Convolutional Neural Networks

Stephen Vardeman

Analytics Iowa LLC

ISU Statistics and IMSE

Image classification

A popular application of neural network ideas is that of image classification. Not surprisingly, success in this realm seems to rely as much upon ideas from image processing as upon ideas from prediction.

A grey-scale image is typically represented by an $L \times M$ matrix $\mathbf{X} = [x_{lm}]$ where each $x_{lm} \in \{0, 1, 2, \ldots, 254, 255\}$ represents a brightness at location (l, m). A color image is often represented by 3 matrices $\underset{L\times M}{\mathbf{X}^{\mathrm{r}}} = [x_{lm}]$, $\underset{L\times M}{\mathbf{X}^{\mathrm{g}}} = [x_{lm}]$, and $\underset{L\times M}{\mathbf{X}^{\mathrm{b}}} = [x_{lm}]$ (again all with entries in $\{0, 1, 2, \ldots, 254, 255\}$) representing intensities in red, green, and blue "channels." The standard machine learning problem is to produce a classifier (based on a training set of N images $\mathbf{X}_i$ or $\left[\mathbf{X}^{\mathrm{r}}, \mathbf{X}^{\mathrm{g}}, \mathbf{X}^{\mathrm{b}}\right]_i$ with corresponding class identities $y_i \in \{1, 2, \ldots, K\}$). (For example, a standard test problem is "automatic" recognition of hand-written digits 0 through 9.)

Generalities

Simple convolutional neural networks with H hidden layers and a softmax output layer are successive compositions of linear and non-linear operations that might be represented as follows. For σ^H operating on $\mathbf{X}$ using some set of real number parameters $\mathbf{A}^H$ to produce some multivariate output (we describe below some kinds of things that are popularly used) a "deepest layer" of the convolutional neural net produces

$$\mathbf{Z}^H \equiv \sigma^H \left(\mathbf{X}, \mathbf{A}^H \right) \tag{4}$$

Then applying another set of operations σ^{H-1} to the result (4) using some set of parameters $\mathbf{A}^{H-1}$, the next layer of values in the convolutional neural net is produced as

$$\mathbf{Z}^{H-1} = \sigma^{H-1} \left(\mathbf{Z}^H, \mathbf{A}^{H-1} \right)$$

Generalities cont.

This continues, with

$$\mathbf{Z}^h = \sigma^h\left(\mathbf{Z}^{h+1}, \mathbf{A}^h\right) \tag{5}$$

for $h = H, H-1, \ldots, 1$ where σ^1 is $\Re^K$-valued (the top layer of hidden values is a K-vector).

Then, with $\mathbf{g}$ the softmax function the output K-vector of class probabilities is

$$\mathbf{g}\left(\mathbf{Z}^1\right)$$

In multi-channel cases, it seems common to develop separate series of compositions based on $\mathbf{X}^{\mathrm{r}}, \mathbf{X}^{\mathrm{g}}$, and $\mathbf{X}^{\mathrm{b}}$ and to bring them together only in the top or top few levels of this kind of hierarchy.

Variants

Variants of this basic structure are possible and have been used. For example, it is sometimes done to make a "direct connection" between layer h and one deeper than layer $h+1$. That is the option to employ a form

$$\mathbf{Z}^h = \sigma^h\left(\mathbf{Z}^{h+1}, \mathbf{Z}^{h+j}, \mathbf{A}^h\right)$$

for some $j > 1$ or even a form

$$\mathbf{Z}^h = \sigma^h\left(\mathbf{Z}^{h+1}, \mathbf{X}, \mathbf{A}^h\right)$$

making a direct connection to the input layer is sometimes employed. (Obviously, even more complicated schemes are possible.)

Convolutional layers: linear filtering

Most of what we have said thus far in this section is not really special to the problem of image classification (and could serve as a high-level introduction to general neural net predictors). What sets the "convolutional" neural network field apart from "generic" neural network practice is the image-processing-inspired forms employed in the functions σ^h. The most fundamental form is one that applies "linear filters" to images followed by some nonlinear operation. This creates what is commonly called a "convolution" layer.

