University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 12: October 9

Lecturer: Alexander Schwing Scribe: Suraj Sanjay Jog

1 Overview

This document scribes Lecture 12: Structured Prediction (Exhaustive Search, Dynamic Programming). The learning objectives of this lecture are:

- Introduction to Structured Prediction and why it is useful
- Two structured inference algorithms, namely, Exhaustive Search and Dynamic Programming

The assigned reading material for this lecture is [1].

2 Preliminaries

2.1 General Framework

From the previous lectures, we have built the general framework for any learning problem to be formulated as follows:

$$\min_w \left(\frac{C}{2}\|w\|_2^2 + \sum_{i \in \mathcal{D}} \epsilon \; ln \sum_{\hat{y}} exp \; \frac{L(y^{(i)}, \hat{y}) + F(w, x^{(i)}, \hat{y})}{\epsilon} - F(w, x^{(i)}, y^{(i)}) \right) \tag{1}$$

where, w are the tunable parameters (weights), $x^{(i)}$ is a data sample in $\mathcal{D}$ and $y^{(i)}$ is the associated label, $\hat{y}$ is the set of all possible classes. Additionally, $F(w, x, y)$ is the scoring function and L is the Taskloss.

We have seen in the previous lectures that using the above general framework, one can obtain several learning paradigms including:

- Logistic Regression
- Binary SVM
- Multiclass Regression
- Multiclass SVM
- Deep Learning

2.2 Inference

Given the trained parameters w, the next step is to perform inference on some new data sample x. This inference process can be formulated as:

$$y^* = \arg\max_{\hat{y}} F(w, x, \hat{y}) \tag{2}$$

A straightforward way of solving the above is to evaluate $F(w, x, \hat{y})$ over all possible $\hat{y}$ exhaustively, and select the value that maximizes the scoring function. Such an approach is suitable when you are dealing with binary or multi-class classification with a few classes.

3 Structured Prediction

3.1 Why is structured information useful?

We will illustrate this point with the help of an example. Consider the task of letter recognition on the 4 examples shown in Fig.1. The black box presents the handwritten letter, and the red box shows the output of the letter recognition program. The predictions shown in Fig.1 look reasonable.

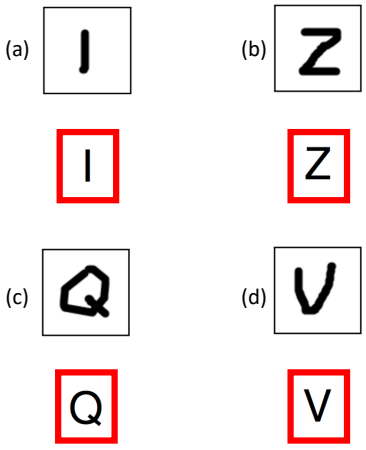


Figure 1: Letter Recognition Task

However, if instead of four such independent examples, we were given the input as shown in Fig.2, one would immediately realize that the second letter should be *'U'* instead of *'V'*, since the first letter is *'Q'*. Therefore, here, with the help of additional information related to context or structure of the word, we are able to refine and improve our prediction. This example demonstrates that leveraging the structure and correlations can help add context and improve prediction accuracy.

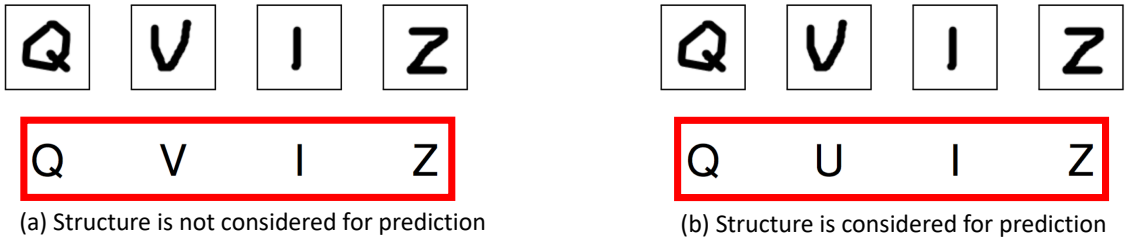


Figure 2: Letter Recognition Task

3.2 More Examples on Structured Prediction

Within the framework of structured prediction, our goal is to predict $\mathbf{y^{(i)}} = (y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, \ldots y_D^{(i)})$ given the input $x^{(i)}$. Therefore, now we are trying to predict the values of all the output variables

$y_j^{(i)}$ jointly, as opposed to independently. This way we can account for the correlations among the output variables, and in turn predict a complex object.

