Queen's University | Faculty of Arts and Science

Classroom **PROCEDURES**

Get vaccinated

Provide vaccination proof

Do the daily COVID screen

Don't attend when ill

Wear a mask

Leave room promptly

Wash hands frequently

Don't consume drinks/food

QUartsci.com/Fall2021

CISC/CMPE452/COGS400
Pocket Algorithm

Farhana Zulkernine

Perceptron Learning

- Two types of learning:
  1. **Simple Feedback learning - Done**

     Uses the correct/incorrect feedback and info about (y>=D) or (y<D) to change weights.

     $$\Delta w = \pm c.x$$

  2. **Error Correction Learning**

     Uses an error measure (D-y) to adapt the weight vector.

     $$\Delta w = \pm c.|E|.x$$

     Here, c is the learning rate, x is input, and |E| is the adjustment for the calculated error (D-y).

General Rule: If D>=0 and y<0, increase weight, otherwise decrease weight.

Error Correction Learning

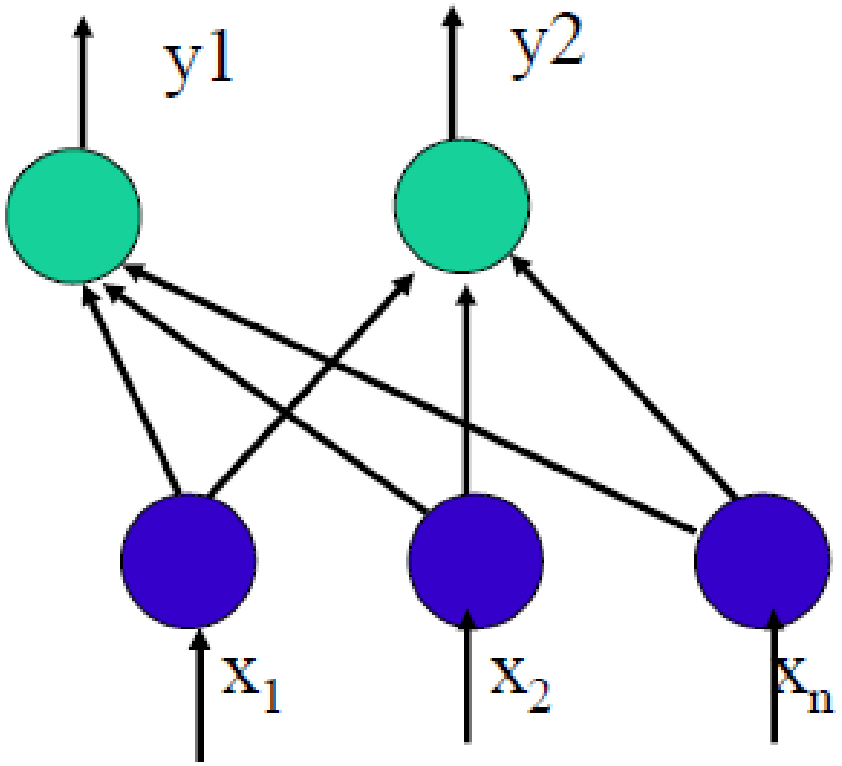
- Range of the input and output values should be taken into account
  - Note that for both D and y ϵ {-1, +1}, (D-y) ϵ {-2, 0, +2}
  - c * (d-y) will be high.
  - so c should be ½ of what it would be for y ϵ {0, 1}
- In most cases, designing a model requires preprocessing the data, deciding on the initial weight values, terminating condition and the learning rate.

Categorical Inputs

- What if input values are categorical?
  - E.g. color ϵ {red, blue, green, yellow}
- The simplest alternative:
  - Generate four new dimensions: red, blue, green and yellow
  - Recode categorical value as a binary vector [red, blue, green, yellow] = $\{0,1\}^4$.
    - For example, if red=0, blue=0, green=1, yellow=0 then "green" can be represented as (0,0,1,0).
- You can use 2 digits to represent the 4 possible values [(0,0),(0,1),(1,0),(1,1)] – Problems
  - Mapping of layers get complicated
  - Results in longer training time and need more neurons in hidden layers.
- In the general case, if an attribute (e.g. color) can take one of *n* different values, then *n* new dimensions are obtained.

