IFT 6132 – Advanced Structured Prediction **Winter 2019**

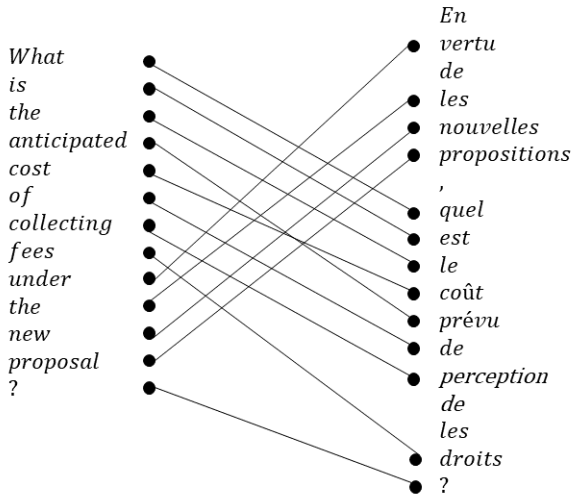
Lecture 1 — January 8, Université de Montréal

Lecturer: Simon Lacoste-Julien *Scribe: Frederic Boileau, Adel Nabli*

1.1 Introduction to structured prediction

The goal of *structured prediction* is to learn a "good" mapping from some arbitrary discrete input $x \in \mathcal{X}$ to a **structured output** $y \in \mathcal{Y}$ which we can think of as an *object made of interrelated parts*. As the size of the output space $\mathcal{Y}$ usually grows exponentially with the length of the input sequence, the expected structure of the output is used to come up with constraints that reduce the number of admissible mappings.

Example 1.1.1 (Word alignments) *Here, the input is a couple of sentences that are in translation of one another and the output is the set of matching pairs of words. We can add a matching constraint such as setting a maximum of one alignment per word.*



Example of word alignments

In this example, if we name $e = e_1, ..., e_E$ the english sentence and $f = f_1, ..., f_F$ the french one and encode the alignments as $y_{ij} = \mathbb{1}(e_i \text{ linked to } f_j) \in \{0, 1\}$ with $(i, j) \in E \times F$, then we have $|\mathcal{Y}| = 2^{E \times F}$. Adding the constraint ($\sum_i y_{ij} \leq 1$ and $\sum_j y_{ij} \leq 1$) restrains the size of the output space to $|\mathcal{Y}| = \sum_{k=0}^{\min(E,F)} \binom{\min(E,F)}{k} \binom{\max(E,F)}{k}$.

More examples of structured prediction problems can be found in Appendix A.

To specify what is a "*good mapping*", we use a *structured error* function $l(y, y')$ that assigns a cost to an error between a prediction y' and the ground truth y. We need an error function on complex labels because not all mistakes are equal. For instance in machine translation making a mistake on a single word is not as bad as getting the whole sentence wrong. In the case of scene labelling it is a much smaller mistake to label a car as a truck than say, a pedestrian as the road. Note that error functions do not necessarily have to be *metrics* as rigorously defined in analysis.

In light of what we introduced, we can characterize structured prediction in contrast to traditional classification with the following:

1. The set of labels $\mathcal{Y}$ is usually **exponential in size**.
2. We need an **encoding function** to represent the labels $y = (y_1, \ldots, y_d)$. The "pieces" of y have constraints.
3. The prediction task is characterized by a **structured error/loss function** $\ell(y, y')$. In other words we do not automatically get an obvious metric for the loss (such as mean squared error in simple linear regression).

1.2 Structured Prediction Basis and Setup

Now we will introduce a more formal setting to structured prediction.
Given a training dataset

$$D \triangleq (x^{(i)}, y^{(i)})_{i=1}^{n} \qquad x^{(i)} \in \mathcal{X} \quad y^{(i)} \in \mathcal{Y} \quad i = 1, \ldots, n \tag{1.1}$$

we want to learn a *prediction mapping* $h_\omega : \mathcal{X} \to \mathcal{Y}$ where ω is a parameter for the mapping. Moreover we want that mapping to have *low generalization error*:

$$L(\omega; P) \triangleq \mathbb{E}_{(x,y)\sim P}[l(y, h_\omega \mid x)] \tag{1.2}$$

In the above equation, we have:

- L is the *generalization error*.
- P is the *unknown "test" distribution* on (x, y): $(X, Y) \sim P$.
- $l : \mathcal{Y}^2 \to \mathbb{R}$ is the *structured error function* which evaluates how bad different prediction mistakes are (not all mistakes are equal).
- $h_w : \mathcal{X} \to \mathcal{Y}$ is the *predictor* parametrized by the random variable w (thus, L is also a random variable).

