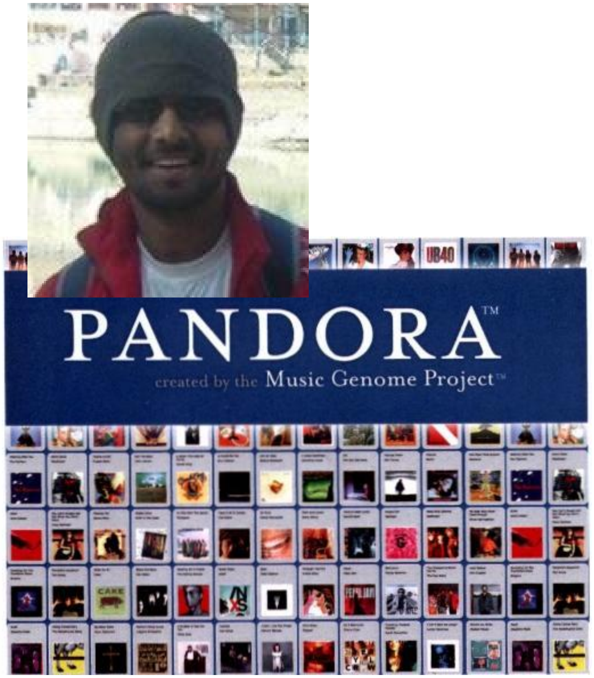
January 31

CSE 6242/CS-4803-DAVA

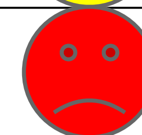
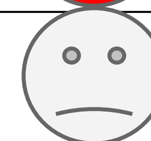
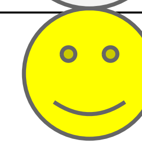
Classification - II

Model combination and visualization

http://poloclub.gatech.edu/cse6242



How will I rate "Chopin's 5th Symphony"?

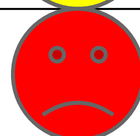
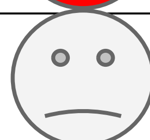
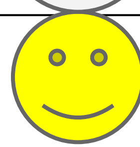
Songs	**Label**
Some nights	
Skyfall	
Comfortably numb	
We are young	
...	...
...	...
Chopin's 5th	???

Classification

What tools do you need for classification?

1. Data $S = \{(x_i, y_i)\}_{i = 1,\ldots,n}$
   - x_i represents each example with d attributes
   - y_i represents the label of each example
2. Classification **model** $f_{(a,b,c,\ldots.)}$ with some parameters $a, b, c, \ldots$
   - a model/function maps examples to labels
3. Loss function $L(y, f(x))$
   - how to penalize mistakes

Features $x_i = (x_{i1}, \ldots, x_{id})$

X	**Y**	**Artist**	**Len.**	**...**	**F_d**
Some nights		Fun	4:23		...
Skyfall		Adele	4:00		...
Comf. numb		Pink Fl.	6:13		...
We are young		Fun	3:50		...
...	...	...	...		...
...	...	...	...		...
Chopin's 5th	??	Chopin	5:32		...

$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

Training a classifier

Q: How do you learn appropriate values for *a, b, c, ...* such that

- (Part I) $y_i = f_{(a,b,c,....)}(x_i), i = 1, ..., n$
  - Low/no error on the training set
- (Part II) $y = f_{(a,b,c,....)}(x),$ for any new x
  - Low/no error on future queries (songs)

Possible A: Minimize $\sum_{i=1}^{n} L(y_i, f_{(a,b,c,..)}(x_i))$ with respect to *a, b, c,...*

Classification loss function

Most common loss: **0-1 loss function**

$$L_{0-1}(y, f(x)) = \mathbb{I}(y \neq f(x))$$

More general loss functions are defined by a *m x m* cost matrix *C* such that

$$L(y, f(x)) = C_{ab}$$

where $y = a$ and $f(x) = b$

Class	T0	T1
P0	0	C_{10}
P1	C_{01}	0

T0 (true class 0), **T1** (true class 1)

P0 (predicted class 0), **P1** (predicted class 1)

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

What to pick?

