Chapter 7

Telling birds from airplanes: Learning from images

Summary

- `Datasets` and `DataLoaders` can be used for loading and sampling datasets in computer vision
- For classification, the ideal loss function is obtained using the output of softmax as the input of a nonnegative log likelihood function (the combination of softmax and this loss is called cross entropy in PyTorch)
- Treating images as vectors of pixel values is an option to use in a fully connected network but that makes it difficult to take advantage of spatial relationships in the data

7.1: A dataset of tiny images

PyTorch `Dataset`: an object that is required to implement two methods: `__len__`, which returns the number of items in the dataset, and `__getitem__`, which returns the item (ie. a sample and its corresponding label)
When `__len__` and `__getitem__` are implemented, we can use the built-in `len()` function and standard Python subscripting, respectively.
We can use the `ToTensor` object from `torchvision.transforms` to convert a PIL image to a PyTorch tensor:

```
from torchvision import transforms
to_tensor = transforms.ToTensor()
img_t = to_tensor(img)
```

Using this object, a 3-channel (RGB) 32 x 32 image will be converted to a 3 x 32 x 32 tensor. Whereas the values in the original PIL image range from 0 to 255 (8 bits per channel), the `ToTensor` transform turns the data into a 32-bit floating-point per channel which scales the values down to a range from 0.0 to 1.0.
To verify that we're still getting the same image:

```
plt.imshow(img_t.permute(1, 2, 0))
plt.show()
```

Transforms can be chained together using `transforms.Compose`, for instance to normalize the dataset so that each channel has zero mean and unitary standard deviation.

7.2 Distinguishing birds from airplanes

Softmax is a function that takes a vector of values and produces another vector of the same dimension, where the values satisfy:

- Each element of the output must be in the [0.0, 1.0] range
- The elements of the output must add up to 1.0

To do this, we take the elements of the vector, compute the elementwise exponential, and divide each element by the sum of exponentials:

```
def softmax(x):
     return torch.exp(x) / torch.exp(x).sum()
```

We can also use the module `nn.Softmax()`

Negative log likelihood (NLL): a loss function that is very high when the likelihood is low and conversely, the loss should be low when the likelihood is higher than the alternatives

The `DataLoader` class can help with shuffling and organizing the data into mini batches.