Module 11: Advanced Machine Learning

Module 11 Outline

- Intro, inc. Module 4 recap
- Engines and algorithms
- Sentiment analysis/voice of the customer exercise
- Relation Extraction exercise

Introduction and Module 4 Recap

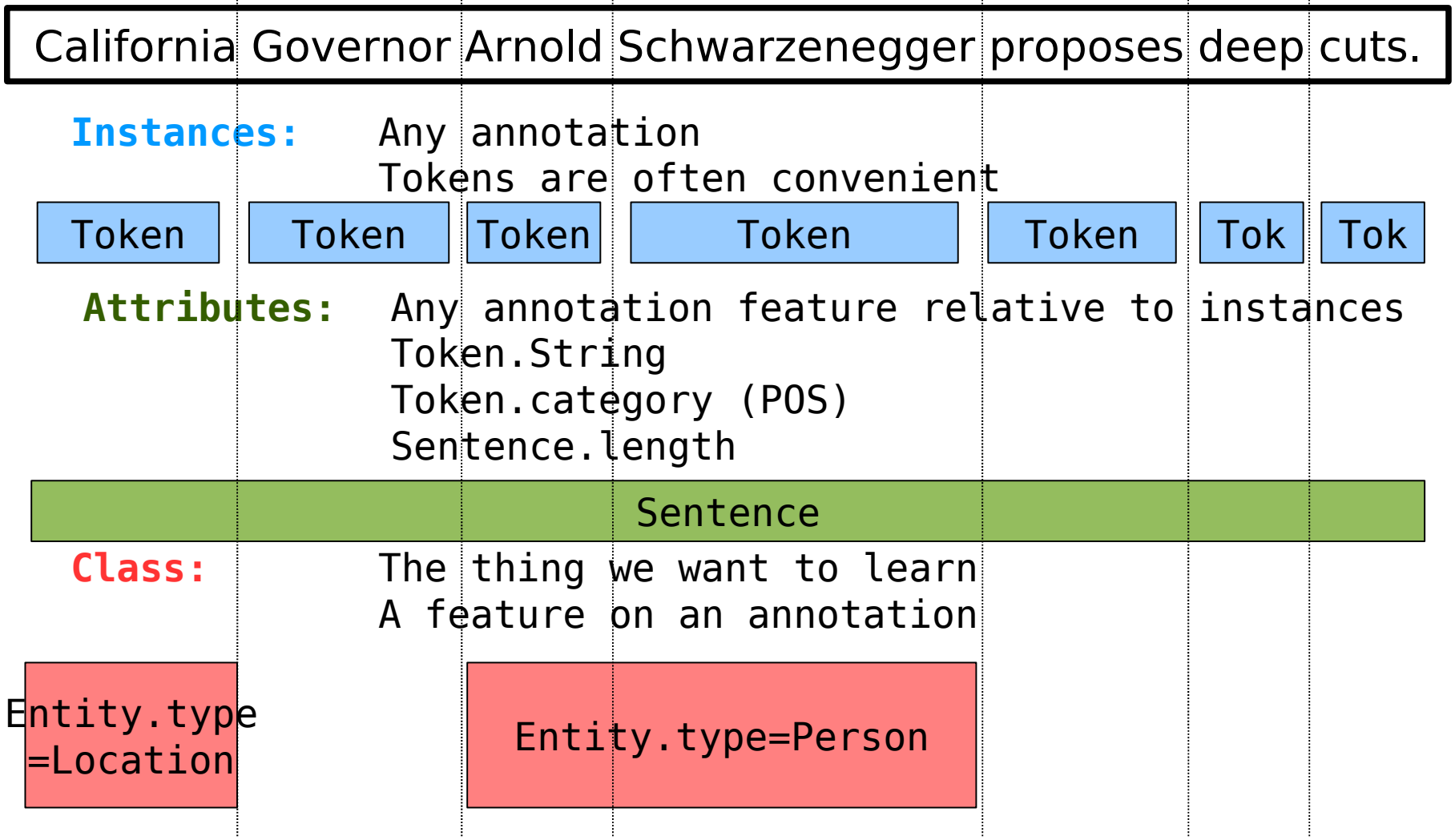
Intro–ML in GATE

- We will be using the “Batch Learning PR” in the “Learning” plugin
- This PR
  - Implements SVM and PAUM engines, as well as supporting algorithms from Weka
  - Offers integrated evaluation
  - Supports named entity extraction, classification and relation extraction
- We will NOT be using the Machine Learning plugin