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

Strategy 1

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 f^*
- Repeat the above step B times to get $f_1, f_2,...,f_B$
- Final classifier $f(x) = \text{majority}\{f_b(x)\}_{j=1,...,B}$

This is called **Bagging**

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 decision trees

Grow a decision tree greedily until there are at most N_{min} points in any node using the following strategy:

- Randomly pick any m of the d attributes available
- Find the best split/attribute from only these m attributes

Bagged random decision trees

= **Random forests**

Points about random forests

Algorithm parameters

- Usual values for *m*: $\sqrt{d}, 1, 10$
- Usual values for $N_{min} \geq 1$ (applicable in classification only)
- 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 $f_1, f_2,...,f_B$
- Final classifier
  $\boldsymbol{f(x) = \text{majority}\{f_b(x)\}_{j=1,...,B}}$

Final words

Advantages

- Efficient and simple training
- Allows you to work with simple classifiers
- ** Random-forests generally useful and quite accurate in practice
- 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

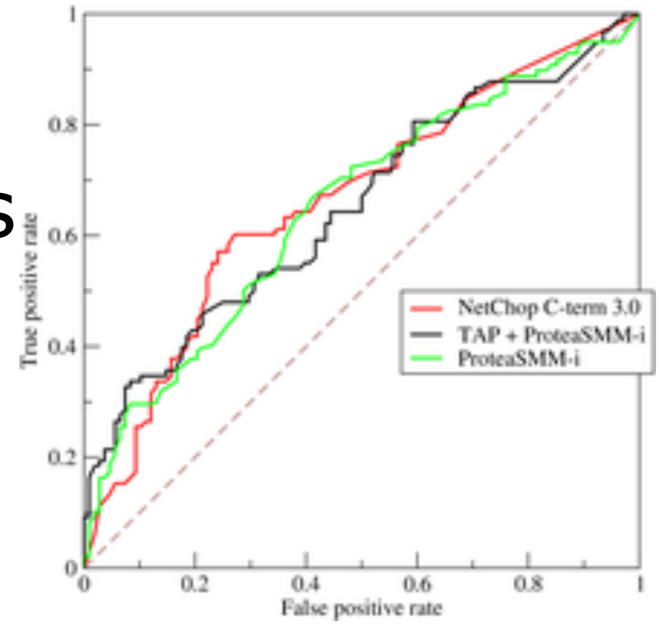
Reading material:

