Incremental Classifier and Representation Learning

Christoph Lampert

Workshop at Genova
March 9–10, 2017
Continuously improving open-ended learning

Life long learner/agent

Task 1

Task 2

Task 3

Lifelong Learning
A few years after the deep learning revolution...
A few years after the deep learning revolution...
Input: a stream of data, examples of different classes occur at different times,
Output: at any time, a competitive multi-class classifier for the classes observed so far,
Conditions: computational requirements and memory footprint remain bounded (or at least grow very slowly) with respect to the number of classes seen so far.
**Input:** a stream of data, examples of different classes occur at different times,

**Output:** at any time, a competitive multi-class classifier for the classes observed so far,

**Conditions:** computational requirements and memory footprint remain bounded (or at least grow very slowly) with respect to the number of classes seen so far.
Class-Incremental Learning

- feature function \( \varphi : \mathcal{X} \rightarrow \mathbb{R}^d \)
- classifiers: \( g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}} \) for each \( y \in \mathcal{Y} \) seen so far
- data comes in class batches: \( X^s, \ldots, X^t \) where all examples in \( X^y \) are of class \( y \)
Class-Incremental Learning

- feature function $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$
- classifiers: $g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \ldots, X^t$ where all examples in $X^y$ are of class $y$

**Idea 1:** incremental multi-class training, e.g., using stochastic gradient descent:

| $x$ | \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{cat1.png} \end{tabular} & \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{cat2.png} \end{tabular} & \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{cat3.png} \end{tabular} & \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{cat4.png} \end{tabular} & \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{dog1.png} \end{tabular} & \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{dog2.png} \end{tabular} & \begin{tabular}{c} \includegraphics[width=0.025\textwidth]{dog3.png} \end{tabular} |
| \hline
| $w_{\text{cat}}$ | 1 & 1 & -1 & 1 & -1 & -1 & 1 |
| $w_{\text{dog}}$ | -1 & -1 & 1 & -1 & 1 & 1 & -1 |

- after being trained on a set of classes, classifier parameters make sense
Class-Incremental Learning

- feature function \( \varphi : \mathcal{X} \to \mathbb{R}^d \)
- classifiers: \( g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}} \) for each \( y \in \mathcal{Y} \) seen so far
- data comes in class batches: \( X^s, \ldots, X^t \) where all examples in \( X^y \) are of class \( y \)

**Idea 1**: incremental multi-class training, e.g., using stochastic gradient descent:

<table>
<thead>
<tr>
<th>( x )</th>
<th>cat</th>
<th>dog</th>
<th>boat</th>
<th>truck</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_{\text{cat}} )</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>( w_{\text{dog}} )</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( w_{\text{boat}} )</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( w_{\text{truck}} )</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- after being trained on a set of classes, classifier parameters make sense
- every later sample is a negative example for earlier classes \( \rightarrow \) parameters \( w_y \) deteriorate
Class-Incremental Learning

- Feature function $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$
- Classifiers: $g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- Data comes in class batches: $X^s, \ldots, X^t$ where all examples in $X^y$ are of class $y$

**Idea 2**: fix classifier parameters after each class batch, train only new ones:

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\times$</th>
<th>$w_{\text{cat}}$</th>
<th>$1$</th>
<th>$1$</th>
<th>$-1$</th>
<th>$1$</th>
<th>$-1$</th>
<th>$-1$</th>
<th>$1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{\text{dog}}$</td>
<td>$-1$</td>
<td>$-1$</td>
<td>$1$</td>
<td>$-1$</td>
<td>$1$</td>
<td>$1$</td>
<td>$-1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_{\text{boat}}$</td>
<td>$-1$</td>
<td>$1$</td>
<td>$1$</td>
<td>$-1$</td>
<td>$-1$</td>
<td>$1$</td>
<td>$1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_{\text{truck}}$</td>
<td>$1$</td>
<td>$-1$</td>
<td>$-1$</td>
<td>$1$</td>
<td>$1$</td>
<td>$-1$</td>
<td>$-1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When $\varphi$ changes but $w_y$ does not, outputs deteriorate → "catastrophic forgetting" 

Class-Incremental Learning

- feature function $\varphi: \mathcal{X} \rightarrow \mathbb{R}^d$
- classifiers: $g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \ldots, X^t$ where all examples in $X^y$ are of class $y$

**Idea 2:** fix classifier parameters after each class batch, train only new ones:

<table>
<thead>
<tr>
<th>$X$</th>
<th>🐾</th>
<th>🐱</th>
<th>🐲</th>
<th>🐶</th>
<th>🐐</th>
<th>🐴</th>
<th>🐳</th>
<th>🚗</th>
<th>🚗</th>
<th>🚗</th>
<th>🚗</th>
<th>🚗</th>
<th>🚗</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{cat}$</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_{dog}$</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_{boat}$</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_{truck}$</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- classifiers of different batches are trained independently $\rightarrow$ batches not separated