3.2.1 Word Prediction

In the word prediction example that we considered previously, we saw that independently predicting each letter was not a good approach. We instead want to do the prediction in a manner that will account for structure in the word. One way to incorporate this is to formulate the problem as a multi-class classification over all possible four letter words. That is, given an input $x^{(i)}$, we want to jointly predict $\mathbf{y^{(i)}} = (y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, y_4^{(i)})$.

However, in this formulation the size of the output space has now blown up to 26^4 (since each letter $(y_j^{(i)})$ has 26 possible classes). Therefore the complexity of our model has increased, but we have also increased the expressivity of the model since it can now account for structure and correlations in the prediction.

3.2.2 Disparity Map Estimation

Disparity Map Estimation is the task of estimating the depth of pixels/objects in an image. This is often also referred to as Stereo Vision. It is possible to solve this problem by estimating the depth of each pixel independently. However, doing so would result in a very noisy prediction as shown in Fig.3(b). Estimating the depth of pixels by looking at neighborhoods of pixels together however, is a more robust approach since we expect that nearby pixels would have similar depth since they are likely to belong to the same object. The output of such a structured prediction approach is shown in Fig.3(c), and we can see that this is less noisier than the independent prediction case. Again here, we should observe that with the structured prediction framework, the size of the output space has become very large. Specifically, if each pixel can be categorized into one of 21 different depth levels and the image has 1 Megapixels, this would mean the size of the output space would be 21^{1e6}, where we jointly predict all $\mathbf{y^{(i)}} = (y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, \ldots y_{1e6}^{(i)})$, given the image $x^{(i)}$. Such large output spaces are a problem that need to be dealt with, and we will talk about potential fixes later in the lecture.

Figure 3: Disparity Map Estimation

3.2.3 Image Segmentation

Image segmentation is another task where jointly predicting the labels for neighboring pixels helps improve prediction accuracy since neighboring pixels are expected to belong to the same object, unless the pixel represents an edge in the image. Using a structured prediction framework is ideal for this problem, and the image and output of such a segmentation task is shown in Fig.4.

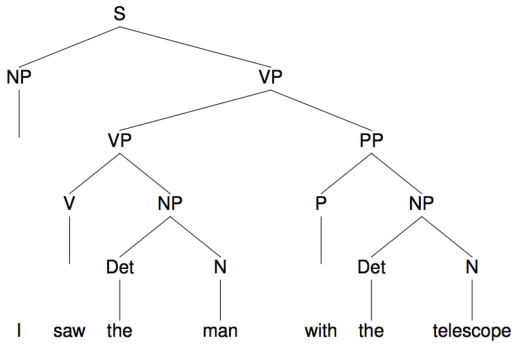
(a) Image (b) Segmentation Output

Figure 4: Image Segmentation

3.2.4 Sentence Parsing

Sentence parsing in NLP is the process of determining the syntactic structure of a text by analyzing its constituent words based on an underlying grammar (of the language). Fig.5 shows an example input sentence, along with the output from the parsing. In such tasks too, structure is very important as global information from across the sentence adds context and hence specific meaning to each word. For instance, if one were to observe the word *'saw'* independently without considering the other words of the sentence, it would not be easy to identify if *'saw'* refers to the verb or the noun (the cutting tool). However by observing the phrase *'I saw'*, it becomes clear that *'saw'* is a verb in this statement. Therefore, making use of structure is extremely beneficial in such tasks.

I saw the man with the telescope.



(a) Input Sentence (b) Sentence Parsing Output

Figure 5: Sentence Parsing

Some additional examples of structured prediction tasks such as image denoising and protein folding are shown in the lecture slides [2].

3.3 Formalizing Structured Prediction Tasks

In previous lectures, we have mostly dealt with "Standard" Prediction frameworks, where our model attempts to predict some output $y \in \mathcal{Y}$, where the y is a scalar number and $\mathcal{Y} = \{1, 2, \ldots, K\}$. However, now in the "Structured" Prediction framework we want to predict an output y which is a structured object and not just a scalar number.[1] In order to formulate the Structured Prediction framework, we will consider a number of approaches and discuss their advantages and disadvantages.

