Convexified Convolutional Neural Networks

Kyle Daruwalla and Akhil Sundararajan

ECE 901 Fall 2016

November 10, 2016

Overview

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:

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

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

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

Optimizing For Low-Rank Matrix (Linear Activation)

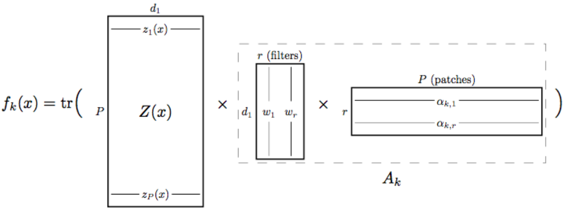


Figure: Reframing the problem in terms of a low-rank matrix, A_k, allows for convex optimization over a nuclear norm ball

Optimizing For Low-Rank Matrix (Linear Activation)

The CNN class of models is then

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

Define the CCNN class of models as

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

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

Reproducing Kernel Hilbert Space (RKHS)

- Hilbert Space: a complete inner-product space
- a reproducing kernel Hilbert space is
  - a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$
  - evaluation functionals are continuous
- reproducing kernel (associated with Hilbert Space $\mathcal{H}$): a function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that
  - $\mathcal{K}(x, \cdot) = \mathcal{K}_x(\cdot) \in \mathcal{H}, \forall x \in \mathcal{X}$
  - $\langle f, \mathcal{K}_x \rangle_{\mathcal{H}} = f(x), \forall f \in \mathcal{H}$ (reproducing property)
-
  - $\mathcal{H}$ is an RKHS $\implies$ $\exists$ unique reproducing kernel $\mathcal{K}$
  - If $\exists$ reproducing kernel $\mathcal{K}$ $\implies$ $\mathcal{H}$ is an RKHS

Framing Problem Using RKHS

- for certain kernels $\mathcal{K}$ and activations σ, Representer Theorem implies that for any patch $z_p(x_i)$

$$h(z_p(x_i)) = \sum_{(i',p')\in[n]\times[p]} c_{i',p'} k(z_p(x_i), z_{p'}(x_{i'}))$$

- choose kernel matrix $K = \mathbb{R}^{nP\times nP}$
- consider $K = QQ^\top$

$$h(z_p(x_i)) = \langle Q_{(i,p)}, w\rangle \text{ where } w := \sum_{(i',p')} c_{(i',p')} Q_{(i',p')}$$

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 $\hat{A} = (\hat{A}_1, \ldots, \hat{A}_{d_2})$

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

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 $\hat{f}_{\text{ccnn}}(x) = \left(\operatorname{tr}\left(Z(x)\hat{A}_1\right), \ldots, \operatorname{tr}\left(Z(x)\hat{A}^{d_2}\right)\right)$ and the convolutional layer output $H(x) = \hat{U}^T(Z(x))^T$.

Solving ERM in CCNN Algorithm

- projected gradient descent

$$A^{t+1} = \Pi_R(A^t - \eta^t \nabla_A \tilde{\mathcal{L}}(A^t))$$

- compute SVD of A, the project singular values onto l_1 ball (Duchi et al)
- proximal adaptive gradient method
- proximal SVRG

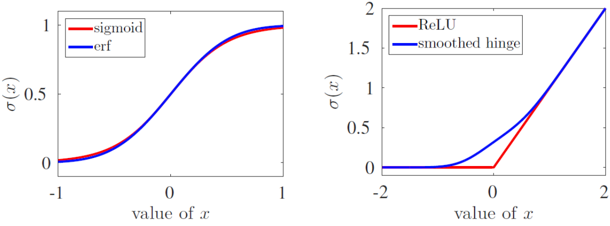
Choice of Kernel

- kernel functions considered must satisfy notion of richness

$$\mathcal{K}(z, z') = \frac{1}{2 - \langle z, z' \rangle}, \quad ||z||_2 \leq 1, ||z'||_2 \leq 1$$
$$\mathcal{K}(z, z') = exp(-\gamma ||z - z'||_2^2), \quad |z||_2 = ||z'||_2 = 1, \gamma \geq 0$$

Valid Activation Functions

- arbitrary polynomial functions
- $\sigma(t) = \sin(t)$
- $\sigma_{erf}(t) = \frac{2}{\sqrt{pi}} \int_0^t e^{-z^2} dz$
- $\sigma_{sh}(t) = \frac{1}{2} \int_{-\infty}^{t} (\sigma_{erf}(z) + 1) dz$



