Supplementary Material

for Pose-Sensitive Embedding by Nonlinear NCA Regression
by Graham W. Taylor, Rob Fergus, George Williams, Ian Spiro, and Christoph Bregler

6 Review: NCA

For linear NCA, it is straightforward to derive the gradient of the NCA loss with respect to the matrix of parameters [14]. However, if we are to use a nonlinear mapping, it is useful to derive the gradient of the NCA loss function with respect to the output (i.e. the low-dimensional representation) so that we may perform backpropagation.

As a review, we perform this derivation for the case of standard NCA before we consider NCA regression. Recall, the loss is

$$L_{\text{NCA}} = -\sum_{i=1}^{N} \sum_{j:y_i=y_j} p_{ij}, \tag{8}$$

where

$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{l \neq i} \exp(-d_{il}^2)}, \quad p_{ii} = 0, \quad d_{ij} = ||\mathbf{z}_i - \mathbf{z}_j||_2. \tag{9}$$

Considering just a single training example, $\mathbf{z}_k = f(\mathbf{x}_k)$, we wish to calculate

$$\frac{\partial L_{\text{NCA}}}{\partial \mathbf{z}_k} = -\sum_{i=1}^{N} \sum_{j:y_i=y_j} \frac{\partial p_{ij}}{\partial \mathbf{z}_k}. \tag{10}$$

Since p_{ij} is a softmax, $\frac{\partial p_{ij}}{\partial \mathbf{z}_k}$ has the form

$$\frac{\partial p_{ij}}{\partial \mathbf{z}_k} = p_{ij} \left(\frac{\partial \gamma_{ij}}{\partial \mathbf{z}_k} - \sum_{l \neq i} p_{il} \frac{\partial \gamma_{il}}{\partial \mathbf{z}_k} \right) \tag{11}$$

where $\gamma_{ij} = -d_{ij}^2$ and

$$\frac{\partial d_{ij}^2}{\partial \mathbf{z}_k} = 2\left(\mathbf{z}_i - \mathbf{z}_j\right)[i=k] + 2\left(\mathbf{z}_j - \mathbf{z}_i\right)[j=k] \tag{12}$$

where $[i=k]$ and $[j=k]$ are indicator variables in $\{0,1\}$. Combining Eq. 10-12 we arrive at

$$\frac{\partial L_{\text{NCA}}}{\partial \mathbf{z}_k} = 2\sum_{i=1}^{N} \sum_{j:y_i=y_j} p_{ij} \left(\left(\mathbf{z}_i - \mathbf{z}_j\right)[i=k] + \left(\mathbf{z}_j - \mathbf{z}_i\right)[j=k] \right.$$
$$\left. - \sum_{l \neq i} p_{il} \left(\left(\mathbf{z}_i - \mathbf{z}_l\right)[i=k] + \left(\mathbf{z}_l - \mathbf{z}_i\right)[l=k] \right) \right) \tag{13}$$

The above expression can be further simplified:

$$\frac{\partial L_{\text{NCA}}}{\partial \mathbf{z}_k} = 2\left(\sum_{j:y_j=y_k} p_{kj}\left(\mathbf{z}_k - \mathbf{z}_j\right) + \sum_{i:y_i=y_k} p_{ik}\left(\mathbf{z}_k - \mathbf{z}_i\right)\right.$$
$$\left. - \sum_{j:y_j=y_k} p_{kj} \sum_{l\neq k} p_{kl}\left(\mathbf{z}_k - \mathbf{z}_l\right) - \sum_i \sum_{j:y_i=y_j} p_{ij}p_{ik}\left(\mathbf{z}_k - \mathbf{z}_i\right)\right) \tag{14}$$
$$= 2\left(\sum_{j:y_j=y_k} p_{kj}\left(\mathbf{z}_k - \mathbf{z}_j\right) + \sum_{i:y_i=y_k} p_{ik}\left(\mathbf{z}_k - \mathbf{z}_i\right)\right.$$
$$\left. - p_k \sum_{l\neq k} p_{kl}\left(\mathbf{z}_k - \mathbf{z}_l\right) - \sum_i p_i p_{ik}\left(\mathbf{z}_k - \mathbf{z}_i\right)\right) \tag{15}$$
$$= 2\left(\sum_{j:y_j=y_k} p_{kj}\left(\mathbf{z}_k - \mathbf{z}_j\right) + \sum_{j:y_j=y_k} p_{jk}\left(\mathbf{z}_k - \mathbf{z}_j\right)\right.$$
$$\left. - p_k \sum_{j\neq k} p_{kj}\left(\mathbf{z}_k - \mathbf{z}_j\right) - \sum_j p_j p_{jk}\left(\mathbf{z}_k - \mathbf{z}_j\right)\right) \tag{16}$$
$$= 2\left(\sum_{j:y_j=y_k} \left(\mathbf{z}_k - \mathbf{z}_j\right)\left(p_{kj} + p_{jk}\right) - \sum_{j\neq k} \left(\mathbf{z}_k - \mathbf{z}_j\right)\left(p_k p_{kj} + p_j p_{jk}\right)\right) \tag{17}$$

