CSE 6242/CX 4242

Model Combination

Or, Ensemble Methods

Based on lecture by Parikshit Ram

Numerous Possible Classifiers!

Classifier	**Training time**	**Cross validation**	**Testing time**	**Accuracy**
kNN classifier	None	Can be slow	Slow	??
Decision trees	Slow	Very slow	Very fast	??
Naive Bayes classifier	Fast	None	Fast	??
…	…	…	…	…

Which Classifier/Model to Choose?

Possible strategies:

- Go from simplest model to more complex model until you obtain desired accuracy
- Discover a new model if the existing ones do not work for you
- Combine all (simple) models

Common Strategy: Bagging

Consider the data set $S = \{(x_i, y_i)\}_{i=1,..,n}$

- Pick a sample S^* with replacement of size n from S
- Train on this set S^* to get a classifier f^*
- Repeat above steps B times to get $f_1, f_2,\ldots,f_B$
- Final classifier $f(x) = \text{majority}\{f_b(x)\}_{j=1,\ldots,B}$

Common Strategy: Bagging

Why would bagging work?

- Combining multiple classifiers reduces the variance of the final classifier

When would this be useful?

- We have a classifier with low bias and high variance (any examples)

Bagging decision trees

Consider the data set S

- Pick a sample S^* with replacement of size n from S
- Grow a decision tree T_b greedily
- Repeat B times to get $T_1,\ldots,T_B$
- The final classifier will be

$$f(x) = \text{majority}\{f_{T_b}(x)\}_{b=1,\ldots,B}$$

Random Forests

Almost identical to bagging decision trees, except we introduce some randomness:

- Randomly pick any *m* of the *d* attributes available
- Grow the tree only using those *m* attributes

That is, Bagged **random** decision trees

= **Random forests**

Points about random forests

Algorithm parameters

- Usual values for *m:* $\sqrt{d}, 1, 10$
- Usual value for *B*: keep increasing *B* until the training error stabilizes

Bagging/Random forests

Consider the data set $S = \{(x_i, y_i)\}_{i=1,..,n}$

- Pick a sample S^* with replacement of size n from S
- Do the training on this set S^* to get a classifier (e.g. random decision tree) f^*
- Repeat the above step B times to get $f_1, f_2,\ldots,f_B$
- Final classifier $\boldsymbol{f(x) = \text{majority}\{f_b(x)\}_{j=1,\ldots,B}}$

Final words

Advantages

- Efficient and simple training
- Allows you to work with simple classifiers
- Random-forests generally useful and accurate in practice (one of the best classifiers)
- Embarrassingly parallelizable

Caveats:

- Needs low-bias classifiers
- Can make a not-good-enough classifier worse

Final words

Reading material

- Bagging: ESL Chapter 8.7
- Random forests: ESL Chapter 15
  http://www-stat.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf

Strategy 2: Boosting

Consider the data set $S = \{(x_i, y_i)\}_{i=1,..,n}$

- Assign a weight $w_{(i,0)} = (1/n)$ to each i
- Repeat for $t = 1,...,T$:
  - Train a classifier f_t on S that minimizes the weighted loss: $\sum_{i=1}^{n} w_{(i,t)} L(y_i, f_t(x_i))$
  - Obtain a weight a_t for the classifier f_t
  - Update the weight for every point i to $w_{(i, t+1)}$ as following:
    - Increase the weights for i: $y_i \neq f_t(x_i)$
    - Decrease the weights for i: $y_i = f_t(x_i)$
- Final:

$$f(x) = \text{sign}\left(\sum_{t=1}^{T} a_t f_t(x)\right)$$

Final words on boosting

Advantages

- Extremely useful in practice and has great theory as well
  - Better accuracy than random forests usually
- Can work with very simple classifiers

Caveats:

