A $-Family Friendly Approach to Prototype Selection

Corey Pittman Eugene M. Taranta II
Joseph J. LaViola Jr.

Interactive Systems & User Experience Lab
Department of Computer Science
University of Central Florida

Intelligent User Interfaces, 2016

Background

- Sketch gesture recognition continues to be a prominent feature in applications
- $-Family recognizers ($1, $P, $N, 1¢) for gesture recognition
  - template matching (1-NN)
  - rapid prototyping
  - low coding overhead
  - error rates on par with state of the art
  - often use large datasets
- Reducing computational overhead is beneficial for mobile devices

Improving Performance

- Execution time and memory usage scale linearly with size of dataset
- Reducing size of dataset is simplest method for decreasing computational overhead

Prototype Selection Methods

- Naive method: Randomly select prototypes
- Two proposed alternatives:
  - Genetic Algorithm (GA)
  - Random Mutation Hill Climb (RMHC)
- More complex alternatives
  - K-medoids
  - Agglomerative Hierarchical Clustering

Genetic Algorithms

- Test the fitness of a population of candidate solutions
- Each candidate solution is a set of prototypes which form a subset of the full dataset
- Fit individuals generate subsequent generations via genetic operators
  - crossover to mix two candidates sets uniformly
  - mutation to change a single prototype in an individual
- Iterate through generations of numerous solutions until an optimal fitness candidate is found

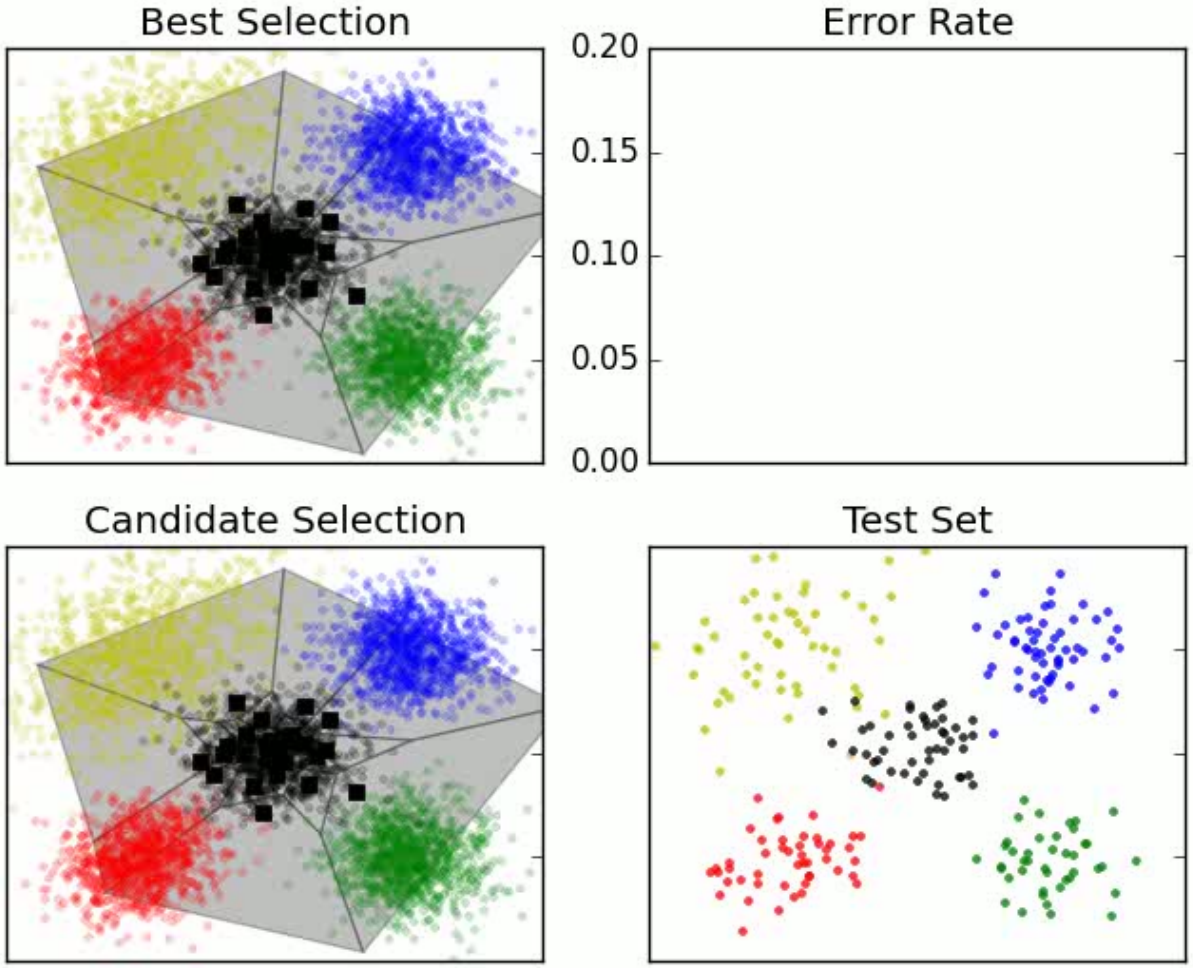
Fitness Evaluation

- A recognizer is constructed for each candidate solution
- Each recognizer is tested on a random selection of samples from the dataset
- The fitness of a candidate is the accuracy of its generated recognizer

