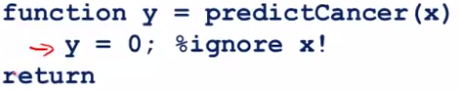
***Handling Skewed Data***

**I. ERROR METRICS FOR SKEWED CLASSES**

* So, it’s important to have **error metrics** = a single real number evaluation metric for a learning algorithm to tell how well it's doing.
* In the context of evaluation of error metrics, there’s 1 important case, where it's particularly tricky to come up with an appropriate error/evaluation metric for a learning algorithm = **skewed classes**.
* Consider the problem of cancer classification, where we have features of medical patients + we want to decide whether they have cancer or not (y = 1 if patient has cancer)
* We train regression classifier + on a test set we get 1% error.
* 99% correct diagnoses seems like a really impressive result, but say we find out only 0.5% of patients in training + test sets actually have cancer.
* In this case, the 1% error no longer looks so impressive.
* In particular, here's a piece of non-learning code that takes as input features x + it ignores it:



* It just sets y = 0 + always predicts “nobody has cancer”
* This algorithm would actually get 0.5% error, which is even better than the 1% error we got
* So when the ratio of positive to negative examples is very close to 1 of 2 extremes, we call it the **case of skewed classes** (ex: # of positive examples is much, much smaller than # of negative examples b/c y = 1 so rarely)
* In it, we have a lot more of examples from 1 class than from the other class.
* By just predicting y = 0 or y = 1 all the time, an algorithm can do pretty well.
* The problem with using classification error/accuracy as our evaluation metric is the following:
* Say you have 1 learning algorithm w/ 99.2% accuracy (a 0.8% error)
* You make a change to the algorithm + now are getting 99.5% accuracy (0.5% error)
* Is this an improvement to the algorithm or not?
* 1 of the nice things about having a single real number evaluation metric is it helps us quickly decide if we made a good change to the algorithm or not.
* *By going from 99.2% accuracy to 99.5% accuracy, did we do something useful or just replace code w/ something that just predicts y = 0?*
* So, w/ very skewed classes, it becomes much harder to use *just* classification accuracy, b/c we can get very high classification accuracies/very low errors + it's not always clear if doing so is really improving quality of the classifier
* Predicting y = 0 all the time doesn't seem like a particularly good classifier.
* Just predicting y = 0 more often can bring error down to maybe as low as 0.5%.
* When faced w/ such skewed classes, we’d want to come up w/ a different error/evaluation metric.
* 1 such evaluation metric is **precision recall.**
* Say we’re evaluating a binary classifier on a test set
* What our learning algorithm will do is predict some class value for each example in the test set
* If the actual class = 1 + predicted class = 1 🡪a **true positive**
* If our learning algorithm predicts negative/class = 0 + the actual class is also class 0, 🡪 **true negative**
* If our learning algorithm predicts class = one but actual class = zero, that's a **false positive**.
* A **false negative** = our algorithm predicted 0, but the actual class was 1.
* Now we compute 2 numbers:
* **Precision** = # of true positives / # we predicted as positive (TP + FP)
* Of all patients where we've predicted they *have* cancer, what fraction *actually has* cancer?
* High precision is good, b/c it means that of the group of patients we said had cancer, most actually do have cancer.
* **Recall** = # of true positives / # *actual* positive (TP + FN)
* Say all patients in in test or CV set actually have cancer, what fraction of them did we correctly detect as having cancer?
* Once again, having a high recall would be a good thing.
* Computing precision + recall will usually give us a better sense of how well our classifier is doing.
* In particular, w/ have a learning algorithm that predicts y = 0 all the time, this classifier will have a recall = 0 b/c there won't be any TP’s
* That's a quick way to recognize that a classifier that predicts y = 0 all the time isn't a very good one
* More generally, even for settings w/ very skewed classes, it's not possible for an algorithm to "cheat" + somehow get a very high precision + very high recall by doing something simple like predicting y = 0 or predicting y = 1 all the time.
* We're much surer a classifier of w/ high precision or high recall is a good classifier
* This gives us a more useful evaluation metric + a more direct way to understand whether our algorithm may be doing well.
* In the definition of precision and recall, usually we use the convention that y = 1 in the presence of the *rarer class*.
* If trying to detect rare conditions such as cancer, precision + recall are defined setting y = 1, rather than y = 0, b/c it’s the presence of that rare class that we're trying to detect.
* By using precision + recall, we find that what happens is, even w/ very skewed classes, it's not possible for an algorithm to "cheat" + predict y = 1 or predict y = 0 all the time
* If a classifier is getting high precision + high recall, we are confident the algorithm must be doing well, even w/ very skewed classes.

