error metrics handling skewed data

error metrics for skewed classes

cancer classification example

train a logistic regression model $h_\theta(x)$ $y = \begin{cases} 1 \text{ if cancer} \\ 0 \; otherwise \end{cases}$

the test set results in 1% error (99% correct diagnoses')

only 50% of the patients contract cancer (the classes are skewed)

```
function y = predictCancer(x)
    y = 0; %ignore x!
return
```

→ 0.5% error

→ 99.2% accuracy (0.8% error) → 99.5% accuracy (0.5% error)

an alternative evaluation metric for the above problem

precision/recall

y = 1 in presence of rare class that is the target to detect

y = 0
recall = 0

actual class

predicted class	**1**	**0**
1	true postives (+)	false positives (+)
0	false negatives (-)	true negatives (-)

precision

(of all patients where prediction = 1, what fraction actually has cancer?)

$$\frac{\text{true positives}}{\#\,\textbf{predicted}\text{ positives}} = \frac{\text{true positives}}{\text{true positives} + \textbf{false positives}}$$

recall

(of all patients that actually have cancer, what fraction was correctly detected as having cancer?)

$$\frac{\text{true positives}}{\#\,\textbf{actual}\text{ positives}} = \frac{\text{true positives}}{\text{true positives} + \textbf{false negatives}}$$

trading off precision and recall

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

predict 1 if $h_\theta(x) \geq 0.5$ (default)

predict 0 if $h_\theta(x) < 0.5$ (default)

suppose prediction of $y = 1$ (cancer) only if very confident

predict 1 if $h_\theta(x) \geq 0.7$ or 0.9

predict 0 if $h_\theta(x) < 0.7$ or 0.9

→ the model will produce higher precision but lower recall

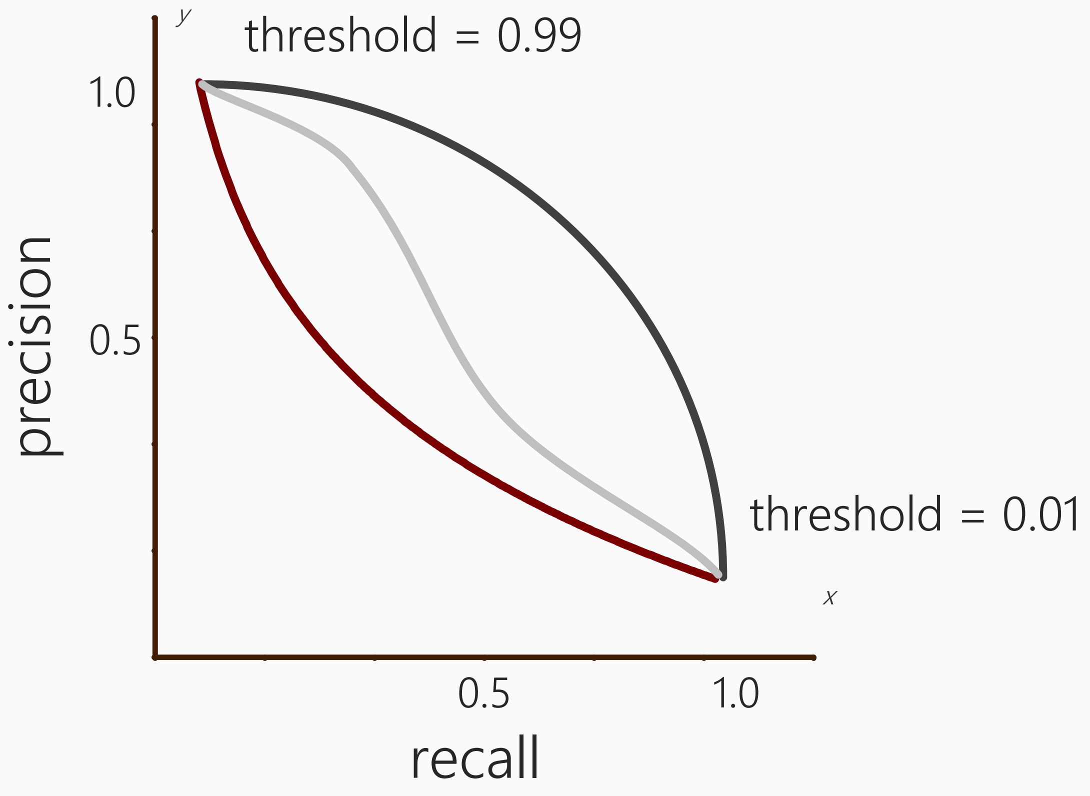
suppose avoidance to missing too many cases of cancer (avoid false negatives)

predict 1 if $h_\theta(x) \geq 0.7$ or 0.9

predict 0 if $h_\theta(x) < 0.7$ or 0.9

→ the model will produce lower precision but higher recall

in general terms: predict 1 if $h_\theta(x) \geq$ **given threshold**



F_1 score

comparing precision/recall metrics

precision and recall illustrate relationships among the two. using these metrics to select algorithms requires performance to be measured as a whole. the table below displays:

	precision (p)	recall (r)	~~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

average $= \frac{p+r}{2}$ F_1 score $= 2\frac{pr}{p+r}$

the average of precision and recall dictate that the 3rd algorithm is most attractive. this is misleading because that model predicted y =1 for the entire sample (recall = 100%) and greatly skewed the average performance between precision and recall

the F_1 score applies a weight to the lower value, making it penalize the metric accordingly. the illustration above shows the first algorithm as the most effective and conversely the 3rd algorithm substantially worse than compared to the average metric.