To make the idea of a convolutional layer precise, consider the following. Let $\mathbf{F}$ be an $R \times C$ matrix (that might be called a linear filter matrix). Typically this matrix is much smaller than the image and square (at least when "horizontal" and "vertical" resolutions in the images are the same), and R and C are often odd.

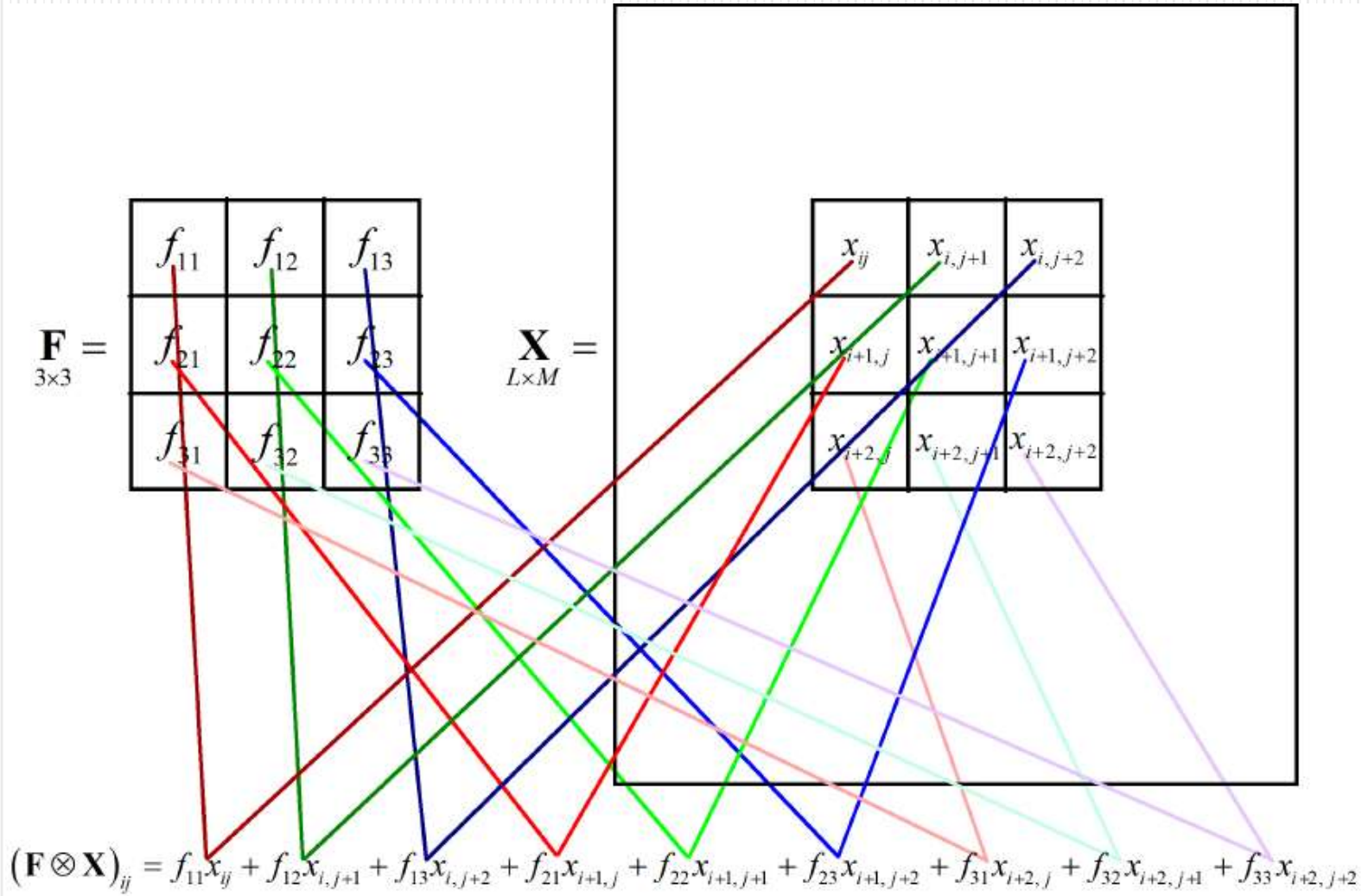
Linear filtering/convolution

One can then make from $\mathbf{F}$ and $\mathbf{X}$ a new matrix $\mathbf{F} \otimes \mathbf{X}$ of dimension $(L - R + 1) \times (M - C + 1)$ with entries

$$(\mathbf{F} \otimes \mathbf{X})_{ij} = \sum_{a=1}^{R} \sum_{b=1}^{C} f_{ab} x_{(i+a-1),(j+b-1)} \tag{6}$$

