Study of Spatial Biases in KNN Distance Metrics

David Pojunas

Advisor: Dr. Ferrer

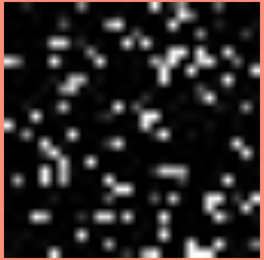
Motivation

- ❑ Spatial biases are usually ignored when classifying images, or extracting features
- ❑ A preliminary investigation on how spatial biases affect the error rate in k-nearest-neighbor classifiers using different distance metrics.
- ❑ Does permuting the order of image pixels significantly alter the classification accuracy?

Metrics

- ❑ Images
- ❑ BRIEF descriptors
- ❑ Dr. Ferrer’s Convolutional Method

$\stackrel{?}{=}$



BRIEF

Goal:

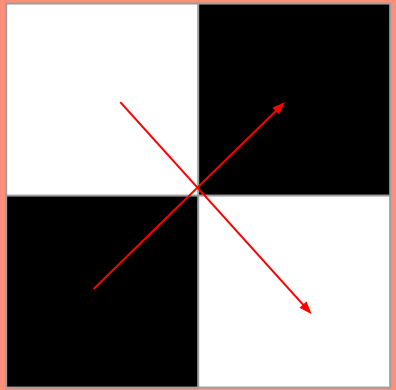
Represent features in an image with a binary representation of relations of pixels

Algorithm:

1. Pick *n* pairs of pixels (x,y)
2. If $x < y$, return 1
   If $x \geq y$, return 0

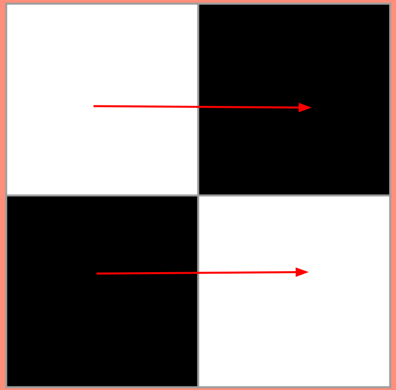
Images

Descriptors



Pairs: {(0,0),(1,1)}

Binary: {0,0}



Pairs: {(0,1),(1,0)}

Binary: {1,0}

Convolutional Method (One Image)

1

Extract 3x3 kernels from an image pixel by pixel

2

Cluster 8 kernels with K-means

3

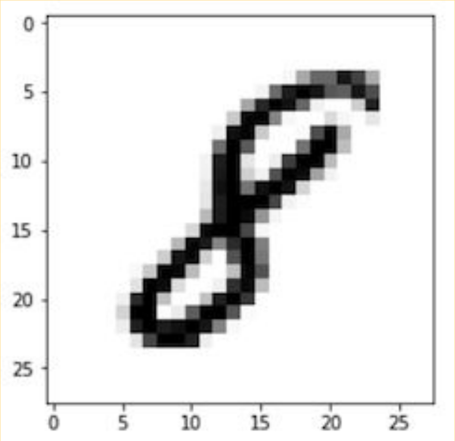
Convolve each kernel with the image

4

For each image we have 8 versions in an array, and we do this for all images

Evaluating KNN Distance Metrics

LOAD LABELED DATA FOR TRAINING AND TESTING



FILTER ALL IMAGES

BRIEF
Returns all images in a binary format defined by the spatial bias.