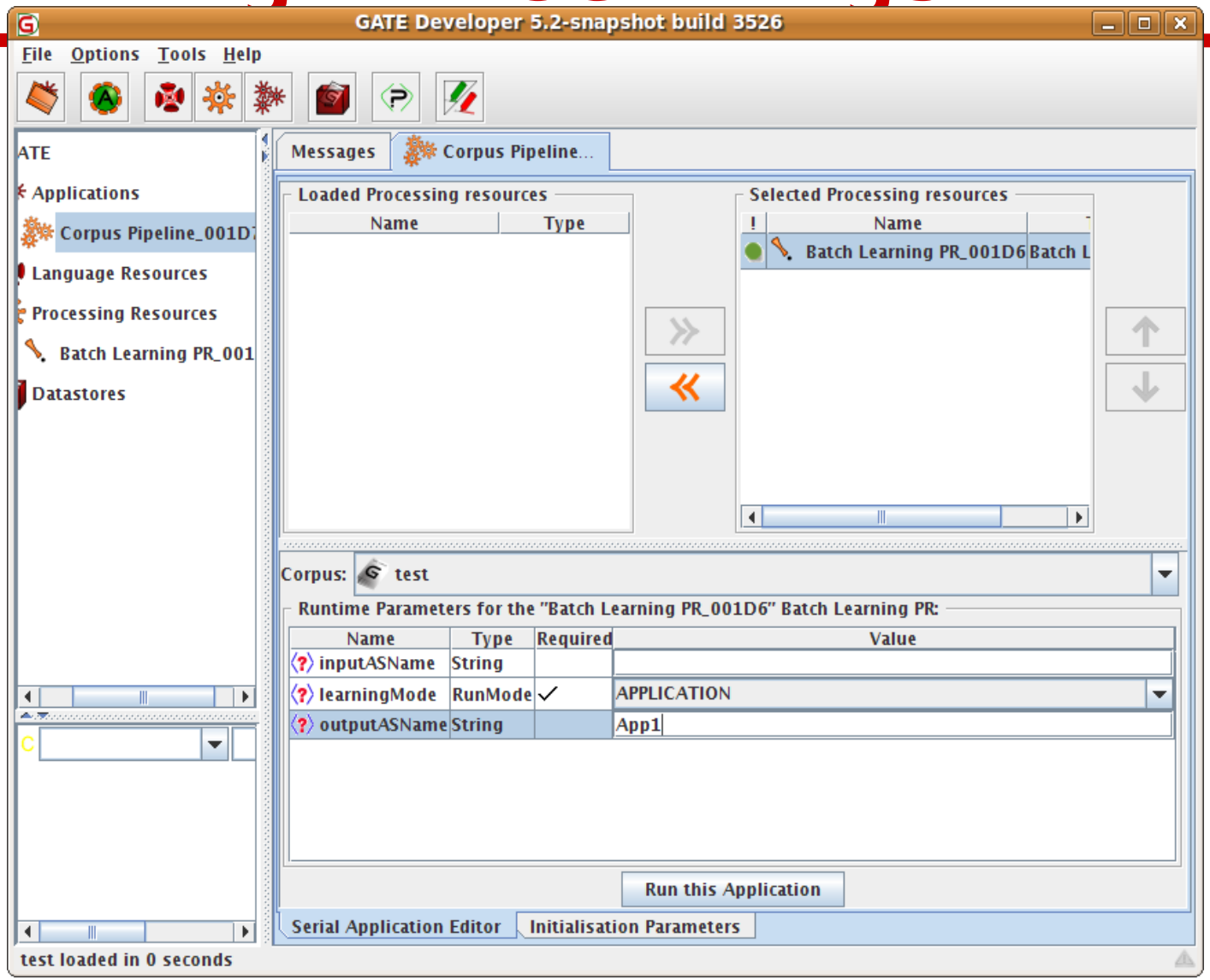
Mod 4 Recap

- Batch Learning PR takes a config file as an init time parameter
  - This is where instances, attributes and class are specified
  - We'll talk a bit about that shortly
- Corpus, learning mode, input and output annotation sets are runtime parameters
- Modes include evaluation, training and application
- There is also a mode for producing feature files, that you could then use outside of GATE

Mod 4 Recap: Instances, attributes, classes



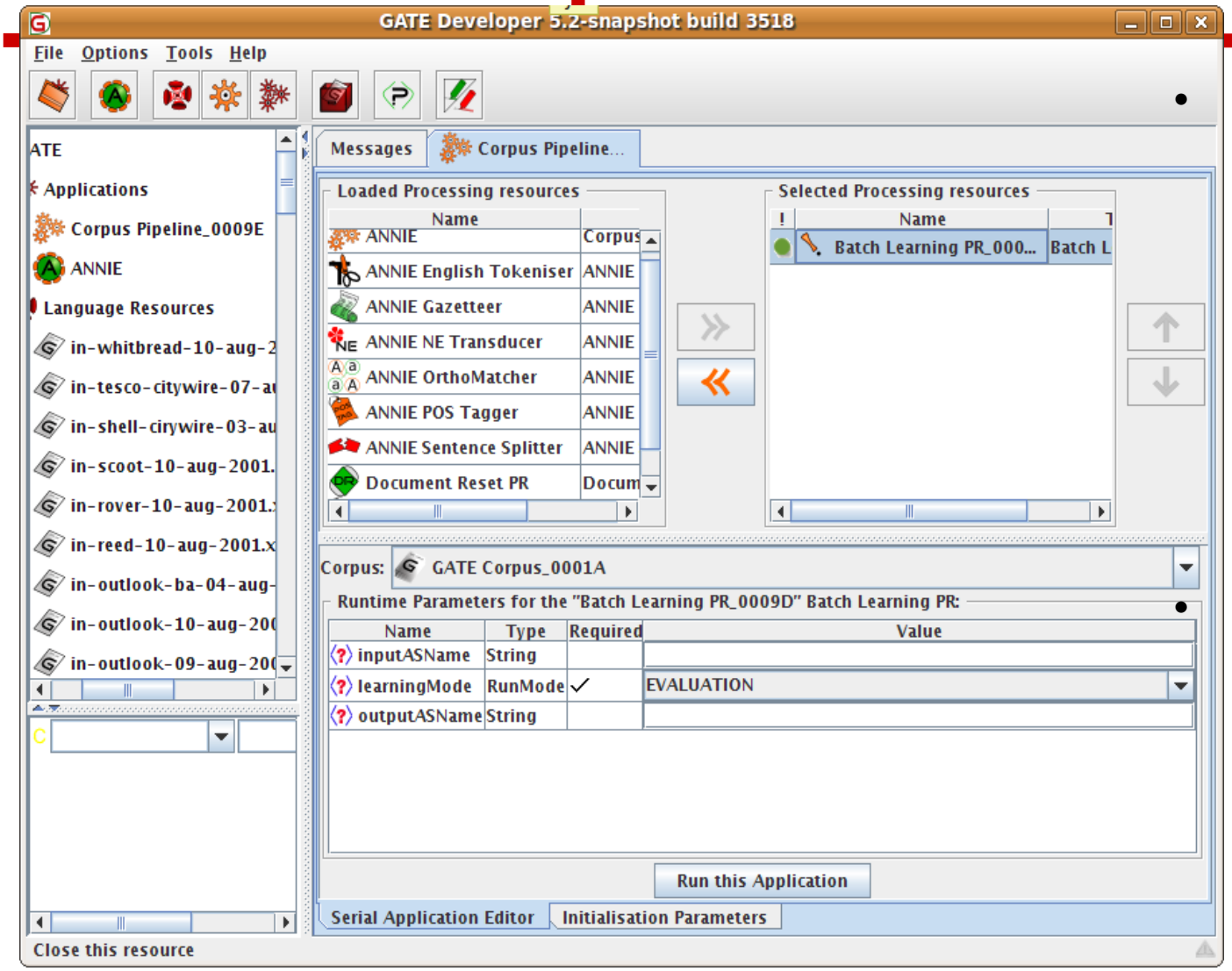
Mod 4 Recap: Batch Learning PR Settings



Mod 4 Recap: Learning Modes

- Evaluation mode runs an evaluation and outputs performance statistics to the messages tab
  - How to evaluate is specified in the config file
  - Hold-out and k-fold cross-validation are available
- Training and application do the obvious!