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

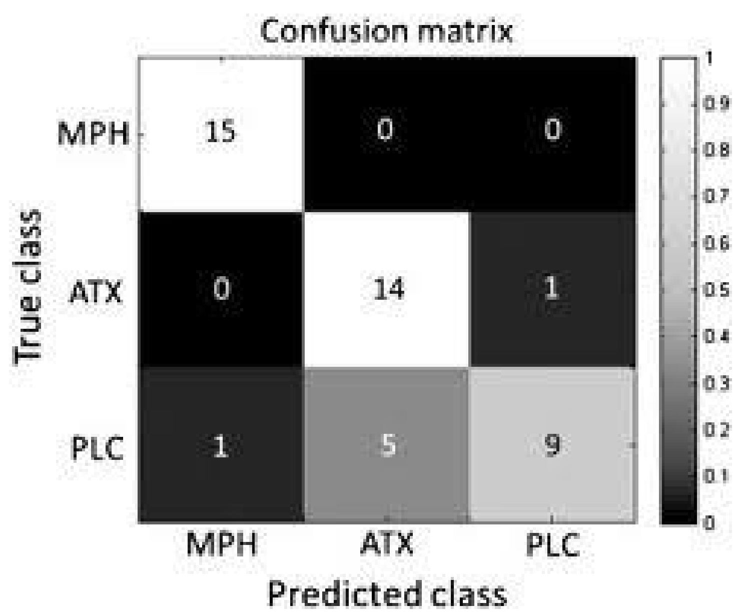
Visualization in Classification

Usual tools

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



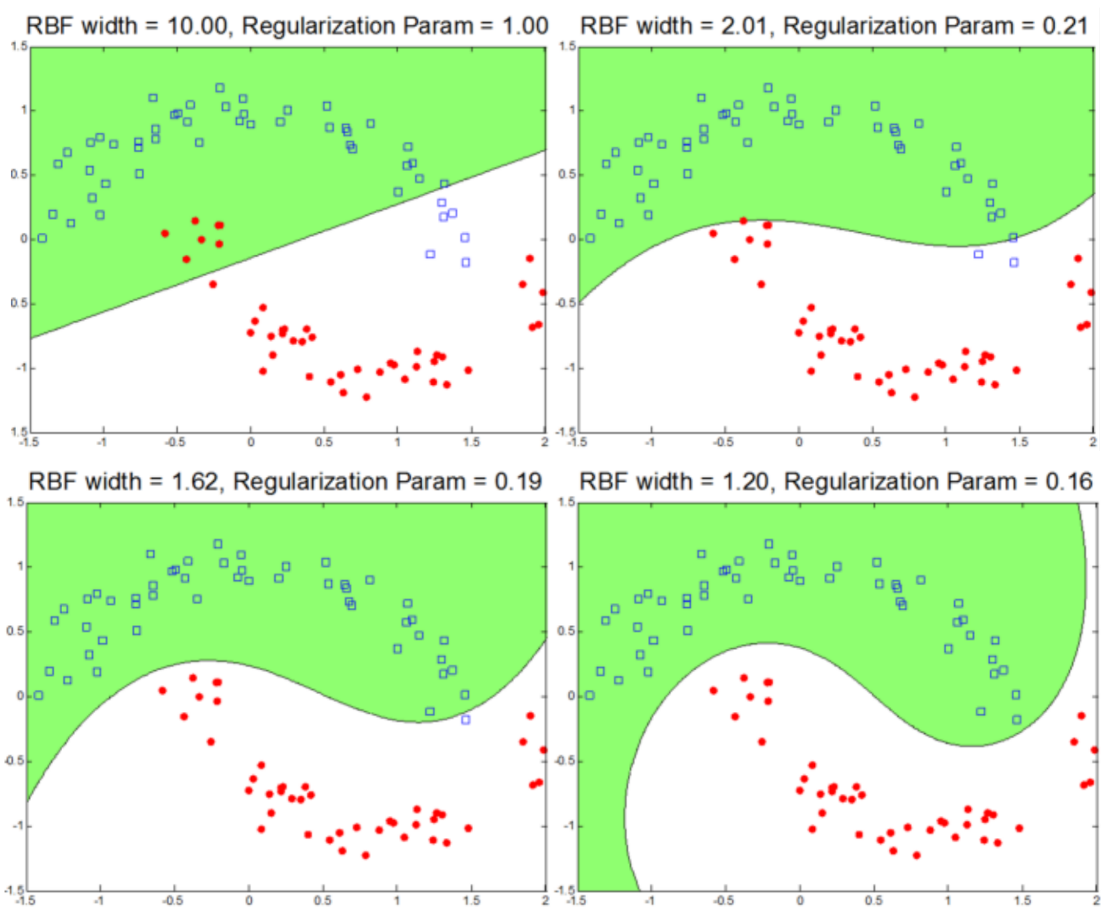
- Confusion matrix



Visualization in Classification

Newer tool

