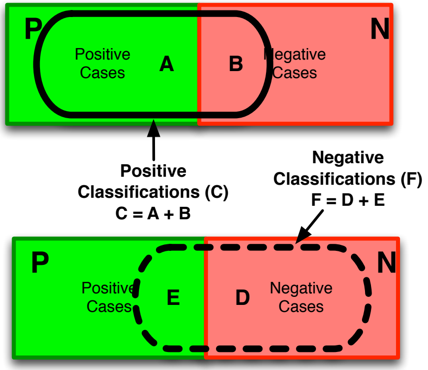
Practical Aspects of Deep Learning

Keith L. Downing

The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
keithd@idi.ntnu.no

November 12, 2017

Accuracy, Precision and Recall



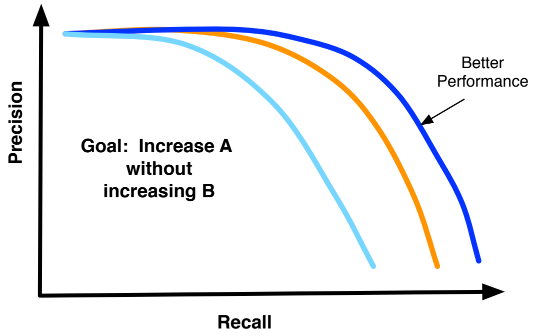
$$\text{Precision} = \frac{A}{A + B}$$

$$\text{Recall} = \frac{A}{P}$$

$$\text{Accuracy} = \frac{A + D}{P + N}$$

- P and N are the positive and negative cases, respectively.
- C and F are the cases classified as positive (negative) by the model.

Precision-Recall Curve



With increased recall, A increases. If this can be done without increasing B (false positives), then precision remains high: the upper plateau remains flat.

General Deep-Learning Tips

- Consider simpler techniques first, and if you use DL, try the vanilla versions (e.g. basic Stochastic Gradient Descent with zero or one hidden layer) before adding all the bells and whistles.
- For image classification, use convolution; for sequences, use LSTMs or Gated Recurrent Units (GRUs).
- Adding more parameters (i.e. layers and weights) can normally reduce training error, but a) increased computational cost, and b) greater risk of overfitting and thus high generalization error.
- If you add more parameters, you may have to add regularization.
- The Adam Optimizer, Batch Normalization and Dropout are powerful (and popular) regularizers.
- But if you have **enough data**, you may not need any regularizers.
- It may be more practical (and effective) to gather more data instead of trying more and more complex DL algorithms.
- For hard image-processing or natural-language tasks, pre-trained layers or embeddings can provide a good head start.

Hyperparameters

- Include number and types of layers, learning rate(s), momentum rate(s), initial weight range, dropout rate, regularization penalty, decay rate(s) in optimizers (RMSProp, Adam), etc.
- Select these manually or automatically.
- These have considerable interaction, so you need to think carefully about these relationships when tuning.
- Standard goal: reduce generalization error at lowest computational cost; bells and whistles can be expensive.
- Learning rate is often the most important hyperparameter. Decaying learning rates normally perform better than constant rates.
- If using automated methods, random search in hyperparameter space works better than structured (grid) search, since the former does not force an artificial discretization of the search space.

Model Capacity

The number of cases that the network can properly handle.

Methods for Increasing Model Capacity

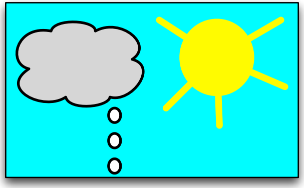
- Optimally tune the learning rate.
- $\Uparrow$ hidden layers and neurons.
- $\Uparrow$ convolution kernels.
- $\Uparrow$ width of convolution filters/kernels.
- $\Uparrow$ zero padding $\rightarrow$ less layer-to-layer size reduction.
- $\Downarrow$ weight decay rate. Larger weight range produces more internal patterns.
- $\Downarrow$ dropout rate.

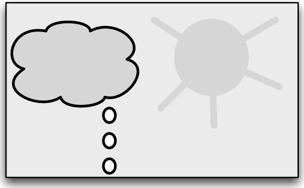
Of course, anything that increases model capacity runs the risk of overfitting.

Debugging Tips

- Since NN's are adaptive, the system may compensate for a bug and thereby hide it.
- Time spent producing informative visualizations is a good investment.
- Monitor changes in weights, internal activations and gradients.
- Start with small datasets.
- Start with small networks and just verify that they are learning (at least a little bit).
- Use comparison of training and testing error to indicate problems such as overfitting.
- 1% Rule - Weight and bias updates should be approximately 1% of the parameter's magnitude.
- Convince yourself that the representation chosen for input features actually supports learning: the NN can achieve differences (of internal state) that make a difference (in classifications).

Contrast Normalization





- For many tasks (e.g. image recognition), these two images should evoke similar responses from a neural network.
- But they may not due to extreme differences in intensity and contrast.
- Solution: Scale all pixel RGB (red, green, blue) intensity values by the average contrast.

Contrast Normalization

Scale pixel R,G,B values by contrast = standard deviation of intensity.

- $X_{i,j,k}$ = intensity of color k at pixel location (i,j), where k = R,G or B.
- Avg Intensity: $\bar{X} = \frac{1}{3mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{3} X_{i,j,k}$
- Intensity Variance: $\sigma_c^2 = \frac{1}{3mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{3} (X_{i,j,k} - \bar{X})^2$
- The scaled pixel intensity, $\tilde{X}_{i,j,k}$:

$$\tilde{X}_{i,j,k} = \frac{X_{i,j,k} - \bar{X}}{\sqrt{\lambda + \sigma_c^2}}$$

  where λ is a constant (e.g. 1) to avoid division by zero when $\sigma_c^2 = 0$.
- These scaled values, typically in the range (-3, 3) make a lot of sense to an NN, but they do not translate directly to RGB values. So they require another round of scaling (e.g. to values in range (0,255)) to actually view the scaled images.