Learning Structure from Annotated Data

In chapter 2, making predictions over given structures was discussed, now we turn our attention to learning structures from data. It is considered that native speakers have the ability to accurately complete tasks of linguistic well-formedness given valid arbitrary inputs.

3.1 Annotated Data

Annotation is usually done in iterations, inter-annotator agreement is measured with disagreement solved through iterations. The usefulness of datasets depends on the consistency of the annotations. When working with an annotated dataset one must assume:

The inputs and outputs by a dataset represent a learnable relation.

Language acquisition in humans has two main tracks of research. The first track argues language is not learnable, but an humans are born with an innate ability to learn language. The second line believes that some features influence abilities to learn language, which is a belief closer to computational NLP.

3.2 Generic formulation of learning

We define $\mathcal{H}$ as the set of possible predictors $\mathcal{X} \to \mathcal{Y}$. We define loss as a function $\mathcal{X} \times \mathcal{Y} \times (\mathcal{X} \to \mathcal{Y}) \to \mathbb{R}$ which defines the badness of the predictions made versus gold labels. We define the learning problem as

$$\min_{h \in \mathcal{H}} \mathbb{E}[loss(X, Y; h)] + complexity(h)$$

True loss (expectation) is approximated through a sample, and we call it empirical risk.

3.3 Generative models

A generative models assigns probabilities p to outcomes in $\mathcal{X} \times \mathcal{Y}$. Generative models typically assume that examples are drawn from the same distribution, so probabilities can be factored:

$$p(\langle x_1, y_1 \rangle, ..., \langle x_n, y_n \rangle) = \prod_{i=1}^{n} p(X = x, Y = y)$$

Generative learning requires a specification of probability models $\mathcal{P} \subset \mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$. The output of a generative model is a distribution p over $\mathcal{X} \times \mathcal{Y}$ which should

approximate the true distribution. If we had the entire distribution (not just a sample), we could simply write probability as:

$$p(x, y) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{x = x_i \wedge y = y_i\}$$

Learning from log loss corresponds to **maximum likelihood estimation** (MLE). Given a model family $\mathcal{P}$ and a sample, we use MLE to choose a model from the family

$$argmin_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} loss(x_i, y_i; h) = argmax_{p \in \mathcal{P}} p(\langle x_i, y_i \rangle_{i=1}^{N})$$

Model complexity is left out here, it will be discussed in 3.3.7. Each $p \in \mathcal{P}$ corresponds to a set of parameters $w \in \mathbb{R}^d$. Now, the problem is framed as parameter estimation.

3.3.1 Decoding rule

Generative models suggest choosing $y*$ given x is by taking $y \in \mathcal{Y}_x$ that is most likely to occur with x under the model given by conditional probability:

$$y* = argmax_{y \in \mathcal{Y}_x} p_w(Y = y | X = x)$$

3.3.2 Multinomial-based models

A multinomial distribution assigns probabilities to a discrete, finite set of events $e \in \varepsilon$: $p_\theta(e) = \theta_e$. Probabilities must be normalized to sum up to 1. This can be thought of a multi-sided die where each side is a single event with its probability.

3.3.3 Hidden Markov Models

Hidden Markov Models are built over sequences. A discrete first-level HMM is defined by a set of states Λ, vocabulary Σ of symbols and real-valued parameters w.