Multiple Output Nodes

- When you have more than one neuron at the output layer, each neuron identifies a specific class (outputs 1 or 0).
- You train y2 the same way you train y1.



Initial Weight Values

- Depends on the distribution of the input data values since weights multiply the input data. If feature value is large (e.g., house price), weight should be small and vice versa.
- Very small or large weight values (less than < 0.05 or >5) may take a long time to adjust.
- If all data feature values are similar, a good value to start with is 1. Determine empirically.
- Additional statistical analysis can be done to determine the range and variance of feature values.

Terminating Condition

- Until all data are correctly classified
  - Problem ***
    - Data may not be linearly separable → results in infinite loop
      - Add a maximum number of iteration
    - The value of *c* may be too high and the weight vector fluctuates too much.
      - Try using a lower value.
- Until a fixed number of iteration has been run
  - One iteration = Running with $X_1..X_p$ once
- Until acceptable error level is reached
  - Error = (misclassified data points / total data points) <= threshold
  - When data is not linearly separable – How do we know that?
- Combine multiple of the above conditions

Choice of c

- If *c* is too small, the algorithm will make very small changes to the weights each time → very long training time
- If *c* is too big, the weight changes will be too much and the data that were previously correctly classified may be misclassified again.
  - The separator line will fluctuate its slope too much and never reach the correct slope.

Choice of c (cont…)

- A common choice is $c = 1$.
- To ensure that the sample x is correctly classified following the weight change
- $(w \pm \Delta w).x$ must be of the opposite sign of $w.x$

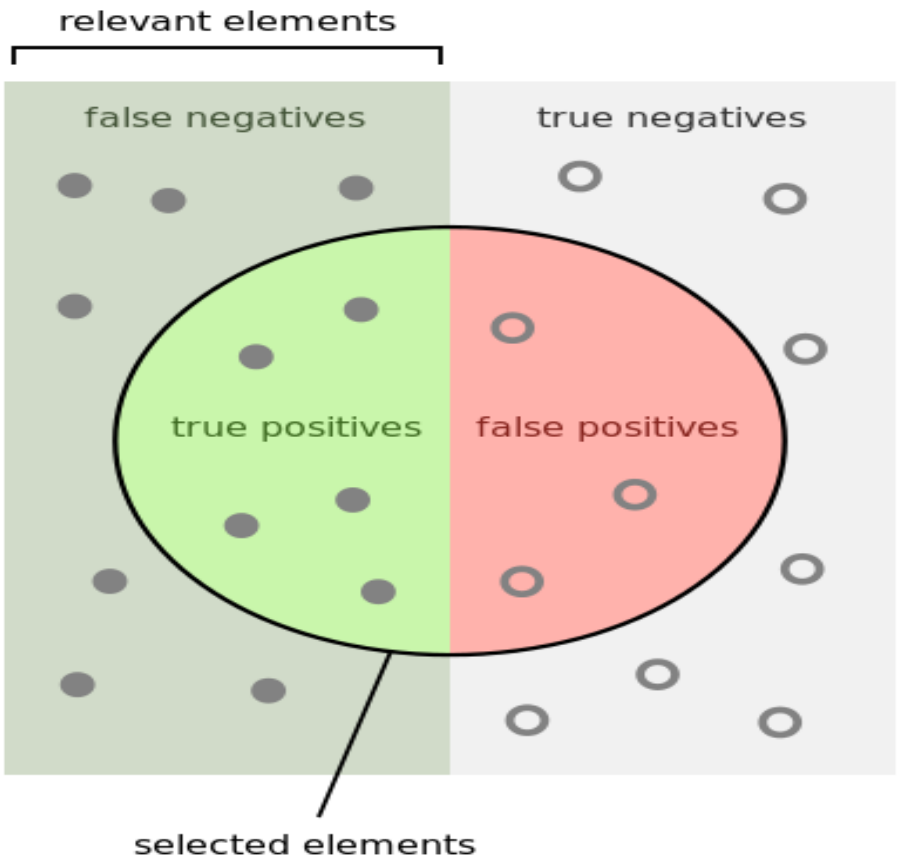
  $\Rightarrow |\Delta w \,.\, x| > |w \,.\, x|$ where |x| means abs(x)

