University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 12: Structured Prediction (exhaustive search, dynamic programming) October 09

Lecturer: Alexander Schwing Scribe: Weicheng Jiang

1 Goals

- Getting to know structured prediction
- Understanding basic structured inference algorithms

Reading material: D. Koller and N. Friedman; Probabilistic Graphical Models: Principles and Techniques;

2 Recap

In the previous lectures, we discussed about various learning models including:

- Linear regression
- Logistic regression
- Binary SVM
- Multiclass regression
- Multiclass SVM
- Deep Learning

The general framework for learning is:

$$\min_{\mathbf{w}} \frac{C}{2} \|\mathbf{w}\|_2^2 + \sum_{i \in D} \epsilon \ln \sum_{\hat{y}} \exp \frac{L(y^{(i)}, \hat{y}) + F(\mathbf{w}, x^{(i)}, \hat{y})}{\epsilon} - F(\mathbf{w}, x^{(i)}, y^{(i)})$$

where $F(\mathbf{w}, x^{(i)}, y^{(i)})$ is the scoring function and the loss function includes log-loss, hinge-loss and taskloss L.

For inference problems, we need to find the configuration with the highest scoring function value, that is to find $y^* = \arg\max_{\hat{y}} F(\mathbf{w}, x, \hat{y})$. So how do we solve it? One straightforward way is to do exhaustive search over all classes, which is easy for binary or few classes. But y could take infinite number of states, making exhaustive search unpractical.

3 Introduction

Structured prediciton is a type of supervised machine learning that involves predicting complex structured objects rather than scalar discrete or real values where the variables in the structured object are somewhat related.

4 Motivation

So why is structured prediciton useful? Let us first take a look at an example:

Correlations not taken into account Correlations taken into account

As we can see above, a four-letter word was predicted from its handwriting. On the left, each of the four letters was predicted independently, only based on the information of each single letter. On the right, the prediction takes correlations into account, that is each letter was predicted with the additional information of other letters, in this case, its prior letter. We can see that the prediction of the four-letter word with correlations taken into consideration generates a more accurate result because it uses the information that a "U" is more likely to follow a "Q" than a "V" in a word.

In the example of predicting a four-letter word from its handwriting, we can formulate the problem as the prediction of all four letter words, that is to formulate the four letter input into an entirety. But the problem is that the output space can be really large, because each letter can take up to 26 different values and a four-letter word can have up to 26^4 different configurations. Let us take a look at more example applications to better understand structured prediction and its potential problems.

Example: Disparity map estimation

Image Independent Prediction **Structured Prediction**

In disparity map estimation, independent prediction estimates the shift of each point between two stereo images independently, which generates a disparity map with a lot of noises. A structured prediction on the other hand, estimates the shift of each point with the additional information of its surrounding points, which results in a much smoother disparity map that is close to the ground truth. Then how big is the output space? For independent prediction, it would be the number of possible values for each pixel times the number of pixels in the image. For structured prediction, the output space would be the number of possible values for each pixel to the power of the number of pixels in the image, which is a lot larger.

Example: De-noising

We can see above that structured prediction can also be useful for de-noising images because it uses the information of neighboring pixels to produce more accurate estimation of the pixel values to smooth out the noises.

More example structured prediction applications:

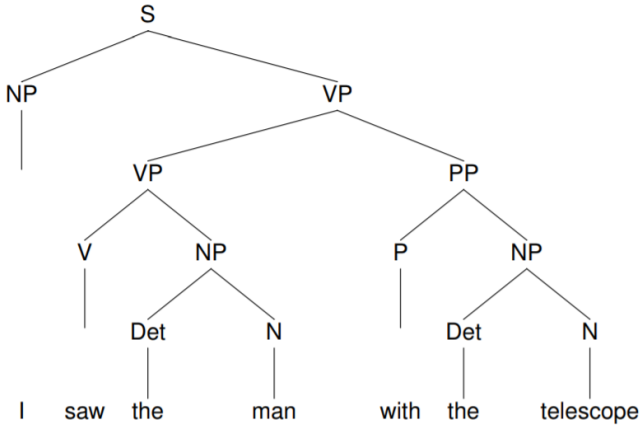
Image segmentation: estimate a label for each pixel to classify it to an object or the background.