Note: It's important to distinguish the *generalization error* (or the *Vapnik risk* as Simon calls it) from the *frequentist risk* which is an expectation over all the possible datasets D and **not** an expectation over all the possible *ground truth pairs* following P as it is here. Indeed, if we make the assumptions that all pairs $(x^{(i)}, y^{(i)})$ are i.i.d samples, thus each dataset D_n is a random variable following a joint distribution $\underbrace{P \otimes .... \otimes P}_{n\ times}$ noted $P^{\otimes n}$. If we have a *decision procedure* (e.g a learning algorithm) $A : D_n \mapsto \hat{w_n}$, then the *frequentist risk* is a summary of the performance of A over all the possible datasets defined as the following:

$$R_P^F(A) \triangleq \mathbb{E}_{D_n \sim P^{\otimes n}} \big[L\big(A(D_n); P\big) \big] \tag{1.3}$$

To go deeper, see lectures 4 and 5 of IFT6269 (Probabilistic Graphical Models) for a review of statistical decision theory[1]

Note: In the Machine Learning field, L is simply called "the risk".

1.3 Empirical Risk Minimization

As stated above our goal is to minimize the generalization error, aka the Vapnik risk as defined in (1.2). However in general we do not have access to the distribution over which the expectation is taken so that we need to approximate it. We thus introduce an *empirical version* of this risk $\hat{L}$ *(the average of the loss function over the training set)* which has the property to converge for every w towards the target L as the number of observations n grows. In practice, we will try to minimize with respect to w a *regularized version of the empirical risk*:

$$\hat{L}(\omega) \triangleq \frac{1}{n} \sum_{i=1}^{n} l(y^{(i)}, h_\omega(x^{(i)})) + R(\omega) \tag{1.4}$$

The term $R(\omega)$ is a regularizer for the parameter in the above (e.g. $||\cdot||^2$). While we have the information necessary to compute the above it is rarely easy to miminize; the structured loss functions (in the summation above) which naturally arise are not convex and/or continuous in general. While there are techniques to deal with non continuous objective functions the absence of convexity is a major problem in optimization.

The solution is to replace the loss function l by a *surrogate loss function* $\mathcal{L}$ (or a contrast function in the statistics literature). While there are many design consideration for such a surrogate loss function the most commonly required feature is convexity. Examples for such functions are the structured hinge loss and the log-loss. Yann LeCun presents and discusses many of these functions in his paper on energy based models[2].

[1] `http://www.iro.umontreal.ca/~slacoste/teaching/ift6269/A18/`

[2] `http://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf`

Substituting those new loss functions into the expression for ERM we get the objective function of our optimization problem:

$$\hat{\mathcal{L}}(\omega) \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(x^{(i)}, y^{(i)}, \omega) + R(\omega) \tag{1.5}$$

1.4 Generative vs. Discriminative models.

Prediction consists in learning a mapping $h_w : \mathcal{X} \longrightarrow \mathcal{Y}$. This can be done at different levels of modelling and robustness. There is a tradeoff: a generative model is more powerful –since it explains the data more completely– but it is less robust –since it makes more assumptions–, whereas a simpler, discriminative model explains less but is more robust.

Types of model from least to most discriminative:

- **joint-probability** $p_w(x, y)$ – generative model of distribution of x and y jointly. The fit to data is the likelihood, and learning is done by finding the parameters w maximizing the likelihood (ML). One way to make predictions to take $h(x) = \text{argmax}_y \, p_w(y|x) = \text{argmax}_y \, p_w(x, y)$.
- **conditional probability** $p_w(y|x)$ – conditional generative model of the distribution of y once x is fixed. The fit to data is the conditional-likelihood, and learning is done by finding the parameters w maximizing the conditional-likelihood (MCL). One way to make predictions to take $h(x) = \text{argmax}_y \, p_w(y|x)$.
- **prediction mapping** $y = h_w(x)$ – discriminative model: instead of a model of x or y, directly learn a mapping from x to a good label y. The model is learned by surrogate loss minimization, which uses information from an **loss function** or **error function** $l(y, y')$, which in turn defines the learning task.

