Structured SVMs and Conditional Random Fields for Semantic Segmentation

Ariyan Zarei

University of Arizona

ariyanzarei@email.arizona.edu

September 17, 2019

Overview

Motivation

Histopathological Image Analysis with Deep Learning

Histopathology is the examination of tissues from the body under a microscope to spot the signs and characteristics of disease.
Currently in our research project we are trying to classify patches of breast tissues into cancerous and normal classes.

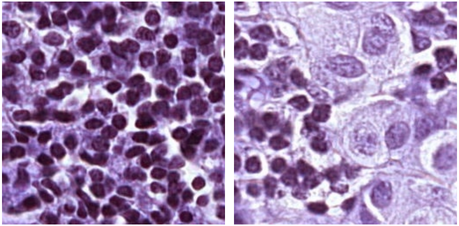

Figure: The left image is a sample of normal patch and the right is a sample of abnormal.

Patch Level Classification Challenges

Existence of darker and smaller normal cells in cancerous regions is possible. Thus, the patch-level classifier (Tuned VGG16) sometimes make False Normal decisions.

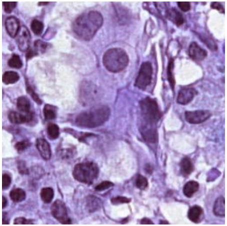

Figure: Existence of normal cells (features) in the abnormal patches

Solution: Take spatial relations between patches into account. (Neighbors)

Classification of Structured Data

Structured Input and Output

Two Types of data:

- Flat Data: Numbers
- Structured Data: Sequences

Input and output of the ML problems can be either flat or structured.

- Simple Image Classification, Stock Market Index Prediction (Structured – Flat)
- Labeling Image Segments, Part of Speech Tagging (Structured – Structured)

Semantic Segmentation

Linking each Pixel/Superpixel to a label.
Structured matrix of Superpixels to Structured matrix of labels (with the same size)

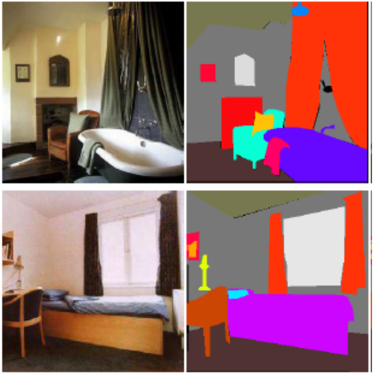

Figure: Real examples of Semantic Segmentation

Naive Approach

A Naive Approach is to create a specific class for each combination of possible labeling and then use a desired classifier.

Problem?

Naive Approach

A Naive Approach is to create a specific class for each combination of possible labeling and then use a multi-class classifier.

Problem? Exponential classes and parameters.

Solution. Use Structured Support Vector Machine.

Binary SVM Review

$$Min \frac{1}{2} W^{\top} W + C \sum_{i=1}^{n} \zeta_i$$

$$s.t. \quad \forall i \qquad y_i(W^{\top} x_i + b) >= 1 - \zeta_i$$

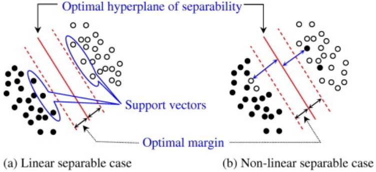

Figure: A simple example of Binary SVM

Multi-class SVM Review

One approach in Multi-class classification using SVM is the "one v.s. all" method.

Prediction: $argmax_m W_m^\top x + b_m$
Training Formula:

$$Min \frac{1}{2} \sum_{m=1}^{K} W_m^\top W_m + C \sum_{i=1}^{n} \sum_{m \neq y_i} \zeta_i^m$$

$$s.t.\ \ \forall i\ \forall m \neq y_i \qquad W_{y_i}^\top x_i + b_{y_i} >= W_m^\top x_i + b_m + 2 - \zeta_i$$

where y_i is the true class for x_i and m is any other class.
Intuition: **for all data points, the inner product of correct label weight vector with input, should be bigger than the inner product of any other weight vector with the input.**

Structured Support Vector Machines

Multi-class SVM for Structured data is not good:

- Exponential parameters, constraints
- No efficient prediction algorithm
- No efficient training algorithm

We need to avoid searching through all possible outputs (all possible labelings in our case).

Structured SVM helps us doing that.

Structured Support Vector Machines

In SSVM we map all combination of input and output instances to a space of fixed dimensionality, called the feature map. Then instead of searching in the output space, we search in this feature space.
Components of SSVM:

- Feature / Compatibility Function : $\psi(x, y)$
- Weight Vector: W
- Discrimination / Scoring Function: $W^\top \psi(x, y)$
- Loss Function (could be optional): $\Delta(y, y_i)$

Structured Support Vector Machines

The form of the optimization equation is:

$$argmin_{W,\zeta} \quad \frac{1}{2}W^\top W + C\sum_{i=1}^{n}\zeta_i$$

$$s.t.\ \forall i\ \forall y \neq y_i \quad W^\top\psi(x_i, y_i) - W^\top\psi(x_i, y) >= \Delta(y_i, y) - \zeta_i$$

$$\forall i\ \zeta_i >= 0$$

Margin Rescaling: Since the margin is not constant.

