Machine learning system design
AI4003-Applied Machine Learning

Dr. Mohsin Kamal

Department of Electrical Engineering
National University of Computer and Emerging Sciences, Peshawar, Pakistan

Building a spam classifier

From: cheapsales@buystufffromme.com
To: mohsin.kamal@nu.edu.pk
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - $100
Med1cine (any kind) - $50
Also low cost M0rgages available.

From: suleman.mir@nu.edu.pk
To: mohsin.kamal@nu.edu.pk
Subject: Research work

Aoa Mohsin,
I hope this email finds you well.
We should discuss the project today.
Suleman Mir

Building a spam classifier

Supervised learning. x =features of email. y = spam (1) or not spam (0).
Features x: Choose 100 words indicative of spam/not spam.
For example: deal, buy, discount, sale, now

From: cheapsales@buystufffromme.com
To: mohsin.kamal@nu.edu.pk
Subject: Buy now!

Deal of the week! Buy now!

Note: In practice, take most frequently occurring n words (10,000 to 50,000) in training set, rather than manually pick 100 words.

Building a spam classifier

How to spend your time to make it have low error?

- Collect lots of data
  - For example: "honeypot" project.
- Develop sophisticated features based on email routing information (from email header).
- Develop sophisticated features for message body, e.g. should "discount" and "discounts" be treated as the same word? How about "deal" and "Dealer"? Features about punctuation?
- Develop sophisticated algorithm to detect misspellings (e.g. m0rtgage, med1cine, w4tches.)

Recommended approach

- Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data.
- Plot learning curves to decide if more data, more features, etc. are likely to help.
- Error analysis: Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

Error analysis

$m_{cv} = 500$ examples in cross validation set
Algorithm misclassifies 100 emails.
Manually examine the 100 errors, and categorize them based on:

1. What type of email it is
2. What cues (features) you think would have helped the algorithm classify them correctly.

Pharma:
Replica/fake:
Steal passwords:
Other:

Deliberate misspellings:
(m0rgage, med1cine, etc.)
Unusual email routing:
Unusual (spamming) punctuation:

The importance of numerical evaluation

Should discount/discounts/discounted/discounting be treated as the same word?
Can use "stemming" software (For example: "Porter stemmer")
universe/university.
Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.
Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.
Without stemming: With stemming:
Distinguish upper vs. lower case (Mom/mom):

Cancer classification example

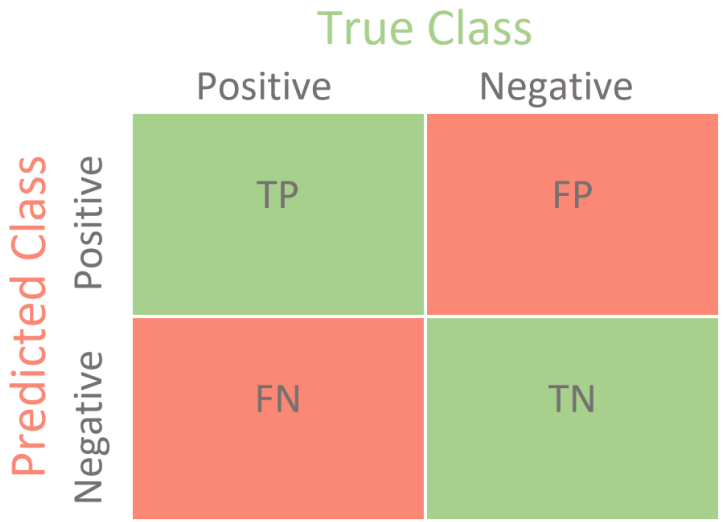
Train logistic regression model $h_\theta(x)$. ($y = 1$ if cancer, $y = 0$ otherwise)
Find that you got 1% error on test set.
(99% correct diagnoses)

Only 0.50% of patients have cancer.

Accuracy is generally not the preferred performance measure for classifiers, especially when you are dealing with skewed datasets (i.e., when some classes are much more frequent than others).

Confusion Matrix

- A much better way to evaluate the performance of a classifier is to look at the **confusion matrix**.
- The general idea is to count the number of times instances of class A are classified as class B.
- To compute the confusion matrix, you first need to have a set of predictions, so they can be compared to the actual targets.



Precision/Recall

$y = 1$ in presence of rare class that we want to detect

Precision

(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

$$\frac{\textit{True positives}}{\textit{No. of predicted positives}} = \frac{\textit{True positives}}{\textit{True pos}+\textit{False pos}}$$

Recall

(Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?)

$$\frac{\textit{True positives}}{\textit{No. of actual positives}} = \frac{\textit{True positives}}{\textit{True pos}+\textit{False neg}}$$

Trading off precision and recall

Logistic regression: $0 \leq h_\theta(x) \leq 1$

Predict 1 if $h_\theta(x) \geq 0.5$

Predict 0 if $h_\theta(x) < 0.5$

Suppose we want to predict $y = 1$ (cancer) only if very confident.

