Class #06:
Non-Binary Classification;
Evaluating ML Models

Machine Learning (COMP 135)

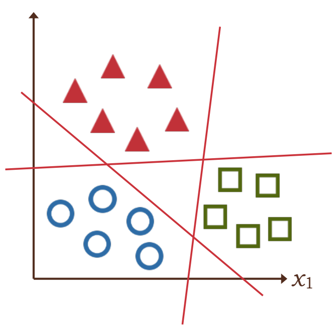
1

Binary and Other Classification

- We will generally discuss binary classifiers, which divide data into one of two classes
- Linear classifiers are inherently binary, defining the classes based on two regions, separated by a linear function
  - Many of the things we discuss can be applied to more than two classes, however

2

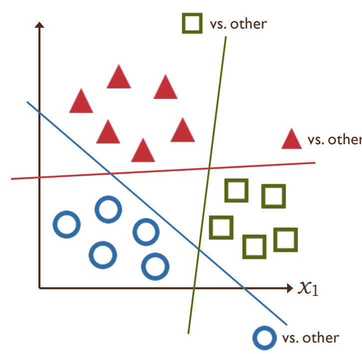
Extending Binary Linear Classification



- In the presence of more than two classes, a single basic linear classifier can't properly divide data
- Even if that data is linearly separable by class, any single line drawn must include elements of more than one class on at least one side
- We can combine ***multiple*** such classifiers, however…

3

One-Versus-All Classification (OVA)

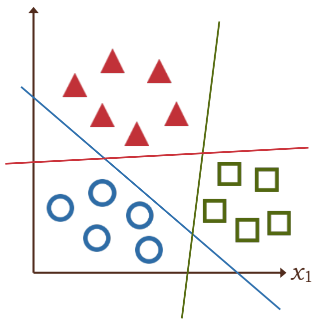


- In an OVA scheme, with k different classes:

1. Train k different $1/0$ classifiers, one for each output class
2. On any new data-item, apply each classifier to it, and assign it the class corresponding to the classifier for which it receives a 1

4

Issues with OVA Classification



- The basic OVA idea requires that each linear classifier separate one class from all others
- As the number of classes increases, this added linear separability constraint gets harder to satisfy

One-Versus-One Classification (OVO)

- Another idea is to train a separate classifier for each possible *pair* of output classes
  - Only requires each such pair to be ***individually*** separable, which is somewhat more reasonable
  - For k classes, it requires a larger number of classifiers:

$$\binom{k}{2} = \frac{k\,(k-1)}{2} = \mathrm{O}(k^2)$$

  - Relative to the size of data sets, this is generally manageable, and each classifier is often simpler than in an OVA setting
- A new data-item is again tested against all the classifiers, and given the class of the ***majority*** of those for which it is given a non-negative (1) value
  - May still suffer from some ambiguities

Evaluating a Classifier

- It is often useful to separate the results generated by a classifier, according to what it gets right or not:
  - True Positives (TP): those that it identifies correctly as relevant
  - False Positives (FP): those that if identifies wrongly as relevant
  - False Negatives (FN): those that are relevant, but missed
  - True Negatives (TN): those it correctly labels as non-relevant
- These categories make sense when we are interested in separating out one relevant class from another (again, we return to binary classification for simplicity)
- Of course, relevance depends upon what we care about:
  - Picking out the actual earthquakes in seismic data (earthquakes are relevant; explosions are not)
  - Picking out the explosions in seismic data (explosions are relevant; earthquakes are not)

Evaluating a Classifier

- It is often useful to separate the results generated by a classifier, according to what it gets right or not:
  - True Positives (TP): those that it identifies correctly as relevant
  - False Positives (FP): those that if identifies wrongly as relevant
  - False Negatives (FN): those that are relevant, but missed
  - True Negatives (TN): those it correctly labels as non-relevant

		Classifier Output	
		Negative (0)	Positive (1)
Ground Truth	Negative (0)	TN	FP
	Positive (1)	FN	TP

Basic Accuracy

- The simplest measure of accuracy is just the fraction of correct classifications:

$$\frac{\#\ \text{Correct}}{|\text{Data-set}|} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Basic accuracy treats both types of correctness—and therefore both types of error—as the same
- This isn't always what we want however; sometimes false positives and false negatives are quite different things

Basic Accuracy

- The simplest measure of accuracy can also be misleading, depending upon the data-set itself:

$$\frac{\#\ \text{Correct}}{|\text{Data-set}|} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- In a data-set of 100 examples, with 99 positive, and only a single negative example, any classifier that simply says positive (1) for everything would have 99% "accuracy"
- Such a classifier might be entirely useless for real-world classification problems, however!