Feature Map / Compatibility Function

Intuition: Somehow represents the compatibility between each input x and output y.

Typically is a combination of two specific function:

- Unary Potential/Compatibility Function: compatibility between local fragments of input and output.
- Binary Portential/Compatibility Function: compatibility between the local labels (output).

Discrimination Function

Scores the compatibility of a given input x and output y.

$$W^\top \psi(x, y)$$

We should define a good feature space under which we can tune W such that the score for correct labels becomes higher than the score for incorrect ones.

Loss Function

Gives us a measure of how different is an incorrect output from a correct output.

$$\Delta(y_i, y)$$

Can simply be the euclidean distance between the two given outputs.

Prediction Using SSVMs

$$\hat{y} = argmax_y W^\top \psi(x_i, y)$$

Intuition: in some sense we are finding an output that is most compatible with our input.

We need to have a good approach on maximizing the score by finding a good y. Iterating over all possible outputs is a waste of time. Having this algorithm highly depends on how we define the feature space and also on the domain itself.

Training SSVMs Using Cutting Plains Alg.

The form of the optimization equation is:

$$\frac{1}{2} W^\top W + C \sum_{i=1}^{n} \zeta_i$$

$$s.t. \;\; \forall i \, \forall y \neq y_i \qquad W^\top \psi(x_i, y_i) - W^\top \psi(x_i, y) >= \Delta(y_i, y) - \zeta_i$$

$$\forall i \; \zeta_i >= 0$$

Problem?

Training SSVMs Using Cutting Plains Alg.

The form of the optimization equation is:

$$\frac{1}{2}W^\top W + C\sum_{i=1}^{n}\zeta_i$$

$$s.t.\ \ \forall i\ \forall y \neq y_i \qquad W^\top\psi(x_i, y_i) - W^\top\psi(x_i, y) >= \Delta(y_i, y) - \zeta_i$$

$$\forall i\ \zeta_i >= 0$$

Problem? We have exponential number of constraints. It is inefficient to use all of them (if not impossible).

Solution: Start with empty set of constraints and iteratively add more constraints. Add the most violating constraint for each input data (Cutting Plains Algorithm). Another method is Gradient Descent.

Training SSVMs Using Cutting Plains Alg.

One algorithm is called Cutting Plains:

CUTTING_PLAINS()
  repeat
    K_old = K
      For i = 1 .. n
        $\hat{y} = argmax_{y \neq y_i} - (W^\top \psi(x_i, y_i) - W^\top \psi(x_i, y)) + \Delta(y_i, y)$
        If $W^\top \psi(x_i, y_i) - W^\top \psi(x_i, \hat{y}) < \Delta(y_i, \hat{y}) - \zeta_i$ Then

$$K = K \cup \{W^\top \psi(x_i, y_i) - W^\top \psi(x_i, \hat{y}) >= \Delta(y_i, \hat{y}) - \zeta_i\}$$

$$\{W, \zeta\} = argmin_{W,\zeta} \frac{1}{2} W^\top W + C \sum_{i=1}^{n} \zeta_i \quad s.t. \quad K$$

        End If
      End For
  Until K = K_old

Training SSVMs Using Cutting Plains Alg.

Sometimes coming up with a good algorithm for finding $\hat{y}$ in

$$\hat{y} = argmax_{y \neq y_i} - (W^\top \psi(x_i, y_i) - W^\top \psi(x_i, y)) + \Delta(y_i, y)$$

is difficult. In some domains, there might be a good algorithm (parsing sentences).

One approach is to not find the wrongest solution but to find a wrong solution. People have used MCMC around the correct solution to find a wrong solution.

Conditional Random Fields

CRF Review

Provides the following model:

$$P(y|x) = \frac{1}{Z} exp(f(x, y))$$

Where f is a feature function and Z is the normalization term. The feature function gives an score to the labeling y for a given x. Feature function can be handcrafted or can be learned through a process.

Conditional Random Fields take the **local relationship** between the fragments of input and output into consideration.

Semantic Segmentation Using CRFs and CNN Features

Methodology

They first group the pixels of their input image into superpixels. Then they form the conditional random field as:

$$P(y|x;w) = \frac{1}{Z} exp(-E(y,x;w))$$

where,

$$E(y,x;w) = \sum_{p \in N} \Psi^{(1)}(y^p, x; w) + \sum_{p,q \in S} \Psi^{(2)}(y^p, y^q, x; w)$$

N is the list of all superpixels, S is the list of all neighboring superpixels, y is a labeling, Ψ functions are the potential functions and (p,q) are a pair of neighboring superpixels. When the input and output are less compatible, energy is high.