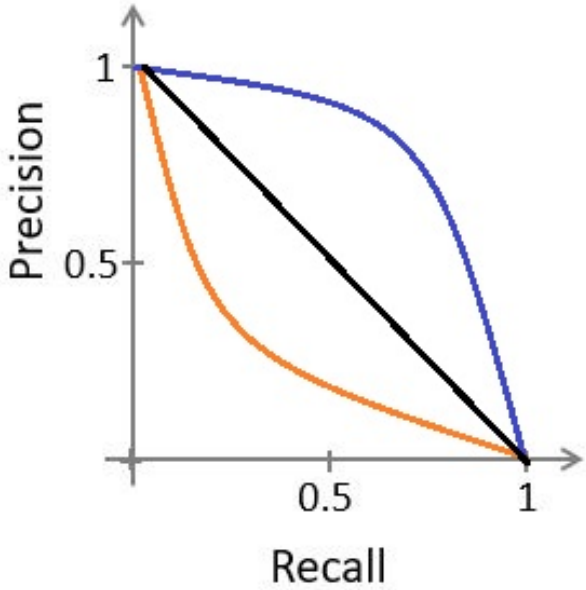
Higher precision, lower recall

Suppose we want to avoid missing too many cases of cancer (avoid false negatives).

Higher recall, lower precision

$$Precision = \frac{True\ positives}{No.\ of\ predicted\ positives}$$

$$Recall = \frac{True\ positives}{No.\ of\ actual\ positives}$$



More generally: Predict 1 if $h_\theta(x) \geq$ threshold.

F_1 Score (F score)

- It is often convenient to combine precision and recall into a single metric called the F1 score, in particular if you need a simple way to compare two classifiers. The F1 score is the harmonic mean of precision and recall.
- Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values.
- As a result, the classifier will only get a high F1 score if both recall and precision are high.

$$\text{Average} = \frac{P+R}{2}$$

$$F_1 \text{ score} = \frac{2}{\frac{1}{P}+\frac{1}{R}} = 2 \times \frac{P \times R}{P+R} = \frac{TP}{TP+\frac{FN+FP}{2}}$$

F_1 Score (F score)

How to compare precision/recall numbers?

	Precision (P)	Recall	Average	F_1 score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

- The F1 score favors classifiers that have similar precision and recall.
- This is not always what you want: in some contexts you mostly care about precision, and in other contexts you really care about recall.
- Suppose you train a classifier to detect shoplifters on surveillance images: it is probably fine if your classifier has only 30% precision as long as it has 99% recall (sure, the security guards will get a few false alerts, but almost all shoplifters will get caught).
- Unfortunately, you can't have it both ways: increasing precision reduces recall, and vice versa. This is called the precision/recall tradeoff.

Designing a high accuracy learning system

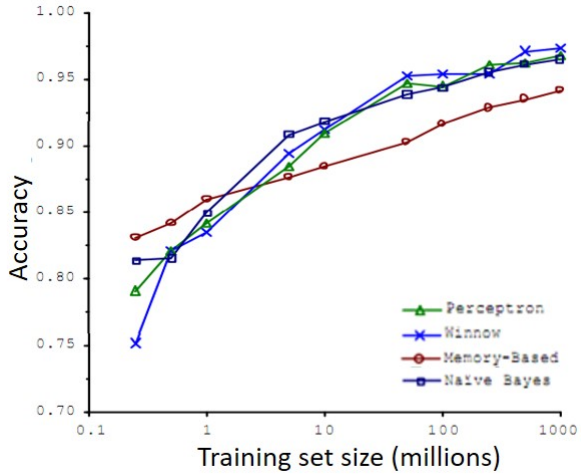
E.g. Classify between confusable words.

{to, two, too}, {then, than}

For breakfast I ate —— eggs.

Algorithms

- Perceptron (Logistic regression)
- Winnow
- Memory-based
- Naive Bayes



"It's not who has the best algorithm that wins. It's who has the most data."

Large data rationale

Assume feature $x \in \Re^{n+1}$ has sufficient information to predict y accurately.

Example: For breakfast I ate —— eggs.
Counterexample: Predict housing price from only size (feet^2) and no other features.

Useful test: Given the input x, can a human expert confidently predict y?

Large data rationale

Use a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units).
The algorithms will be low bias algorithms because of large parameters which can fit many complex functions.
If we run this algorithms on training set then hopefully,

$$J_{train}(\theta) \text{ will be small.}$$

If we use a very large training set then it will be unlikely to overfit.
Hence, $J_{train}(\theta) \approx J_{test}(\theta)$
If we get low variance as well then hopefully we will get a small $J_{test}(\theta)$
This guarantees that we have enough features due to which we are getting low bias and low variance.