- Training is inherently sequential
  - Hard to parallelize

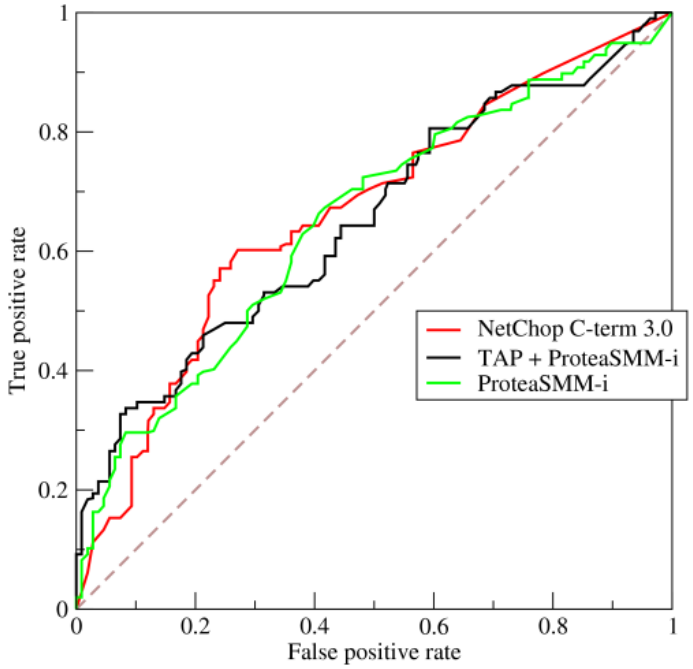
Reading material:

- ESL book, Chapter 10
- Le Song's slides:
  http://www.cc.gatech.edu/~lsong/teaching/CSE6704/lecture9.pdf

Visualizing Classification

Usual tools

- ROC curve / cost curves
  - True-positive rate vs. false-positive rate



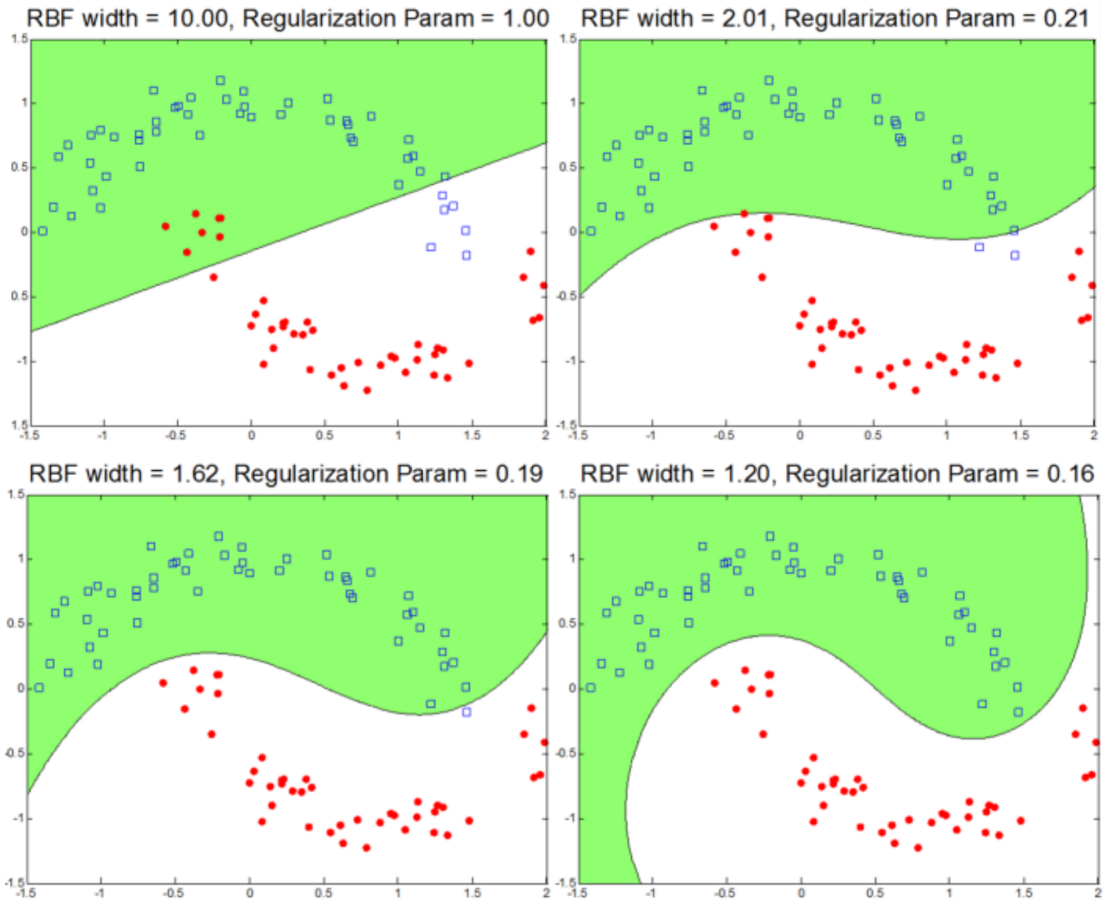
- Confusion matrix

		Predicted class		
		Cat	**Dog**	**Rabbit**
Actual class	**Cat**	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Visualizing Classification

Newer tool

- Visualize the data and class boundary with 2D projection (dimensionality reduction)



Weights in combined models

Bagging / Random forests

- Majority voting

Let people play with the weights?

