Loss Functions for Deep Structured Models

Presenter: Jack Lanchantin

University of Virginia
https://qdata.github.io/deep2Read/

12/12/18

Outline

SVM Losses

SVM Loss for Binary Classification

$D = \{(\mathrm{x}_i, y_i)\}_{i=1}^{n}$, $\mathrm{x}_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$

$$\mathcal{L}(\mathrm{w}) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \langle \mathrm{w}, \mathrm{x}_i \rangle)$$

If $y_i = 1$, $\langle \mathrm{w}, \mathrm{x}_i \rangle$ should be ≥ 1
If $y_i = -1$, $\langle \mathrm{w}, \mathrm{x}_i \rangle$ should be ≤ -1
both of these result in $1 - 1 = 0$, giving an SVM loss of 0.

SVM Loss for Multi-Class Classification

$$D = \{(\mathrm{x}_i, y_i)\}_{i=1}^{n},\ \mathrm{x}_i \in \mathbb{R}^d,\ y_i \in \{0, 1, 2, ..., c\}$$

$$\mathcal{L}(\mathrm{w}) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 + \max_{t \neq y_i} \langle \mathrm{w}_t, \mathrm{x}_i \rangle - \langle \mathrm{w}_{y_i}, \mathrm{x}_i \rangle)$$

This maximizes the margin between the true label y_i and the next biggest label's prediction ($\max_{t \neq y_i}$).

Alternatively, we could push down *all* labels which aren't the true label (as opposed to only the next max):

$$\mathcal{L}(\mathrm{w}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t \neq y_i} \max(0, 1 + \langle \mathrm{w}_t, \mathrm{x}_i \rangle - \langle \mathrm{w}_{y_i}, \mathrm{x}_i \rangle)$$

Structured SVM for Multi-label Classification

$D = \{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^{n}$, $\mathrm{x}_i \in \mathbb{R}^d$, $\mathrm{y}_i \in \{0, 1\}^L$, where L the number of labels. $\hat{\mathrm{y}}$ is the predicted output set, and Δ is hamming loss

$$\mathcal{L}(\mathrm{w}) = \frac{1}{n} \sum_{i=1}^{n} \max(0, \max_{\hat{\mathrm{y}}}(\Delta(\mathrm{y}, \hat{\mathrm{y}}) + \langle \mathrm{w}, \phi(\mathrm{x}_i, \hat{\mathrm{y}}) \rangle) - \langle \mathrm{w}, \phi(\mathrm{x}_i, \mathrm{y}) \rangle)$$

Finding this $\max_{\hat{\mathrm{y}}}$ is typically intractable, but gradient descend can be used to approximate.

SPEN and SPEN InfNet

Energy Models

Instead of maximizing the score of $\langle \mathrm{w}, \phi(\mathrm{x}_i, \hat{\mathrm{y}})\rangle$, we can instead minimize the energy of the joint pair (x, y)

$$\min_{\mathrm{y}} \mathrm{E}(\mathrm{x}, \mathrm{y}) \quad \text{subject to} \quad \mathrm{y} \in \{0, 1\}^L$$

This could be rendered tractable by assuming certain structure (e.g., a tree) for the energy function $\mathrm{E}(\mathrm{x}, \mathrm{y})$. Instead, we consider a general $\mathrm{E}(\mathrm{x}, \mathrm{y})$, but optimize over a convex relaxation of the constraint set

$$\min_{\mathrm{y}} \mathrm{E}(\mathrm{x}, \mathrm{y}) \quad \text{subject to} \quad \mathrm{y} \in [0, 1]^L$$

Structured Prediction Energy Networks (SPEN)

The energy of a pair (x, y) is represented as:

$$\mathrm{E}(\mathrm{x}, \mathrm{y}) = \mathrm{E}(\mathrm{x}, \mathrm{y})^{local} + \mathrm{E}(\mathrm{x}, \mathrm{y})^{label}$$

$$\mathrm{E}(\mathrm{x}, \mathrm{y})^{local} = \sum_{i=1}^{L} y_i b_i^\top F(\mathrm{x})$$

$$\mathrm{E}(\mathrm{x}, \mathrm{y})^{label} = c_2^\top \sigma(C_1 \mathrm{y})$$

Structured Prediction Energy Networks (SPEN)

If we were to use the CRF framework in SPENs, the label, or global energy network would be

$$\mathrm{E}(\mathrm{x}, \mathrm{y})^{label} = \mathrm{y}^\top S_1 \mathrm{y} + s \top \mathrm{y}$$

which doesn't use the $[0, 1]^L$ relaxation on y and only considers pairwise dependencies as opposed to the C_1 matrix in the original label network, which allows for any learned dependencies.