- Visualize the data and the class boundary with some 2D projection



Weights in combined models

Bagging / Random forests

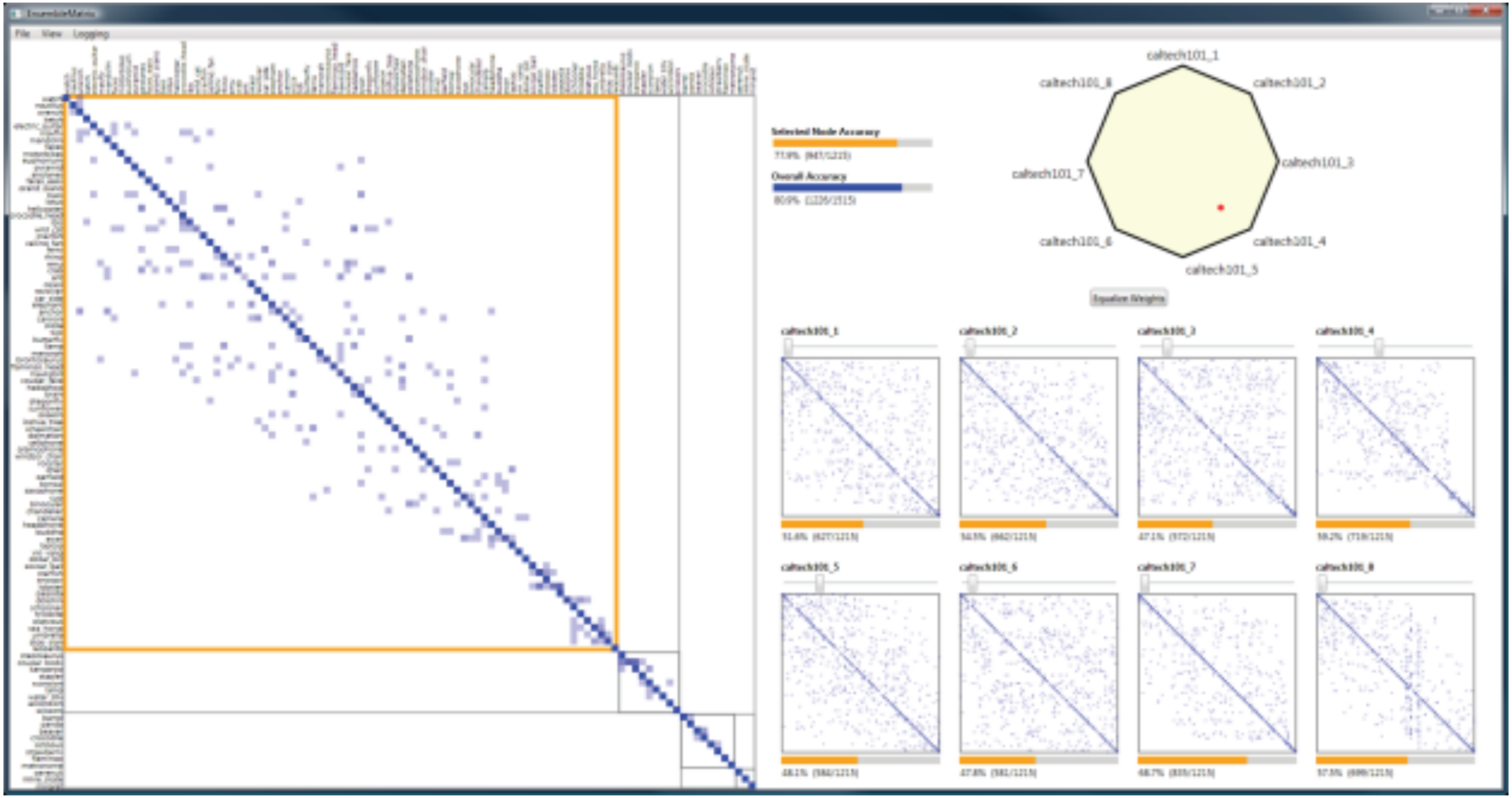
- Majority voting

Boosting

- Systematic weighting based on its individual performance

Would it be useful to allow humans to play with these weights?