$$x^{(i)} \rightarrow \boldsymbol{y}^{(i)} = (y_1^{(i)}, \ldots, y_D^{(i)})$$

Sentence parsing: estimate a parse tree for a sentence that decomposes the sentence into different grammatical components trying to understand the meaning of the sentence.

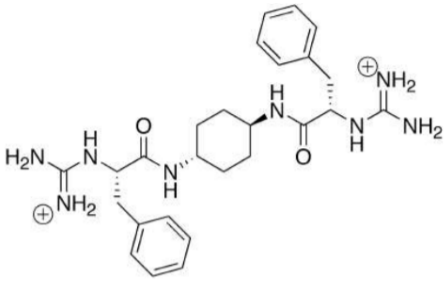
$$x^{(i)} \rightarrow \boldsymbol{y}^{(i)} = (y_1^{(i)}, \ldots, y_D^{(i)})$$

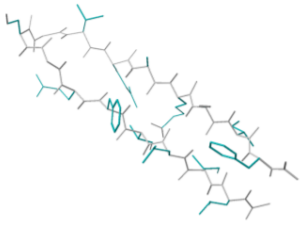
I saw the man with the telescope.



Protein folding: estimate a protein structure.

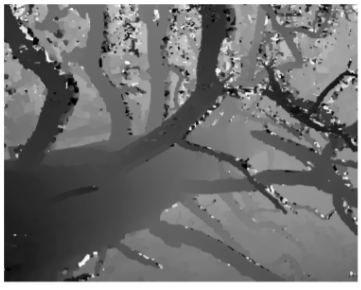
$$x^{(i)} \rightarrow \boldsymbol{y}^{(i)} = (y_1^{(i)}, \ldots, y_D^{(i)})$$





Stereo vsion: estimate a disparity map.

$$x^{(i)} \rightarrow \boldsymbol{y}^{(i)} = (y_1^{(i)}, \ldots, y_D^{(i)})$$



5 From Standard Prediction to Structured Prediction

In "Standard" Prediction, output y is a scalar number, so $y \epsilon \mathcal{Y}$, where $\mathcal{Y} = \{1, ..., K\}$ or $\mathcal{Y} = \mathbb{R}$.

In "Structured" Prediction, output $\mathbf{y}$ is a structrured output, so $\mathbf{y} = (y_1, ...y_D)$ and $y_d \epsilon \{1, ...K\}$. Now the Inference/Prediction problem becomes finding the structured configuration that maximizes the scoring function, that is to find

$$\mathbf{y}^* = \arg\max_{\hat{\mathbf{y}}} F(\mathbf{w}, x, \hat{\mathbf{y}}) = \arg\max_{\hat{\mathbf{y}}} F(\mathbf{w}, x, \hat{y}_1, \ldots, \hat{y}_D)$$

Then how many possible values do we have to store and explore? Since the structured output $\mathbf{y}$ has D variables and each variable can take K possible values, the output space is K^D which can be really large. So what can we do? One possible solution is to have separate predictions for the variables in the structured object.

If the scoring function of the structured prediction can be decomposed into the sum of scoring functions of each single variable in the structured object, that is

$$F(\mathbf{w}, x, \hat{y}_1, \ldots, \hat{y}_D) = \sum_{d=1}^{D} f_d(\mathbf{w}, x, \hat{y}_d)$$

then we can maximize the scoring function of the structured output by maximizing the the scoring function of each individual variable in the structured output, that is

$$\max_{\hat{\mathbf{y}}} F(\mathbf{w}, x, \hat{y}_1, ..., \hat{y}_D) = \max_{\hat{\mathbf{y}}} \sum_{d=1}^{D} f_d(\mathbf{w}, x, \hat{y}_d) = \sum_{d=1}^{D} \max_{\hat{y}_d} f_d(\mathbf{w}, x, \hat{y}_d)$$

So why do we not predict each variable y_d from $\mathbf{y} = (y_1, ..., y_D)$ separately? Because then correlations between neighboring variables are not explicitly taken into account, which can decrease the prediction accuracy as we can see in the previous examples.