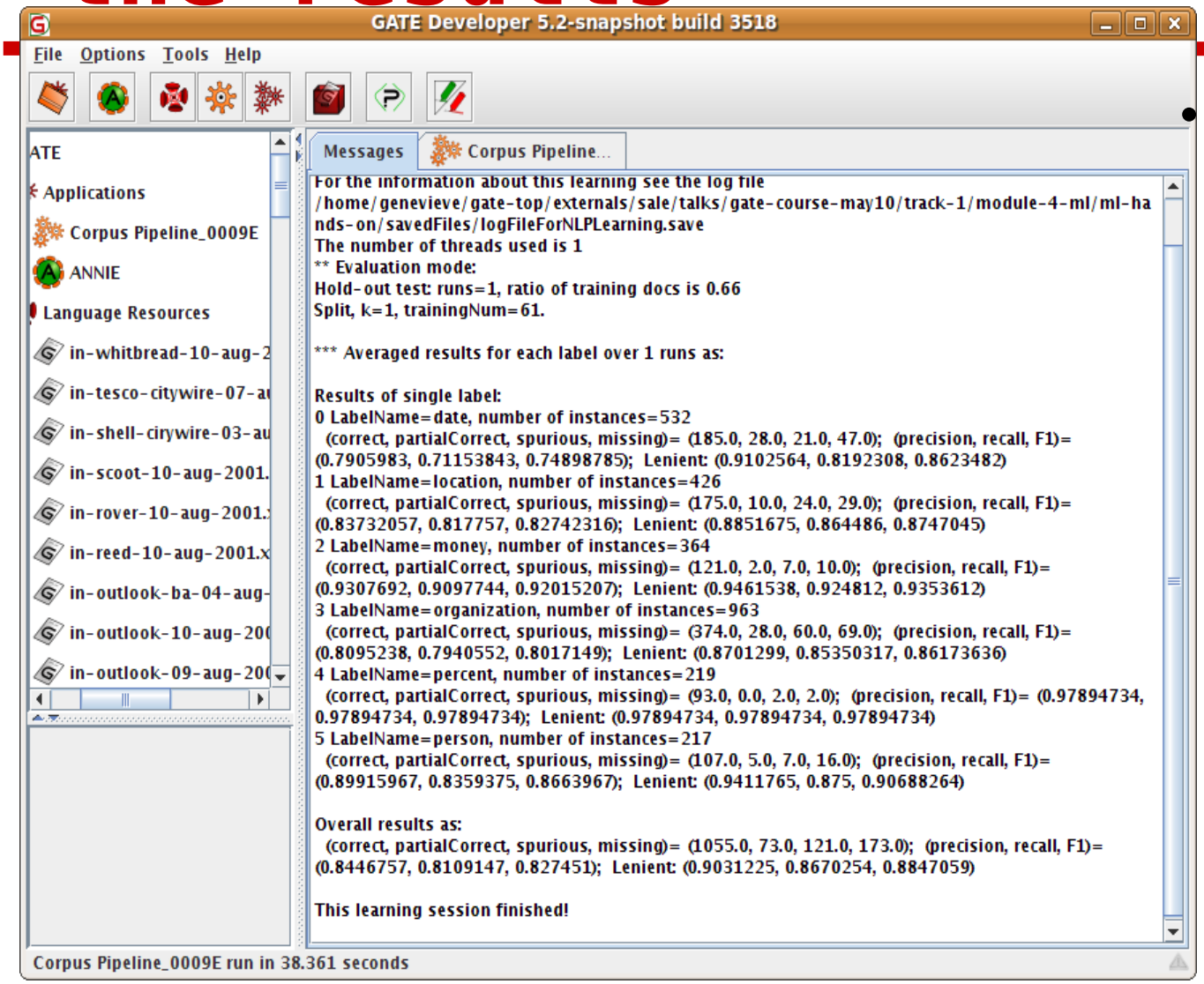
Mod 4 Recap: Evaluation Mode Example



The inputASName is blank because the attributes and class are in the default annotation set

OutputASName should be the same as inputASName in evaluation mode

Mod 4 Recap: Inspecting the results



Eval mode results are output to the messages tab

Mod 4 Recap

- You can also run your own evaluation using GATE's other evaluation tools
  - Corpus Quality Assurance
  - Corpus Benchmark Tool

Mod 4 Recap: Configuration File

- In the hands-on materials, open ne-config-file.xml using a text editor
- Have a look at the configuration file
- This is a configuration file for the task of learning named entities
- We'll go through a few of the things you can specify in the configuration file

Surround mode and filtering


```
<SURROUND value="true"/>
<FILTERING ratio=”0.0” dis=”near”/>
```

- Surround mode tells the API to build classifiers for the begin and end boundaries when learning chunks such as named entities.
- Filtering is used to remove negative examples in cases where they heavily outweigh positives. It could be relevant in relation learning, but not in the example we will use.

Evaluation

```
<EVALUATION method="kfold" runs="10"/>
```

OR

```
<EVALUATION method="holdout" ratio="0.66"/>
```

- Holdout can be used for speed, especially if you have lots of complex features
- But k-fold will give you more reliable results

Engine

```
<ENGINE   nickname=”SVM”
          implementationName=”SVMLibSvmJava”
          options=” -c 0.7 -t 0 -m 100 -tau 0.6" />
```

- Next we specify what machine learning algorithm we wish to use
- In the example config we specify PAUM
- Above, we specify SVM
- Parameters are also passed according to which engine we are using

Confidence Thresholds

```
<PARAMETER name="thresholdProbabilityEntity" value="0.2"/>
<PARAMETER name="thresholdProbabilityBoundary" value="0.42"/>
<PARAMETER name="thresholdProbabilityClassification"
value="0.5"/>
```

- Learner will provide confidence ratings—how likely is a result to be correct
- We must determine how certain is good enough
- Depending on the application we might prefer to include or exclude annotations for which the learner is not too sure
- thresholdProbabilityBoundary and thresholdProbabilityEntity are thresholds for chunk learning, and not relevant here
- thresholdProbabilityClassification is the threshold for classification tasks, such as relation learning

Multiple classes

```
<multiClassification2Binary method="one-vs-others" />
```

- SVM and Perceptron aim to learn to distinguish between **two** classes only
- In e.g. named entity extraction we may have several classes (person, date, location etc)
- Therefore the problem must be converted so that we can use binary algorithms to solve it
- **one-vs-others**
  - person vs date + location / date vs person +location / location vs date + person
- **one-vs-another**
  - person vs date / location vs person / date vs location