**II. TRADING OFF PRECISION AND RECALL**

* For many applications, we'll want to somehow control the trade-off between precision + recall
* Let's continue our cancer classification example, where y = 1 if patient has cancer + y = 0 otherwise.
* Say we've trained in logistic regression classifier which outputs probability between 0-1.
* As usual, predict y = 1 if h(x) >= 0.5 + predict y = 0 if h(x) < 0.5
* But, suppose we want to predict that a patient has cancer *only if we're very confident that they really do*.
* B/c if you tell a patient they have cancer, it's going to give them a huge shock
* It’s very bad news, + they may end up going through a pretty painful treatment process + so on.
* 1 way to do this would be to modify the algorithm + set this threshold at 0.5 to 0.7
* This is like saying “we'll tell someone they have cancer only if we think there's a >= 70% chance they do”
* We end up w/ a classifier w/ higher precision, b/c out of all patients we think have cancer, a higher fraction will actually turn out to have cancer b/c we made predictions only if we were very confident
* In contrast, this classifier will have *lower recall* b/c now we predict y = 1 on a *smaller number* of patients
* Now, can even take this further. Instead of setting the threshold at 0.7, we can set this at 0.9. Now we'll predict y=1 only if we are more than 90% certain that the patient has cancer. And so, a large fraction of those patients will turn out to have cancer. And so this would be a higher precision classifier will have lower recall because we want to correctly detect that those patients have cancer. Now consider a different example. Suppose we want to avoid missing too many actual cases of cancer, so we want to avoid false negatives. In particular, if a patient actually has cancer, but we fail to tell them that they have cancer then that can be really bad. Because if we tell a patient that they don't have cancer, then they're not going to go for treatment. And if it turns out that they have cancer, but we fail to tell them they have cancer, well, they may not get treated at all. And so that would be a really bad outcome because they die because we told them that they don't have cancer. They fail to get treated, but it turns out they actually have cancer. So, suppose that, when in doubt, we want to predict that y=1. So, when in doubt, we want to predict that they have cancer so that at least they look further into it, and these can get treated in case they do turn out to have cancer.
* 4:04
* In this case, rather than setting higher probability threshold, we might instead take this value and instead set it to a lower value. So maybe 0.3 like so, right? And by doing so, we're saying that, you know what, if we think there's more than a 30% chance that they have cancer we better be more conservative and tell them that they may have cancer so that they can seek treatment if necessary.
* 4:31
* And in this case what we would have is going to be a higher recall classifier, because we're going to be correctly flagging a higher fraction of all of the patients that actually do have cancer. But we're going to end up with lower precision because a higher fraction of the patients that we said have cancer, a high fraction of them will turn out not to have cancer after all.
* 5:00
* And by the way, just as a sider, when I talk about this to other students, I've been told before, it's pretty amazing, some of my students say, is how I can tell the story both ways. Why we might want to have higher precision or higher recall and the story actually seems to work both ways. But I hope the details of the algorithm is true and the more general principle is depending on where you want, whether you want higher precision- lower recall, or higher recall- lower precision. You can end up predicting y=1 when h(x) is greater than some threshold. And so in general, for most classifiers there is going to be a trade off between precision and recall, and as you vary the value of this threshold that we join here, you can actually plot out some curve that trades off precision and recall. Where a value up here, this would correspond to a very high value of the threshold, maybe threshold equals 0.99. So that's saying, predict y=1 only if we're more than 99% confident, at least 99% probability this one. So that would be a high precision, relatively low recall. Where as the point down here, will correspond to a value of the threshold that's much lower, maybe equal 0.01, meaning, when in doubt at all, predict y=1, and if you do that, you end up with a much lower precision, higher recall classifier. And as you vary the threshold, if you want you can actually trace of a curve for your classifier to see the range of different values you can get for precision recall. And by the way, the precision-recall curve can look like many different shapes. Sometimes it will look like this, sometimes it will look like that. Now there are many different possible shapes for the precision-recall curve, depending on the details of the classifier. So, this raises another interesting question which is, is there a way to choose this threshold automatically? Or more generally, if we have a few different algorithms or a few different ideas for algorithms, how do we compare different precision recall numbers? Concretely, suppose we have three different learning algorithms. So actually, maybe these are three different learning algorithms, maybe these are the same algorithm but just with different values for the threshold. How do we decide which of these algorithms is best? One of the things we talked about earlier is the importance of a single real number evaluation metric. And that is the idea of having a number that just tells you how well is your classifier doing. But by switching to the precision recall metric we've actually lost that. We now have two real numbers. And so we often, we end up face the situations like if we trying to compare Algorithm 1 and Algorithm 2, we end up asking ourselves, is the precision of 0.5 and a recall of 0.4, was that better or worse than a precision of 0.7 and recall of 0.1? And, if every time you try out a new algorithm you end up having to sit around and think, well, maybe 0.5/0.4 is better than 0.7/0.1, or maybe not, I don't know. If you end up having to sit around and think and make these decisions, that really slows down your decision making process for what changes are useful to incorporate into your algorithm.