3.3.1 Approach 1: Joint Prediction

As a naive first step, we can model the Structured Prediction framework as follows. As stated previously, we now want to jointly predict all outputs as $\mathbf{y^{(i)}} = (y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, \ldots y_D^{(i)})$, where each output variable can belong to one of K classes, that is, $y_d \in \{1, \ldots, K\}$. The inference step can be formulated as:

$$y^* = \arg\max_{\hat{y}} F(w, x, \hat{y}) = \arg\max_{\hat{y}} F(w, x, \hat{y}_1, \ldots, \hat{y}_D) \tag{3}$$

where the scoring function F is a joint function of $\hat{y}_1, \ldots, \hat{y}_D$. This formulation is extremely expressive since it accounts for all possibilities of $\hat{\mathbf{y}}$. However, on the other hand, with this above formulation we will have to store and explore K^D possibilities for $\hat{\mathbf{y}}$. This is very large and clearly formulating the problem in this way does not scale for large D.[2] Therefore we want to make the formulation more tractable, albeit at the expense of model expressivity.

One way to reduce this K^D complexity is to approximate the scoring function F as a decomposition into multiple component functions, where each component function depends only on a subset of output variables. We will consider two such approaches next.

3.3.2 Approach 2: Separate Prediction

As a simple case, let us approximate the scoring function F to decompose as follows:

$$F(w, x, \hat{y}_1, \ldots, \hat{y}_D) = \sum_{d=1}^{D} f_d(w, x, \hat{y}_d) \tag{4}$$

Therefore the inference step in this case can be formulated as:

$$y^* = \arg\max_{\hat{y}} F(w, x, \hat{y}_1, \ldots, \hat{y}_D) = \arg\max_{\hat{y}} \sum_{d=1}^{D} f_d(w, x, \hat{y}_d) = \sum_{d=1}^{D} \arg\max_{\hat{y}_d} f_d(w, x, \hat{y}_d) \tag{5}$$

One should observe that such a decomposition of F is equivalent to predicting each $\hat{y}_d$ independent of the other output variables. Therefore, such a formulation does not take into account correlations and structure, and hence, does not provide the benefits of structured prediction that we have seen.

[1] One should however note that it is possible to transition from the "Standard" Prediction framework to the "Structured" Prediction framework by considering a standard multi-class classification problem where all the combinations for the output are considered as the potential classes. However, in order to do this, K would have to very large.

[2] Consider an image of 12 MP resolution. D would be 12e6 here, and even for just $K = 10$ classes, the output space is too large to be practical.

3.3.3 Approach 3: Non-trivial decompositions for F

Clearly, the previous two approaches were two opposite extremes, and we want to achieve some sort of a middle ground, where we are accounting for correlations in our model, but at the same time, the store and explore complexity is tractable. With this objective in mind, let us consider the following less trivial decomposition of F where:

$$F(w, x, \hat{y}_1, \ldots, \hat{y}_D) = \sum_{r \in \mathcal{R}} f_r(w, x, \hat{y}_r) \tag{6}$$

where each r is a *Restriction Set* with $r \subseteq \{1, \ldots, D\}$, and $\mathcal{R}$ is the set of all restriction sets.[3]

To see how this above formulation is useful, let us consider the word recognition problem in Fig.2. Let $\hat{y}_1$, $\hat{y}_2$, $\hat{y}_3$ and $\hat{y}_4$ be the output predictions for the four letters in the word. Clearly, each $\hat{y}_j$ can belong to one of 26 classes, that is, one of the 26 letters in the alphabet. Now, for this problem, let us choose $\mathcal{R} = \big\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{2, 3\}, \{3, 4\}\big\}$. Therefore, our score function F decomposes as

$$\begin{aligned} F(w, x, \hat{y}_1, \ldots, \hat{y}_4) = f_1(w, x, \hat{y}_1) + f_2(w, x, \hat{y}_2) + f_3(w, x, \hat{y}_3) + f_4(w, x, \hat{y}_4) \\ + f_{1,2}(w, x, \hat{y}_1, \hat{y}_2) + f_{2,3}(w, x, \hat{y}_2, \hat{y}_3) + f_{3,4}(w, x, \hat{y}_3, \hat{y}_4) \end{aligned} \tag{7}$$

The last three terms in the above decomposition capture the structure and correlations among the $\hat{y}_j$'s. For instance, let us look at the term $f_{1,2}(w, x, \hat{y}_1, \hat{y}_2)$. This term scores the possibility of some $y_1^{(i)}$ and $y_2^{(i)}$ to appear together as the first and the second letter respectively. However, $f_1(w, x, \hat{y}_1)$ on the other hand, only scores the possibility of the first letter being some $y_1^{(i)}$ without considering any other letter in the word. One should observe that the above formulation makes it feasible to predict the word in Fig.2 correctly, since it is expected that the score $f_{1,2}(w, x, \hat{y}_1 = \text{`}Q\text{'}, \hat{y}_2 = \text{`}U\text{'})$ will be much higher than $f_{1,2}(w, x, \hat{y}_1 = \text{`}Q\text{'}, \hat{y}_2 = \text{`}V\text{'})$.