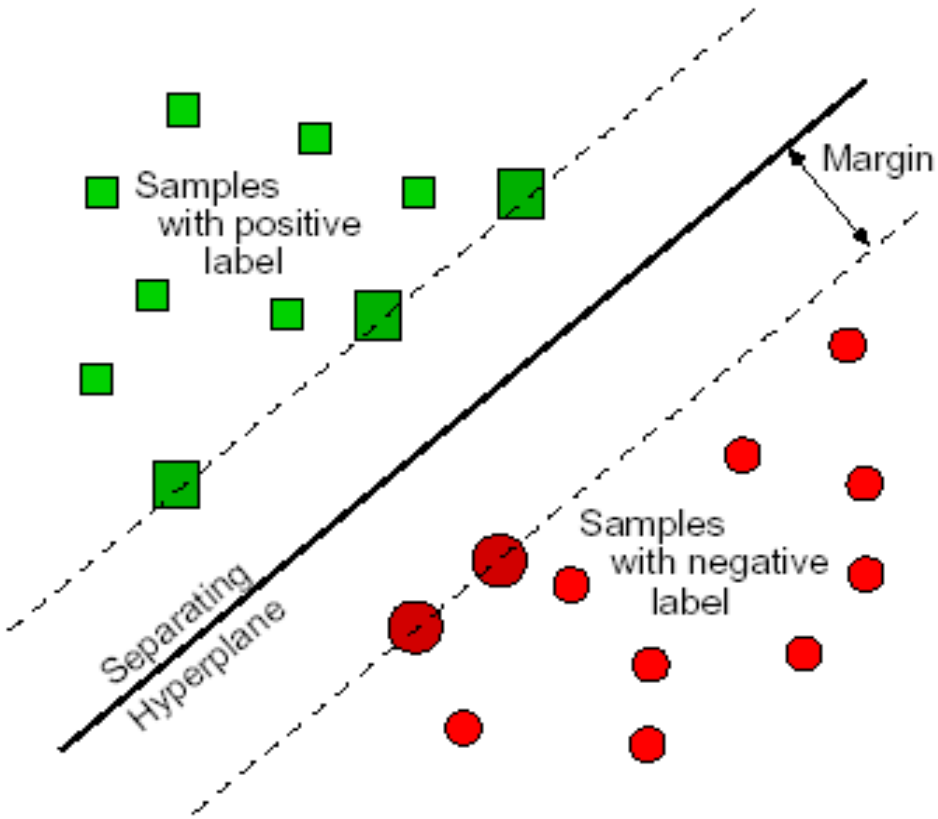
What will we cover in this module?

- In Module 4 we focused on named entity recognition, and we used the PAUM engine
- In this module we will cover other engines and tasks
- Engines and algorithms—a bit about how they work
  - SVM
  - Perceptron
  - Calling some Weka engines (Weka is a popular ML program)
- Task styles:
  - Classification (sentiment analysis/voice of the customer)
  - Relation extraction (finding entities that are connected by a relationship)

Engines and Algorithms

Support Vector Machines

- Attempt to find a hyperplane that separates data
- Goal: maximize margin separating two classes
- Wider margin = greater generalisation



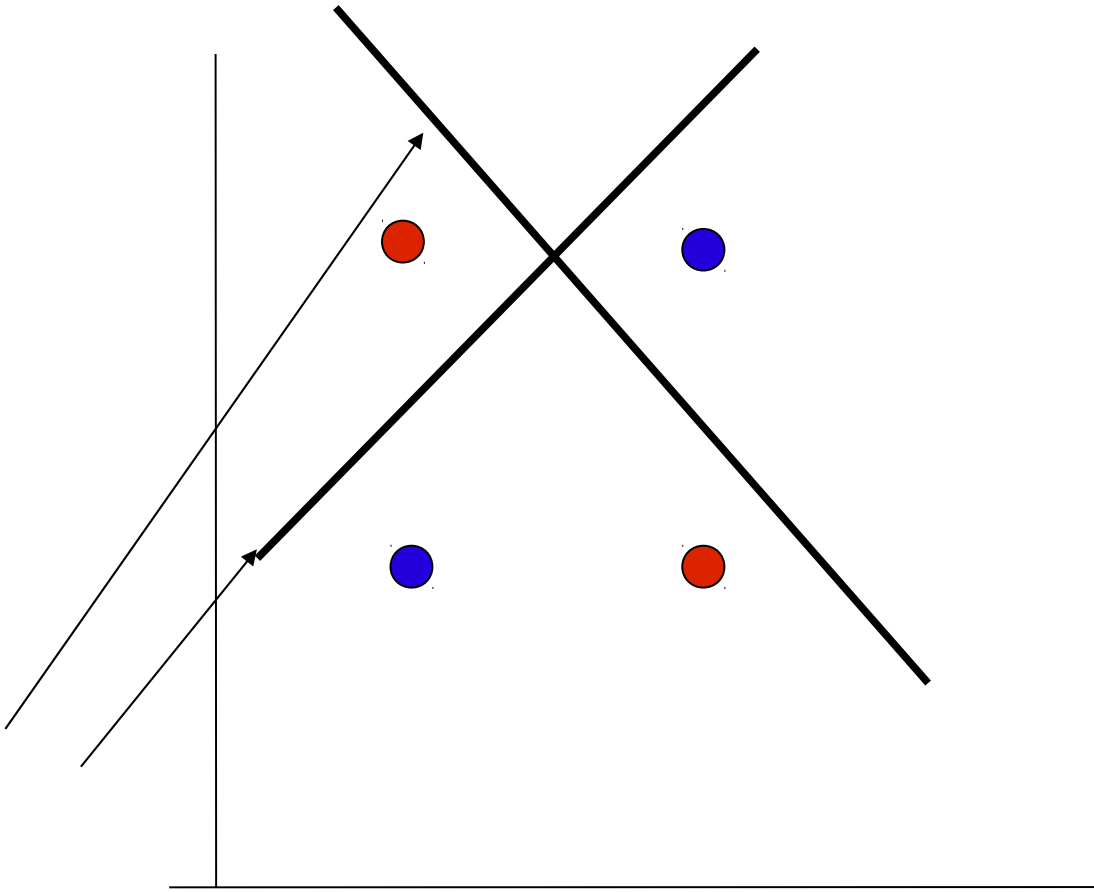
Support Vector Machines

- Points near decision boundary: support vectors (removing them would change boundary)
- Points far from boundary not important for decision
- What if data doesn't split?
  - Soft boundary methods exist for imperfect solutions
  - However linear separator may be completely unsuitable

Support Vector Machines

- What if there is no separating hyperplane?
- See example:
- Or class may be a globule

They do not work!



Kernel Trick

- Map data into different dimensionality
- http://www.youtube
- As shown in the video, due to polynomial kernel elliptical separators can be created nevertheless.
- Now the points are separable!

