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
```

# Prediction vs hyperparameters

→ Accuracy is on increasing trend with increase in number of hidden neurons

→ The white space between blue and yellow curve shows very little over-fitting.



# Clustering

K means

# clusters = 260

between_class_SS / total_SS = 0.857

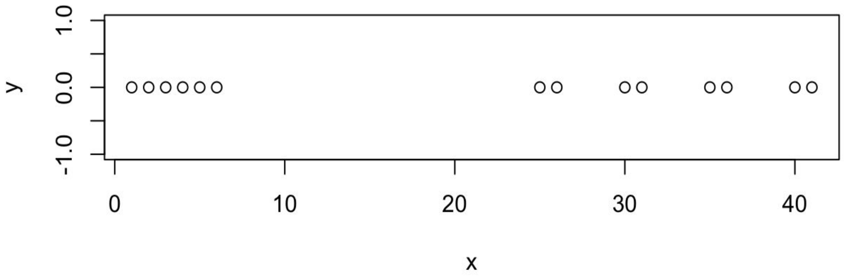
Average clusters per letter = 10

# Clustering

Cutoff Clustering

Each point is clustered with its “neighbors”: points within a certain cutoff distance

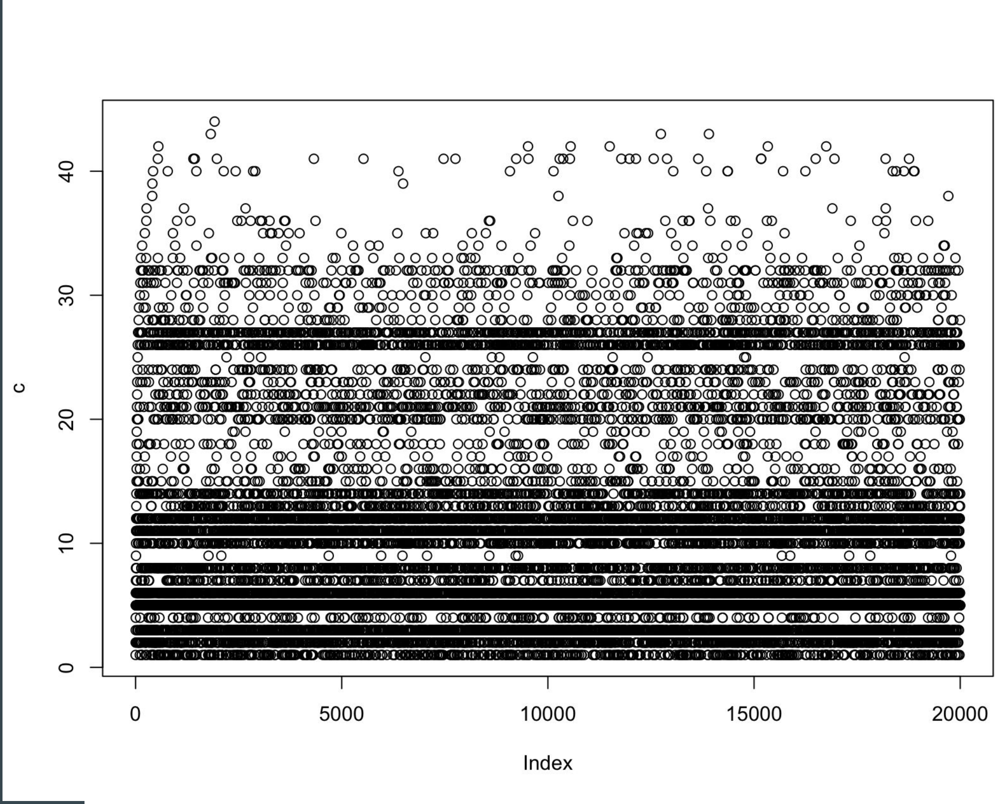
Based on cluster density, not size



# Clustering

Cutoff Clustering Results

Cutoff = 3.5



# Two-layer Ensemble

Clustering + Logistic Regression

Predictors = k-means label + cutoff-clustering label

# clusters in k-means = 26

Accuracy = 37%

# Performance

## Metrics

Accuracy

Maximal confusion rate

Average confusion rate

# Confusion Rate

| | G | C |
|---|---|---|
| G | 8 | 2 |
| C | 4 | 6 |

| | G | C |
|---|---|---|
| G | .8 | .2 |
| C | .4 | .6 |

Confusion rate
= 0.8*0.6 - 0.4*0.2 = 0.4

= P(G | G)*P(C | C) - P(G | C)*P(C | G)

# Performance

Logistic Regression

Accuracy: 73%

MCR: 0.65 (confusing O and H)

ACR: 0.97

# Performance

Multilayer Perceptron

Accuracy: 95%

MCR: 0.88 (confusing G and C)

ACR: 0.9875

# Performance

Ensemble Model

Accuracy: 37%

MCR: 0.007 (confusing S and Z)

ACR: 0.875

# Conclusion

Letter recognition can be hard, especially for noisy data

Good confusion rate shows low systematic misprediction

# Questions?

Models:

1. Logistic Regression
2. Multilayer Perceptron
3. K-means clustering
4. Cutoff clustering

Metrics:

1. Accuracy
2. Maximal Confusion Rate
3. Average Confusion Rate