This above formulation now requires us to store and explore only $3 \cdot 26^2 + 4 \cdot 26$ values of $\hat{\mathbf{y}}$, as opposed to 26^4 values of $\hat{\mathbf{y}}$ required in Approach 1. Therefore the formulation in Eq. 7 is computationally more efficient. However, this comes at a cost. Our model in Eq. 7, is not as expressive as the model in Approach 1, since we are considering only pairwise correlations and that too only for consecutive pairs of letters. But this tradeoff is favorable for us, since the problem now becomes tractable and it captures enough correlation to give us good prediction results. Therefore, the model proposed in Approach 3 with a suitable $\mathcal{R}$ is a good choice to model our structured prediction problem.

4 Graphical Models for Structured Prediction

We will now talk about how we can depict the models discussed previously in terms of a graphical representation. Let us begin by considering the model defined in the previous section for the word recognition task. The corresponding decomposition is:

$$\begin{aligned} F(w, x, \hat{y}_1, \ldots, \hat{y}_4) = f_1(w, x, \hat{y}_1) + f_2(w, x, \hat{y}_2) + f_3(w, x, \hat{y}_3) + f_4(w, x, \hat{y}_4) \\ + f_{1,2}(w, x, \hat{y}_1, \hat{y}_2) + f_{2,3}(w, x, \hat{y}_2, \hat{y}_3) + f_{3,4}(w, x, \hat{y}_3, \hat{y}_4) \end{aligned} \tag{8}$$

[3]This formulation is very general, and the formulation in Approach 1 and Approach 2 is in fact a special case of Eq. 6. However, due to reasons stated in the previous subsections, we will avoid choosing $\mathcal{R}$ that makes the problem boil down to Eq. 5 or Eq. 3.

The above decomposition can be visualized by the graph shown in Fig.6, where each node represents a function and the variables inside the node denote the variables that the function depends on. The edges in the graph denote the subset relationship.

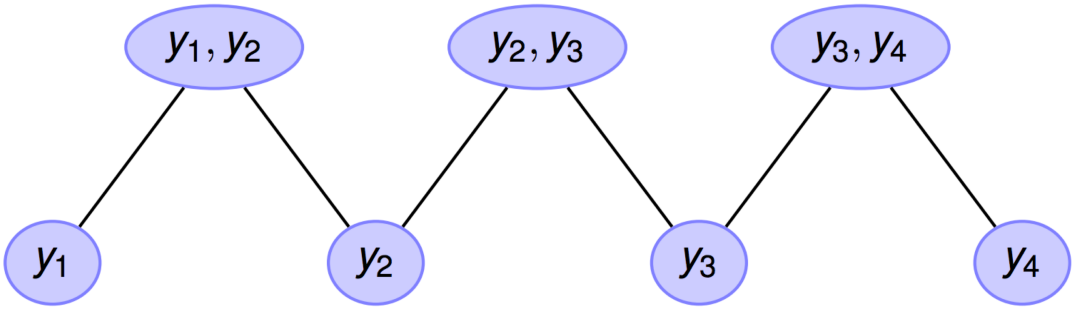


Figure 6: Graphical Representation

4.1 Special Cases

We will also look at the graphical representation for few special cases.

1. **Separate Prediction for variables**

   For the case of separate prediction, the scoring function can be decomposed as

   $$F(w, x, \hat{y}_1, \ldots, \hat{y}_D) = \sum_{d=1}^{D} f_d(w, x, \hat{y}_d) \tag{9}$$

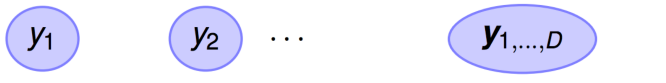
   This can be represented by the graph in Fig.7(a). Since there is no correlation/structure being accounted for in this model, we can observe that there are no connections between the nodes in the graphical representation. The graph in Fig.7(a) is a *Markov Random Field with only unary variables.*