Kernel Trick in GATE and NLP

- Binomial kernel allows curved and elliptical separators to be created
- These are commonly used in language processing and are found to be successful
- Linear and polynomial kernels are implemented in Batch Learning PR's SVM

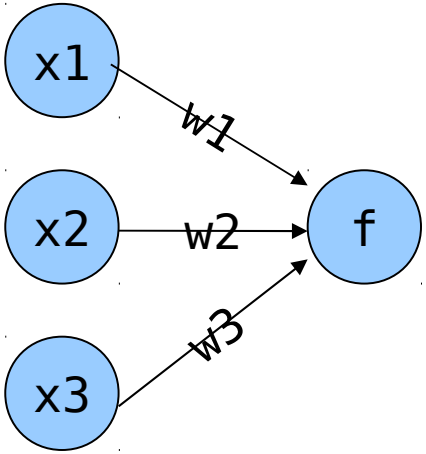
Support Vector Machines

- SVMs combined with kernel trick provide a powerful technique
- Multiclass methods simple extension to two class technique (one vs. another, one vs. others)
- Widely used with great success across a range of linguistic tasks

Perceptron and PAUM

- Perceptron is one of the oldest ML methods (invented in the 50s!)
- Has some similarities to SVM (implements a linear separator)
- Theoretically SVM works a little better because it calculates the optimal separator
- However in practice there is minimal difference
- Perceptron is a lot faster!

Perceptron



- You might think of perceptrons as being these things (correct)
- What this is actually calculating is a dot product w.x

More perceptron

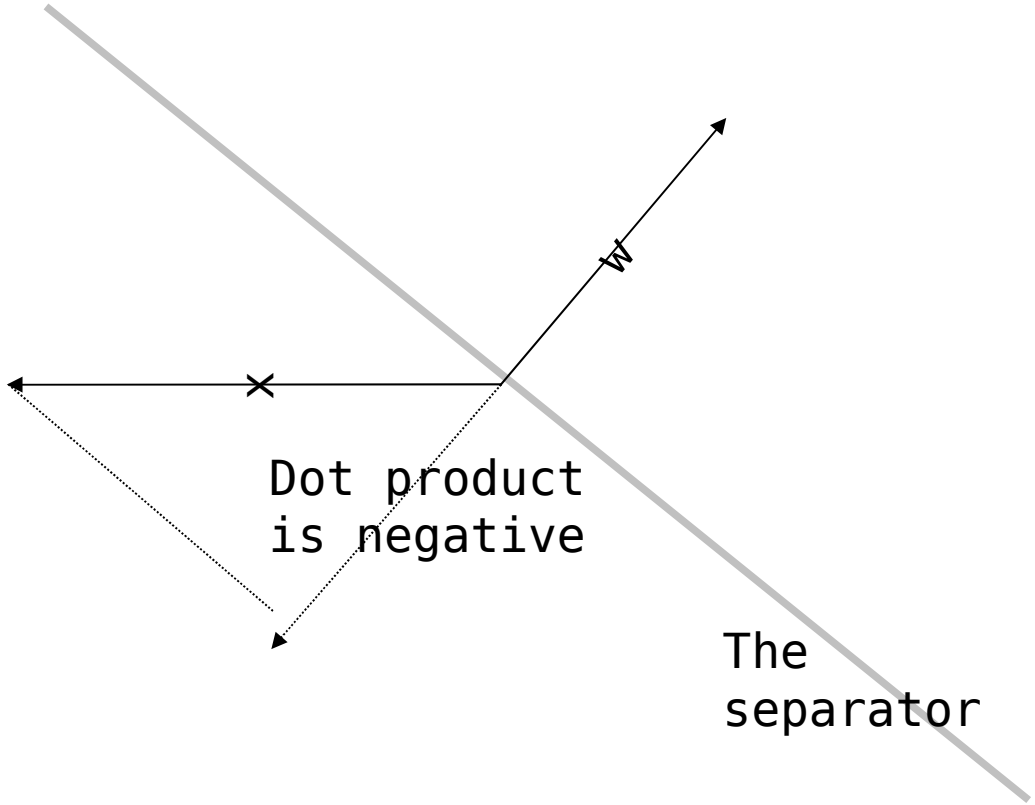
$$f(x) = \begin{cases} 1 & \text{if } \mathbf{w}.\mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- In English?
  - x is a datapoint represented as a vector
  - w is a vector that defines the separating hyperplane (it is perpendicular to it)
  - This function tells you which side of the hyperplane your point lies
  - (b defines an offset from the origin)