where we have used the shorthand $p_i = \sum_{j:y_i=y_j} p_{ij}$. We can also rearrange the above expression to match the expression given in the Appendix of [34] (here, using our notation):

$$-\frac{\partial L_{\text{NCA}}}{\partial \mathbf{z}_k} = -2\left(\sum_{j:y_j=y_k} p_{kj}\left(\mathbf{z}_k - \mathbf{z}_j\right) - p_k \sum_{j\neq k} p_{kj}\left(\mathbf{z}_k - \mathbf{z}_i\right)\right)$$
$$+ 2\left(\sum_{j:y_j=y_k} p_{jk}\left(\mathbf{z}_j - \mathbf{z}_k\right) - \sum_{j\neq k} p_j p_{jk}\left(\mathbf{z}_j - \mathbf{z}_k\right)\right). \tag{18}$$

7 NCA regression

We can similarly derive an expression for the gradient of the NCAR objective with respect to the output units. Recall that the NCAR objective is

$$L_{\text{NCAR}} = \sum_{i=1}^{N} \sum_{j\neq i} p_{ij} ||\mathbf{y}_i - \mathbf{y}_j||_2^2. \tag{19}$$

where $\mathbf{y_k}$ are real-valued rather than discrete. Again we consider a single training example, $\mathbf{z}_k = f(\mathbf{x}_k)$, and aim to calculate

$$\frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}_k} = \sum_{i=1}^{N} \sum_{j\neq i} y_{ij}^2 \frac{\partial p_{ij}}{\partial \mathbf{z}_k} \tag{20}$$

where we have used the shorthand $y_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||_2^2$. Note the similarity between Eq. 20 is to Eq. 10. Combining Eq. 20 with Eq. 11 and Eq. 12, we arrive at

$$\frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}_k} = -2 \sum_{i=1}^{N} \sum_{j \neq i} y_{ij}^2 p_{ij} \left((\mathbf{z}_i - \mathbf{z}_j) [i = k] + (\mathbf{z}_j - \mathbf{z}_i) [j = k] \right.$$
$$\left. - \sum_{l \neq i} p_{il} \left((\mathbf{z}_i - \mathbf{z}_l) [i = k] + (\mathbf{z}_l - \mathbf{z}_i) [l = k] \right) \right) \tag{21}$$

The above expression can be further simplified

$$\frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}_k} = -2 \left(\sum_{j \neq k} y_{kj}^2 p_{kj} (\mathbf{z}_k - \mathbf{z}_j) + \sum_{i \neq k} y_{ik}^2 p_{ik} (\mathbf{z}_k - \mathbf{z}_i) \right.$$
$$\left. - \sum_{j \neq k} y_{kj}^2 p_{kj} \sum_{l \neq k} p_{kl} (\mathbf{z}_k - \mathbf{z}_l) - \sum_{i} \sum_{j \neq k} y_{ij}^2 p_{ij} p_{ik} (\mathbf{z}_k - \mathbf{z}_i) \right) \tag{22}$$
$$= -2 \left(\sum_{j \neq k} y_{kj}^2 p_{kj} (\mathbf{z}_k - \mathbf{z}_j) + \sum_{i \neq k} y_{ik}^2 p_{ik} (\mathbf{z}_k - \mathbf{z}_i) \right.$$
$$\left. - \delta_k \sum_{l \neq k} p_{kl} (\mathbf{z}_k - \mathbf{z}_l) - \sum_{i} \delta_i p_{ik} (\mathbf{z}_k - \mathbf{z}_i) \right) \tag{23}$$
$$= -2 \left(\sum_{j \neq k} y_{kj}^2 p_{kj} (\mathbf{z}_k - \mathbf{z}_j) + \sum_{j \neq k} y_{jk}^2 p_{jk} (\mathbf{z}_k - \mathbf{z}_j) \right.$$
$$\left. - \delta_k \sum_{j \neq k} p_{kj} (\mathbf{z}_k - \mathbf{z}_j) - \sum_{j} \delta_j p_{jk} (\mathbf{z}_k - \mathbf{z}_j) \right) \tag{24}$$
$$= -2 \sum_{j \neq k} (\mathbf{z}_k - \mathbf{z}_j) \left(y_{kj}^2 p_{kj} + y_{jk}^2 p_{jk} - \delta_k p_{kj} - \delta_j p_{jk} \right) \tag{25}$$
$$= -2 \sum_{j \neq k} (\mathbf{z}_k - \mathbf{z}_j) \left(p_{kj} \left(y_{kj}^2 - \delta_k \right) + p_{jk} \left(y_{jk}^2 - \delta_j \right) \right) \tag{26}$$