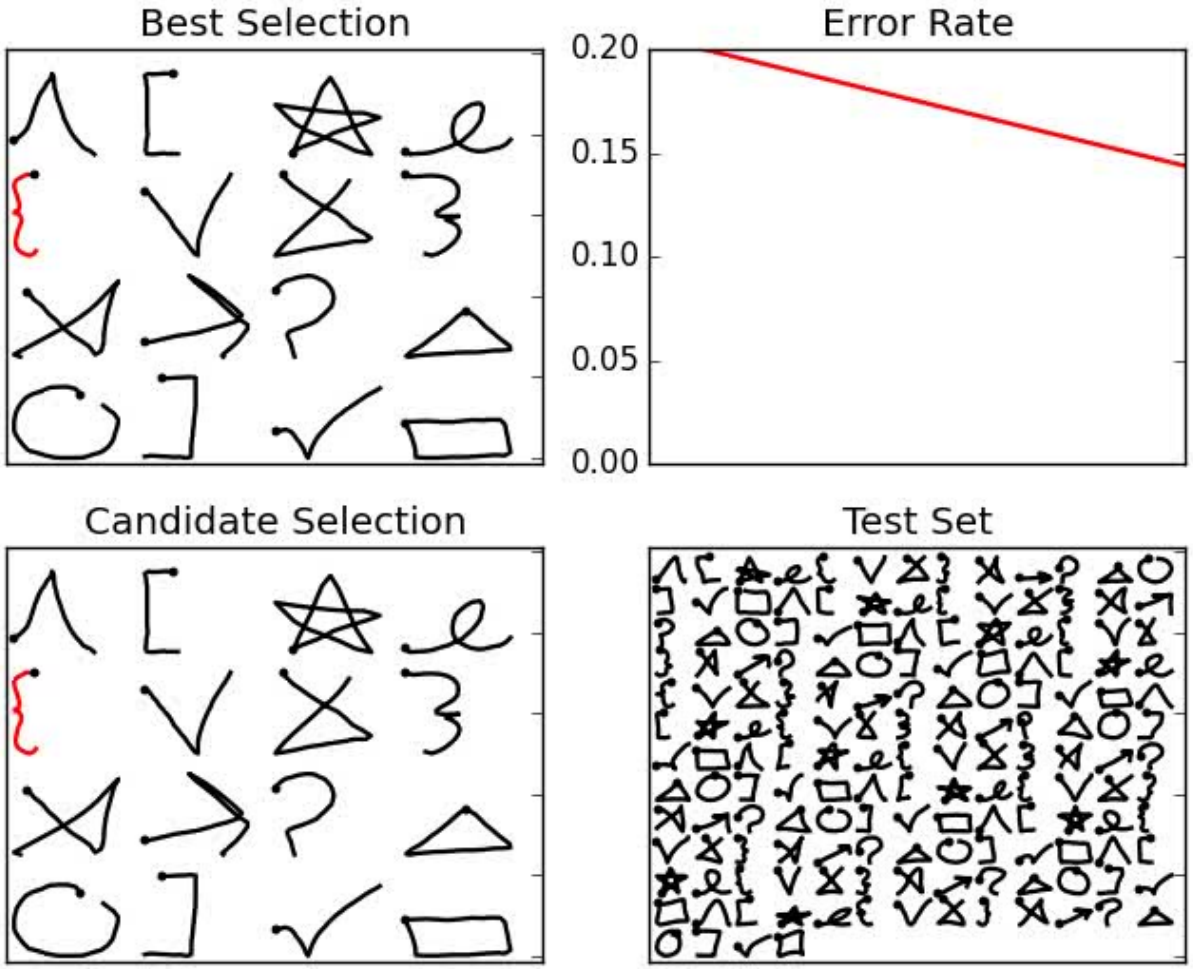
Random Mutation Hill Climb

- Similar representation of candidate solution
- Based on Skalak's approach to prototype selection
- Repeatedly mutate a single member of the subset for a predetermined number of iterations
- Store highest fitness individual.