Structured Prediction Energy Networks (SPEN)

SPENs minimize the following SSVM Loss, where $[\cdot]_+$ represents $\max(0, \cdot)$

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \Big[\max_{\hat{y}} (\Delta(y_i, \hat{y}) - \mathrm{E}(x_i, \hat{y}) + \mathrm{E}(x_i, y_i)) \Big]_+$$

During training, the $\max_{\hat{y}}$ is found by *cost-augmented inference*, where $\mathcal{P}$ projects values into [0,1]:

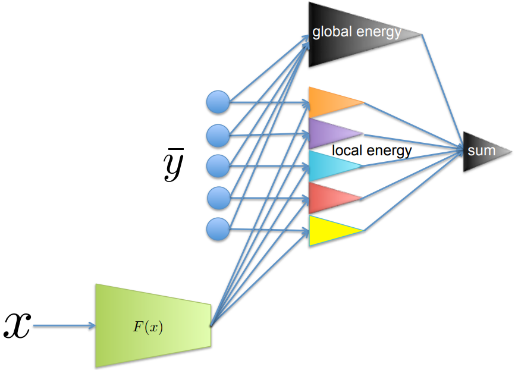
$$y^{t+1} = \mathcal{P}\Big(y^t - \eta \frac{\mathrm{d}}{\mathrm{d}y} (-\Delta(y_i, y^t) + \mathrm{E}(x, y^t)) \Big) \tag{1}$$

Structured Prediction Energy Networks (SPEN)

At test time, given x_i, $\hat{\mathsf{y}}$ is found by minimizing the energy $\mathrm{E}(\mathsf{x}_i, \hat{\mathsf{y}})$.
This is done via gradient descent:

$$\mathsf{y}^{t+1} = \mathcal{P}\Big(\mathsf{y}^t - \eta \frac{\mathrm{d}}{\mathrm{d}\mathsf{y}} \mathrm{E}(\mathsf{x}, \mathsf{y}^t))\Big)$$

SPEN Model



Learning Approximate Inference Networks for Structured Prediction (SPEN InfNet)

Instead of using (13), learn an *inference network* G_Ψ with the goal that

$$G_\Psi(\mathrm{x}) \approx \arg\min_{\hat{\mathrm{y}}} E(\mathrm{x}, \hat{\mathrm{y}})$$

Given energy function E, we seek to minimize the following:

$$\arg\min_{\Psi} E(\mathrm{x}, G_\Psi(\mathrm{x}))$$

Learning Approximate Inference Networks for Structured Prediction (SPEN InfNet)

Training InfNet requires two steps: InfNet turns the loss into a minimax problem to learn a cost augmented inference network G_Φ

$$\min_{\Theta} \max_{\Phi} \frac{1}{n} \sum_{i=1}^{n} \left[\Delta(\mathsf{y}_i, G_\Phi(\mathsf{x}_i)) - \mathrm{E}_\Theta(\mathsf{x}_i, G_\Phi(\mathsf{x}_i)) + \mathrm{E}_\Theta(\mathsf{x}_i, \mathsf{y}_i) \right]_+$$

$$\hat{\Theta} \leftarrow \arg\min_{\Theta} \left[\Delta(\mathsf{y}_i, G_\Phi(\mathsf{x}_i)) - \mathrm{E}_\Theta(\mathsf{x}_i, G_\Phi(\mathsf{x}_i)) + \mathrm{E}_\Theta(\mathsf{x}_i, \mathsf{y}_i) \right]_+$$

$$\hat{\Phi} \leftarrow \arg\min_{\Phi} \left[- \Delta(\mathsf{y}_i, G_\Phi(\mathsf{x}_i)) + \mathrm{E}_\Theta(\mathsf{x}_i, G_\Phi(\mathsf{x}_i)) - \mathrm{E}_\Theta(\mathsf{x}_i, \mathsf{y}_i) \right]_+$$

Learning Approximate Inference Networks for Structured Prediction (SPEN InfNet)