EnsembleMatrix

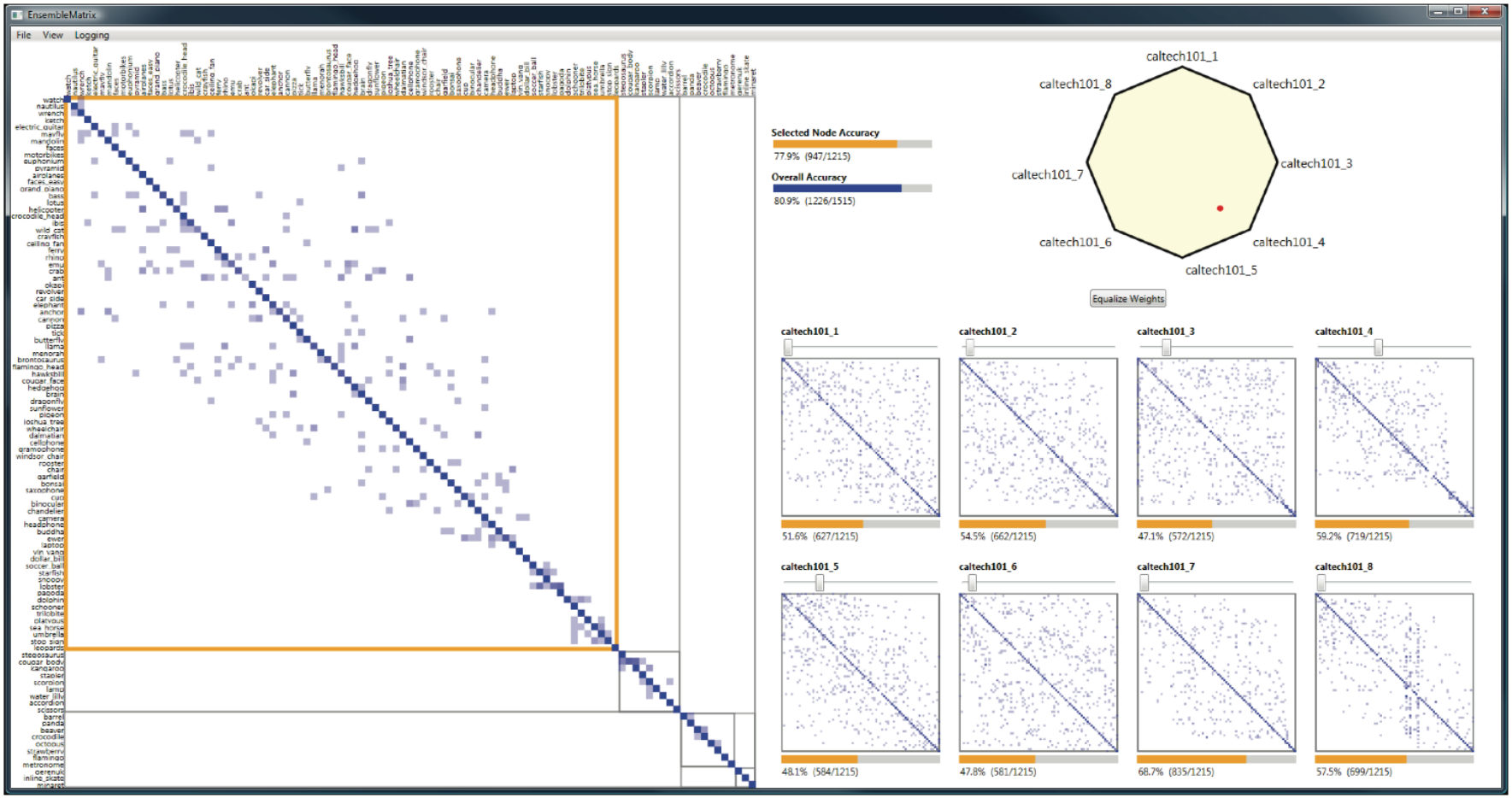


Figure 1. Primary view in EnsembleMatrix. Confusion matrices of component classifiers are shown in thumbnails on the right. The matrix on the left shows the confusion matrix of the current ensemble classifier built by the user.