---

Class-Incremental Learning

- feature function $\varphi: X \rightarrow \mathbb{R}^d$

- classifiers: $g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far

- data comes in class batches: $X^s, \ldots, X^t$ where all examples in $X^y$ are of class $y$

**Idea 2:** fix classifier parameters after each class batch, train only new ones:

<table>
<thead>
<tr>
<th>$x$</th>
<th>cat</th>
<th>dog</th>
<th>boat</th>
<th>truck</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{cat}$</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>$w_{dog}$</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>$w_{boat}$</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$w_{truck}$</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

> classifiers of different batches are trained independently $\rightarrow$ batches not separated

If data representation is also trained, even fixing old $w_y$ is not enough!

When $\varphi$ changes but $w_y$ does not, outputs deteriorate $\rightarrow$ "catastrophic forgetting"

Idea 3: Nearest-Class-Mean (NCM) classifier [Mensink et al. 2013]

\[ y^* = \arg\min_{y \in \mathcal{Y}} \| \varphi(x) - \mu_y \|^2 \]

for \( \mu_y = \frac{1}{|\{i : y_i = y\}|} \sum_{\{i : y_i = y\}} \varphi(x_i) \)

Advantage:
- class mean \( \mu_y \) does not deteriorate when new samples come in
- even classes within different batches 'compete'

Problem:
- when \( \varphi \) changes, we have to recompute \( \mu_y \)
  - we need to store all training examples
  - does not fulfill condition for 'class-incremental'

[Mensink, Verbeek, Perronnin, Csurka. "Distance-based image classification: generalizing to new classes at near-zero cost.", TPAMI 2013]
**Proposal:**

**iCaRL (Incremental Class and Representation Learning)** [Rebuffi et al., CVPR 2017]

**Internal representation:**
- feature function $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$, weight vectors $w_y$ for $y \in \mathcal{Y}$
- for each seen class, $y$, a set of exemplar samples, $P_y$, (in total up to $K$ samples)

**For representation learning:**
- probabilistic outputs: $g_y(x) = \frac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far

**For classification:**
- classify samples by their distance to a class prototype (like NCM does), but using the mean of exemplars, not the mean of all training examples

---

iCaRL: Classification

\[ y^* \leftarrow \text{NearestMeanOfExemplars}(x) \]

<table>
<thead>
<tr>
<th>input ( x )</th>
<th>// sample to be classified</th>
</tr>
</thead>
<tbody>
<tr>
<td>require ( \mathcal{P} = (P_1, \ldots, P_t) )</td>
<td>// class exemplar sets of classes 1, \ldots, ( t ) seen so far</td>
</tr>
<tr>
<td>require ( \varphi : \mathcal{X} \rightarrow \mathbb{R}^d )</td>
<td>// feature map</td>
</tr>
</tbody>
</table>

for \( y = 1, \ldots, t \) do
  \[ \mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p) \] // mean-of-exemplars
end for

\( y^* \leftarrow \text{argmin}_{y=1,\ldots,t} \| \varphi(x) - \mu_y \| \) // nearest prototype

output class label \( y^* \)
**IncrementalTrain**($X^s, \ldots, X^t, K$)

**input** $X^s, \ldots, X^t$  // new training examples in per-class sets (all $x \in X^y$ are of class $y$)

**input** $K$  // maximum number of exemplars

**require** $\Theta$  // current model parameters

**require** $\mathcal{P} = (P_1, \ldots, P_{s-1})$  // current exemplar sets

$\Theta \leftarrow \text{UpdateRepresentation}(X^s, \ldots, X^t; \mathcal{P}, \Theta)$

$m \leftarrow \lfloor K/t \rfloor$  // number of exemplars per class

for $y = 1, \ldots, s - 1$ do

$P_y \leftarrow \text{ReduceExemplarSet}(P_y, m)$

end for

for $y = s, \ldots, t$ do

$P_y \leftarrow \text{ConstructExemplarSet}(X^y, m, \Theta)$

end for

$\mathcal{P} \leftarrow (P_1, \ldots, P_t)$  // new exemplar sets
**UpdateRepresentation**($X^s, \ldots, X^t$)

**input** $X^s, \ldots, X^t$  // training samples of classes $s, \ldots, t$

**require** $\mathcal{P} = (P_1, \ldots, P_{s-1})$  // exemplar sets

**require** $\Theta$  // current model parameters

$$D \leftarrow \bigcup_{y=s,\ldots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\ldots,s-1} \{(x,y) : x \in P^y\}$$  // combined training set

**for** $y = 1, \ldots, s - 1$ **do**

$$q^y_i \leftarrow g_y(x_i) \quad \text{for all } (x_i, \cdot) \in D$$  // store outputs of pre-update network

