SNoW:
Sparse Network of Winnows

Presented by Nick Rizzolo
Introduction

- Multi-class classification
- Infinite attribute domain
- User configurable linear threshold unit networks
- Variety of update rules, algorithmic extensions
Outline

- The Basic System
- Training
- Algorithmic Extensions
- Testing
- Tuning
The Basic System

Targets (concepts)

Weighted edges, instead of weight vectors

Features

- SNoW only represents the targets and weighted edges
- Prediction is “one vs. all”
Update Rules

- **Winnow** – mistake driven
  - Promotion: if \( \sum_{i \in A_t} w_{t,i} s_i < \theta_t \), \( \forall i \in A_t \), \( w_{t,i} \leftarrow w_{t,i} \cdot \alpha_t^{s_i} \)
  - Demotion: if \( \sum_{i \in A_t} w_{t,i} s_i \geq \theta_t \), \( \forall i \in A_t \), \( w_{t,i} \leftarrow w_{t,i} \cdot \beta_t^{s_i} \)

- **Perceptron** – mistake driven
  - Promotion: if \( \sum_{i \in A_t} w_{t,i} s_i < \theta_t \), \( \forall i \in A_t \), \( w_{t,i} \leftarrow w_{t,i} + \alpha_t s_i \)
  - Demotion: if \( \sum_{i \in A_t} w_{t,i} s_i \geq \theta_t \), \( \forall i \in A_t \), \( w_{t,i} \leftarrow w_{t,i} - \alpha_t s_i \)

- **Naïve Bayes** – statistical
Command Line Interface

```
snow -train -I <example file> -F <network file>
    -W <α>,<β>,<θ>,<initial weight>:<targets>
    -P <α>,<θ>,<initial weight>:<targets>

snow -test -I <example file> -F <network file> ...

snow -train -I learnMe.snow -F learnMe.net
    -W 2,0.5,3.5,1:1-3 -z + -s s

snow -test -I testMe.snow -F learnMe.net -o winners
```
A Training Example

1, 1001, 1006:
2, 1002, 1007, 1008:
1, 1004, 1007:
3, 1006, 1004:
3, 1004, 1005, 1009:
1001, 1005, 1007:

Update rule: Winnow

$$\alpha = 2, \beta = \frac{1}{2}, \theta = 3.5$$
Bells & Whistles

- Eligibility
  - `e count:1`
- Discarding
  - `d abs:0.1`
- “Fixed feature”
  - `f -`
- Feature conjunctions
  - `g -`
- Prediction threshold
  - `p 0.5`
- Smoothing
  - `b 5`
Outline

- The Basic System
- Training
- Algorithmic Extensions
- Testing
- Tuning
Clouds (Voting)

- Multiple target nodes per concept
- Learning still independent
- Cloud activation is weighted sum of its targets’ activations
- Decreasing function of mistakes gives those weights

```
snow -train -I learnMe.snow -F learnMe.net
-W :1-3 -P :1-3
```
The Sequential Model

- Larger confusion set $\rightarrow$ lower prediction accuracy
- Put prior knowledge in each example

1, 1001, 1005, 1007; 1, 3:

1001, 1005, 1007; 1, 3:
The Thick Separator

- Theory is similar to SVM
- Increases “margin”
- Improves generalization

Almost always improves performance
Constraint Classification

- Promotion / demotion depends on activation comparisons
- More expressive than independent update rules
- Sometimes difficult to realize performance improvements
Other Training Policies

- Regression
  - Function approximation
  - (Exponentiated) Gradient Descent

- Threshold relative updating
  - Each update moves hyperplane to example

- Single target training and testing
Outline

- The Basic System
- Training
- Algorithmic Extensions
- Testing
- Tuning
Sigmoid Functions

- How to compare activations from different algorithms?
- Sigmoids map activations to (0:1)

**Winnow / Perceptron**

$$\sigma(\theta, \Omega) = \frac{1}{1 + e^{\theta - \Omega}}$$

**Naïve Bayes already yields (0:1)**

**Softmax:**

$$\frac{e^{\Omega_t(e)}}{\sum_{t' \in T} e^{\Omega_{t'}(e)}}$$
Output Modes

- **accuracy**  (the default)
  - Assumes labeled testing examples
  - Simply reports SNoW’s performance

- **winners**

- **softmax**

- **allactivations**

Example 47  Label: 0

0: 0.70005  4.8475  0.77582*
1: 0.30915  3.1959  0.14876
2: 0.18493  2.5167  0.075420
Outline

- The Basic System
- Training
- Algorithmic Extensions
- Testing
- Tuning
Tuning

- Parameter settings make a big difference
  - Want to pick the right algorithm
  - Don’t want to over-fit
- Trial & error
- The major players:
  -W <α>,<β>,<θ>,<initial weight>:<targets>
  -P <α>,<θ>,<initial weight>:<targets>
  -S <thickness>
  -r <rounds>  -u <+ | ->

  `tune.pl -train <file> -arch W:<targets> ...`