T-tests for difference from a [redacted] value

It's more common to compare populations, but conceivably you could want to decide whether a value is different from a target value, like 5. This works exactly the same, except one of the numbers in the "difference" has no variance and no additional degrees of freedom. Thus the standard error used is just $\frac{\sigma}{\sqrt{N}}$ instead of $\sqrt{\frac{2\sigma^2}{N}} = \sqrt{\frac{\sigma^2}{N} + \frac{\sigma^2}{N}}$. (Again, N is the number of individuals in one group.)

Chi-square tests

T-tests are good for finding significant effects for continuous values: finding that one mean is bigger than another, for example. Chi-square tests are good for finding significance in boolean or discrete values. For a Boolean example, we might wonder whether Mac users are more likely than Windows users to install our product. Each sample is just an answer to two Boolean questions: Mac? And, Software Installed? Our datapoints aren't continuous, and treating them as 0 or 1 won't result in a normal distribution, so we need a different approach to decide whether a claim like "Mac users install our software less" has statistical significance.

The first step in a chi-square test is to make a table of outcomes. Let's say we have 50 Mac users and 150 windows users. 1/5 the Mac users installed the software, but a third of the Windows users installed it. Is this difference significant, or is it fairly likely to happen even if (null hypothesis) the two populations are the same?

A 2×2 Table:

Software? \ Mac?	Y	N
Y	10	50
N	40	100
	↑ 50 Mac	↑ 150 Windows users

"Contingency Table"

The next step is to calculate, from these counts, the probability of each Boolean ignoring the other distinction, by counting the total outcomes in each column or row. $Pr(Mac) = 50/200 = 1/4$. $Pr(Software) = \frac{60}{200} = \frac{3}{10}$.

(50 ↑ first column; 60 ← first row)

Next, we calculate what the table would look like if every cell held the expected value and the two variables were independent. These are the expected contents of the table under the null hypothesis that the variables are independent. Each cell is equal to $p_i q_i N$, where p is the probability of the first value, q is the probability of the second, and N is the total number of samples.

"Expected" table

Software? \ Mac?	Y (1/4)	N (3/4)	
Y	15	45	(3/10)
N	35	105	(7/10)

$3/4 \cdot 3/10 \cdot 200 = 45$

What remains is to check whether the deviation from these expected values is very unlikely to happen from chance alone. Now, to finish the example, I'll take some steps that won't be well-justified at first. We want to see how far off we are from the expectation table, in a way that can be compared to threshold values (similar to the t distribution). Our deviation from expectation is computed as $\chi^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$ where O_{ij} is the cell in the observed table (i,j) and E_{ij} is the cell in the expected table (i,j).

For our example, this would be

$$\frac{(10-15)^2}{15} + \frac{(50-45)^2}{45} + \frac{(40-35)^2}{35} + \frac{(100-105)^2}{105}$$

which comes out to 3.175. The table/function we use to determine whether this is significant is the chi-square distribution, the distribution of a sum of squared normals. It has one additional input, the degrees of freedom, which is 1 for a 2×2 test because

the deviation in one cell determines the deviations in the rest of its row and column, which here determines the whole thing. The "critical value" that would make these results <5% likely under the null hypothesis is 3.84. We got less than that, so we shouldn't bare any important decisions on thinking Mac users use the product less. This is a difference that is well within the realm of possibility under the null hypothesis.

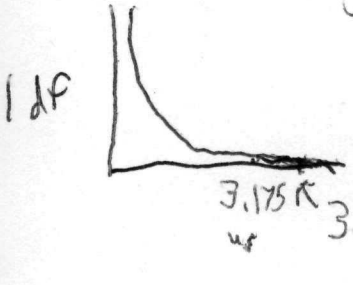
Okay, we've worked through an example. What is the χ^2 (chi) distribution? It's the distribution you get from a sum of normal distributions, squared: $\mu=0$