2. **Joint Prediction for variables**

   For the case of joint prediction of all the variables, the scoring function can be written as

   $$F(w, x, \hat{y}_1, \ldots, \hat{y}_D) = f_{1,\ldots,D}(w, x, \hat{y}_{1,\ldots,D}) \tag{10}$$

   The above can be represented by the graph in Fig.7(b). Since there is only one function in the decomposition, the graph contains just one node.



a) Graphical Representation for Separate Prediction

b) Graphical Representation for Joint Prediction

Figure 7: Graphical Representation for Special Case 1 and 2

3. **Practical Problems**

(a) Stereo Vision (Input and Output) (b) Image Segmentation (Input and Output)

Figure 8:

Finally, we will consider two real-life problems, namely Stereo Vision and Image Segmentation as shown in Fig.8. As mentioned before, structured prediction is a well suited paradigm for solving these problems. However, we need to constrain the store and explore complexity of the output space to keep the problem tractable. Therefore we use the following decomposition for F where:

$$F(w, x, \hat{y}_1, \ldots, \hat{y}_D) = \sum_{d=1}^{D} f_d(w, x, \hat{y}_d) + \sum_{i,j} f_{i,j}(w, x, \hat{y}_i, \hat{y}_j) \tag{11}$$

where i and j are neighboring pixels. The graphical representation for this decomposition can be visualized as Fig.9

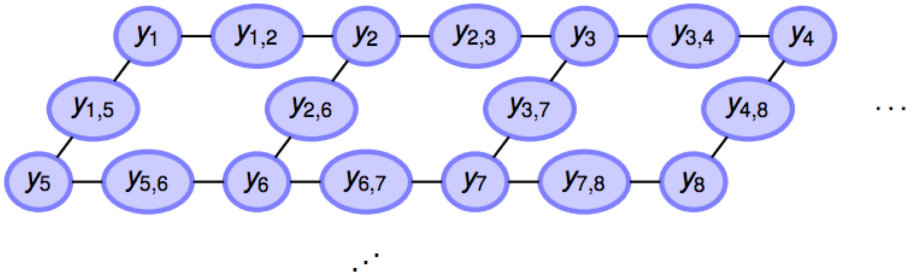


Figure 9: Graphical Representation for Eq. 11

This above model considers correlation/structure only for neighboring pixels. This makes sense because it is expected that far away pixels in the image will not be very correlated and it is in fact OK to predict their values independent of each other. However, by considering only neighboring pixels, the complexity is reduced to $O(DK^2)$ from $O(K^D)$. This makes the problem tractable, specially when we are dealing with large D as is often the case with images.

Finally, we should notice that the expression in Eq. 11 consists of unary terms (depending on just 1 output variable) and pairwise terms. The unary terms in the expression encode the image/local evidence whereas the pairwise terms encode the smoothness prior, i.e., the notion of nearby pixels having similar values.

5 Structured Inference Algorithms

As seen previously, the inference step can be formulated as

$$y^* = \arg\max_{\hat{y}} \sum_r f_r(w, x, \hat{y}_r) \tag{12}$$

There are various inference algorithms we can use to compute the above y^*, namely

- Exhaustive Search
- Dynamic Programming
- Integer Linear Program
- Linear Programming Relaxation
- Message Passing
- Graph-Cut

In this document, we will discuss Exhaustive Search and Dynamic Programming.

5.1 Exhaustive Search

The exhaustive search algorithm works as follows:

1. Enumerate the value of $\sum_r f_r(w, x, \hat{y}_r)$ for all possible configurations $\hat{y} \in \mathcal{Y}$
2. Keep the highest scoring element among all possible $\hat{y}$

The advantage of such an exhaustive scheme is that it is very simple to implement. However, the algorithm will be very slow for reasonably sized problems since you need to exhaustively evaluate for K^D possibilities of $\hat{y}$.

5.2 Dynamic Programming

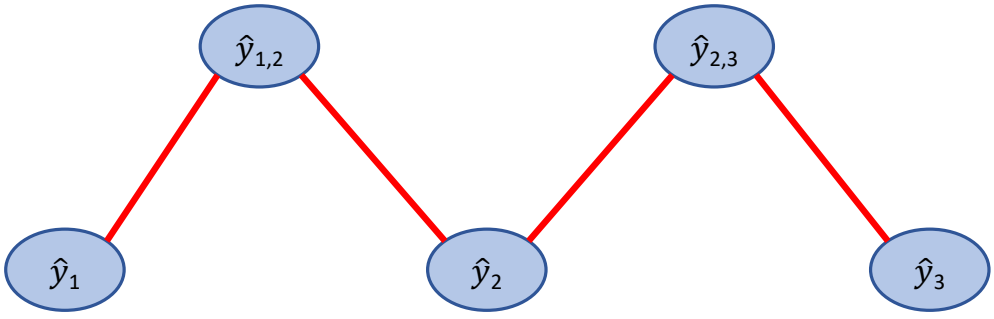


Figure 10:

