Machine Learning Cheatsheet

Janet Matsen's Machine Learning (ML) notes from CSE 446, Winter 2016. http://courses.cs.washington.edu/courses/cse446/16wi/
Used LaTeX template from an existing Statistics cheat sheet: https://github.com/wzchen/probability_cheatsheet, by William Chen (http://wzchen.com) and Joe Blitzstein.

Last Updated January 14, 2016

Essential ML ideas

- Never ever touch the test set
- You know you are overfitting when there is a big test between train and test results. E.g. metric of percent wrong.
- Need to be comfortable taking a hit on fitting accuracy if you can get a benefit on the result.

Math/Stat Review

Random Variable X belongs to set Ω

Conditional Probability *is* Probability $P(A|B)$ is a probability function for any fixed B. Any theorem that holds for probability also holds for conditional probability. $P(A|B) = P(A \cap B)/P(B)$

Bayes' Rule - Bayes' Rule unites marginal, joint, and conditional probabilities. We use this as the definition of conditional probability.

$$P(\mathbf{A}|\mathbf{B}) = \frac{P(\mathbf{A} \cap \mathbf{B})}{P(\mathbf{B})} = \frac{P(\mathbf{B}|\mathbf{A})P(\mathbf{A})}{P(\mathbf{B})}$$

$$P(A = a \mid B) = \frac{P(A = a)P(B \mid A = a)}{\sum_{a'} P(A = a)P(B \mid A = a)}$$

Law of Total Probability : $\sum_x P(X = x) = 1$

Product Rule : $P(A, B) = P(A \mid B) \cdot P(B)$

Sum Rule : $P(A) = \sum_{x \in \Omega} P(A, B = b)$

Vocab:

- **likelihood function** $L(\theta|O)$ is called as the likelihood function. θ = unknown parameters, O is the observed outcomes. The likelihood function is conditioned on the observed O and that it is a function of the unknown parameters θ. Not a probability density function.
- **"likelihood" vs "probability"**: if discrete, $L(\theta|O) = P(O|\theta)$. If continuous, $P(O|\theta) = 0$ so instead we estimate θ given O by maximizing $L(\theta|O) = f(O|\theta)$ where f is the pdf associated with the outcomes O.

Law of Total Probability (LOTP)

Let $B_1, B_2, B_3, ...B_n$ be a *partition* of the sample space (i.e., they are disjoint and their union is the entire sample space).

$P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \cdots + P(A|B_n)P(B_n)$

$P(A) = P(A \cap B_1) + P(A \cap B_2) + \cdots + P(A \cap B_n)$

For **LOTP with extra conditioning**, just add in another event C!

$P(A|C) = P(A|B_1, C)P(B_1|C) + \cdots + P(A|B_n, C)P(B_n|C)$

$P(A|C) = P(A \cap B_1|C) + P(A \cap B_2|C) + \cdots + P(A \cap B_n|C)$

Special case of LOTP with B and B^c as partition:

$$P(A) = P(A|B)P(B) + P(A|B^c)P(B^c)$$

$$P(A) = P(A \cap B) + P(A \cap B^c)$$

Bayes' Rule

Bayes' Rule, and with extra conditioning (just add in C!)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A|B, C) = \frac{P(B|A, C)P(A|C)}{P(B|C)}$$

We can also write

$$P(A|B, C) = \frac{P(A, B, C)}{P(B, C)} = \frac{P(B, C|A)P(A)}{P(B, C)}$$

Odds Form of Bayes' Rule

$$\frac{P(A|B)}{P(A^c|B)} = \frac{P(B|A)}{P(B|A^c)} \frac{P(A)}{P(A^c)}$$

The *posterior odds* of A are the *likelihood ratio* times the *prior odds*.

Practice: What is $P(disease \mid +test)$ if P(disease) = 0.01, P(+ | disease) = 0.99, P(+ | no disease) = 0.01?

Expectation

f(X) probability distribution function of X

X ~ P : X is distributed according to P.