where we have used the shorthand $\delta_i = \sum_{j \neq i} y_{ij}^2 p_{ij}$. Note that the matrix formed by elements y_{ij} is symmetric while the one formed by elements p_{ij} is not.

8 Parameter updates

In the case of linear NCAR, we have a single parameter matrix, A, where $\mathbf{z}_k = A\mathbf{x}_k$. The gradient of L_{NCAR} with respect to A is given by

$$\frac{\partial L_{\text{NCAR}}}{\partial A} = \sum_{k=1}^{N} \frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}_k} \frac{\partial \mathbf{z}_k}{\partial A} = \sum_{k=1}^{N} \frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}_k} \mathbf{x}_k^T \tag{27}$$

where $\frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}_k}$ is given by Eq. 26.

In the case of nonlinear NCAR (e.g. C-NCAR), we can update parameters recursively using back-propagation. Here, we present the weight updates for the specific convolutional network that we used in our experiments: two convolution and subsampling layers, and one fully-connected layer (Fig. 2). A more general, practical treatment for arbitrary convolutional nets is given in [7].

8.1 Forward pass

We now generalize our notation to accommodate multiple layers of representation. Let $\mathbf{x}_\mathbf{j}^\ell$ be the j^{th} feature map at layer l. This is a 3D array: the first two dimensions are spatial, and the third is the feature, indexed by j. We use greyscale images, so the first layer, $\mathbf{x}^0$, represents the preprocessed input which does not have a 3rd dimension. Note that previously we used a subscript to represent cases in our training set. In the discussion that follows, we will always discuss a single case, and therefore the subscript indexes features.

The first step is to convolve the input with a series of learned 2D filters, $\mathbf{k}_j^1$, add a per-feature map bias, b_j^1, and apply a nonlinearity:

$$\mathbf{x}_j^1 = f\left(\mathbf{x}^0 * \mathbf{k}_j^1 + b_j^1\right) \tag{28}$$

where $f\left(\mathbf{u}_j\right) = \text{abs}\left(\tanh\left(\mathbf{u}_j\right)\right)$. Each feature map is then averaged and downsampled (in our experiments, by a factor of five):

$$\mathbf{x}_j^2 = \beta_j^2 \text{down}\left(\mathbf{x}_j^1, 5\right) \tag{29}$$

where β_j is a per-map learned coefficient (the superscript being a layer index, not a square) and $\text{down}\left(\cdot, m\right)$ is an operator that performs average downsampling by a factor of m. The second convolutional layer is similar to the first:

$$\mathbf{x}_j^3 = f\left(\sum_{i \in M_j} \mathbf{x}_i^2 * \mathbf{k}_{ij}^3 + b_j^3\right), \tag{30}$$

except that now we have multiple feature maps as input, instead of a single greyscale image. Rather than fully connecting all input maps to all output maps, we use sparse random connectivity [21]. Each output map is connected to four randomly chosen input maps specified by a map index M_j which is constructed prior to training. The nonlinearity, $f\left(\cdot\right)$, remains the same. We then downsample by a factor of four:

$$\mathbf{x}_j^4 = \beta_j^4 \text{down}\left(\mathbf{x}_j^4, 4\right). \tag{31}$$

The output is converted from a 3D array to a vector, and is processed by a final "fully-connected" layer:

$$\mathbf{z} = \mathbf{x}^5 = A^5 \times \text{flatten}\left(x_1^4, \ldots, x_J^4\right) \tag{32}$$

where $\text{flatten}\left(\cdot\right)$ is a flattening operator that converts all J feature maps of layer 4 into a single vector. The output, $\mathbf{z}$, subsequently is a vector.