OR

CONVOLUTIONAL
Returns all images as a list of convolved images from each kernel

EVALUATE DISTANCES THROUGH KNN

BRIEF
Distance is defined as the hamming distance

OR

CONVOLUTIONAL
Distance is defined as the sum of distances between all kernized images

Dr. Ferrers Results

Error Rates (%)		
Distance Metric	Original	Permuted
Euclidean	3.12	3.08
Convolutional Euclidean	2.72	6.09
Uniform Classical BRIEF	3.47	3.54
Gaussian Classical BRIEF	3.98	5.54
3x3 Neighbor BRIEF	5.42	3.39
Uniform Neighbor BRIEF	3.44	3.47
Gaussian Neighbor BRIEF ($\frac{1}{3}$)	3.23	3.44
Gaussian Neighbor BRIEF ($\frac{1}{7}$)	3.47	3.41

Table 1: MNIST Error Rates (%)

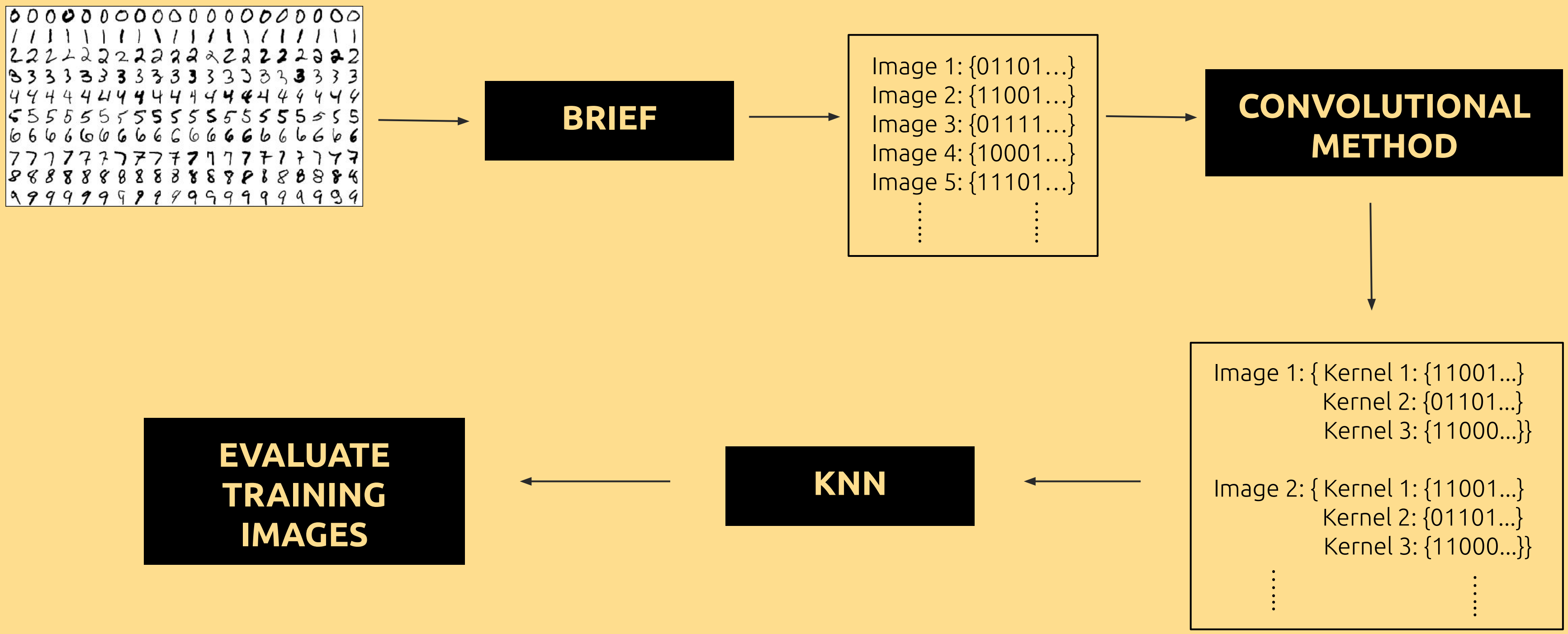
Execution Times (s)		
Distance Metric	Conversion	Evaluation
Euclidean	0	914
Convolutional Euclidean	3358.00	15414
Uniform Classical BRIEF	3.18	651
Gaussian Classical BRIEF	4.17	621
3x3 Neighbor BRIEF	3.76	655
Uniform Neighbor BRIEF	3.24	622
Gaussian Neighbor BRIEF ($\frac{1}{3}$)	3.25	637
Gaussian Neighbor BRIEF ($\frac{1}{7}$)	3.40	640