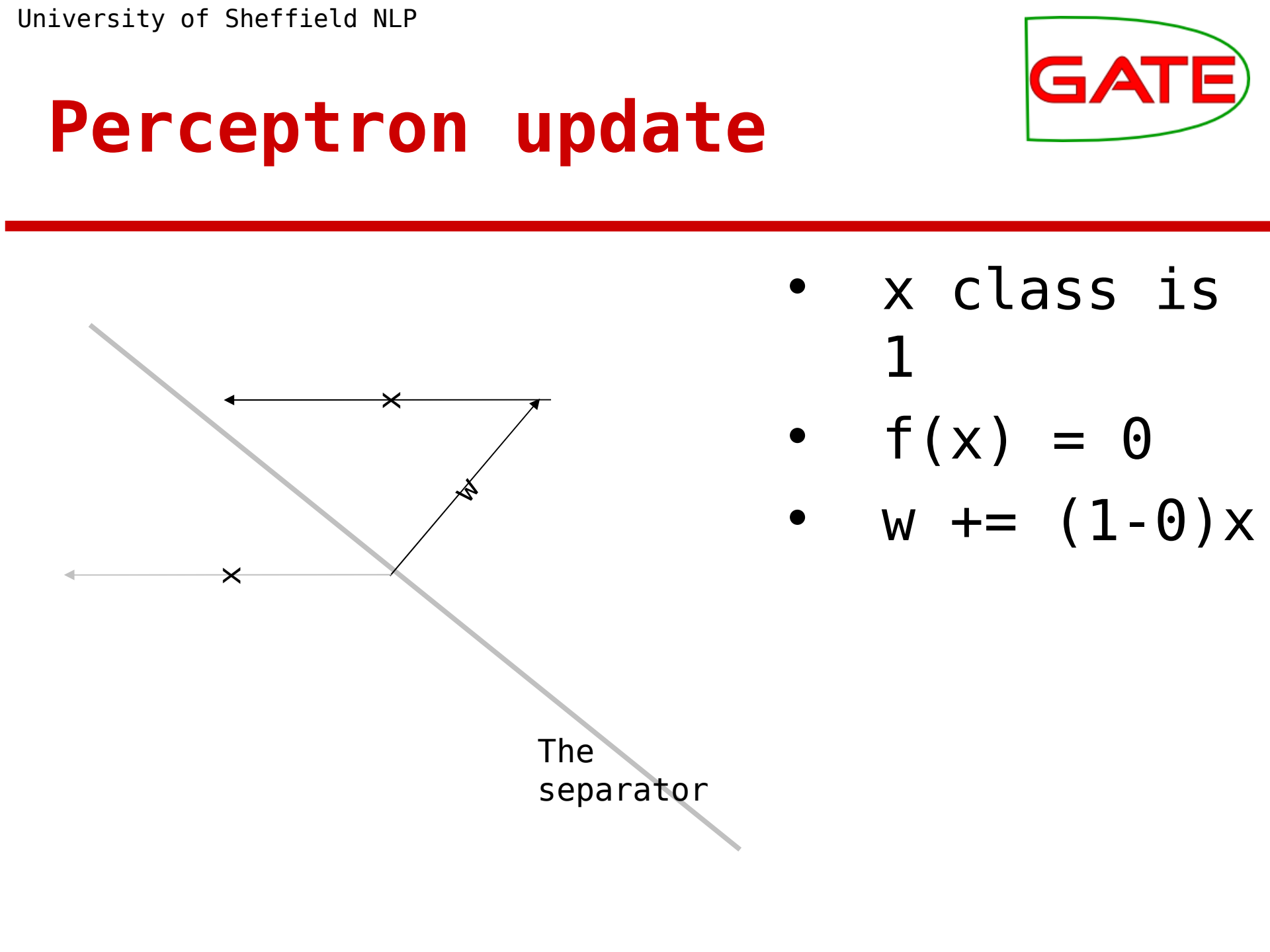
More perceptron

- How does it learn?
  - Each datapoint is annotated with class value 1 or 0
  - Function returns 1 or 0 depending on which side of the separator the point lies
  - Calculate difference between actual and desired output
  - Multiply input vector by this delta and add it to the weight vector
  - Given sufficient iterations the separator will find a solution

Perceptron update



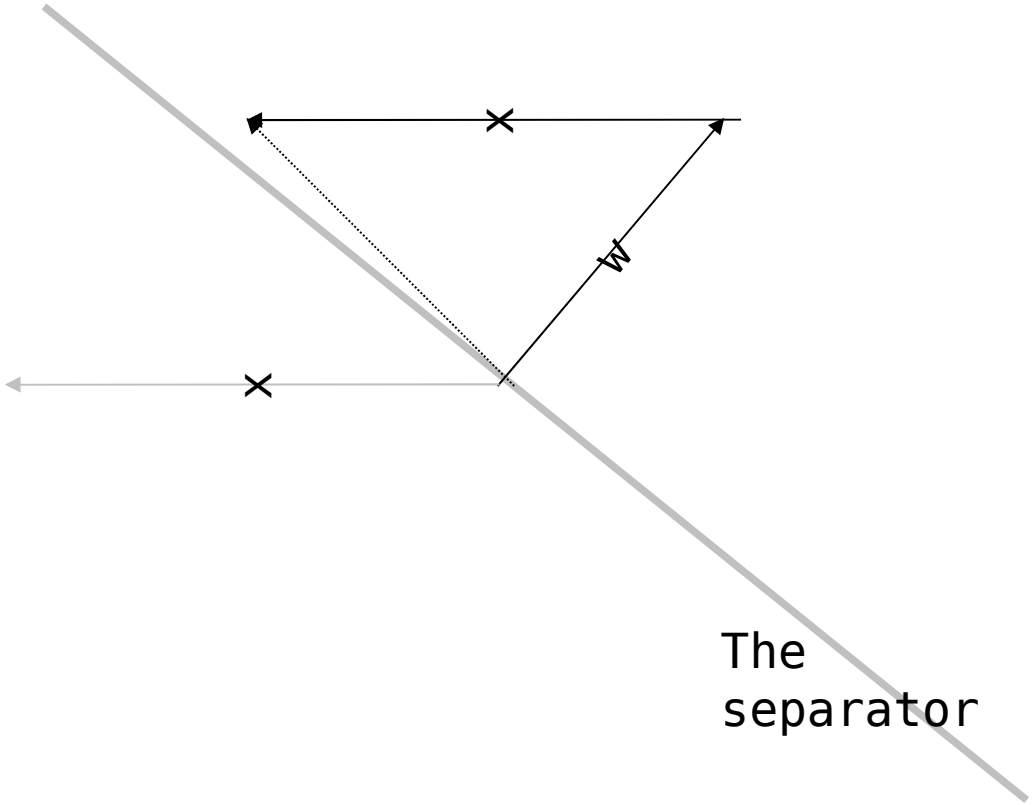
- Dot product is negative, so f=0
- But x is a positive example!
- Oh no! Must update