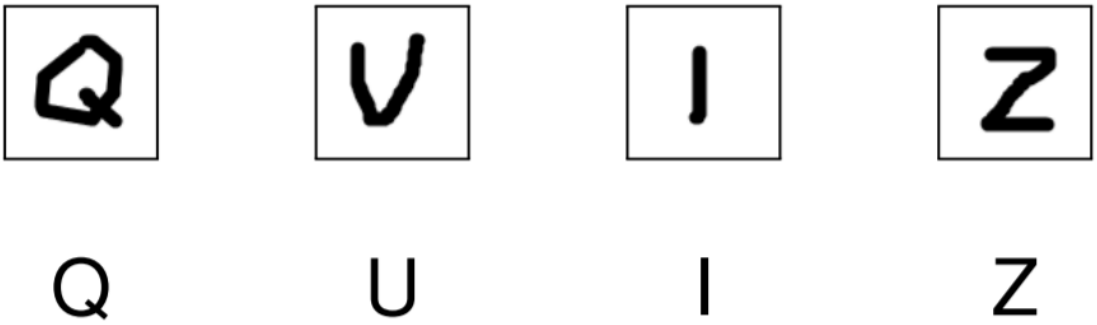
Now let us try to decompose the scoring function less trivially, that is

$$F(\mathbf{w}, x, y_1, \ldots, y_D) = \sum_{r \epsilon \mathcal{R}} f_r(\mathbf{w}, x, \mathbf{y}_r)$$

where $r \subseteq \{1, ..., D\}$ is the Restriction set and $\mathcal{R}$ is the set of all restriction sets.

For example, if $r = \{1, 2\}$, then $f_{\{1,2\}}(\mathbf{y}_{\{1,2\}}) = f_{\{1,2\}}(y_1, y_2) = [f_{\{1,2\}}(1, 1), f_{\{1,2\}}(1, 2), ...]$

If we go back to the previous example of predicting a four-letter word from its handwriting,



Q U I Z

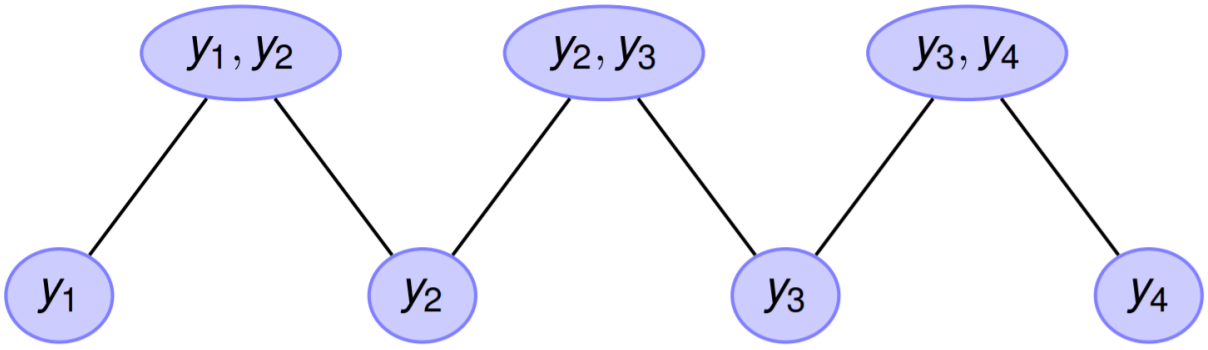
decomposing the scoring function would yield

$$F(\mathbf{w}, x, y_1, \ldots, y_4) = f_1(\mathbf{w}, x, y_1) + f_2(\mathbf{w}, x, y_2) + f_3(\mathbf{w}, x, y_3) + f_4(\mathbf{w}, x, y_4)$$

$$+ f_{1,2}(\mathbf{w}, x, y_1, y_2) + f_{2,3}(\mathbf{w}, x, y_2, y_3) + f_{3,4}(\mathbf{w}, x, y_3, y_4)$$