http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/CHI2009-EnsembleMatrix.pdf

Understanding performance

	1	2	3	4	5	6	7	8	9	0
1	91	0	1	2	0	1	3	1	0	1
2	1	89	1	1	1	1	2	1	2	1
3	1	2	48	1	2	0	3	40	1	2
4	2	0	0	83	0	3	7	2	3	0
5	10	7	12	10	30	4	5	11	5	6
6	0	1	1	1	1	95	0	0	1	0
7	2	0	1	0	1	1	94	0	1	0
8	1	2	47	1	1	1	2	43	0	2
9	4	0	0	0	1	0	0	3	92	0
0	2	1	1	1	2	1	2	1	2	87

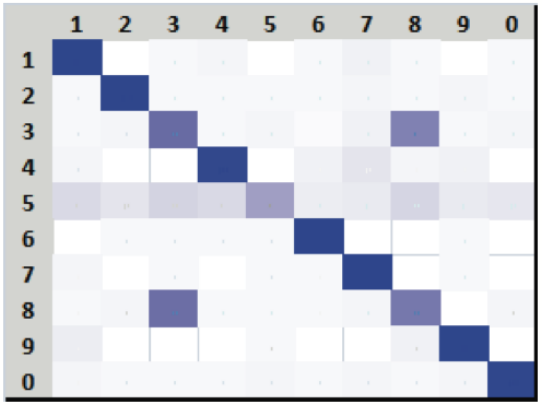
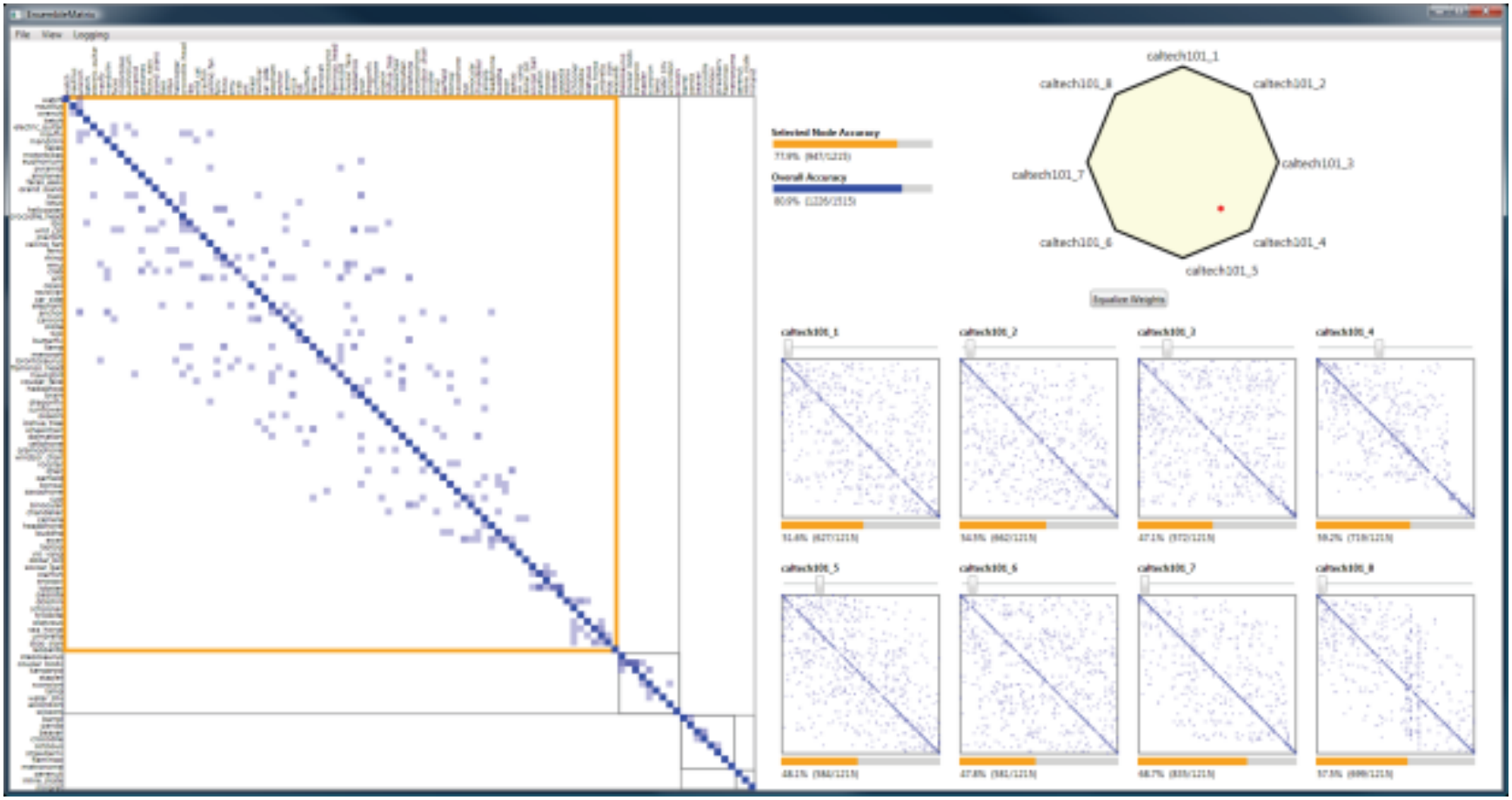