(a) sigmoid v.s. erf (b) ReLU v.s. smoothed hinge loss

Figure: approximations to activations which are not smooth enough

Theorem 1: Bound on Generalization Error

- loss function $\mathcal{L}(;y)$ is L-Lipschitz continuous for every $y \in [d_2]$
- $\mathcal{K}$ is the inverse polynomial kernel or the Gaussian RBF kernel
- valid activation function σ
- $c > 0$
- radius $R := C_\sigma(B_1)B_2r$

$\exists C_\sigma(B_1)$ such that

$$\mathbb{E}_{X,Y}[\mathcal{L}(\hat{f}_{ccnn}(X);Y)] \leq$$

$$\inf_{f\in\mathcal{F}_{cnn}} \mathbb{E}_{X,Y}[\mathcal{L}(f(X);Y)] + \frac{cLC_\sigma(B_1)B_2r\sqrt{log(nP)\mathbb{E}_X[||K(X)||_2]}}{\sqrt{n}}$$

Proof Sketch

1. consider a relaxed function class

$$\mathcal{F}_{ccnn} := \left\{ x \mapsto \sum_{j=1}^{r^*} \sum_{p=1}^{P} \alpha_{j,p} h_j(z_p(x)) : r^* < \infty \right.$$
$$\left. \text{and } \sum_{j=1}^{r^*} ||\alpha_j||_2 ||h_j||_{\mathcal{H}} \leq C_\sigma(B_1) B_2 d_2 \right\}$$

2. characterize Rademacher complexity of $\mathcal{F}_{ccnn}$ to upper bound generalization error of $\hat{f}_{ccnn}$

Further Proof Details (1/4)

Lemma 1:

$$\text{For any valid } \sigma(\cdot),\ \exists C_\sigma(B_1) \text{ s.t. } \mathcal{F}_{cnn} \subset \mathcal{F}_{ccnn}$$

Lemma 4:

With CCNN hyper-parameter $R = C_\sigma(B_1)B_2 d_2$,

$\hat{f}_{ccnn}$ is guaranteed to satisfy $\hat{f}_{ccnn} \in \arg\min_{f \in \mathcal{F}_{ccnn}} \sum_{i=1}^{n} \mathcal{L}(f(x_i); y_i)$

Further Proof Details (2/4)

the Rademacher complexity of $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$ with respect to n i.i.d. samples $\{X_i\}_{i=1}^n$ is given by

$$\mathcal{R}_n(\mathcal{F}) := \mathbb{E}_{X,\epsilon}\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(X_i)\right]$$

where $\{\epsilon_i\}_{i=1}^n$ are an i.i.d. sequence of uniform $\{-1, +1\}$-valued random variables

Further Proof Details (3/4)

Lemma 5:
$\exists$ universal constant c such that

$$\mathcal{R}_n(\mathcal{F}_{ccnn}) \leq \frac{cC_\sigma(B_1)B_2 r\sqrt{log(nP)\mathbb{E}[||K(X)||_2]}}{\sqrt{n}}$$

generalization bound:

$$\mathbb{E}[\mathcal{L}(\mathcal{F}_{ccnn}(X); Y)] \leq \inf_{f\in\mathcal{F}_{ccnn}} \mathbb{E}[\mathcal{L}(f(x); y)] + 2L\mathcal{R}_n(\mathcal{F}_{ccnn}) + \frac{c}{\sqrt{n}}$$

Further Proof Details (4/4)

By Lemma 3,

$$\inf_{f \in \mathcal{F}_{ccnn}} \mathbb{E}[\mathcal{L}(f(x); y)] \leq \inf_{f \in \mathcal{F}_{cnn}} \mathbb{E}[\mathcal{L}(f(x); y)]$$

It follows that

$$\mathbb{E}_{X,Y}[\mathcal{L}(\hat{f}_{ccnn}(X); Y)] \leq \inf_{f \in \mathcal{F}_{cnn}} \mathbb{E}_{X,Y}[\mathcal{L}(f(X); Y)] + \frac{cLC_{\sigma}(B_1)B_2 r \sqrt{log(nP)\mathbb{E}_X[||K(X)||_2]}}{\sqrt{n}}$$

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