  $\Rightarrow c\,|x \,.\, x| > |w \,.\, x|$ since $\Delta w = cx$

  $$\Rightarrow c > \frac{|w \,.\, x|}{|x \,.\, x|}$$

Validation of a Classification Model

- How do you validate that your model works?
- Popular measures in classification
  - Precision
  - Recall
  - Sensitivity
  - Specificity
  - F-measure
  - Confusion matrix
  - ROC



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Wikipedia

Precision – Recall – Sensitivity - Specificity

- Precision – relevant/total selected

$$\frac{TP}{TP + FP}$$

- Recall – relevant/total relevant = Sensitivity

$$\frac{TP}{TP + FN}$$

- Specificity – ratio of negatives (medical domain)

$$\frac{TN}{TN + FP}$$

F-measure

- The traditional F-measure or balanced F-score (also called harmonic mean of precision and recall):

$$\frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion Matrix

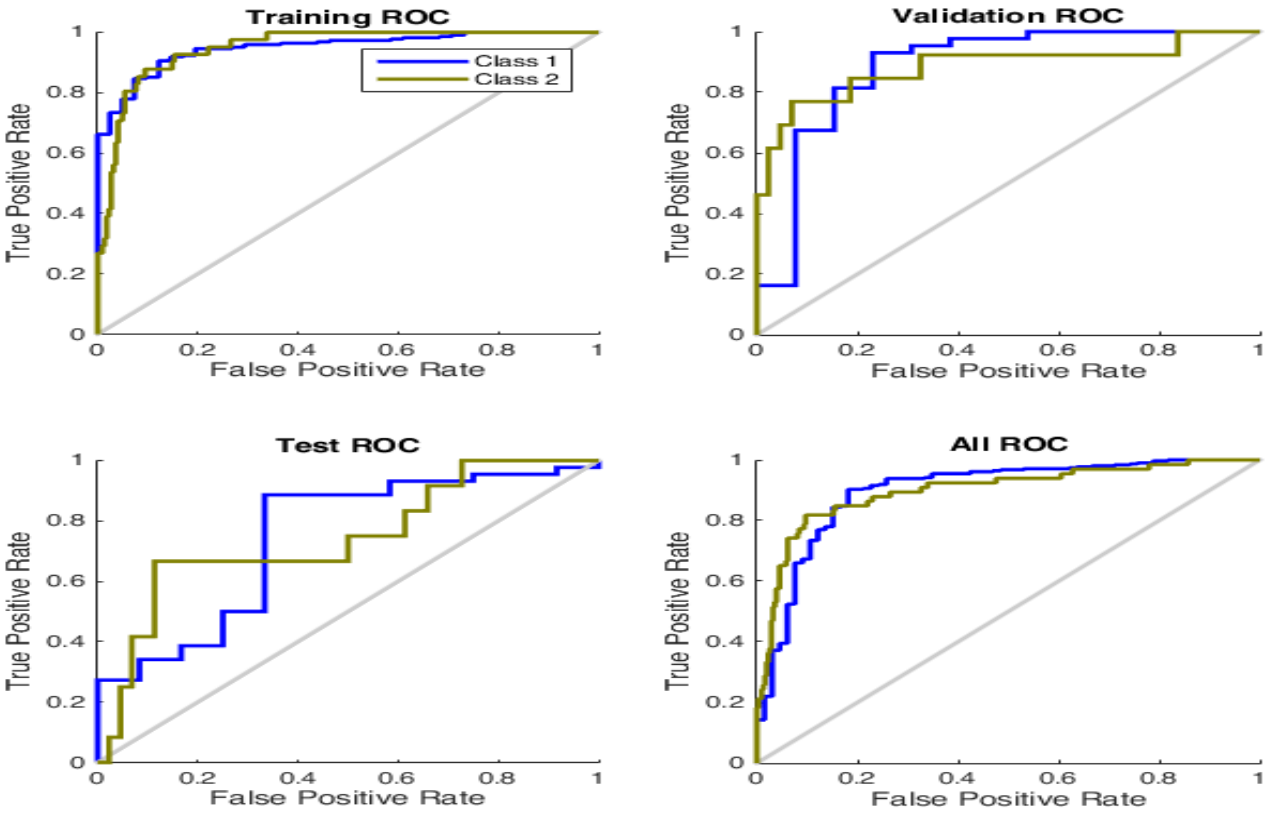
The column on the far right of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified. These metrics are often called the precision (or positive predictive value) and false discovery rate, respectively. The row at the bottom of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified. These metrics are often called the recall (or true positive rate) and false negative rate, respectively. The cell in the bottom right of the plot shows the overall accuracy.