Figure 2. Representations of confusion matrix for a handwritten digit classification task. (top) standard confusion matrix; (bottom) heat-map confusion matrix. It is much easier to identify underlying patterns in the visual representation; 3 and 8 are often misclassified as each other and 5 is misclassified as many different numbers.

http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/CHI2009-EnsembleMatrix.pdf

Improving performance



http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/CHI2009-EnsembleMatrix.pdf

Improving performance

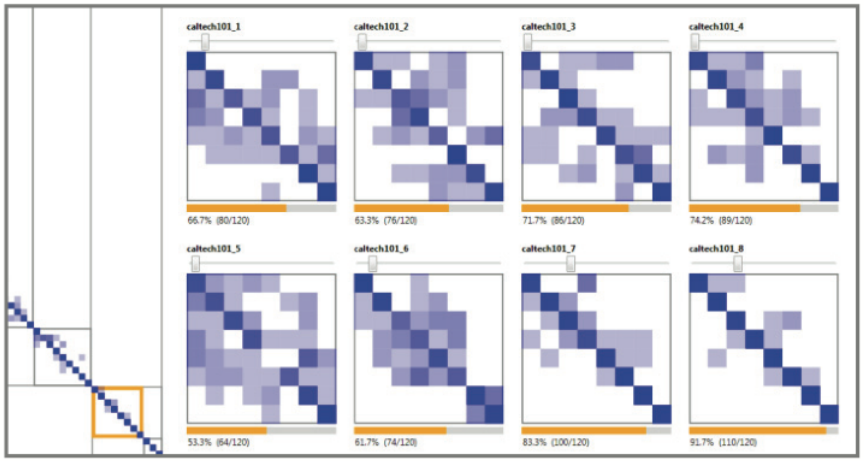
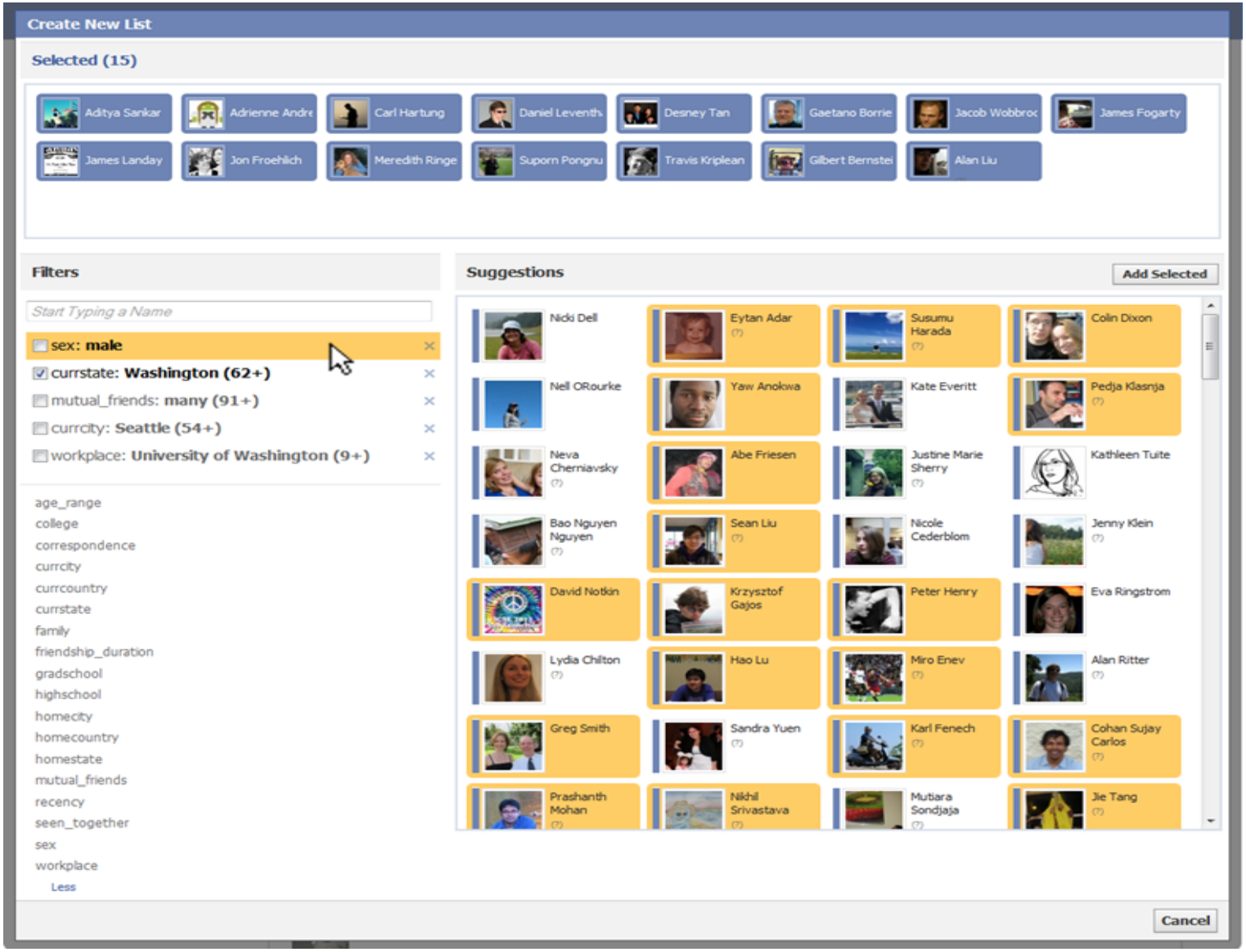