EnsembleMatrix



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	0	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	91	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

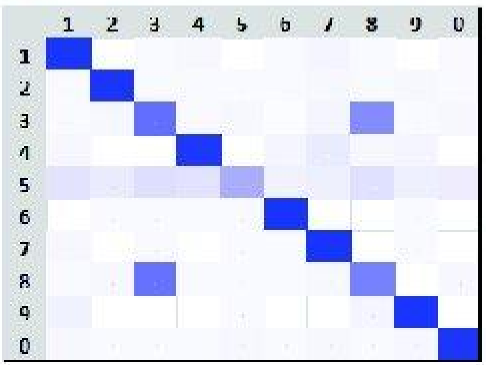
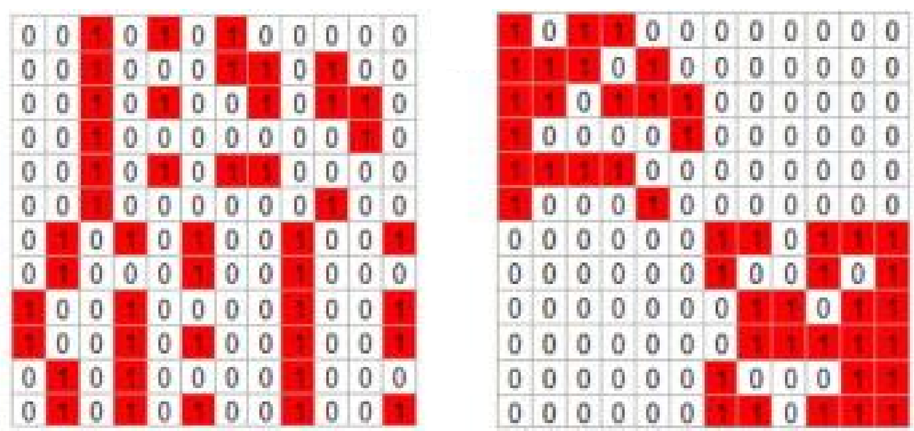


