Lecture 1 — January 8, Université de Montréal

Lecturer: Simon Lacoste-Julien *Scribe: Frederic Boileau*

1.1 Structured Prediction Basis and Setup

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 L is the generalization error[1] which takes as input the parameter of the prediction mapping and P which is an (arbitrary?) 'test distribution'. The function $l(\cdot, \cdot)$ on the RHS is the *structured error function* which evaluates how bad the different mistakes are.

Note: See lectures 4 and 5 of IFT6269 (Probabilistic Graphical Models) for a review of statistical decision theory[2]

Two aspects characterize structured prediction in contrast to traditional classification:

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).

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.

[1] also known as the Vapnik risk in ML literature

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

1.2 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 expctation is taken so that we need to approximate it. We thus define the empirical risk minimization which is just the average of the loss function over the training set.

$$\hat{L}(\omega) \triangleq \frac{1}{n} \sum_{i=1}^{n} l(y^{(i)}, h_\omega(x^{(i)})) + R(\omega) \tag{1.3}$$

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 by a *surrogate loss function* (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[3].

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

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

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

1.3 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, 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 $h(x) = \text{argmax}_y\, p_w(y|x)$.

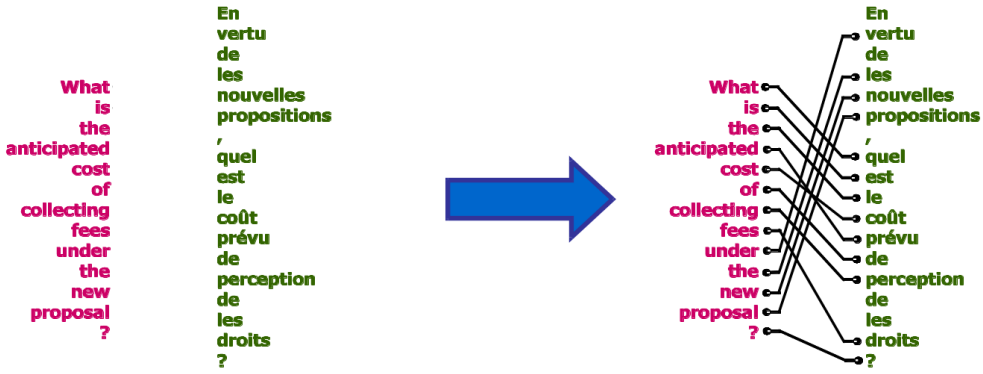
- **prediction mapping** $y = h_w(x)$ – discriminative model: instead of a model of x or y, directly learn a mapping from x to a good label y. The model is learned by surrogate loss minimization, which uses information from an **loss function** or **error function** $l(y, y')$, which in turn defines 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.

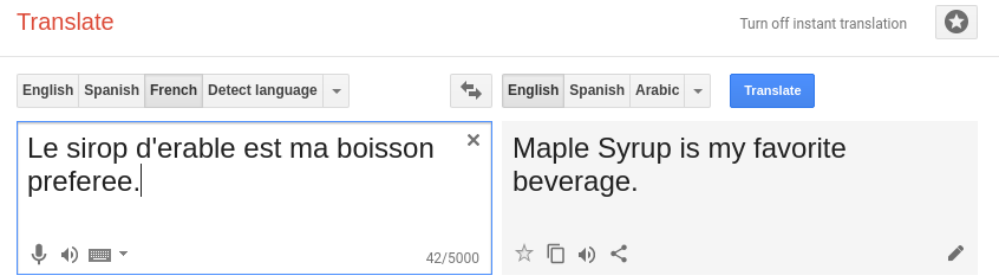
1.4 Examples of structured prediction problems

Example 1.4.1 (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 1.4.2 (word alignment) *Align sentences in two languages.*



Example 1.4.3 (machine translation) *translate a sentence from one language to the other.*



Google translate interface.

Example 1.4.4 (3D object recognition) *from a depth map x predict an object map y which maps each pixel location to an object class.*

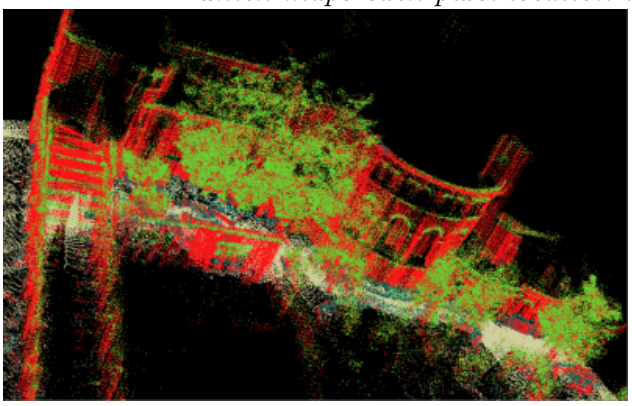


Image courtesy of Ben Taskar.

Example 1.4.5 (protein folding) *from a sequence of amino-acids x predict the 3D structure y of the corresponding protein.*

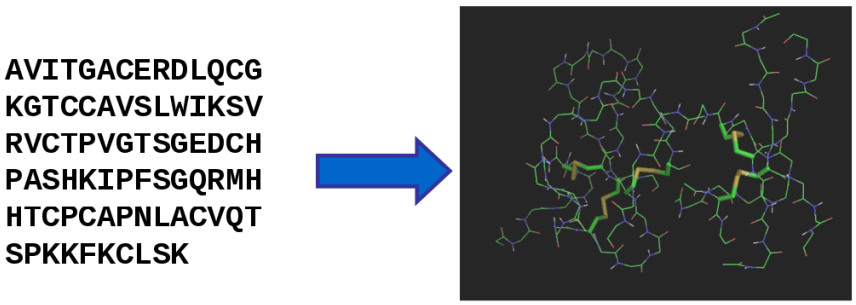


Image courtesy of Ben Taskar.

Example 1.4.6 (text parsing) *from a sentence of words x predict the parse tree(s) y that generated x.*

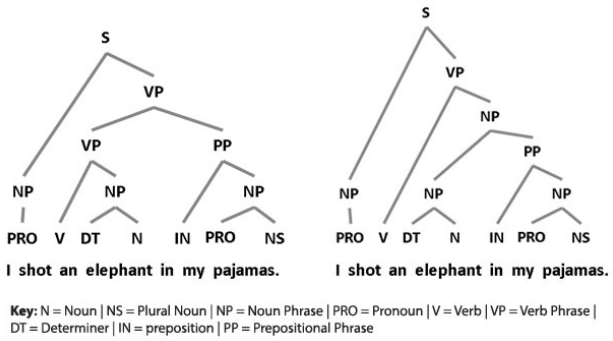


Image courtesy of Ron Daniel[4].

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