Training only gives us a cost-augmented inference network G_Φ, but we want inference network G_Ψ. So we initialize G_Ψ with G_Φ and minimize the original Energy:

$$\arg\min_{\Psi} E(\mathrm{x}, G_\Psi(\mathrm{x}))$$

Deep Value Networks

Deep Value Networks

Train a value function $v(\mathrm{x}, \hat{\mathrm{y}}; \theta)$ that approximates an oracle function v^* (dependent on task):

$$v(\mathrm{x}, \hat{\mathrm{y}}; \theta) \approx v^*(\mathrm{y}, \hat{\mathrm{y}})$$

For MLC,

$$v^*(\mathrm{y}, \hat{\mathrm{y}}) = v_{F1}(\mathrm{y}, \hat{\mathrm{y}}) = \frac{2(\mathrm{y} \cap \hat{\mathrm{y}})}{(\mathrm{y} \cap \hat{\mathrm{y}}) + (\mathrm{y} \cup \hat{\mathrm{y}})}$$

At inference time, $\hat{\mathrm{y}}$ is found by initializing to 0 vector $\hat{\mathrm{y}}^0 = [0]^L$ and then updating via gradient ascent on $v(\mathrm{x}, \hat{\mathrm{y}}^t; \theta)$

$$\hat{\mathrm{y}}^{t+1} = \mathcal{P}\big(\hat{\mathrm{y}}^t + \eta \frac{\mathrm{d}}{\mathrm{d}\hat{\mathrm{y}}} v(\mathrm{x}, \hat{\mathrm{y}}^t; \theta)\big)$$

Deep Value Networks

Our training objective aims at minimizing the discrepancy between $v(\mathrm{x}, \hat{\mathrm{y}}; \theta)$ and $v^*(\mathrm{y}, \hat{\mathrm{y}})$ on a training set of triplets (input, output, value) denoted $D = \{\mathrm{x}^i, \mathrm{y}^i, v^{*i}\}_{i=1}^N$

This can be done using binary cross entropy between $v(\mathrm{x}, \hat{\mathrm{y}}; \theta)$ and $v^*(\mathrm{y}, \hat{\mathrm{y}})$:

$$\mathcal{L} = \sum_{i=1}^{n} -v^*(\mathrm{y}_i, \hat{\mathrm{y}}) \log(v(\mathrm{x}_i, \hat{\mathrm{y}}; \theta)) - (1 - v^*(\mathrm{y}_i, \hat{\mathrm{y}})) \log(1 - v(\mathrm{x}_i, \hat{\mathrm{y}}; \theta))$$

Deep Value Networks

Choosing $\hat{y}$ can be done in several ways:

- **Ground Truth**: $\hat{y} = y^i$ and $v * (\hat{y}, y^i) = 1$
- **Inference**: Gradient ascent on the current v^*
- **Adversarial Samples**: Maximize the cross entropy loss using gradient ascent

Other

Adversarial Training for Segmentation

$$\hat{\Theta} \leftarrow \arg\min_{\Theta} \Big(\ell_{BCE}(D(\mathrm{x}_i, G_\Phi(\mathrm{x}_i), 0) + \ell_{BCE} D(\mathrm{x}_i, \mathrm{y}_i), 1) \Big)$$

$$\hat{\Phi} \leftarrow \arg\min_{\Phi} \Big(-\ell_{BCE}(D(\mathrm{x}_i, G_\Phi(\mathrm{x}_i), 0) + \ell_{avgBCE}(\mathrm{y}, G_\Phi(\mathrm{x}_i)) \Big)$$

Other Loss Functions

N samples, L labels. True label y_j^i and predicted label $\hat{y}_j^i$ for sample i and label j.

BCE

$$\mathcal{L}_{BCE} = \sum_{i=1}^{N} \sum_{j=1}^{L} -\left(y_j^i \log \hat{y}_j^i + (1 - y_j^i) \log(1 - \hat{y}_j^i)\right) \tag{2}$$

L2

$$\mathcal{L}_{L2} = \sum_{i=1}^{N} \sum_{j=1}^{L} (y_j^i - \hat{y}_j^i)^2 \tag{3}$$

KL

$$\mathcal{L}_{KL} = \sum_{i=1}^{N} \sum_{j=1}^{L} y_j^i \log \left(\frac{y_j^i}{\hat{y}_j^i}\right) \tag{4}$$