A natural way to think about this operation is to align an $R \times C$ integer grid with values in $\mathbf{F}$ on the grid points with a (larger) corresponding grid for the image $\mathbf{X}$, setting the upper left $(1, 1)$ corner of the $\mathbf{F}$ grid at the (i, j) location on the $\mathbf{X}$ grid, and to then sum products of aligned matrix entries. The entries of $\mathbf{F}$ serve as weights on the values in the $R \times C$ part of the image aligned with the filter matrix. The next figure illustrates this process for a simple case where $\mathbf{F}$ is 3×3 (and so ultimately $\mathbf{F} \otimes \mathbf{X}$ has 2 fewer columns and 2 fewer rows than $\mathbf{X}$ and is thus $(L - 2) \times (M - 2)$).

Convolution



$$(\mathbf{F}\otimes\mathbf{X})_{ij} = f_{11}x_{ij} + f_{12}x_{i,j+1} + f_{13}x_{i,j+2} + f_{21}x_{i+1,j} + f_{22}x_{i+1,j+1} + f_{23}x_{i+1,j+2} + f_{31}x_{i+2,j} + f_{32}x_{i+2,j+1} + f_{33}x_{i+2,j+2}$$

Introducing non-linearity

This convolution operation is linear and it is typical practice to introduce non-linearity by following convolution operations in a layer with the hinge function $\max(u, 0)$ applied to each element u of the resulting matrix. Sometimes people (apparently wishing to not lose rows and columns in the convolution process) "0 pad" an image with extra rows and columns of 0s before doing the convolution–a practice that strikes this author as lacking sound rationale.

Types of filter matrices

Multiple convolutions are typically created in a single convolution layer. Sometimes the filter matrices are filled with parameters to be determined in fitting (i.e. are part of $\mathbf{A}^h$ in the representation (5)). But they can also be fixed matrices created for specific purposes. For example the 3×3 matrices

$$\mathbf{S}^{\text{vert}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{S}^{\text{horiz}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

are respectively the vertical and horizontal Sobel filter matrices, commonly used in image processing when searching for edges of objects or regions. And various "blurring" filters (ordinary arithmetic averaging across a square of pixels and weighted averaging done according to values of a Gaussian density set at the center of an integer grid) are common devices meant to suppress noise.

Dimension reduction: sampling

As multiple layers each with multiple new convolutions are created, there is potential explosion of the total dimensionality of the sets of $\mathbf{Z}^h$ and $\mathbf{A}^h$. Two devices for controlling that explosion are the notions of sampling and pooling to reduce the size of a $\mathbf{Z}$. First, instead of creating and subsequently using an entire filtered image $\mathbf{F} \otimes \mathbf{X}$, one can use only every sth row and column. In such a "sampling" operation s is colloquially known as the "stride." Roughly speaking, this reduces the size of a $\mathbf{Z}$ by a factor of s^2.

Dimension reduction: pooling

Another possibility is to choose some block size, say $s \times t$, and divide an image into roughly

$$\left(\frac{L}{s}\right)\left(\frac{M}{t}\right)$$

non-overlapping blocks, within a block applying a "pooling" rule like "simple averaging" or "maximum value." One then uses the rectangular array of these pooled values as a layer output. This, of course, reduces the size of a $\mathbf{Z}$ by a factor of roughly st.

It seems common to apply one of these ideas after each one or few convolution layers in a network, and especially before reaching the top and final one or few layers. The final hidden layers of a convolutional neural net are of the "ordinary" type described earlier and if the dimensionality of their inputs are too large, numerical and fitting problems will typically ensue.