Now let us look at how many function values are needed to be stored if each letter can take 26 possible values. Earlier when each letter is dependent on all four letters, the number of possible function values are 26^4. Now we are able to decompose the scoring function into four unary terms that only depend on one single letter and three pairwise terms that depend on two neighboring letters, the resulting number of different function values are $3 \cdot 26^2 (+4 \cdot 26)$ which is much smaller. Note that the unary terms are not necessary because they can be incorporated into the pairwise terms.

If we try to visualize the decomposition of the scoring function, it would look like the following:



where the nodes denote the scoring function terms and the edges denote the subset relationships.

There are two special cases for the decomposition:

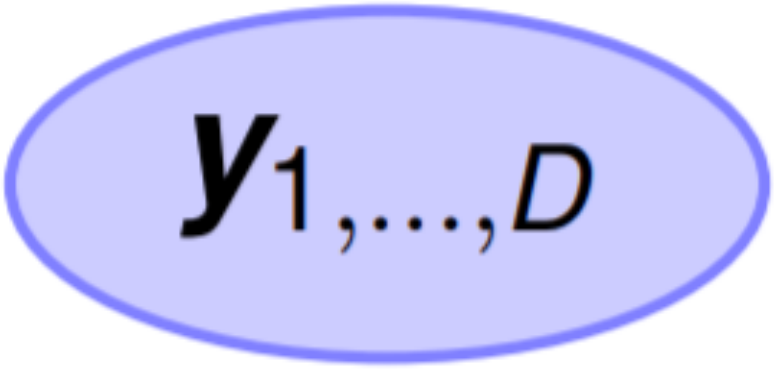
One extreme is to predict every variable of the structured output separately, that is

$$F(\mathbf{w}, x, y_1, \ldots, y_D) = \sum_{d=1}^{D} f_d(\mathbf{w}, x, y_d)$$

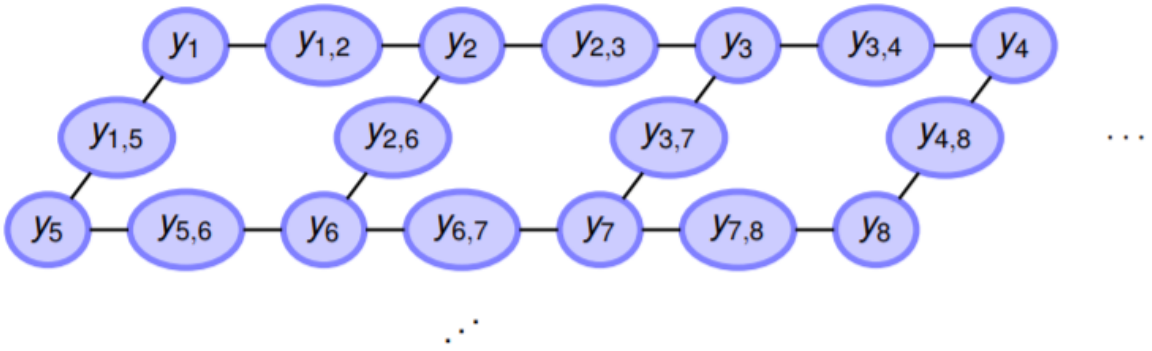
which is similar to Markov random field with only unary variables.