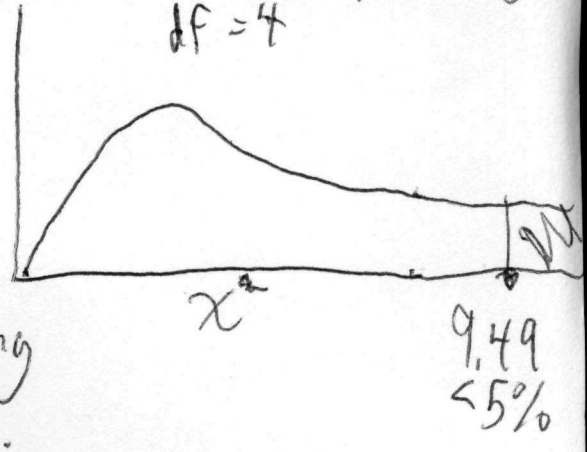
$Z_1^2 + Z_2^2 + \cdots + Z_k^2$ where Z's are normal and $\sigma=1$

The number of normal distributions involved is the degrees of freedom. If you have enough distributions involved, χ^2 starts to look normal itself, but for our contingency tables, it looks more like

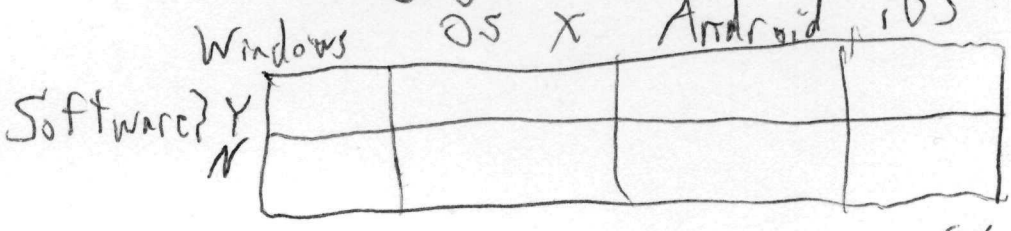
By the central limit theorem, differences in one cell from expectation will tend toward being normally distributed. But in a table with r rows and c columns, the final row is determined by N and the previous entries, and the same for the final column and previous columns. So the degrees of freedom are equal to $(r-1)(c-1)$, the nonfinal cells.

When we compare to a critical value, we are asking how unlikely such results are under the null hypothesis, meaning, the area under the chi-square to the right of our value. Since the distribution changes with the degrees of freedom, the critical value necessary changes as well.





Why would you have more degrees of freedom? The variables you're considering may have more than one possible value.



4 columns, 2 rows here means $(4-1)(2-1) = 3$ df. The process is the same summing $\sum \frac{(O-E)^2}{E}$ across all cells, and we compare to a 3df critical value. Notice that the results of a big table test may be more vague than you'd like. A significant result would tell you that there is or is not interaction between these variables, but it won't tell you which OS is responsible. That might be clear from inspection - or it might not.

Incidentally, the formula for the Chi-square is a bit complex, but thankfully, not very relevant. You'll basically always interact with it through tables, software, or a library.

Chi-square for goodness-of-fit ("one-way tables")

There's another major use case for the chi-square is deciding whether an observed distribution fits your theoretical expectations. For example, suppose you roll a 6-sided die 10 times and roll a 6 half the time. Is the die really fair? Or, suppose we draw samples from a population that we thought would be normally distributed, and it looks skewed instead. Is it really normal, with the skew from random fluctuations? Or is a different distribution a better fit?

This is going to work exactly like the contingency tables, only now, there's effectively just one row.

Die roll	1	2	3	4	5	6
Count	1	0	2	1	1	5

Population measure	$< -2\sigma$	-2σ to $-\sigma$	$-\sigma$ to 0	0 to $+\sigma$	$+\sigma$ to $+2\sigma$	$> +2\sigma$
	2	10	21	25	4	1

On the left, our expected table is $\frac{10}{6}$ in every cell. (Null hypothesis of uniform dist.) On the right, we'd calculate expected counts from the CDF. (Null hypothesis of normality.)

The other tricky sticking point here is that we lose a degree of freedom for each parameter we had to calculate on the way to making the observed bins. The table on the right loses 2 - one for μ, one for σ - and still loses one for the final column. 3 df.