Table 2: Execution Times (s)

What’s Next

1) Generalize the convolutional method for any we can define a distance between
2) Test and research new distance metrics for BRIEF descriptors
3) Deploy these methods for classifying real time images
    a) Use image datasets derived from robots in realistic environments

Generalizing the Convolutional Method



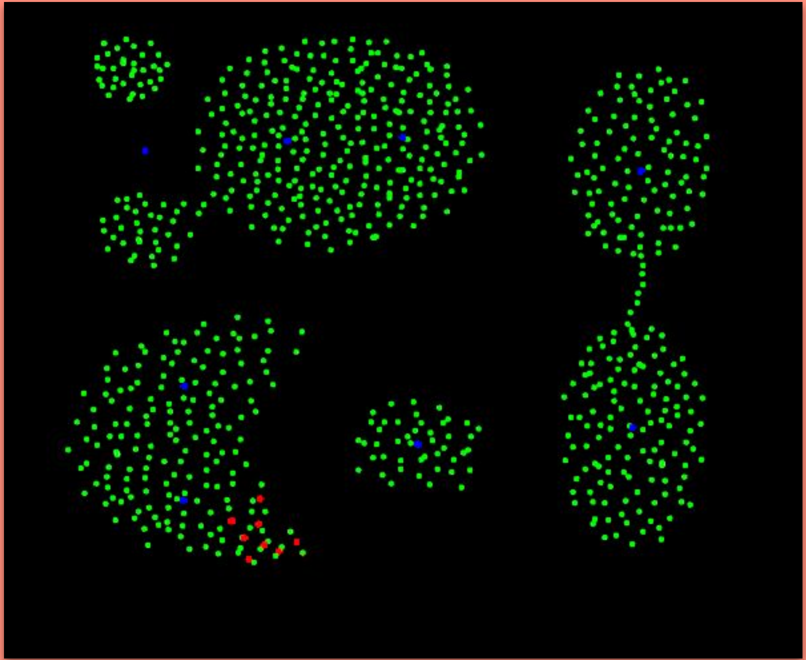
Issues with the Generalization

- **Convolving binary images was challenging**
  - Each convolution defined the distance between the kernel and neighborhood of pixels
- **Generalizing the code slowed down performance and caused BUGS!**
- **Increased the number of images needed to store in memory**
  - All training images, all BRIEF descriptors, and all convolved images
- **Slow and memory intensive process for running BASIC Images through the Convolutional Method**
  - Clustering images through k-means took a long time

New Direction

- We started to blame k-means for poor performance
- Dr. Ferrer had a replacement clustering algorithm
    - **Self Organizing Custers (SOC)**
- *Next step:* Implement and deploy new clustering algorithm