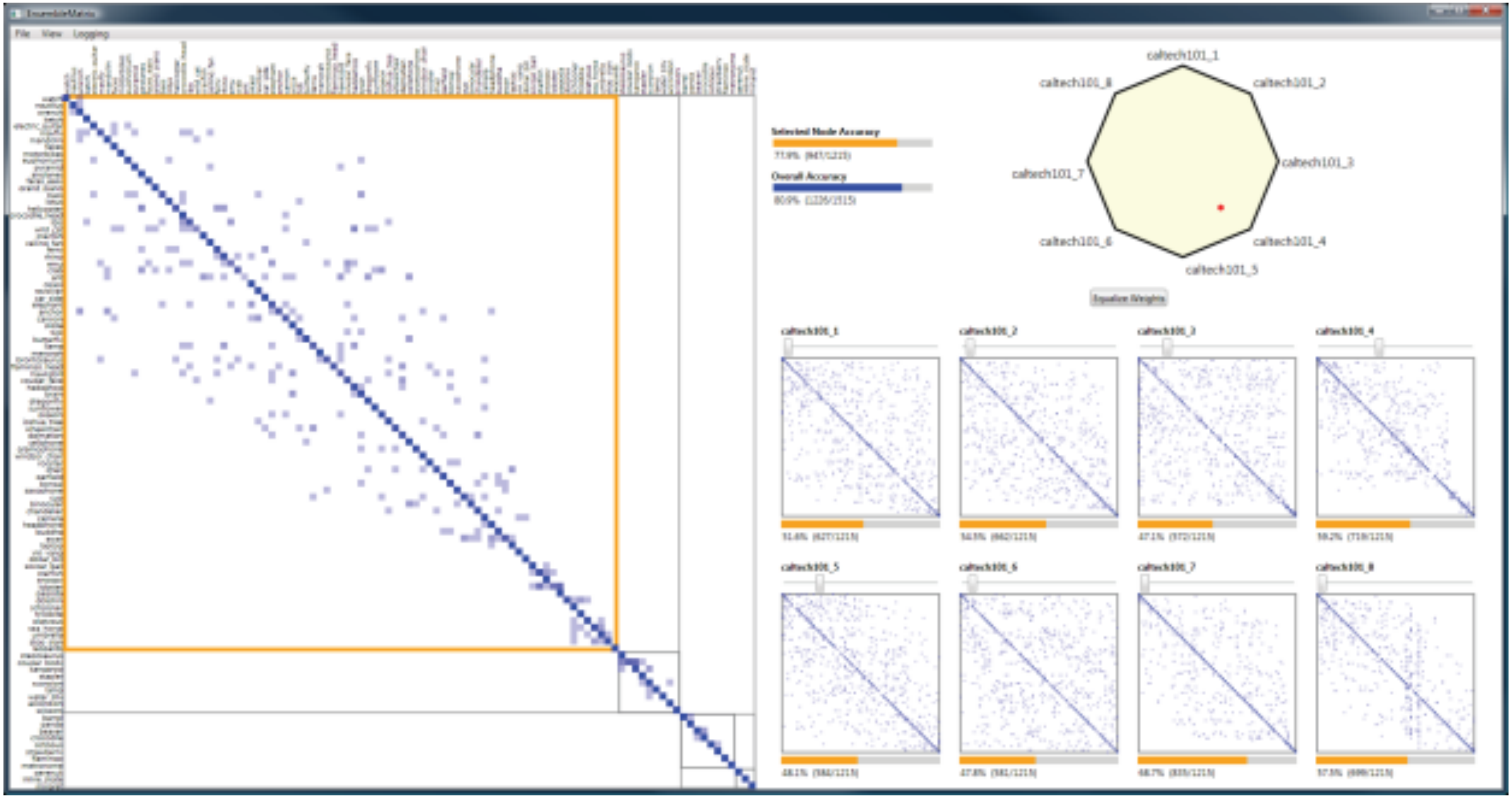
Figure 2. Representations of confusion matrix for a handwritten digit classification task. (top) standard confusion matrix; (bottom) heat-map confusion matrix. It is much easier to identify underlying patterns in the visual representation; 3 and 8 are often misclassified as each other and 5 is misclassified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

- Identify problem areas
- Reorder rows/columns to put confused classes together
  - Can use a graph clustering algorithm