Expected value of f under P : $E_P[f(x)] = \sum_x p(x)f(x)$

E.g. unbiased coin. x = 1, 2, 3, 4, 5, 6. p(X=x) = 1/6 for all x.
$E(X) = \sum_x p(x) \cdot x = (1/6) \cdot [1 + 2 + 3 + 4 + 5 + 6] = 3.5$

Entropy

$X \sim P$, $x \in \Omega$

First define **Surprise**: $S(x) = -\log_2 p(x)$
$S(X = \text{heads}) = -\log_2(1/2) = 1$.

Axiom 1 : S(1) = 0. (If an event with probability 1 occurs, it is not surprising at all.)

Axiom 2 : S(q) > S(p) if q < p. (When more unlikely outcomes occur, it is more surprising.)

Axiom 3 : S(p) is a continuous function of p. (If an outcomes probability changes by a tiny amount, the corresponding surprise should not change by a big amount.)

Axiom 4 : S(pq) = S(p) + S(q). (Surprise is additive for independent outcomes.)

Surprise of 7 = pretty surprised. Probability of $1/2^7$ of happening (Shannon) **Entropy**:

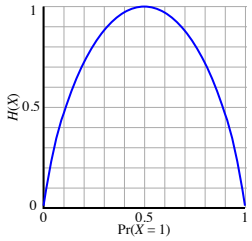
$$H[X] = -\sum_x p(x) \cdot \log_2 p(x)$$
$$= -\sum_x p(x)S(x)$$
$$= E[S(x)]$$

The entropy is the expectation of the surprise. Throw out x for $p(x) = 0$ because log(0) is ∞.

Entropy of an unbiased coin flip:
X is a coin flip. $P(X = \text{heads}) = 1/2$, $P(X = \text{tails}) = 1/2$
Note: $\log_2(1/2) = -1$, $-\log_2(1/2) = \log_2(2) = 1$
$H[X] = -[1/2 \log_2(1/2) + 1/2 \log_2(1/2)] = 1$

Entropy of a coin that always flips to heads:
$P(X = \text{heads}) = 1$, $P(X = \text{tails}) = 0$
Note: $\log_x(0) = 0$
$H[X] = -[1 \log_2(1) + 0] = 0$
No surprise: you are sure what you are going to get.

Binary entropy plot.



Canonical example:

X	Y
0	1
1	0
1	1

If you want to estimate entropy of X, you can use P(X=0).

$$H[X] = -[\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3}]$$
$$= \frac{1}{3} \log_2 3 + \frac{2}{3} \log_2 3 - \frac{2}{3} \log_2 2$$
$$= \log_2 3 - \frac{2}{3} \approx 0.91$$

This time H[X] = H[Y] because of symmetry.

Conditional Entropy

If you don't know x: (this is kind of an average).
$H[Y \mid X = x] = -\sum_y P(Y = y \mid X = x) \cdot \log_2 P(y \mid X = x)$

$H[Y \mid X = x] = E[S(Y \mid X = x)]$
Note that we are summing over y because we are specifying x.

For a particular value of X:
$H[Y \mid X] = \sum_x p(x)H[Y \mid X = x]$

Back to table above:

$H[Y \mid X = 0] =?$
look only at X=0 in table.
$= -[0 + 1 \log_2]$

Now that you know X=0, entropy goes to 0.

$H[Y \mid X = 1] = 1$: You know *less* if you know X=1.

Now use $H[Y \mid X] = \frac{1}{3}(0) + \frac{2}{3}(1) = 2/3$
Given X, you know more. Average our the more certain case and the less certain case.

Note: $H[Y \mid X] \leq H[Y]$: knowing something can't make you know less.

Entropy and Information Gain

Information Gain - $IG(X) = H(Y) - H(Y \mid X)$
Y is the node on top. X are the nodes below. He might have used lower case.
Conditional Information Gain: $H(y|x) = -sumP(y|x)log_2$

- Low uncertainty $\leftrightarrow$ Low entropy.
- Lowering entropy $\leftrightarrow$ More information gain.

The discrete distribution with maximum entropy is the uniform distribution. For K values of X, $H(X) = \log_2 K$
Conversely, the distribution with minimum entropy (which is zero) is any delta-function that puts all its mass on one state. Such a distribution has no uncertainty.
Binary Entropy Function: $p(X = 1) = \theta$ and $p(X = 0) = 1 - \theta$

$$\begin{aligned} H(X) &= -[p(X = 1) \log_2 p(X = 1) + p(X = 0) \log_2 p(X = 0)] \\ &= -[\theta \log_2 \theta + (1 - \theta) \log_2 (1 - \theta)] \end{aligned}$$

Bits

If you use log base 2 for entropy, the resulting units are called bits (short for binary digits).
How many things can you encode in 15 bits? 2^{25}.