ROC Curve

Change in diagnostic ability of a binary classifier as its discrimination threshold is varied.

TPR = Sensitivity or Recall FPR = (1- Specificity)

Receiver Operating Characteristic (ROC) Curve from Matlab

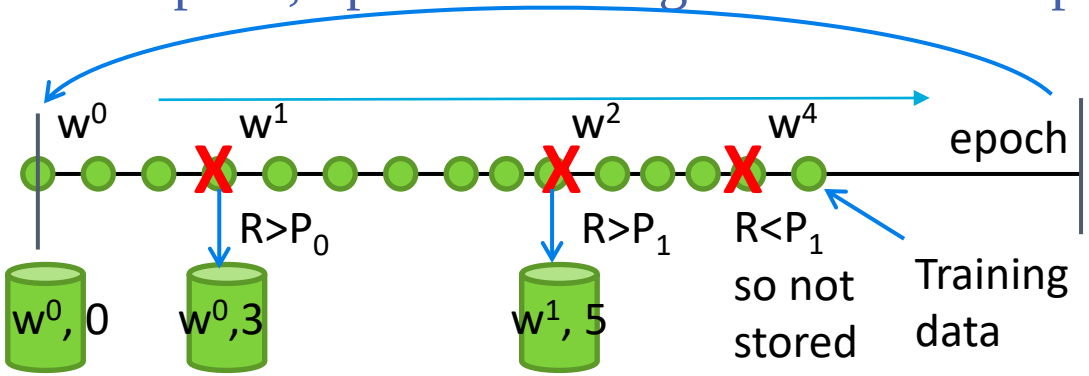


Not Linearly Separable – Algorithms

- The "Pocket" and "Least Mean Squares" (LMS) algorithms attempt to achieve robust classification when two classes are not linearly separable.

The Pocket Algorithm

- The pocket algorithm is a useful modification of the perceptron training algorithm
  - Weight change mechanism is the same.
  - Identifies the weight vector $w^i=(w_1,\ldots,w_n)$ with the **longest unchanged run** (# of points correctly classified: P) as the best solution so far.
  - Stores *the best weight vector* in the "pocket" as well as *the best run length* associated with the weight vecors (w^i,P_i).
  - Pocket contents are replaced with a new weight vector when a longer successful run R (R points are correctly classified) is found.
  - After each epoch, update the weight vectors with pocket w^p



Pocket Algorithm with *Ratchet*

- A lucky run of several successes may allow a poor solution to replace a better solution in the pocket.
- To avoid this, the Pocket algorithm with ratchet ensures that the pocket weights always "ratchet up"
  - w^1 in the pocket is replaced by w^2 that has longer successful run *only after testing on all training samples* whether w^2 does correctly classify a greater number of samples in the training data than w^1.
  - Expensive computation.

Results

- The Pocket algorithm gives good results, although there is no guarantee of reaching the optimal weight vector in a reasonable number of iterations due to the distribution of data points.
- The best solution may never get saved in the pocket and hence ratchet version in that case will also not be able to find that solution.
  - If the best solution ever gets selected to be put into the Pocket then ratchet will test it for all solutions and keep it in the pocket.

Perceptron and Linear Separability

- In general, networks of perceptron-like processors can solve most non-linearly separable tasks by using more than one layer of processors.
- Rosenblatt (and others) realized this in the 1960's, but did not have a learning rule that would work effectively with more than one level (it wasn't invented until the mid 1970's).