Multiple columns example

05

Happy	Mac	Windows	Linux
Y	10	8	2
N	5	8	0

Probs: $P(\text{happy}) = 20/33$

$P(\text{Mac}) = 15/33$

$P(\text{Windows}) = 18/33$

$P(\text{Linux}) = 2/33$

Expected

9.1	9.7	1.2
4.5	6.3	0.6

Chi-square vals

0.09	0.30	0.53
0.06	0.46	0.6

Sum 2.04

2 df thresh 5.99 – no significance

Die roll example

1	2	3	4	5	6
1	0	2	1	1	5

Expected $\frac{10}{6}$

$$S_{um} \quad \left(\frac{4}{6}\right)^2 / \frac{10}{6} + \left(\frac{10}{6}\right)^2 / \frac{10}{6} + \left(\frac{2}{6}\right)^2 / \frac{10}{6} + \left(\frac{4}{6}\right)^2 / \frac{10}{6} + \left(\frac{4}{6}\right)^2 / \frac{10}{6} + \left(\frac{20}{6}\right)^2 / \frac{10}{6}$$

$= 9.2$

5 df threshold 11.07

Not significant

You might think this test would get used all the time, because you should know whether normality holds before doing a t-test. But, many fields (at least in CS) just assume that normality is a good enough assumption unless there's an obvious reason to doubt it. Failing to disprove a null hypothesis is also often not considered interesting, so groups that do the test may not report it (one could always claim there weren't enough samples). Still, one could argue that researchers should perform this test before just assuming their data follows a particular distribution — especially since deviations may be interpreted as significant results.

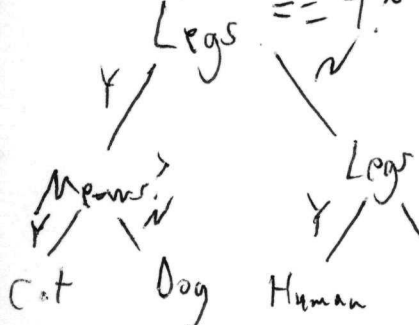
-end wk 11-

Decision Tree Learning & Entropy

(wk 12)

Last time, we explored chi-square tests as a means of deciding whether a dependence between two variables was significant. We can also use tests for significance to decide whether a machine learning model has introduced a feature that is not actually helpful, and remove that feature from the model to produce better performance. That is, chi-square tests can perform a similar function to regularization, removing the portions of the model that correspond to nothing but noise. The method we'll examine for this goes by a few names, including C4.5, CART, or simply, decision trees.

Decision trees are a useful ML method if you want to be able to examine the sequence of decisions that led to a particular outcome. A trained decision tree model is a tree where each node contains a yes or no decision — "Is the number of legs = 4?" — and the children are either further decisions — "Does it meow?" — or outcomes — "It's a cat." Given an example of a set of features, the trained model follows yes or no branches until reaching a leaf, which gives the classification.



(Decision trees can theoretically have more than two children, but in practice training such trees creates problems — features with more than two outcomes can have an unfair advantage.)

We can train such trees from labeled examples, engaging in supervised learning. Decision trees are more limited in what they can learn than neural networks, but have the advantage of being inspectable - so if a tree is unfairly using an attribute like race or sex, it's clear that it's doing so.

The fundamental way that a decision tree is created is by using information gain - it want's decisions that reduce the chaos of the pile of examples by separating it into two piles that are less chaotic. If, for example, it took a pile that was half DOG and half CAT, and it managed to find a decision such that the YES's were all DOGs and the NO's were all CATs, that would be the best decision we could possibly choose. In practice, we're unlikely to find a decision that is so clear-cut, but we can try to find a decision that makes the split as well as possible.