Decison Trees

Summary:

- One of the most popular ML tools. Easy to understand, implement, and use. Computationally cheap (to solve heurisRcally).
- Uses informaton gain to select attributes (ID3, C4.5,)
- Presented for classification, but can be used for regression and density estimation too
- Decision trees will overfit!!!
- Must use tricks to find simple trees, e.g., (a) Fixed depth/Early stopping, (b) Pruning, (c) Hypothesis testing
- Tree-based methods partition the feature space into a set of rectangles.
- Interpretability is a key advantage of the recursive binary tree.

Pros:

- easy to explain to people
- more closely mirror human decision-making than do the regression and classification approaches
- can be displayed graphically, and are easily interpreted even by a non-expert
- can easily handle qualitative predictors without the need to create dummy variables

Cons:

- trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches
- can be very non-robust. A small change in the data can cause a large change in the final estimated tree

Vocab:

- **classification tree** - used to predict a qualitative response rather than a quantitative one
- **regression tree** - predicts a quantitative (continuous) variable.
- **depth of tree** -the maximum number of queries that can happen before a leaf is reached and a result obtained
- **split** -
- **node** - synonymous with split. A place where you split the data.
- **node purity** -
- **univariate split** - A split is called univariate if it uses only a single variable, otherwise multivariate.
- **multivariate decision tree** - can split on things like A + B or Petal.Width / Petal.Length ¡ 1. If the multivariate split is a conjunction of univariate splits (e.g. A and B), you probably want to put that in the tree structure instead.
- **univariate decision tree** - a tree with all univariate splits/nodes. E.g. only split on one attribute at a time.
- **binary decision tree**
- **argmax** - the input that leads to the maximum output
- **greedy** - at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.
- **threshold splits** -

Protocol:

1. Start from empty decision tree
2. Split on next best attribute (feature).
   - Use something like information gain to select next attribute. $\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y|X_i)$
3. Recurse

When do we stop decision trees?

- Dont split a node if all matching records have the same output value
- Only split if all of your bins will have data in them. His words: "none of the attributes can create multiple nonempty children." He also said "no attributes can distinguish", and showed that for the remaining training data, each category only had data for one label. And third, "If all records have exactly the same set of input attributes then dont recurse"

He noted that you might not want to stop splitting just because all of your information nodes have zero information gain. You would miss out on things like XOR.
Decision trees will overfit. If your labels have no noise, the training set error is always zero. To prevent overfitting, we must introduce some bias towards simpler trees. Methods available:

- Many strategies for picking simpler trees
- Fixed depth
- Fixed number of leaves
- Or something smarter

One definition of overfitting: If your data is generated from a distribution $D(X, Y)$ and you have a hypothesis space H:
Define errors for hypothesis $h \in H$: training error = $error_{train}(h)$, Data (true) error = $error_D(h)$. The hypothesis h overfits the training data if there exists an h' such that $error_{train}(h) < error_{train}(h')$ and $error_D(h) > error_{train}(D)$. In plain english, if there is an alternative hypothesis that gives you more error on the training data but less error in the test data then you have overfit your data.

How to Build Small Trees
Two reasonable approaches:

- Optimize on the held-out (development) set. If growing the tree larger hurts performance, then stop growing. But this requires a larger amount of data
- Use statistical significance testing. Test if the improvement for any split it likely due to noise. If so, dont do the split. Chi Square test w/ MaxPchance = something like 0.05.

Pruning Trees
Start at the bottom, not the top. The top is most likely to have your best splits. In this way, you only cut high branches if all the branches below were cut.

Classification vs. Regression Trees
In class we mostly discussed nodes with categorical attributes. You can have continuous attributes (see HW1). You can also have either discrete or continuous output. When output is discrete, you can chose your splits based on entropy. If it is continuous, you need to do something more like least squares. For regression trees, see pg 306 from ISL or pg 307 of ESLII.

Let's do this thing.