k-means Self Organizing Clusters

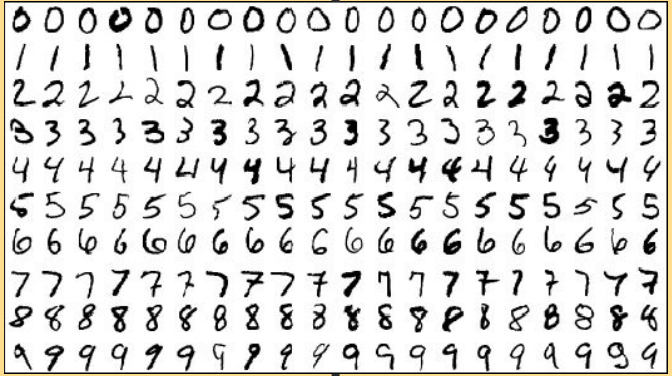


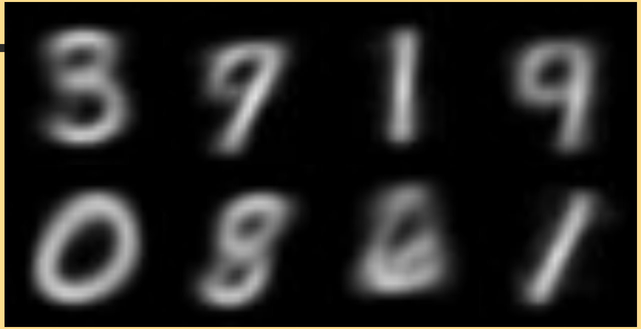
- k-means
  - Non-deterministic
  - Multiple passes of data
  - Batch clustering
  - Works well with images
- Self Organizing Clusters
  - Deterministic
  - One pass of data
  - Works incrementally
  - Does not work well with images

