Convexified Convolutional Neural Networks

Kyle Daruwalla and Akhil Sundararajan

ECE 901 Fall 2016

November 6, 2016

Overview

Background

Convex Relaxations
- Linear Activation Functions
- Non-Linear Activation Functions

Algorithm

Theoretical Results

Experimental Results

Paper Overview

1. Start generic two-layer CNN
2. Convex relaxation
   2.1 Linear activation – optimize for a low-rank matrix A instead of filter weights and coefficients
   2.2 Non-linear activation – frame problem in terms of RKHS
3. Introduce a kernel-based algorithm for CCNNs
4. Provide theoretical guarantees on the generalization error
5. Explain extensions like pooling and multi-layer CNNs
6. Provide experimental results on MNIST and CIFAR-10

Convolutional Neural Networks

For an input vector, $x \in \mathbb{R}^{d_0}$, and output vector, $y \in \mathbb{R}^{d_2}$, define

$$\{z_p(x) \mid z_p(x) \in \mathbb{R}^{d_1}\}_{j=1}^{P}$$

to be the set of P patches of x.
For a given $\sigma : \mathbb{R} \mapsto \mathbb{R}$, the output of a filter is

$$h_j(z) = \sigma(w_j^T z)$$

The output of a CNN is $f = (f_1(x), f_2(x), \ldots, f_{d_2}(x))$ is defined as

$$f_k(x) = \sum_{j=1}^{r} \sum_{p=1}^{P} \alpha_{k,j,p} h_j(z_p(x)) \tag{1}$$

Convolutional Neural Networks

CNNs are described by the class of models:

$$\mathcal{F}_{\text{cnn}}(B_1, B_2) = \{f \text{ of the form Eq. } 1 \mid \max_{j \in [r]} \|w_j\|_2 \leq B_1 \quad \text{and} \quad \max_{k \in [d_2], j \in [r]} \|\alpha_{k,j}\|_2 \leq B_2\} \tag{2}$$

Given a set of samples $\{(x_i, y_i)\}_{i=1}^n$, we want to solve the ERM:

$$\hat{f}_{\text{cnn}} = \arg\min_{f \in \mathcal{F}_{\text{cnn}}} \sum_{i=1}^n \mathcal{L}(f(x_i), y_i) \tag{3}$$

Optimizing For Low-Rank Matrix (Linear Activation)

For $x \in \mathbb{R}^{d_0}$, define

$$Z(x) = \begin{bmatrix} z_1(x)^T \\ \vdots \\ z_P(x)^T \end{bmatrix} \text{ and } \alpha_{k,j} = \begin{bmatrix} \alpha_{k,j,1} \\ \vdots \\ \alpha_{k,j,P} \end{bmatrix}$$

Then rewrite Eq. 1 with activation function, $\sigma(t) = t$, as

$$f_k(x) = \sum_{j=1}^{r} \alpha_{k,j}^T Z(x) w_j = \operatorname{tr}\left(Z(x) \sum_{j=1}^{r} w_j \alpha_{k,j}^T \right) = \operatorname{tr}\left(Z(x) A_k\right) \tag{4}$$

Optimizing For Low-Rank Matrix (Linear Activation)

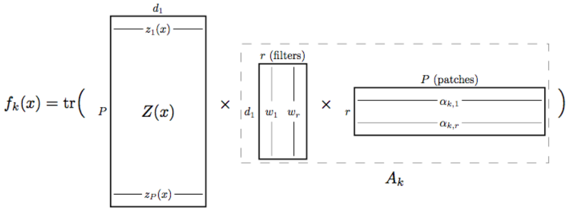

Figure: Reframing the problem in terms of a low-rank matrix, A_k, allows for convex optimization over a nuclear norm ball

Optimizing For Low-Rank Matrix (Linear Activation)

The CNN class of models is then

$$\begin{aligned}\mathcal{F}_{\mathrm{cnn}}(B_1, B_2) = \{f \text{ of the form Eq. } 4 \mid & \max_{j\in[r]} \|w_j\|_2 \le B_1 \\ & \text{and} \max_{k\in[d_2], j\in[r]} \|\alpha_{k,j}\|_2 \le B_2\} \\ & \text{and } \operatorname{rank}(A) = r\end{aligned} \tag{5}$$

Define the CCNN class of models as

$$\mathcal{F}_{\mathrm{ccnn}}(B_1, B_2) = \{f \text{ of the form Eq. } 4 \mid \|A\|_* \le B_1 B_2 r \sqrt{d_2}\} \tag{6}$$

where $\mathcal{F}_{\mathrm{cnn}}(B_1, B_2) \subseteq \mathcal{F}_{\mathrm{ccnn}}(B_1, B_2)$.