**end for**

run network training (e.g. BackProp) with loss function

$$\mathcal{L}(\Theta) = -\sum_{(x_i,y_i) \in D} \left[ \sum_{y=s}^t \ell(\delta_{y=y_i}, g_y(x_i)) + \sum_{y=1}^{s-1} \ell(q^y_i, g_y(x_i)) \right]$$

- **classification loss**
- **distillation loss**
Two difference to ordinary network learning/finetuning:

- **training set**

\[
\mathcal{D} \leftarrow \bigcup_{y=s,\ldots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\ldots,s-1} \{(x,y) : x \in P^y\}
\]

consists of samples of new classes, but also exemplars of old classes
→ representation is 'reminded' of old classes regularly

- **loss function**

\[
\mathcal{L}(\Theta) = - \sum_{(x_i,y_i) \in \mathcal{D}} \left[ \sum_{y=s}^{t} \ell(\delta_{y=y_i}, g_y(x_i)) + \sum_{y=1}^{s-1} \ell(q^y_i, g_y(x_i)) \right]
\]

contains not just ordinary classification term, but also *distillation* term [Hinton et al., 2014] → encourage network to preserve its output values across training steps [Li, Hoiem. 2016]
\[ P \leftarrow \text{REDUCEEXEMPLARSET}(P, m) \]

**Input**
- \( P = (p_1, \ldots, p_{|P|}) \) // current exemplar set
- \( m \) // target number of exemplars

**Output**
- exemplar set \( P = (p_1, \ldots, p_m) \) // keep only first \( m \) exemplars

\[ P \leftarrow \text{CONSTRUCTEXEMPLARSET}(X, m) \]

**Input**
- sample set \( X = \{x_1, \ldots, x_n\} \) of class \( y \)
- \( m \) target number of exemplars

**Require**
- current feature function \( \varphi : \mathcal{X} \rightarrow \mathbb{R}^d \)

\[
\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x) \quad \text{// current class mean}
\]

**For** \( k = 1, \ldots, m \)

\[
p_k \leftarrow \underset{x \in X}{\text{argmin}} \left\| \mu - \frac{1}{k} [\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\| \quad \text{// next exemplar}
\]

**Output**
- exemplar set \( P = (p_1, \ldots, p_m) \)
CIFAR-100 dataset:

- 60K low-resolution images (50K train, 10K test)
- 100 classes

iCaRL setup:

- 32-layer ResNet [He et al. 2015]
- $K = 2000$ exemplars
- 2, 5, 10, 20 or 50 classes per batch

iCaRL: Experiments

**ImageNet ILSVRC-2012 dataset:**
- >1.2M high-resolution images (1.2M train, 50K val)
- 1000 classes

**iCaRL setup:**
- 18-layer ResNet [He et al. 2015]
- $K = 20000$ exemplars
- 10 or 100 classes per batch

Alternative methods:

- **finetuning:**
  - train network using stochastic gradient descent
  - no measures to prevent catastrophic forgetting

- **fixed representation:**
  - on the first batch of classes, train the complete network,
  - then, freeze the data representation (all network layers except the last one),
  - for subsequence batches of classes, train only new classifiers

- **LwF.MC:**
  - train network including distillation loss, but do not use exemplars anywhere
  - resembles multi-class version of "Learning with Forgetting" (LwF) [Li and Hoiem, 2016]
Multi-class accuracies over 10 repeats (average and standard deviation) for class-incremental training on CIFAR-100 with 2 (top left), 5 (top middle), 10 (top right), 20 (bottom left) or 50 (bottom right) classes per batch.
Top-5 accuracies for class-incremental training on ILSVRC 2012 with 10 (left) or 100 (right) classes per batch.
Class-Incremental Learning: Results

Confusion matrices of different methods on CIFAR-100 after training for 100 classes with 10 classes per batch (entries are transformed by $\log(1 + x)$ for better visibility).

- **iCaRL**: predictions spread homogeneously over all classes
- **LwF.MC**: prefer recently seen classes $\rightarrow$ some long-term memory loss
- **fixed representation**: prefer batch of classes seen first $\rightarrow$ lack of neural plasticity
- **finetuning**: predict only most recently seen classes $\rightarrow$ catastrophic forgetting
Summary and Conclusion

Class-incremental learning is very reasonable, but it is far from solved:

- how to learn a multi-classifier and a representation jointly?
- how to avoid catastrophic forgetting?

iCaRL is our proposal:

- keep a small set of exemplars for each class
- use a mean-of-exemplars classifier rule instead of network outputs
- using distillation during representation learning to avoid catastrophic forgetting

Open questions:

- can other learning strategies be made class-incremental?
- will exemplars be as beneficial for other problem settings?
- what if we cannot store exemplars, e.g. due to privacy/copyright?