Self Organizing Clusters vs. *K-means*

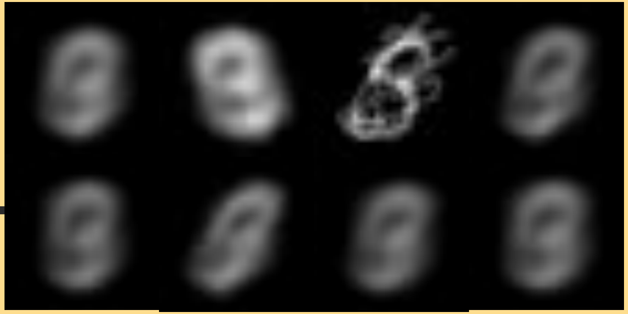
Testing Method: SOC vs k-means

60,000 Images





<u>K-MEANS</u>
3557.204 seconds



<u>SOC</u>
123.619 seconds

Initial Findings

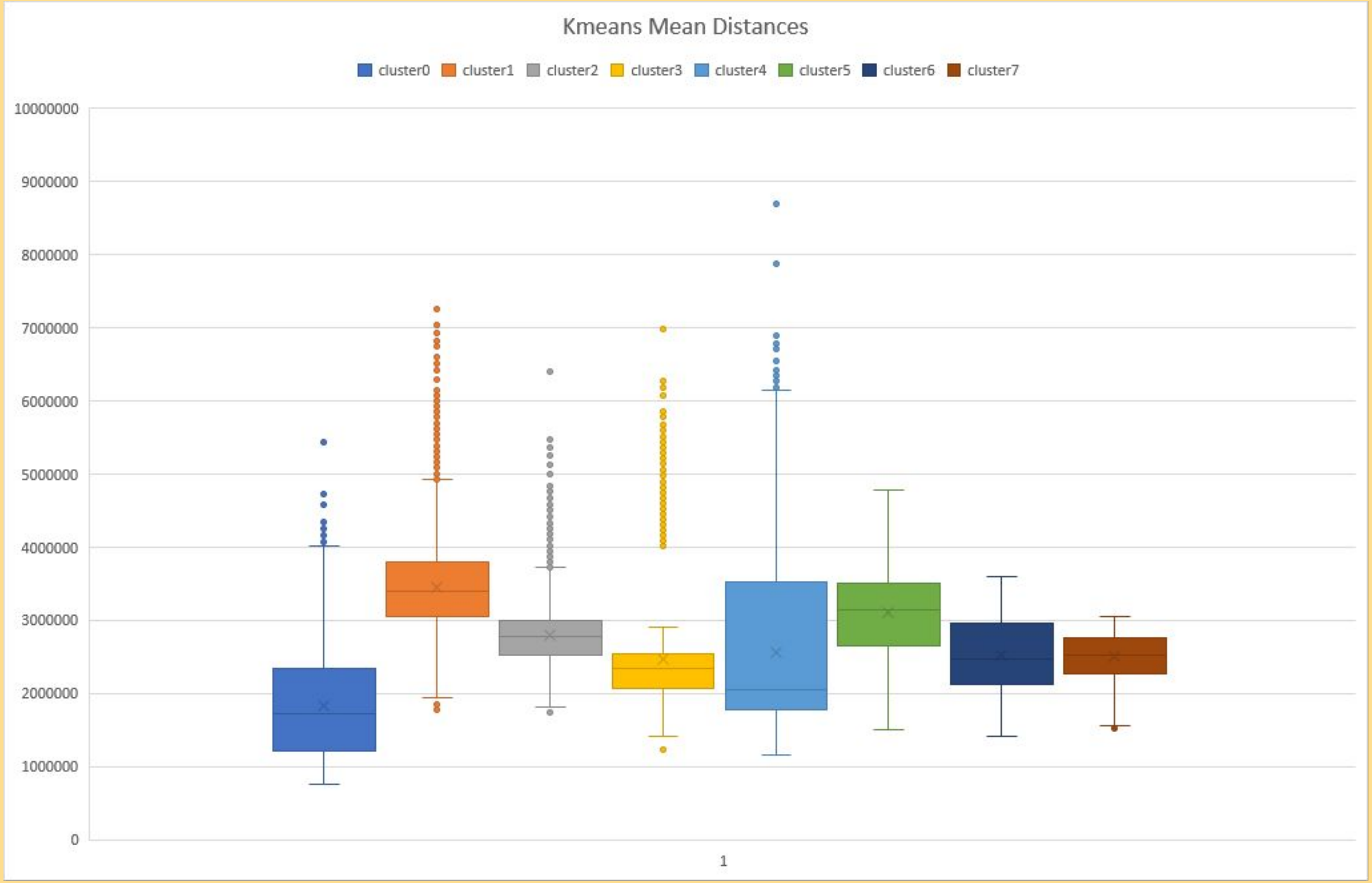
Clusters as KNN backend

Clusters 8	**Correct**	**Incorrect**
k-means	*5844*	*4156*
soc	*3408*	*6592*
uniform random	*3488*	*6512*

Clusters 12	**Correct**	**Incorrect**
k-means	*6102*	*3898*
soc	*2589*	*7411*
uniform random	*4054*	*5946*

Clustering Counts

Cluster	*C1*	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*
k-means:	*[7291,*	*8950,*	*5057,*	*8017,*	*10690,*	*9268,*	*5842,*	*4885]*
soc:	*[7437,*	*32761,*	*4446,*	*13563,*	*1354,*	*116,*	*327,*	*4]*
uniform random:	*[7531,*	*7490,*	*7476,*	*7451,*	*7534,*	*7458,*	*7524,*	*7528]*


Kmeans Mean Distances
cluster0
cluster1
cluster2
cluster3
cluster4
cluster5
cluster6
cluster7
10000000
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
1

Can We Find Outliers?

- For each cluster what label would it be?
- Looked at every cluster and counted what label is assigned to each image in the cluster
- The reason why k-means was a better backend to KNN than the self organizing cluster

k-means	***gini***	***% of total***
cluster-0	*0.1301603927*	*0.084*
cluster-1	*0.2525019984*	*0.081*
cluster-2	*0.71091586*	*0.12*
cluster-3	*0.5945320733*	*0.17*
cluster-4	*0.6647465133*	*0.13*
cluster-5	*0.2681977512*	*0.10*
cluster-6	*0.6890316238*	*0.15*
cluster-7	*0.729119827*	*0.15*

soc	***gini***	***% of total***
cluster-0	*0.8553339997*	*0.070*
cluster-1	*0.7159045192*	*0.076*
cluster-2	*0.8847461958*	*0.20*
cluster-3	*0.8726457832*	*0.21*
cluster-4	*0.8678705434*	*0.07*
cluster-5	*0.875632851*	*0.12*
cluster-6	*0.8938628415*	*0.23*
cluster-7	*0.6761667066*	*0.03*

Final Thoughts

- We attempted to remove outliers in the self-organizing clusters
- Concluded that k-means still wins for clustering images
- We are still looking to improve the self-organizing clusters
- Eventually tie these clustering methods back in with the convolution method