8.2 Backpropagation

Now that we have defined the forward pass, we can apply backpropagation to compute the updates for the parameter set $\{\mathbf{k}^1, \mathbf{b}^1, \beta^2, \mathbf{k}^3, \mathbf{b}^3, \beta^4, A^5\}$. Backpropagation alternates between computing the "sensitivity"[1] of layer l, δ_ℓ, and computing the gradient of the loss with respect to the parameters at layer l.

The first step of backpropagation, that is, computing the sensitivity of the output layer, has already been done: $\delta_5 = \frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}}$ (see Eq. 26 noting that we have dropped the case index). Since our output units are linear, the gradient with respect to A (for notational simplicity, we drop the superscript) is given by Eq. 27. Backpropagation continues as follows:

[1] Sensitivity is the derivative of the loss with respect to the input of a single unit of layer l.

$$\delta_j^4 = A^T \frac{\partial L_{\text{NCAR}}}{\partial \mathbf{z}}, \tag{33}$$

$$\frac{\partial L_{\text{NCAR}}}{\partial \beta_j^4} = \sum_{u,v} \left(\delta_j^4 \circ \text{down}\left(\mathbf{x}_j^3, 4\right)\right)_{uv} \tag{34}$$

where u and v are spatial indices and $\circ$ is element-wise multiplication. Continuing,

$$\delta_j^3 = \beta_j^4 \left(f'\left(\mathbf{u}_j^3\right) \circ \text{up}\left(\delta_j^4, 4\right)\right) \tag{35}$$

where $\mathbf{u}_j^\ell$ is the input to feature map j at layer l (i.e. before the nonlinearity) and $\text{up}\left(\cdot, m\right)$ is an upsampling operator that reverses $\text{down}\left(\cdot, m\right)$. In the following equations, we use Matlab notation for the 2D convolution operations to be explicit about boundary conditions:

$$\frac{\partial L_{\text{NCAR}}}{\partial b_j^3} = \sum_{u,v} \left(\delta_j^3\right)_{uv} \tag{36}$$

$$\frac{\partial L_{\text{NCAR}}}{\partial \mathbf{k}_{ij}^3} = \text{rot180}\left(\text{conv2}\left(\mathbf{x}_i^2, \text{rot180}\left(\delta_j^3\right), '\text{valid}'\right)\right) \tag{37}$$

where $\text{rot180}\left(\cdot\right)$ flips the filter in both dimensions to perform correlation rather than convolution. To calculate the sensitivity maps for layer 2, we must consider the connectivity map. Explicitly, we must sum over all output maps j to which input map i is connected via the connectivity maps $M_j, \forall j$:

$$\delta_i^2 = \sum_{j:i \in M_j} \text{conv2}\left(\delta_j^3, \text{rot180}\left(\mathbf{k}_{ij}^3\right), '\text{full}'\right). \tag{38}$$

The remaining expressions are similar to the ones above:

$$\frac{\partial L_{\text{NCAR}}}{\partial \beta_j^2} = \sum_{u,v} \left(\delta_j^2 \circ \text{down}\left(\mathbf{x}_j^1, 5\right)\right)_{uv}, \tag{39}$$

$$\delta_j^1 = \beta_j^2 \left(f'\left(\mathbf{u}_j^1\right) \circ \text{up}\left(\delta_j^2, 5\right)\right), \tag{40}$$

$$\frac{\partial L_{\text{NCAR}}}{\partial b_j^1} = \sum_{u,v} \left(\delta_j^1\right)_{uv}, \tag{41}$$

$$\frac{\partial L_{\text{NCAR}}}{\partial \mathbf{k}_j^1} = \text{rot180}\left(\text{conv2}\left(\mathbf{x}^0, \text{rot180}\left(\delta_j^1\right), '\text{valid}'\right)\right). \tag{42}$$

Now that we have the gradient expressions for each set of parameters, we can optimize with respect to the loss. We use an implementation of the nonlinear conjugate gradient method.