Simple RMHC Example



Actual RMHC Example



$1-GDS from Wobbrock et. al. (2007)

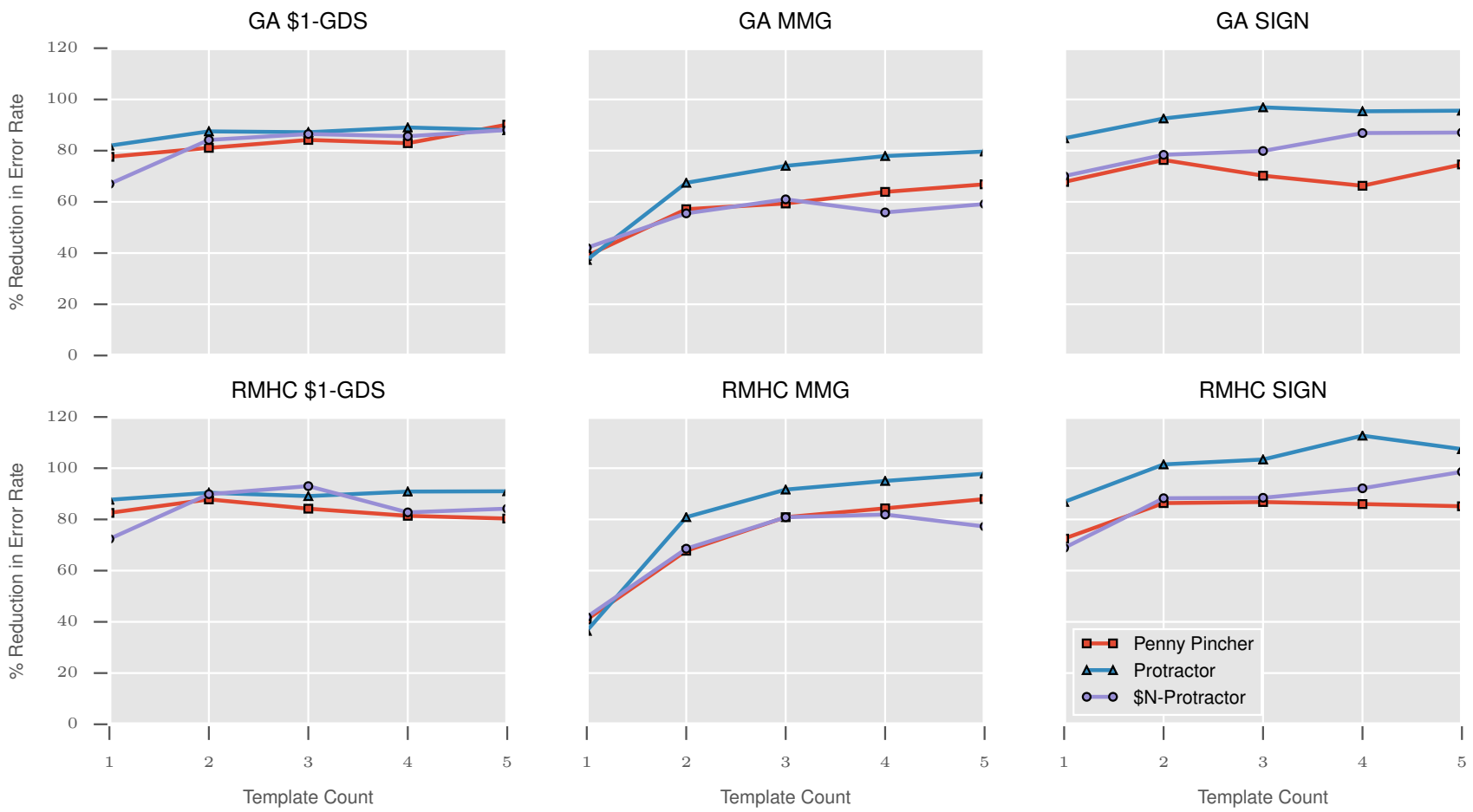
Design of Evaluation

- Evaluated effect of selection methods on error rates for three recognizers:
  - Protractor
  - $N-Protractor
  - Penny Pincher
- Three datasets were included in evaluation ($-GDS, SIGN, MMG)
- Four selection methods were included (random, RMHC, GA, full dataset)
- Each recognizer was tested with all datasets, selection methods, and per class template counts ($k = [1, 5]$)

Procedure

- Randomly generated tests by selecting a random subset to be recognized by candidate recognizers
- Attempted to find optimal subset of prototypes to maximize recognition rate
- Repeated test 500 times for each configuration

Error Rates Reduced with Little Tradeoff



Dramatic reduction in computation time and memory

	$1-GDS		**SIGN**		**MMG**	
Recognizer	Mem	Time	Mem	Time	Mem	Time
Penny Pincher	98.3	95.7	99.7	99.5	97.5	95.2
Protractor	98.3	97.7	99.7	99.7	97.5	96.8
$N-Protractor	98.3	97.4	99.7	99.6	97.5	97.7

Percent reduction in memory consumption and runtime for $k = 5$ compared to baseline.

Conclusion

- While the results for the two methods are similar, we recommend RMHC.
  - straightforward to implement
  - mutation operator is exploratory component of GA
- Optimizing the subset of samples can result in near baseline error rates
- Selection methods serve as a preprocessing step to reduce spatial and temporal constraints

Acknowledgments

- NSF career award IIS-0845921
- ISUE lab members
- Anonymous reviewers