* 8:23
* Whereas in contrast, if we have a single real number evaluation metric like a number that just tells us is algorithm 1 or is algorithm 2 better, then that helps us to much more quickly decide which algorithm to go with. It helps us as well to much more quickly evaluate different changes that we may be contemplating for an algorithm. So how can we get a single real number evaluation metric?
* 8:47
* One natural thing that you might try is to look at the average precision and recall. So, using P and R to denote precision and recall, what you could do is just compute the average and look at what classifier has the highest average value.
* 9:00
* But this turns out not to be such a good solution, because similar to the example we had earlier it turns out that if we have a classifier that predicts y=1 all the time, then if you do that you can get a very high recall, but you end up with a very low value of precision. Conversely, if you have a classifier that predicts y equals zero, almost all the time, that is that it predicts y=1 very sparingly, this corresponds to setting a very high threshold using the notation of the previous y. Then you can actually end up with a very high precision with a very low recall. So, the two extremes of either a very high threshold or a very low threshold, neither of that will give a particularly good classifier. And the way we recognize that is by seeing that we end up with a very low precision or a very low recall. And if you just take the average of (P+R)/2 from this example, the average is actually highest for Algorithm 3, even though you can get that sort of performance by predicting y=1 all the time and that's just not a very good classifier, right? You predict y=1 all the time, just normal useful classifier, but all it does is prints out y=1. And so Algorithm 1 or Algorithm 2 would be more useful than Algorithm 3. But in this example, Algorithm 3 has a higher average value of precision recall than Algorithms 1 and 2. So we usually think of this average of precision and recall as not a particularly good way to evaluate our learning algorithm.
* 10:38
* In contrast, there's a different way for combining precision and recall. This is called the F Score and it uses that formula. And so in this example, here are the F Scores. And so we would tell from these F Scores, it looks like Algorithm 1 has the highest F Score, Algorithm 2 has the second highest, and Algorithm 3 has the lowest. And so, if we go by the F Score we would pick probably Algorithm 1 over the others.
* 11:04
* The F Score, which is also called the F1 Score, is usually written F1 Score that I have here, but often people will just say F Score, either term is used. Is a little bit like taking the average of precision and recall, but it gives the lower value of precision and recall, whichever it is, it gives it a higher weight. And so, you see in the numerator here that the F Score takes a product of precision and recall. And so if either precision is 0 or recall is equal to 0, the F Score will be equal to 0. So in that sense, it kind of combines precision and recall, but for the F Score to be large, both precision and recall have to be pretty large. I should say that there are many different possible formulas for combing precision and recall. This F Score formula is really maybe a, just one out of a much larger number of possibilities, but historically or traditionally this is what people in Machine Learning seem to use. And the term F Score, it doesn't really mean anything, so don't worry about why it's called F Score or F1 Score.
* 12:09
* But this usually gives you the effect that you want because if either a precision is zero or recall is zero, this gives you a very low F Score, and so to have a high F Score, you kind of need a precision or recall to be one. And concretely, if P=0 or R=0, then this gives you that the F Score = 0. Whereas a perfect F Score, so if precision equals one and recall equals 1, that will give you an F Score,
* 12:43
* that's equal to 1 times 1 over 2 times 2, so the F Score will be equal to 1, if you have perfect precision and perfect recall. And intermediate values between 0 and 1, this usually gives a reasonable rank ordering of different classifiers.
* 13:00
* So in this video, we talked about the notion of trading off between precision and recall, and how we can vary the threshold that we use to decide whether to predict y=1 or y=0. So it's the threshold that says, do we need to be at least 70% confident or 90% confident, or whatever before we predict y=1. And by varying the threshold, you can control a trade off between precision and recall. We also talked about the F Score, which takes precision and recall, and again, gives you a single real number evaluation metric. And of course, if your goal is to automatically set that threshold to decide what's really y=1 and y=0, one pretty reasonable way to do that would also be to try a range of different values of thresholds. So you try a range of values of thresholds and evaluate these different thresholds on, say, your cross-validation set and then to pick whatever value of threshold gives you the highest F Score on your crossvalidation [INAUDIBLE]. And that be a pretty reasonable way to automatically choose the threshold for your classifier as well.