Figure 3. After partitioning the matrix, selecting a partition, outlined in orange, causes the thumbnails to display only the data instances in that partition. The component classifiers demonstrate very different behavior in this partition, including clustering and large differences in accuracy.

- Adjust the weights of the individual classifiers
- Data partition to separate problem areas
  - Adjust weights just for these individual parts
- State-of-the-art performance, on one dataset

http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/CHI2009-EnsembleMatrix.pdf

ReGroup - Naïve Bayes at work



http://www.cs.washington.edu/ai/pubs/amershiCHI2012_ReGroup.pdf

ReGroup

Gender, Age group
Family
Home city/state/country
Current city/state/country
High school/college/grad school
Workplace
Amount of correspondence
Recency of correspondence
Friendship duration
of mutual friends
Amount seen together

Features to represent each friend

Y - In group?

X - Features of a friend

$P(Y = true|X) = ?$

Compute $P(X_d|Y = true)$ for each feature d using the current group members (how?)

http://www.cs.washington.edu/ai/pubs/amershiCHI2012_ReGroup.pdf

ReGroup

Y - In group?

X - Features of a friend

$P(Y|X) = P(X|Y)P(Y)/P(X)$

$P(X|Y) = P(X_1|Y)*...*P(X_d|Y)$

Compute $P(X_i|Y = true)$ for every feature *d* using the current group members

- Use simple counting

Not exactly classification!

- Reorder remaining friends with respect to P(X|Y=true)
- "Train" every time a new member is added to the group

http://www.cs.washington.edu/ai/pubs/amershiCHI2012_ReGroup.pdf

Some additional reading

- Interactive machine learning
  - http://research.microsoft.com/en-us/um/redmond/groups/cue/iml/
  - http://research.microsoft.com/en-us/um/people/samershi/pubs.html
  - http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/CHI2009-EnsembleMatrix.pdf
  - http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/AAAI2012-PnP.pdf
  - http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/AAAI2012-L2L.pdf