Recitation: Homework 3

Topics: Structured SVMs, MAP Inference via ILP
Semantic Segmentation

Given an RGB image we hope to perform semantic segmentation. You can think of this as running a classifier on each pixel to determine boundaries between each object.

Our dataset will be a cropped and downsampled version of the PASCAL VOC dataset.
Model Architecture: FCN

Our baseline is based on a simple fully convolutional neural network (FCN) based on AlexNet with the dense layers removed. The FCN has been provided for you in your starter code. Take a moment to run the starter code now.
Model Architecture: Linear SVM

A small improvement we could make is to train a maximum margin model on top of our CNN features. The simplest of these will still only consider a single pixel at a time.

This is called a linear SVM and is the SVM you’re used to hearing about.

**Primal SVM Optimization Problem**

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i \\
\text{subject to} & \quad y_i((x_i, w) + b) \geq 1 - \xi_i, \quad (i = 1, \ldots, m) \\
& \quad \xi_i \geq 0, \quad (i = 1, \ldots, m)
\end{align*}
\]
Things to Consider for Linear SVM

1. What is our kernel function?

2. How would we write a linear SVM in PyTorch?

3. What is our loss function?
Model Architecture: Structured SVM

Primal Structured SVM Optimization Problem

\[
\min_w \|w\|^2 + C \sum_{n=1}^{\ell} \max_{y \in Y} (0, \Delta(y_n, y) + \langle w, \Psi(x_n, y) \rangle - \langle w, \Psi(x_n, y_n) \rangle)
\]

We need to define loss and score functions!
Defining a scoring function for structured SVM

$$\sum_i \sum_c \phi_{i,c} x_{i,c} + \sum_i \sum_j \sum_c \phi_{i,j,c} y_{i,j,c}$$

**Pixel assignment scores:**
X’s are boolean variables, set to 1 if pixel at position i is labeled by class c, 0 otherwise.

Phi’s are potentials computed by multiplying features for each pixel (from FCN) with weights ->

What can we use to do this in PyTorch?
Defining a scoring function for structured SVM

Pixel assignment scores:
X’s are boolean variables, set to 1 if pixel at position i is labeled by class c, 0 otherwise

Phi’s are potentials computed by multiplying features for each pixel (from FCN) with weights -> Use nn.Linear()!

Edge assignment scores:
y’s are boolean variables, set to 1 if pixels at position i and j are both labeled class c, 0 otherwise

Phi’s are potentials computed by multiplying concatenated features of pixel pairs (from FCN) with weights -> What can we use to do this in PyTorch?
Constraints on the scoring function

\[ \sum_{c} x_{i,c} = 1 \]

What do these constraints represent?

\[ y_{i,j,c} \leq x_{i,c} \]
\[ y_{i,j,c} \leq x_{j,c} \]
\[ y_{i,j,c} \geq x_{i,c} + x_{j,c} - 1 \]
Loss-Augmented Inference

Recall that structured SVM has a loss term in addition to the scoring function. For this assignment, we define that as the Hamming Loss.

Loss-augmented objective:

\[
\max_{x,y} \sum_i \sum_c \phi_{i,c} x_{i,c} + \sum_i \sum_j \sum_c \phi_{i,j,c} y_{i,j,c} + \sum_i \sum_c \left( x_{i,c}^* (1 - x_{i,c}) + (1 - x_{i,c}^*) x_{i,c} \right)
\]

This is the hamming loss term.
Structured Hinge Loss

$$\max(0, \max_{y \in Y} (\Delta(y, t) + \langle w, \phi(x, y) \rangle) - \langle w, \phi(x, t) \rangle)$$

Remember that delta is the hamming loss for our assignment!
Let's put everything together!

What is the training procedure?
Let's put everything together!

What is the training procedure?

1. For an example, run FCN
2. Compute edge/pixel potentials using FCN outputs
3. **Perform loss-augmented inference:** Optimize loss-augmented objective from the previous slide to find highest-scoring assignment of pixels and edges (x’s/y’s)
4. Compute structured hinge loss between highest-scoring assignment and gold output
5. Backpropagate
Let's put everything together!

What is the testing procedure?

1. For an example, run FCN
2. Compute edge/ pixel potentials using FCN outputs
3. **Perform MAP inference:** Optimize the score-based objective from slide 7 to find highest-scoring assignment of pixels and edges (x’s/y’s)
Building your own loss function in PyTorch

- Step 4 in the training process requires you to build your own loss function
- Two ways to do this:
  1. Subclass `nn.Module()` and write loss computation in the `forward()` function
  2. Write a custom loss function which takes in variables returned by the model’s `forward` function
- Reference follows option 2
- Let’s change the loss function to MSE for our FCN!
Using OR-Tools for ILP/ LP optimization

- Let’s consider the problem of assigning students to project teams
- Number of students: 20
- Number of teams: 7
- Each team can have between 2-4 students

Questions:
- What are the variables?
- What are some possible constraints that we can have?
- How do we formulate these constraints as inequalities?
Using OR-Tools for ILP/ LP optimization

Let’s make the problem a little more complex:

- Each team is associated with a specific topic
- Each student has preferences for topics they would like to work on
- We want to maximize student “happiness”
- How can we add this to our existing ILP?
Programming tips:

- **DO NOT** convert variables that you want to backprop through into numpy arrays

- **DO NOT** iterate/ recurse during score computation

- Avoid using PyTorch Variables in ILP optimization

- You need not consider both (i,j) and (j,i) when computing edge potentials