Consider a decomposition of F represented by the graph in Fig.10. The inference step can be formulated as

$$y^* = \arg\max_{\hat{y}} F(w, x, \hat{y}) = \arg\max_{\hat{y}_1, \hat{y}_2, \hat{y}_3} \left(f_3(\hat{y}_3) + f_{2,3}(\hat{y}_2, \hat{y}_3) + f_2(\hat{y}_2) + f_{1,2}(\hat{y}_1, \hat{y}_2) + f_1(\hat{y}_1) \right) \tag{13}$$

For ease of notation, we will consider the maximization of the above objective instead of arg max. Rearranging the above terms, we get

$$\max_{\hat{y}} F(w, x, \hat{y}) = \max_{\hat{y}_3} \left(f_3(\hat{y}_3) + \max_{\hat{y}_2} \Big(f_{2,3}(\hat{y}_2, \hat{y}_3) + f_2(\hat{y}_2) + \max_{\hat{y}_1} \{ f_{1,2}(\hat{y}_1, \hat{y}_2) + f_1(\hat{y}_1) \} \Big) \right) \quad (14)$$

We can write the maximization this way since the innermost maximization collects all functions that depend on $\hat{y}_1$, the second maximization collects all functions that depend on $\hat{y}_2$, whereas the outermost function collects all functions that depend on $\hat{y}_3$. In order to solve the optimization problem in Eq. 14, we would start by first solving the innermost maximization/optimization problem to get $\mu_{1,2\to 2}(\hat{y}_2) = \max_{\hat{y}_1} \{ f_{1,2}(\hat{y}_1, \hat{y}_2) + f_1(\hat{y}_1) \}$. Since we are optimizing only with respect to $\hat{y}_1$, the output of the optimization will be a function of $\hat{y}_2$. Eq. 14 can now be written as:

$$\max_{\hat{y}} F(w, x, \hat{y}) = \max_{\hat{y}_3} \left(f_3(\hat{y}_3) + \max_{\hat{y}_2} \Big(f_{2,3}(\hat{y}_2, \hat{y}_3) + f_2(\hat{y}_2) + \mu_{1,2\to 2}(\hat{y}_2) \Big) \right) \quad (15)$$

By solving the next innermost optimization problem, we obtain $\mu_{2,3\to 3}(\hat{y}_3) = \max_{\hat{y}_2} \Big(f_{2,3}(\hat{y}_2, \hat{y}_3) + f_2(\hat{y}_2) + \mu_{1,2\to 2}(\hat{y}_2) \Big)$. Therefore, finally, Eq. 14 boils down to

$$\max_{\hat{y}} F(w, x, \hat{y}) = \max_{\hat{y}_3} \left(f_3(\hat{y}_3) + \mu_{2,3\to 3}(\hat{y}_3) \right) \quad (16)$$

Therefore, in the dynamic programming approach, in each step we are solving a local maximization problem and passing the information forward, and this step is repeated multiple times depending on the number of variables we have. The advantage of dynamic programming is that they have better complexity as compared to exhaustive search, since with pairwise terms the complexity in dynamic search reduces to $D \cdot K^2$. However, one should note that this approach is suitable only when the graph model of the decomposition is a *tree*. For all other general loopy graphs, we need to use one of the other four techniques listed earlier in this section.

6 Quiz

- *Why structured output spaces?*

  Ans: The example illustrated in Section 3.1 provides good motivation for why we would want to leverage structured output spaces.

- *What makes computation with structured spaces hard?*

  Ans: The number of possible outputs to store and explore can grow exponentially, and we need to model the problem in a way that it can leverage the structure/correlations while remaining tractable.

- *Inference algorithms for structured output spaces?*

  Ans: We have discussed two inference algorithms in this document, namely Exhaustive Search and Dynamic Programming.

References

[1] D. Koller, N. Friedman, and F. Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[2] A. Schwing. *ECE 544 NA: Pattern Recognition: L12: Structured Prediction (Exhaustive Search, Dynamic Programming) (Slides)*. UIUC, 2018.