Aim: Principled Algorithm for the Optimization of Piecewise Linear CNNs

Piecewise Linear CNN:

![Example of Piecewise Linear CNN architecture](image)

![Max-Pooling function](image)

Large class of CNNs, includes Batch Normalization if statistics are fixed. Strong structure to be exploited in optimization algorithms.

Learning Objective:

- Network with weights $W$ outputs feature vector $\Phi(x_i, W)$ for input $x_i$.
- For label $y_i$ in label set $\mathcal{Y}$, linear classifier $W^\text{Em}$ scores $\Phi(x_i, W)$.
- Hinge loss upper bound on $\Delta$ (zero-one classification).

$$\min_{W,\Delta} \frac{1}{N} \sum_{i=1}^{N} \max (\Delta(y_i, \Phi(x_i, W)))$$

Objective Function of Latent SVM:

$$\min_{W,\Delta} \frac{1}{2} ||W||^2 + \frac{1}{N} \sum_{i=1}^{N} \max (\Delta(y_i, \Phi(x_i, W)))$$

Related Works: Latent SVMs

Concave-Convex Procedure:

- For a Difference of Convex (DC) function: iteratively minimize convex upper bound.
- Monotonic decrease at each iteration.

Block-Co-ordinate Frank Wolfe Algorithm:

- Conditional gradient: Direction to minimize linearized objective function.
- Block-coordinate version: Direction search within blocks of coordinates.
- Optimal step-size: In closed-form for SVMs thanks to:
  - $i$: quadratic objective function,
  - $ii$: decomposability of dual.

Layer-wise Optimization

Difference of Convex Objective Function:

Learning objective is a DC function of $W$ (quadratic regularization + piecewise linear loss).

Relationship with Latent SVMs:

Explicit DC-decomposition reveals a latent SVM:

$$\min_{W} \frac{1}{2} ||W||^2 + \frac{1}{N} \sum_{i=1}^{N} \max (\Delta(y_i, \Phi(x_i, W))) - \max_{h \in \mathcal{H}} ((W^\text{Em})^T \Phi(x_i, y_i, h))$$

Latent Space & DC Networks:

- Selection of activations: “path” in network.
- Given an input $x_i$ one-to-one correspondence between a path and a feature vector $\Phi(x_i, y_i, h)$.
- Feature vectors computed with DC networks.

Improvements on BCFW:

- Memory Requirements:
  - $\text{I}c\text{ue}$: Storing one primal variable per block of coordinates (each of same size as $W^\text{Em}$).
  - Unfeasible for dense layers.
  - Solution: with $\lambda$ input of layer $l$, and $H$ hinge loss:

$$\Delta \approx W^\text{Em} \cdot \chi^l$$

- Warm Start with a Proximal Term:
  - $\text{I}c\text{ue}$: Initialization in the dual sets initial primal variable to zero.
  - Solution: Add proximal term $\frac{\lambda}{2} ||W^\text{Em} - W^\text{Em}||^2$ to objective function.
  - $\text{KKT}$ conditions give initial $W^\text{Em}$ instead of $0$.

- Reduction of Search Space:
  - $\text{I}c\text{ue}$: Size of latent search space $H$ grows exponentially with depth.
  - Solution: Solve problem on increasingly large subsets of $H$, stop when dual gap is below $\varepsilon$.
  - $\text{I}n\text{it}i\text{on}$: Very large feature vectors if very different paths of activations.
  - $\rightarrow$ Paths of convex and concave parts must be similar at optimality.
  - $\rightarrow$ Small search space for convex part (similar to concave part).

Experiments

- **Setup:**
  - Test ability of LWSVM (Layer-wise SVM) to improve on SGD.
  - Comparison with Adagrad, Adam, and Adam.
  - Pre-train with SGD solver and tune with LW-SVM, compare final performances.
  - Experiments on GPU single Nvidia Titan X with Theano.

- **Schedule:**
  - New pass on every layer (from SVM to first layer) until no improvement on validation accuracy.
  - On every layer, one iteration of CCCP: linearize concave part and solve SVM problem until no improvement (by more than 1%) on dual objective function.

MNIST:

- [Image](image)

CIFAR-10 & CIFAR-100:

- [Image](image)

ImageNet (ILSVRC 2012):

- **Specific Setup:**
  - Use publicly available pre-trained VGG-16 and tune fully connected layers.
  - Same regularization $\lambda$ as in the original model and $\mu = 10.\lambda$
  - Fixed budget of five epochs per layer (2 days of training).

- **Results:**
  - Validation Top-1 Error: 26.70% to 26.19%
  - Validation Top-5 Error: 8.67% to 8.37%

Discussion:

- LWSVM systematically improves on performance of SGD-based solvers.
- Best testing result on each data set.
- Proximal term helps better generalization (more regularization).
- Optimal step size, no learning rate to tune (and line search is infeasible!).
- Robust hyper-parameters.
- Scales to large data and large model settings.