Confusion Matrices

- One way to separate out positive and negative examples, and better analyze the behavior of a classifier is to break down the overall success/failure case by case
- For 100 data-points, 50 of each type, we might have behavior as shown in the following table:

		Classifier Output	
		Negative (0)	Positive (1)
Ground Truth	Negative (0)	40	10
	Positive (1)	1	49

- What can this tell us?

Confusion Matrices

		Classifier Output	
		Negative (0)	Positive (1)
Ground Truth	Negative (0)	40	10
	Positive (1)	1	49

- In this data, the *overall* accuracy is $89/100$ = 89%
- However, we see that the accuracy over the two types of data is quite different:
  1. For negative data, accuracy is just $40/50$ = 80%, with a 20% rate of false positives
  2. For positive data, accuracy is $49/50$ = 98%, with only a 2% rate of false negatives

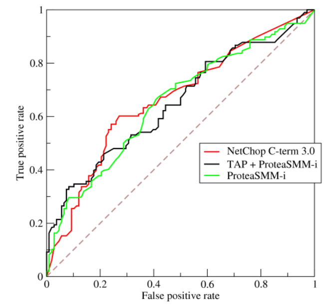
Other Measures of Accuracy

- We can focus on a variety of metrics, depending upon what we care about
  - "C = X" is "Classifier says X", & "T = Y" is "Truth is Y"

Metric	Formula	How often…	Probability
True Positive Rate (Recall)	$\frac{TP}{TP+FN}$	positive examples are correctly labeled	P(C = 1 | T = 1)
True Negative Rate (Specificity)	$\frac{TN}{TN+FP}$	negative examples are correctly labeled	P(C = 0 | T = 0)
Positive Predictive Value (Precision)	$\frac{TP}{TP+FP}$	examples labeled positive actually are positive	P(T = 1 | C = 1)
Negative Predictive Value	$\frac{TN}{TN+FN}$	examples labeled negative actually are negative	P(T = 0 | C = 0)

13

ROC Curves



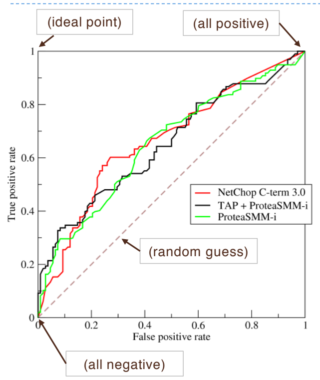
$$TPR = \frac{TP}{TP+FN}$$

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

- Another way to look at classifier performance is the *ratio* of the rates of true positives and false ones
- That is, we compare the percentage of the true positives the classifier gives the right result for, and the percentage of errors it makes by mistakenly classifying negative examples as positive

14

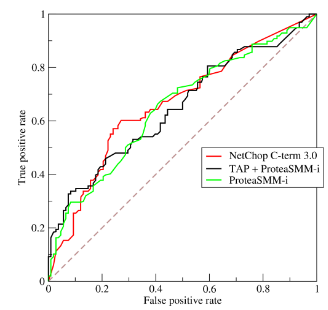
ROC Curves



- Some obvious facts :
1. A ***perfect*** classifier would give us 100% success for true positives, with a 0% rate of false ones
2. A ***coin-flip*** classifier would achieve equal rates of each
3. Any classifier that is always positive hits all true ***and*** false positives
4. One that is always negative has ***no*** true or false positives

15

Area Under ROC Curves (AUROC or AUC)



- The ROC curve can be very nuanced, and it is not always obvious from the curve itself how different algorithms measure up and compare
- A metric for comparing multiple curves is the area under them
  - A larger area means the curve gets a higher true positive success rate ***earlier*** (i.e., with fewer false positives) than one of smaller area

16

Probabilistic Classifiers

- The basic perceptron linear classifier assigns each data-item to a single specific class
- Other approaches generate probability distributions over the data: that is, they assign each data-item a probability of being in the positive class
  - A probability of 1.0 means the data-item is *definitely* positive
  - A probability of 0.0 means the data-item is *definitely* negative
  - A probability $0.0 < p < 1.0$ means the data-item has *some chance* of being in *either* class
- **Question**: how can we turn the outputs of a probabilistic classifier *back into* a discrete $(1/0)$ classification?
  - One possibility is a threshold: pick a probability T such that everything assigned a probability $p \geq T$ is assigned positive, all else negative