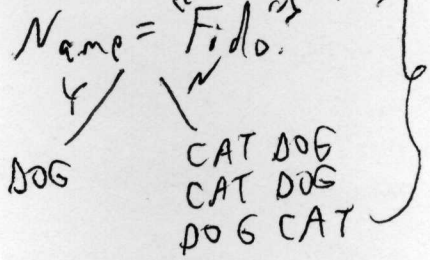
What is information? It's fundamentally a measure of surprise, and is defined as $-\log_2 p$, where p is the probability of the event. So if $p = 1/2$, $I(p) = 1$, and if $p = \frac{1}{1024}$, $I(p) = 10$. More unlikely means more information. The reason for the $\log_2$ is that the

information is the number of bits we would need in an optimal encoding to record the event. Thus a coin flip requires 1 bit while a $\frac{1}{1024}$ chance event requires 10 bits. (1024 outcomes requires 10 bits - and if an outcome happens with probability $\frac{1}{512}$, the optimal thing to do is make the code one bit shorter, 9 bits.)

Decision tree learning wants to find decisions that maximally reduce surprise - information - after we learn the outcome of the decision. The entropy is the expected value of the information, $E[-\log_2 p] = \sum_i -p_i \log_2 p_i$. For a coin flip, this is $\frac{1}{2}(1) + \frac{1}{2}(1) = 1$. If we are certain of an outcome, it's 0. The information gain is the difference between the entropy before we know a fact, and the entropy after we know a fact. So if we have a batch of examples that is half dogs and half cats, and then asking "Does it meow?" separates the examples into a "YES" pile that is all cats, and a "NO" pile that is all dogs, the information gain is 1, because the entropy of the classification went from 1 to 0. (If the entropies of the two outcome piles aren't the same, we find the expected entropy,

$P_r(\text{answer is YES}) \cdot \text{Entropy}(\text{YES examples}) + P_r(\text{answer is NO}) \cdot \text{Entropy}(\text{NO examples})$

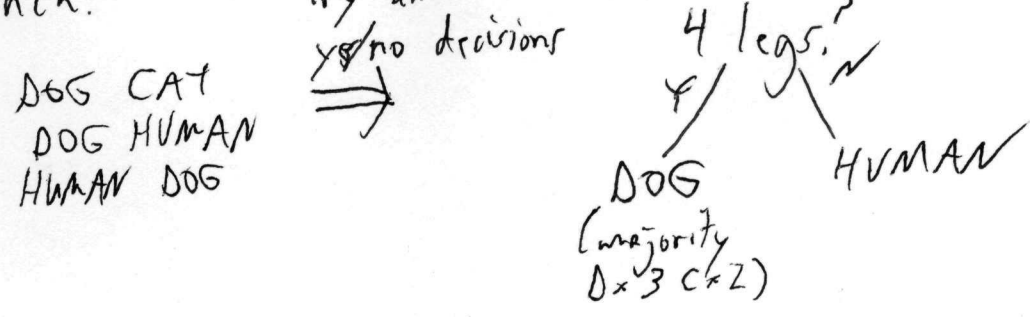
This means of selection does a couple things. One, it tends to choose decisions that result in piles of examples with more uniform classifications. And two, but using the expected value of the resulting entropy, it avoids selecting decisions that only help classify a handful of examples. A decision that just shaves off one example isn't valued as much as one that gives a boost to classifying most examples.



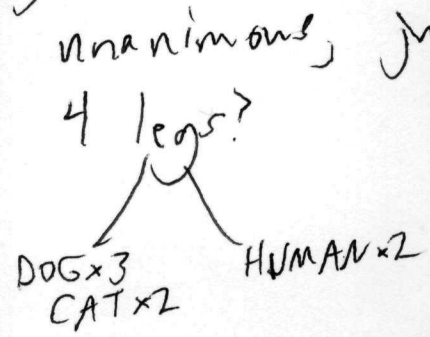
Not that great a feature.