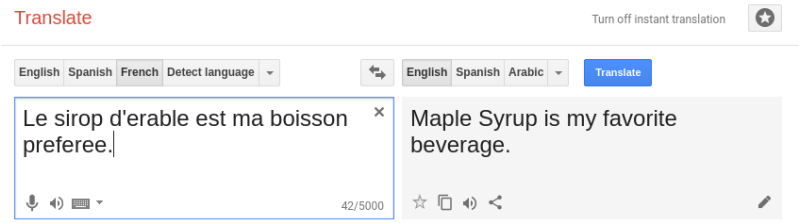
The terminology "structured prediction" traditionally focuses on prediction mappings (vs. generative modelling), and will be the approach we take in this course. Loosely speaking, because pointwise estimators are more discriminative, they are simpler which gives them more power to model more complex labels y. Even though the learned prediction function is not the correct one, we can still try to find the best one – in terms of classification error – among a class of functions.

Appendix A

Examples of structured prediction problems

Example A.0.1 (optical character recognition) *from an input image x representing a word or sentence, predict the vector y of corresponding letters where each dimension corresponds to successive letters.*

Example A.0.2 (machine translation) *translate a sentence from one language to the other.*



Google translate interface.

Example A.0.3 (3D object recognition) *from a depth map x predict an object map y which maps each pixel location to an object class.*

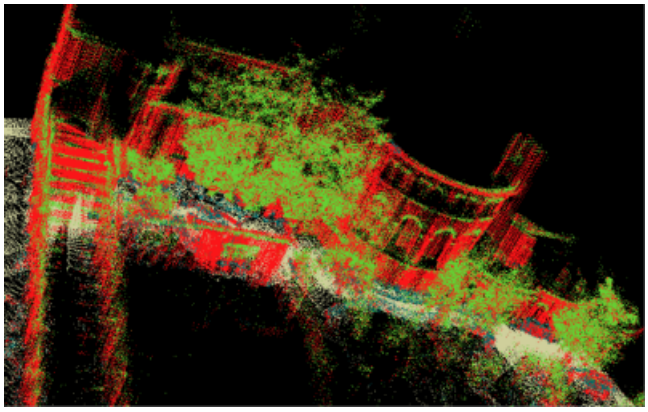


Image courtesy of Ben Taskar.

Example A.0.4 (protein folding) *from a sequence of amino-acids x predict the 3D structure y of the corresponding protein.*

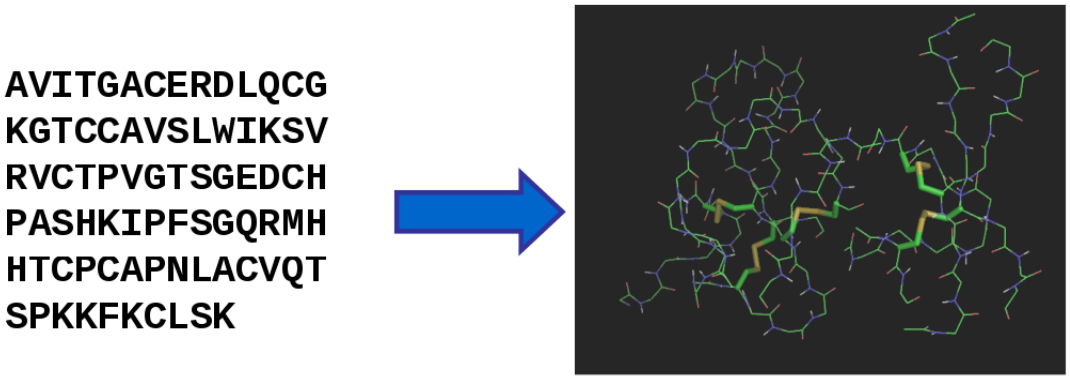


Image courtesy of Ben Taskar.

Example A.0.5 (text parsing) *from a sentence of words x predict the parse tree(s) y that generated x.*

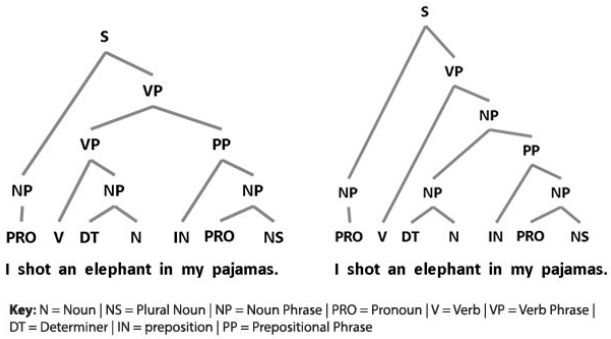


Image courtesy of Ron Daniel[1].

[1] https://www.elsevier.com/connect/new-open-access-resource-will-support-text-mining-and-natural-language-processing