The other extreme is to have one single multi-variable prediction that depends on all the variables without any decomposition of the structured output, that is

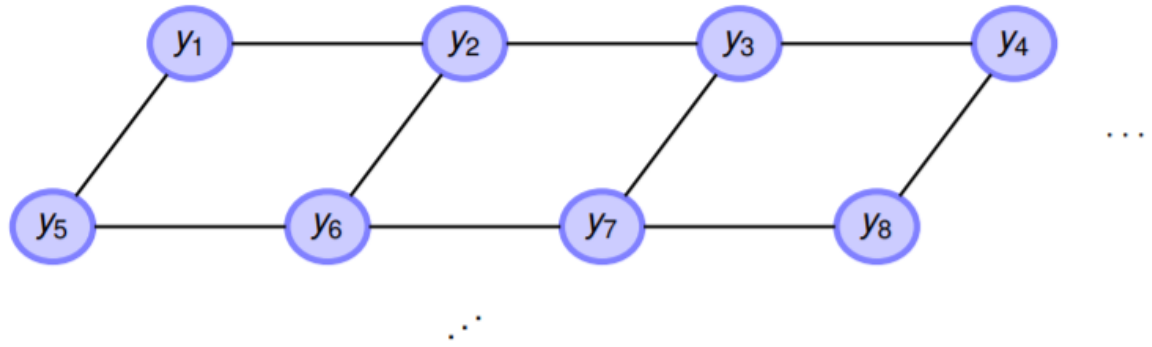
$$F(\mathbf{w}, x, y_1, \ldots, y_D) = f_{1,\ldots,D}(\mathbf{w}, x, \mathbf{y}_{1,\ldots,D})$$



Let us take a look at more examples. In the case of stereo vision, pixels are the variables we want to predict. We also want to take neighboring pixels into account so we have the following grid type structure, where unary variables are connected by pairwise functions.



Sometimes people just use edges to denote the pairwise functions between neighboring pixels for simplicity:



$$F(\mathbf{w}, x, \mathbf{y}) = \sum_{d=1}^{D} f_d(\mathbf{w}, x, y_d) + \sum_{i,j} f_{i,j}(\mathbf{w}, x, y_i, y_j)$$

As a result, the decomposition consists of unary terms that encode individual pixel scores and some pairwise terms such as the smoothness between two neighboring pixels. For example, you can run edge detection on the entire image. If no edge is detected, the pairwise function should encourage a pixel to have the same value as its neightbor while if an edge is detected, the pairwise function score should discourage two neighboring pixels to have the same value. Note in this case, the Markov/Conditional random field is not fully connected because neighboring pixels are much more likely to have stronger influence than pixels that are far away. But in general it really depends on the specific applications to determine which connections are useful and which are not. Similar decomposition can apply to semantic segmentation where unary terms encode some local image evidence such as individual pixel values and the pairwise terms encode some smoothness priors like image edges.

Now we have our general formulation for inference:

$$\mathbf{y}^* = \arg\max_{\hat{\mathbf{y}}} \sum_{r} f_r(\mathbf{w}, x, \hat{\mathbf{y}}_r)$$

How do we find the highest scoring configuration? Some inference algorithms are:

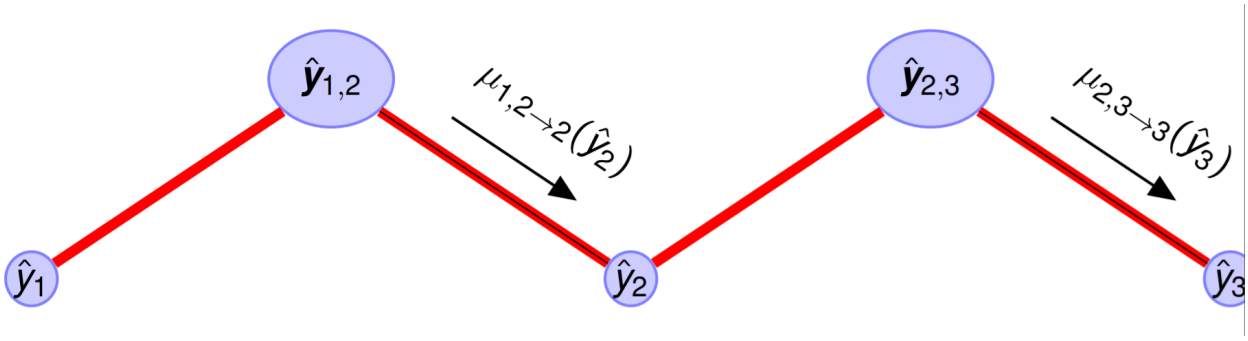
- Exhaustive search
- Dynamic programming
- Interger linear program
- Linear programming relaxation
- Message passing
- Graph-cut