We're almost ready to define the decision tree learning algorithm. If we wanted to make what's called a "decision stump" with just one decision, we could do the following:

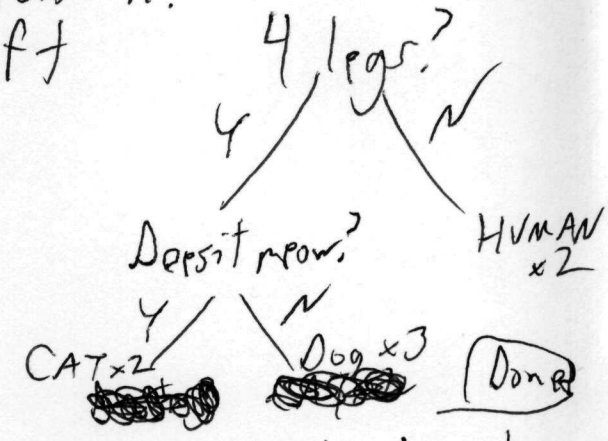
- Calculate the info gain for every possible decision.
- Pick the best decision. (Most info gain)
- For each branch, give the majority-rule decision based on the training examples fitting that branch.



The full version of the algorithm is like that but recursive. Once we choose the best decision, we hand off the YES examples to one recursive call, and the NO examples to a different recursive call. Each recursive call makes a new decision subtree using just those examples. Once a branch's examples are unanimous, just return that decision.

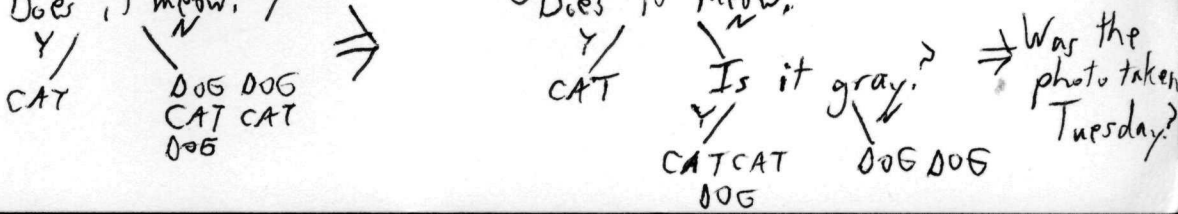


next best decision on left



In the case of non-Boolean features, we just break that up into a series of Y/N decisions. So if # legs = 2, 4, 8 I can either break that up into legs ≤ 2? legs ≤ 4? legs ≤ 8? or legs = 2? legs = 4? legs = 8?

A problem arises if we try to use the algorithm as described so far: noise and overfitting. If the algorithm is forced to achieve unanimous votes for each classification it will start to seize on features that are actually meaningless to do this.



To avoid overfitting, there is typically some pruning that takes place that collapses the leaves of the tree when the decisions seem to be meaningless. One key tool we can use to prune decision trees is the chi-square test. If there isn't really a reliable connection between a feature and a classification, then the association won't be significant in a 2×2 chi-square test.

Gray? \ Cat?	Y	N
Y	10	31
N	30	29

For the frequencies at left, we would expect the chi-square to return non-significance but the baseline decision tree algorithm would still use "gray" as a feature if that were the best feature available. Pruning back the decision if it's not significant is one way to avoid overfitting.

In short, decision tree learning works as follows:

```
Decision Tree (Examples):
    If Examples have all the same classification, create
        leaf with that decision
    For each possible decision,
        calculate the expected information gain, remembering
        if it's best
    Split examples into YES pile and NO pile according to
        best decision
    Left branch = Decision Tree (NO examples)
    Right branch = Decision Tree (YES examples)
    If both left and right are leaves, try chi-square to prune
```

While we could try pruning as we go, rather than building out children first, "early stopping" like that could lead to pruning too soon. If only the AND of two features is meaningful, pruning before building out the children could result in never discovering that rule.

Decision trees aren't quite as powerful as neural networks in the hypotheses they can entertain, because they can't make functions of features that are more complex than ANDs and ORs. (No squares or powers for example.) But they do have the advantage of inspectability: it's easy for even a layperson to see exactly why a decision was made.

Covariance & Correlation

Our statistics so far has dealt with tests of single variables in controlled environments. This is good for many kinds of experiments, but doesn't really fully capture what's going on in many real situations, where we have multiple interacting variables. For example, we'd probably be remiss if we didn't explain what a correlation was, or how to find the strength of one.

A correlation is partly derived from the covariance, a generalization of variance to the multivariable case. If we have observations of pairs (X, Y),