Reproducing-Kernel Hilbert Space (RKHS)

Background on RKHS and representer theorem

Framing Problem Using RKHS

Framing problem using RKHS

CCNN Algorithm

Algorithm 1 CCNN Algorithm

Require: Data $\{(x_i, y_i)\}_{i=1}^n$, kernel function $\mathcal{K}$, regularization parameter $R > 0$, number of filters r

1. Construct a matrix $K \in \mathbb{R}^{nP \times nP}$ such that the entry at column (i, p) and row (i', p') is $\mathcal{K}(z_p(x_i), z_{p'}(x_{i'}))$. Compute the factorization $K = QQ^T$ or an approximation, $K \approx QQ^T$, where $Q \in \mathbb{R}^{nP \times m}$.
2. For each x_i, construct a patch matrix $Z(x_i) \in \mathbb{R}^{P \times m}$ whose p-th row is the (i, p)-th row of Q.
3. Solve the following optimization problem to obtain a matrix $\hat{A} = (\hat{A}_1, \ldots, \hat{A}_{d_2})$

$$\hat{A} = \arg \min_{\|A\|_* \leq R} \tilde{\mathcal{L}}(A) \text{ where } \tilde{\mathcal{L}}(A) = \sum_{i=1}^{n} \mathcal{L}\left((\mathrm{tr}\left(Z(x_i)A_1\right), \ldots, \mathrm{tr}\left(Z(x_i)A_{d_2}\right)), y_i\right)$$

4. Compute a rank-r approximation $\tilde{A} \approx \hat{U}\hat{V}^T$ where $\hat{U} \in \mathbb{R}^{m \times r}$ and $\hat{V} \in \mathbb{R}^{Pd_2 \times r}$.
5. **return** The predictor $\hat{f}_{\text{ccnn}}(x) = \left(\mathrm{tr}\left(Z(x)\hat{A}_1\right), \ldots, \mathrm{tr}\left(Z(x)\hat{A}^{d_2}\right)\right)$ and the convolutional layer output $H(x) = \hat{U}^T(Z(x))^T$.

Solving ERM in CCNN Algorithm

Details on solving minimization problem embedded in algorithm

Bounds on Generalization Error

Theorem 1

Proof of Theorem 1

Proof of theorem 1

Experimental Results

On MNIST dataset:

	basic	rand	rot	img	img+rot
SVM_{rbf}	3.03%	14.58%	**11.11%**	22.61%	55.18%
NN-1	4.69%	20.04%	18.11%	27.41%	62.16%
CNN-1 (ReLU)	3.37%	9.83%	18.84%	14.23%	45.96%
CCNN-1	**2.38%**	**7.45%**	13.39%	**10.40%**	**42.28%**
TIRBM	-	-	**4.20%**	-	35.50%
SDAE-3	2.84%	10.30%	9.53%	16.68%	43.76%
ScatNet-2	1.27%	12.30%	7.48%	18.40%	50.48%
PCANet-2	**1.06%**	6.19%	7.37%	10.95%	35.48%
CNN-2 (ReLU)	2.11%	5.64%	8.27%	10.17%	32.42%
CNN-2 (Quad)	1.75%	5.30%	8.83%	11.60%	36.90%
CCNN-2	1.38%	**4.32%**	6.98%	**7.46%**	**30.23%**

Table: Classification error with a Gaussian kernel for CCNNs

Experimental Results

On CIFAR-10 dataset:

	Error Rate
CNN-1	34.14%
CCNN-1	**23.62%**
CNN-2	24.98%
CCNN-2	**20.52%**
$\text{SVM}_{\mathit{Fastfood}}$	36.90%
PCANet-2	22.86%
CKN	21.70%
CNN-3	21.48%
CCNN-3	**19.56%**

Table: Error rate with a Gaussian kernel for CCNNs

References

Y. Zhang, et al. (2016, Sept. 4). *Convexified Convolutional Neural Networks* (v1) [Online]. Available: https://arxiv.org/abs/1609.01000

The End