In this lecture, we will focus on exhaustive search and dynamic programming.

6 Exhaustive Search

Exhaustive search is simple and straightforward and is almost always the first method we can try out. The basic idea is to try all possible configurations $\hat{\mathbf{y}} \epsilon \mathcal{Y}$, calculate corresponding scoring funciton values and keep the configuration with the highest score. The advantage is that it is very simple and straightforward to implement. The disadvantage, however is that it becomes very slow even for reasonably sized problems because the number of possible configurations is K^D, which increases exponentially and is not practical to do exhaustive search on.

7 Dynamic Programming



In dynamic programming, we decomposes the scoring function and maximize the terms with respect to one variable at a time and and keep going until the end. As a result, we have:

$$\max_{\hat{\mathbf{y}}} F(\mathbf{w}, x, \hat{\mathbf{y}}) = \max_{\hat{y}_1, \hat{y}_2, \hat{y}_3} \left(f_3(\hat{y}_3) + f_{2,3}(\hat{y}_2, \hat{y}_3) + f_2(\hat{y}_2) + f_1(\hat{y}_1) + f_{1,2}(\hat{y}_1, \hat{y}_2)\right)$$

$$= \max_{\hat{y}_3} \left(f_3(\hat{y}_3) + \max_{\hat{y}_2} \left(f_{2,3}(\hat{y}_2, y_3) + f_2(\hat{y}_2) + \underbrace{\max_{\hat{y}_1} \{f_1(\hat{y}_1) + f_{1,2}(\hat{y}_1, \hat{y}_2)\}}_{\mu_{1,2\to 2}(\hat{y}_2)} \right) \right)$$

$$= \max_{\hat{y}_3} \left(f_3(\hat{y}_3) + \max_{\hat{y}_2} \left(f_{2,3}(\hat{y}_2, \hat{y}_3) + f_2(\hat{y}_2) + \mu_{1,2\to 2}(\hat{y}_2) \right) \right)$$

The advantage is that it reduces the computational complexity from K^D for exhaustive search to $D \cdot K^2$ for using dynamic programming on pairwise models. However, dynamic programming is not always suitable and we can only reorganize the terms and maximize them using dynamic programming when the graph is a tree, which is its limitation.

8 Algorithms for general loopy graphs

In the following lectures, we will talk about algorithms that can be useful when solving structured predictions that have general loopy graphs. They are:

- Integer Linear Programs
- Linear Programming relaxations
- Dynamic programming extensions (message passing)
- Graph cut algorithms

9 Quiz and Answers

- Why structured output spaces? **Because sometimes we want to use correlations (information of neighboring variables) to make more accurate predictions of complex structured objects.**
- What makes computation with structured spaces hard? **The output space can have size K^D which increases exponentially and can be really large even with responably sized input.**
- Inference algorithms for structured output spaces? **Exhaustive search, Dynamic programming, Integer linear program, Linear programming relaxation, Message passing, Graph-cut**

References

[1] Schwing, Alexander. *ECE544NA Fall 2018: Pattern Recognition, Lecture 12: Structured Prediciton (exhaustive search, dynamic programming)*. Retrieved from `https://courses.engr.illinois.edu/ece544na/fa2018/`