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

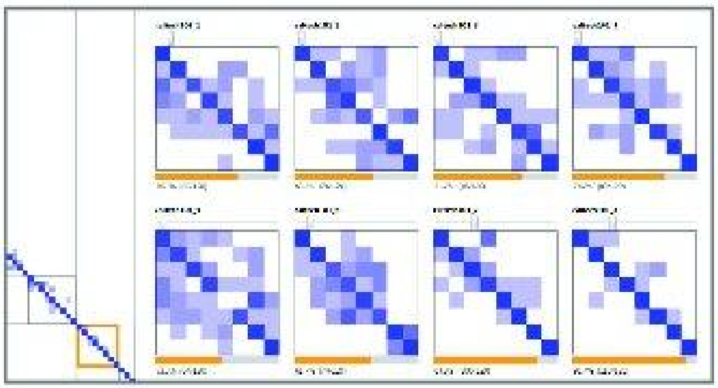
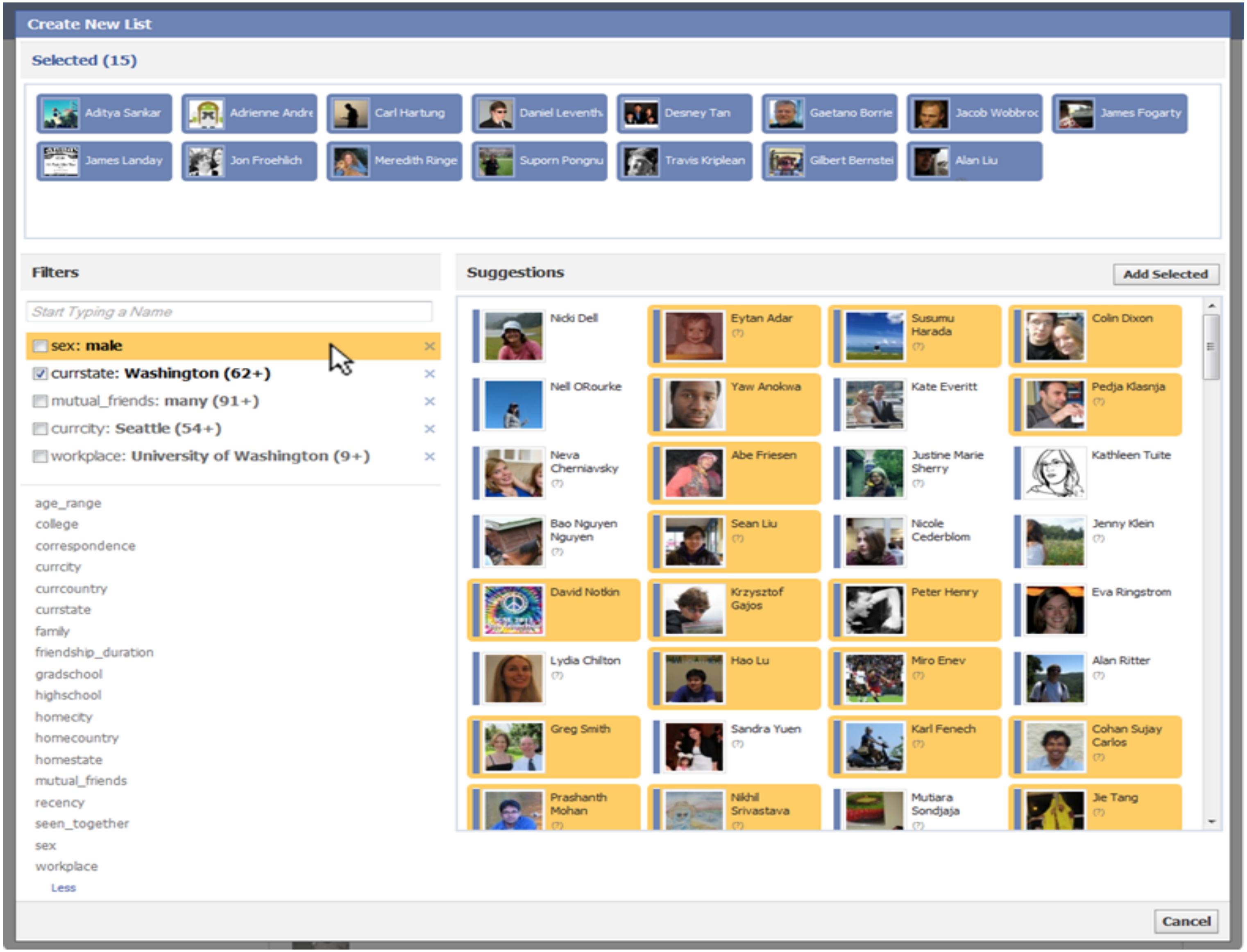


Figure 3. After partitioning the matrix, selecting a partition, outlined in orange, causes the thumbnails to display only the data instances in that partition. The component classifiers demonstrate very different behavior in this partition, including clustering and large differences in accuracy.

- Adjust the weights of the individual classifiers
- Data partition to separate out problem areas
  - Adjust weights just for these individual parts
- Claimed state-of-the-art performance!
  - *on one dataset

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

ReGroup - Naive 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

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

$P(X|Y)$

$= P(X_1|Y) * \ldots * P(X_d|Y)$

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