if $P = 0$ or $R = 0 \rightarrow F_1$ score $= 0$

if $P = 1$ or $R = 1 \rightarrow F_1$ score $= 1$

example

a logistic c regression classifier has been trained as follows:

predict y = 1 if $h_\theta(x) \geq$ threshold

predict y = 0 if $h_\theta(x) <$ threshold

different values for the threshold parameter will yield different values of precision (p) and recall (r). a reasonable way to determine the threshold would be to:

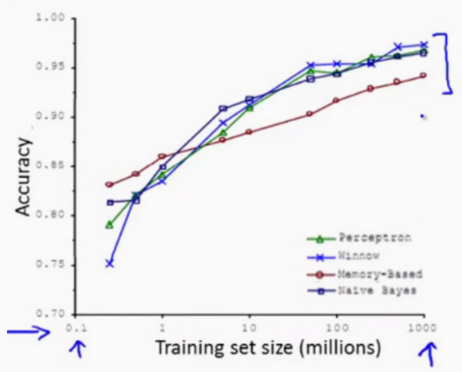
measure precision (p) and recall (r) on the **cross validation set** and choose the value of threshold which maximizes the F_1 score $= 2\frac{pr}{p+r}$

using large data sets

data for machine learning

determining how much data to train on

designing a high accuracy learning system



example classify between confusable words

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

→ "For breakfast I ate _two_ eggs."

the algorithms used to predict the word that belongs in the sentence above:

· perceptron (logistic regression)
· winnow
· memory-based
· naïve bayes

"it is not who has the best algorithm that wins, but who has the most data"

large data rationale

assuming feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately

example: "For breakfast I ate _two_ eggs."

counterexample" "Predict housing price from only the size (feet^2) as a feature."

useful test: given the input x, can a human expert confidently predict y?

using a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units).

→ low bias algorithms

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

using a very large training set (unlikely to overfit)

→ low variance algorithms

$J_{train}(\theta) \approx J_{test}(\theta)$

$J_{test}(\theta)$ will be small

large training sets can help improve a learning algorithm's performance, except: when the features x do not contain enough information to predict y, regardless if the chosen algorithm is simple (logistic regression) or complex (neural networks)

Machine Learning System Design

1. You are working on a spam classification system using regularized logistic regression. "Spam" is a positive class (y = 1) and "not spam" is the negative class (y = 0). You have trained your classifier and there are m = 1000 examples in the cross-validation set. The chart of predicted class vs. actual class is:

	Actual Class: 1	Actual Class: 0
Predicted Class: 1	85	890
Predicted Class: 0	15	10

For reference:

- Accuracy = (true positives + true negatives) / (total examples)
- Precision = (true positives) / (true positives + false positives)
- Recall = (true positives) / (true positives + false negatives)
- *F*1 score = (2 * precision * recall) / (precision + recall)

What is the classifier's *F*1 score (as a value from 0 to 1)? 0.16

2. Suppose a massive dataset is available for training a learning algorithm. Training on a lot of data is likely to give good performance when:

The features *x* contain sufficient information to predict *y* accurately. (For example, one way to verify this is if a human expert on the domain can confidently predict *y* when given only *x*).

We train a learning algorithm with a large number of parameters (that is able to learn/represent fairly complex functions).

3. Suppose you have trained a logistic regression classifier which is outputing $h\theta(x)$.

Currently, you predict 1 if $h\theta(x) \geq$threshold, and predict 0 if $h\theta(x) <$threshold, where currently the threshold is set to 0.5.

Suppose you **increase** the threshold to 0.9:

The classifier is likely to now have higher precision.

4. Suppose you are working on a spam classifier, where spam emails are positive examples (y=1) and non-spam emails are negative examples (y=0). You have a training set of emails in which 99% of the emails are non-spam and the other 1% is spam:

If you always predict non-spam (output y=0), the classifier will have accuracy of 99%.

If you always predict non-spam (output y=0), your classifier will have 99% accuracy on the training set, and it will likely perform similarly on the cross validation set.

5. Using a **very large** training set makes it unlikely a model to overfit the training data.

On skewed datasets (e.g., when there are more positive examples than negative examples), accuracy is not a good measure of performance and should instead use *F*1 score based on the precision and recall.