17

Log-Loss for Probabilistic Classification

- For any data-item x_i (of N total), let y_i be the correct class-label $(1/0)$, and let p_i be the probability assigned by the classifier that the data-item is in fact 1
- We can then define the logarithmic loss (log-loss) for this classifier across the entire data-set:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i)$$

- This measures cross entropy between the true distribution of labels in our data and the classifier's label distribution (that is, it measures the amount of extra noise introduced by the classifier, relative to the true noisiness of the data-set)

18

Log-Loss for Probabilistic Classification

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i)$$

- If the true class of a data-item is 1, then the log-loss only sums up the *first* term in the right-hand equation
  - The closer probability p_i is to 1 in this case, the closer loss is to 0
- If the true class of a data-item is 0, then the log-loss only sums up the *second* term in the right-hand equation
  - The closer probability p_i is to 0 in this case, the closer loss is to 0
  - Remember that by convention, we let $0 \log 0 = 0$

19

AUC for Probabilistic Classification

- If we are using a probabilistic classifier, then the area under the ROC curve for the classifier actually measures something else of real interest:

$$\text{AUC} = P(p_i > p_j \mid y_i = 1 \text{ AND } y_j = 0)$$

  - Here, again, let p_i is the probability assigned by the classifier that the data-item is positive (1)
- This measures, for any given data-items x_i and x_j, one positive and one negative, the chance that the classifier gives the positive one a higher probability than then negative one

20

A Problem Case for AUC

- Suppose we have data as shown, and two classifiers, C_1 and C_2 that assign probabilities as given in this table:

	y_1	C_1	C_2
x_1	0	0.10	0.15
x_2	0	0.20	0.25
x_3	0	0.30	0.35
x_4	0	0.45	0.50
x_5	1	0.60	0.55
x_6	1	0.75	0.65
x_7	1	0.80	0.70
x_8	1	0.95	0.85

- Although the classifiers differ in the values they assign each data-point, they are ***both*** in one sense ***perfect***
  - There are threshold values for which each classifies every input correctly
  - In fact, for any threshold value ($0.50 < T \le 0.55$) ***both*** will classify everything correctly

21

A Problem Case for AUC

- Varying threshold T does change the TPR and FPR of each classifier
  - However, each always has TPR = 1.0 or FPR = 0.0 (or both)
  - It is easy to verify that AUC = 1.0 (the same) for each classifier

	y_1	C_1	C_2
x_1	0	0.10	0.15
x_2	0	0.20	0.25
x_3	0	0.30	0.35
x_4	0	0.45	0.50
x_5	1	0.60	0.55
x_6	1	0.75	0.65
x_7	1	0.80	0.70
x_8	1	0.95	0.85

T	TPR_1	FPR_1	TPR_2	FPR_2
0.1	4/4	4/4	4/4	4/4
0.2	4/4	3/4	4/4	3/4
0.3	4/4	2/4	4/4	2/4
0.4	4/4	1/4	4/4	1/4
0.5	4/4	0/4	4/4	1/4
0.6	4/4	0/4	3/4	0/4
0.7	3/4	0/4	2/4	0/4
0.8	2/4	0/4	1/4	0/4
0.9	1/4	0/4	0/4	0/4
1.0	0/4	0/4	0/4	0/4

22

Choosing an Appropriate Measure

	y_1	C_1	C_2
x_1	0	0.10	0.15
x_2	0	0.20	0.25
x_3	0	0.30	0.35
x_4	0	0.45	0.50
x_5	1	0.60	0.55
x_6	1	0.75	0.65
x_7	1	0.80	0.70
x_8	1	0.95	0.85

- AUC is not a useful metric here, since it rates each classifier the same
- Instead, we can compare the log-loss, which is better (lower) for C_1 because it consistently outputs a probability that is ***closer*** to the correct value (i.e., higher for the 1's and lower for the 0's)

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1 - y_i)\log(1 - p_i)$$

$$\mathcal{L}(C_1) \approx 0.2945$$

$$\mathcal{L}(C_2) \approx 0.3902$$

23