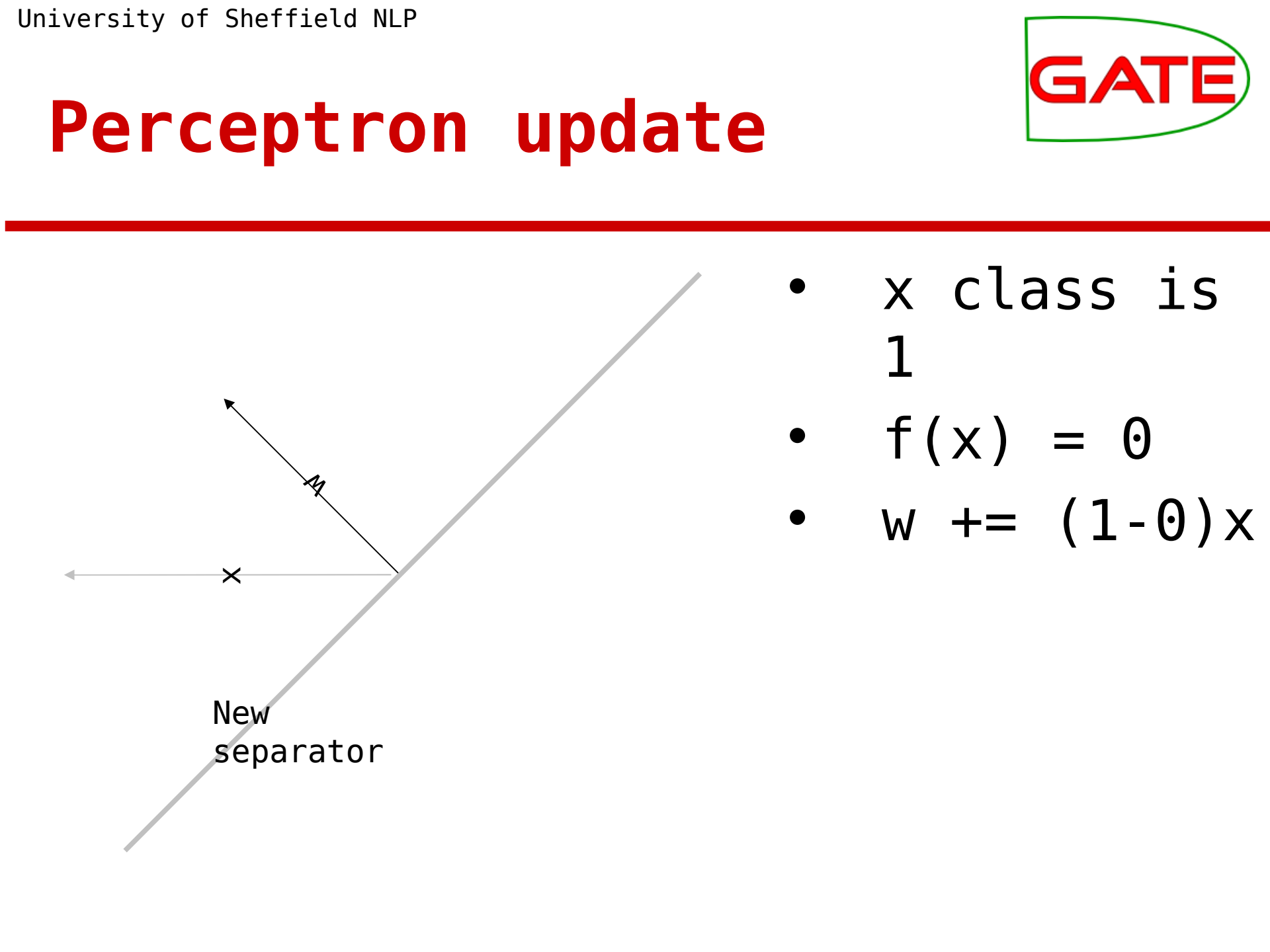
Perceptron update

- x class is 1
- f(x) = 0
- w += (1-0)x

Perceptron update



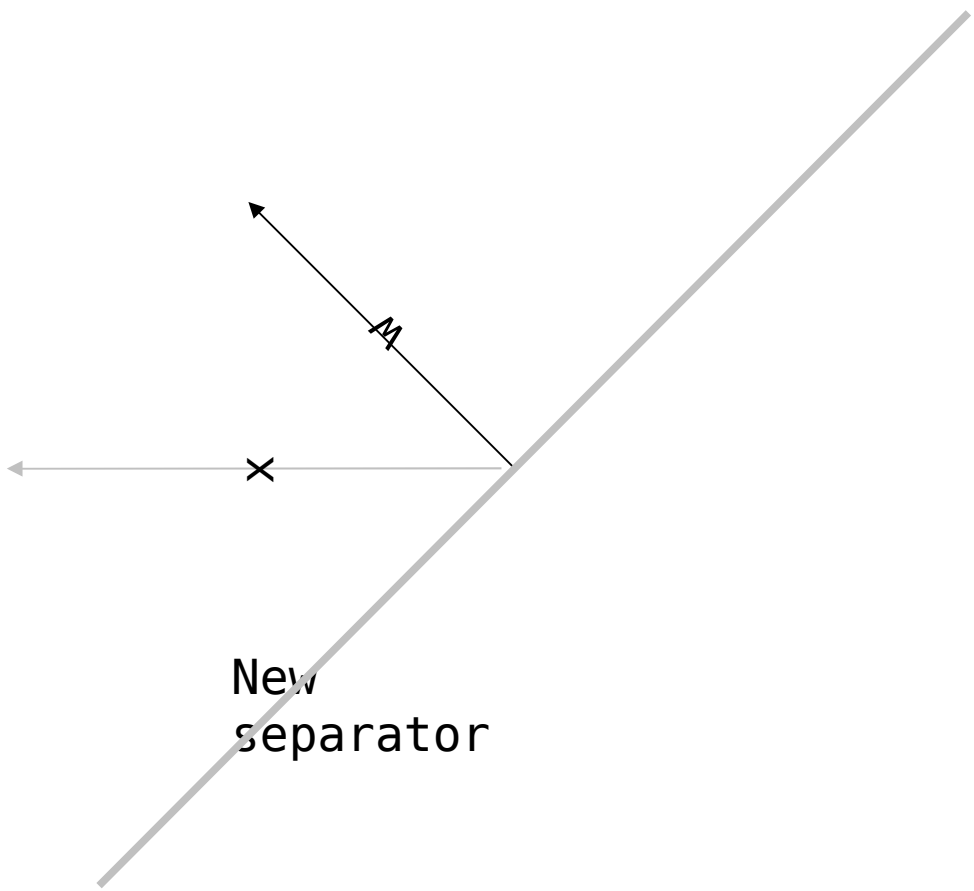
- x class is 1
- f(x) = 0
- w += (1-0)x



Perceptron update

- x class is 1
- f(x) = 0
- w += (1-0)x

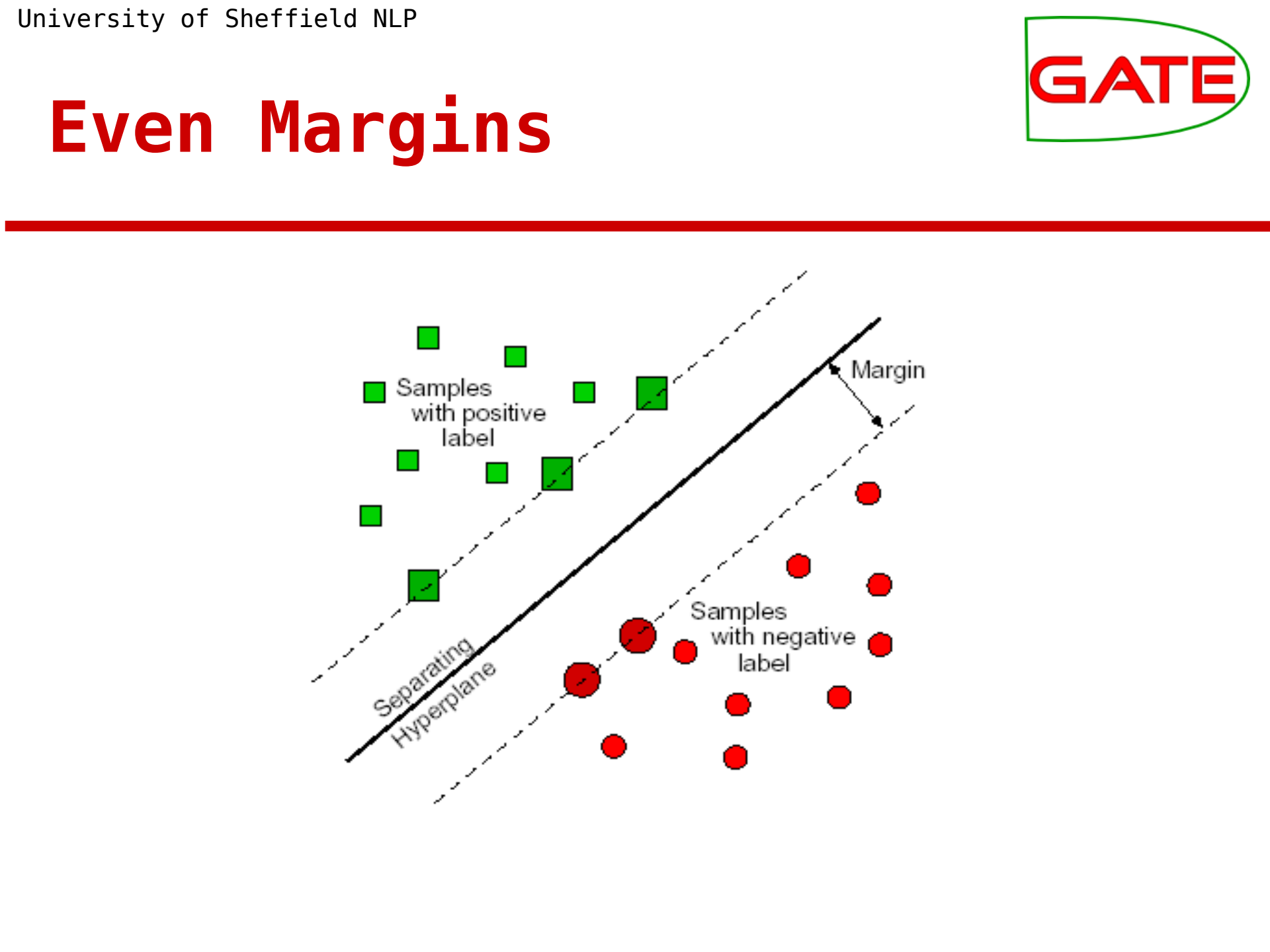
Perceptron update



- Now x is on the right side of the separator!

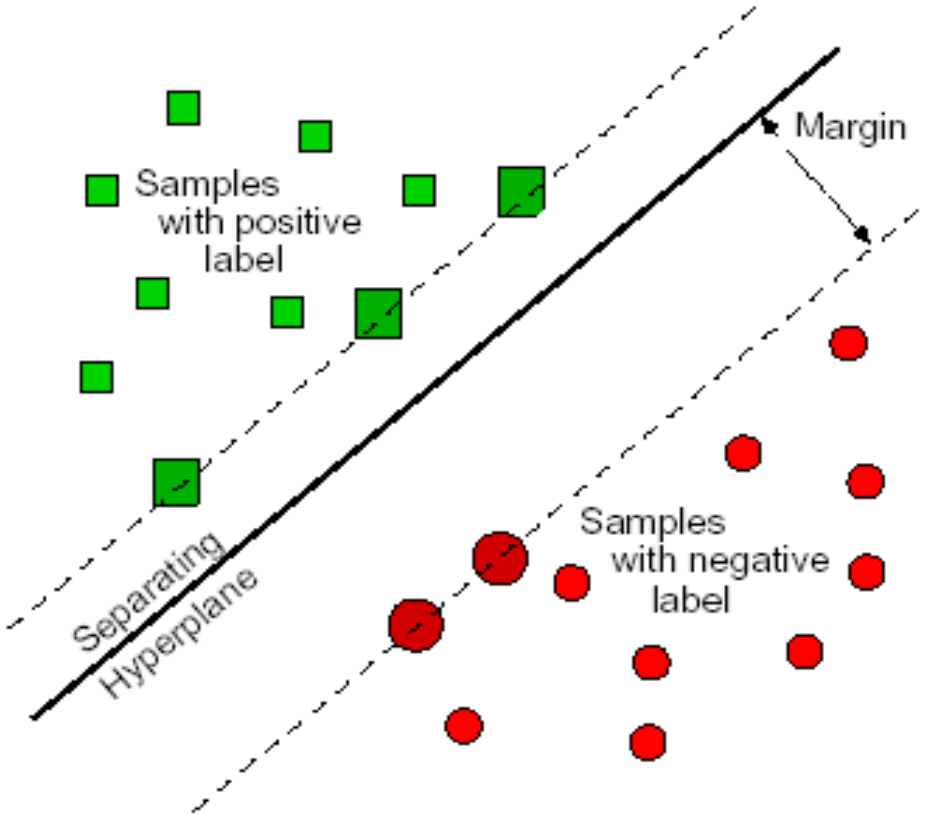
Perceptron with Uneven Margins

- Both Perceptron and SVM implement “uneven margins”
- (PAUM stands for Perceptron Algorithm with Uneven Margins)
- This means that it doesn't position the separator right between the points, but over one side



Even Margins

Uneven Margins



Why Uneven Margins?

- In NLP the datasets are often very imbalanced
- For example if you are finding instances of “Person”, you will have very many words that are not people and only a few that are
- Uneven margins may help with this
- Y. Li, K. Bontcheva, and H. Cunningham. Using Uneven Margins SVM and Perceptron for Information Extraction. Proceedings of Ninth Conference on Computational Natural Language Learning (CoNLL-2005), pp. 72-79. 2005.

Some Other Algorithms

- Batch Learning PR also includes the following from Weka
  - Naïve Bayes
    - Uses Bayes' theorem (probabilities) to determine the most likely class given attributes and training corpus
  - K-Nearest Neighbour
    - Determines class of a point based on k training points positioned geometrically closest to it
  - C4.5 (decision tree)
    - Makes a series of binary decisions that determine the class of a point based on its attribute values (e.g. “is string length > 3?”)