Methodology

$$P(y|x;w) = \frac{1}{Z} exp(-E(y,x;w))$$

$$E(y,x;w) = \sum_{p \in N} \Psi^{(1)}(y^p, x; w) + \sum_{p,q \in S} \Psi^{(2)}(y^p, y^q, x; w)$$

Finding a best labeling with CRF and MAP: (1) learn model parameters, (2) infer a most likely label using a method. Thus segmentation problem is now reduced to minimizing the Energy function using the learned parameters as:

$$\hat{y} = argmin_{y \in Y} E(y,x;w)$$

Learning the Weights and Predict Using SSVM

We can learn the parameters of our CRF by creating and training the following Structured SVM:

$$min_{w,\zeta} \quad \frac{1}{2}w^\top w + C\sum_{i=1}^{n}\zeta_i$$

$$s.t. \;\; \forall i \, \forall y \neq y_i \in Y \qquad E(y, x_i; w) - E(y_i, x_i; w) >= \Delta(y_i, y) - \zeta_i$$

$$\forall i \; \zeta_i >= 0$$

Intuition: Energy of GT label lower than the energy of any incorrect label by at least a margin determined by the loss function. Loss function used is weighted Hamming loss. (fraction of wrong labels to the total number of labels)

Now we can learn the parameters using the cutting plains algorithm by iteratively adding the most violating constraint to the list of constraints.

The Potential Functions

They defined their potential functions as linear functions in the parameter w. Thus we will have:

$$\Psi^{(1)}(y^p, x; w) =< w^{(1)}, \psi^{(1)}(y^p, x) >$$

$$\Psi^{(2)}(y^p, y^q, x; w) =< w^{(2)}, \psi^{(2)}(y^p, y^q, x) >$$

where $<, >$ indicates the inner product and the vector w is made by stacking the two vectors $w^{(1)}$ and $w^{(2)}$.

The Potential Functions

Unary potential function:

$$\psi^{(1)}(y^p, x) = [I(y^p = 1){x^p}^\top \dots I(y^p = K){x^p}^\top]$$

where, I(.) is the indicator function, ${x^p}^\top$ is the deep learning feature vector for superpixel p, K is the number of classes.

The Potential Functions

Binary potential function:

$$\psi^{(2)}(y^p, y^q, x) = \psi_1^{(2)}(y^p, y^q, x) + \psi_2^{(2)}(y^p, y^q, x) + \psi_3^{(2)}(y^p, y^q, x) + \psi_4^{(2)}(y^p, y^q, x)$$

Where,

$$\psi_i^{(2)}(y^p, y^q, x) = L_{pq}.I(y^p \neq y^q).g_i(y^p, y^q)$$

where L_{pq} is either the shared boundary length between the two superpixels or inversed color difference and g_i responds to the inverse of the frequency of co-occurrences of the two labels in four spatially related neighboring. (Next Slide)

The Potential Functions

$$g_i(y^p, y^q) = \frac{1}{f^i_{co-occurance}(y^p, y^q)}$$

$$f^i_{co-occurance}(y^p, y^q) = \frac{N^i_{pq}}{N_{pq}}$$

where N_{pq} is the number of training images in which y^p and y^q coexist, N^i_{pq} are the number of training images in which y^p and y^q appear in four spatially related neighborings.

They have a hyper-parameter that determines the effect of co-occurrence function g on the binary potential.

Results

Fig. 2. Segmentation examples on Weizmann horse. 1st row: test images; 2nd row: ground truth; 3rd row: segmentation results produced by SSVM based CRF learning with bag-of-words feature; 4th row: segmentation results produced by SSVM based CRF learning with unsupervised feature learning; 5th row: segmentation results produced by SSVM based CRF learning with the 6th layer CNN features.

Figure: Effect of choosing different features for $x^{p^\top}$

Results

Fig. 4. Segmentation examples on MSRC. 1st row: test images; 2nd row: ground truth; 3rd row: segmentation results produced by SSVM based CRF learning with bag-of-words feature; 4th row: segmentation results produced by SSVM based CRF learning with unsupervised feature learning; 5th row: segmentation results produced by our method with co-occurrence pairwise potentials.

Figure: Effect of choosing different features for ${x^p}^\top$

Results

Average Per-Category pixel accuracy on MSRC-21 and Stanford Background datasets are 90.5% and 76.9%.

pixel accuracy : $\frac{TP+TN}{TP+TN+FP+FN}$

Results

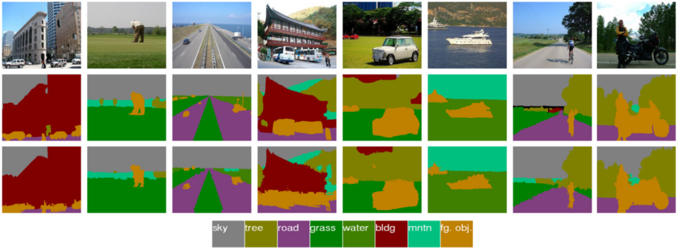

Fig. 5. Segmentation examples on the Stanford Background dataset. 1st row: test images; 2nd row: ground truth; 3rd row: segmentation results produced by our method with co-occurrence pairwise potentials.

Figure: Segmentation Example on Stanford Background dataset

The End