Admistrivia


Probabilistic CFG

- Context free grammar epitomizes independence of surrounding material
- Likewise a probabilistic CFG builds in independence
- But it isn’t as natural—since it isn’t making pass / fail statements.
- So PCFG are dogmatically independent
• Regular CFG say the correct model is “absolutely continuous” we respect to the independent distribution

Example

• Figure 1: S->BC (with B->aa C->aa). S->C (with C->aaa). S->B (with B->a).

• Equally likely

• Estimate on arbitrarily large amounts of data—and we don’t recover the truth

• We estimate the closest independent model

Solutions: Models vs Fits

• First approach: better probabilistic model
  – Conditional random fields
  – build in dependencies
  – Enter the land of MLE and “bayesian” conditional probability statements

• Second approach: Fit rather than model
  – People call regression a model. But this is stupid.
– Models are strong statements about what is and isn’t the case
– Models should be falsifiable. The CAN be wrong
– regression is never “wrong.” It just needs more variables.

Predicting the parse

• Treat the parse as “Y” and the sentence as “X”
• Run a “regression”
• Crazy multi-dimensional!

Revisit PCFG

• We can write down the likelihood as:

\[
P(T|\alpha, \theta) = \prod_{r \in R} p(r|\theta)^{c(T, \alpha, r)}
\]

where \(T\) is the parse tree, \(\theta\) are our parameters, \(R\) is the set of production rules, \(c()\) is the count of the number of such rules in our tree, \(\alpha\) is the sentence to be parsed.

• This is an exponential family. Not one that Larry might recognize though.

\[
\log(P(T|\alpha, \theta)) = \sum_r \phi(T, \alpha) \theta
\]

where we have now identifies \(\theta_r = \log p(r|\theta)\).
Positive and negative instances

- We can now think of the following regression problem.
  - Look at the top 2 suggests made by say a PCFG
  - Only consider sentence for which the true parse is one of these two
  - Try to predict which one is the correct parse
  - We can think of $\phi(T_1, \alpha) - \phi(T_2, \alpha)$ as regression variables.
  - The slope then is an estimate of $\theta$.
  - Now we are doing a regular logistic regression.

- New “model” created by adding new features

Using such a model

- Factored code
- Search is now sold separately from fitting
- We can’t estimate any trees – only good trees.
- Advantage is we can now add new features without having to modify our search algorithm
- Disadvantage: we don’t really have a probabilistic model over all trees–just trees that are contenders for being the right parse
It could all go bad

- Problem: really bad trees might beat some of the good trees
- we haven’t ever trained on them
- so the question we learn is “how to improve a parse” not what the right parse is

More from Collins’ paper

- He also discusses
  - max margin
  - perceptron algorithm
  - support vector machines
  - boosting