Data Normalization Effect

Kamran Chitsaz

May 2021

1 Introduction

Talking about closed-form analysis of neural networks is difficult due to their complexity. Here we first clarify the effect of normalization on a simple linear regression problem. then will return to our case.

So assume that we have a linear regression model which tries to predict scalar-valued t as a function of input vector x with feature representation $\phi(x)$:

$$y = \mathbf{w}^\top \phi(\mathbf{x}) + b \tag{1}$$

We can use a homogeneous coordinate representation, where a 1 is appended to the feature vector. I.e., we take $\breve{\phi}(\mathbf{x}) = \left(\phi(\mathbf{x})^\top 1\right)^\top$ and $\breve{\mathbf{w}} = \left(\mathbf{w}^\top b\right)^\top$ so that:

$$y = \breve{\mathbf{w}}^\top \breve{\phi}(\mathbf{x}) \tag{2}$$

So given training set $\left\{\left(\mathbf{x}^{(i)}, t^{(i)}\right)\right\}_{i=1}^{N}$, and quadratic cost function, we can fit our model as below:

$$\begin{aligned}
\breve{\mathbf{w}}_\star &\in \arg\min_{\breve{\mathbf{w}}} \mathcal{J}(\breve{\mathbf{w}}) \\
&= \arg\min_{\mathbf{w}} \frac{1}{2N} \sum_{i=1}^{N} \left(\breve{\mathbf{w}}^\top \breve{\phi}\left(\mathbf{x}^{(i)}\right) - t^{(i)}\right)^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{2N} \left\|\breve{\Phi}\breve{\mathbf{w}} - \mathbf{t}\right\|^2
\end{aligned} \tag{3}$$

Considering K_{th} step of gradient descent we have:

$$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} \mathcal{J}\left(\mathbf{w}^{(k)}\right) \tag{4}$$

So for the convex quadratic objective we have:

$$\begin{aligned}
\mathbf{w}^{(k+1)} &\leftarrow \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} \mathcal{J}\left(\mathbf{w}^{(k)}\right) \\
&= \mathbf{w}^{(k)} - \alpha \left[\mathbf{A}\mathbf{w}^{(k)} + \mathbf{b}\right] \\
&= (\mathbf{I} - \alpha \mathbf{A})\mathbf{w}^{(k)} - \alpha \mathbf{b}
\end{aligned} \tag{5}$$

Which $\mathbf{A} = \frac{1}{N}\breve{\boldsymbol{\Phi}}^\top\breve{\boldsymbol{\Phi}}$ is a positive semidefinite matrix and b is a vector such that $\mathbf{b} = -\frac{1}{N}\breve{\boldsymbol{\Phi}}^\top\mathbf{t}$. To understand the gradient descent dynamics, consider the curvature matrix $\mathbf{A} = \frac{1}{N}\breve{\boldsymbol{\Phi}}^\top\breve{\boldsymbol{\Phi}}$:

$$\begin{aligned}
\mathbf{A} &= \frac{1}{N}\breve{\boldsymbol{\Phi}}^\top\breve{\boldsymbol{\Phi}} \\
&= \frac{1}{N}\begin{pmatrix} \boldsymbol{\Phi}^\top \\ \mathbf{1}^\top \end{pmatrix}\begin{pmatrix} \boldsymbol{\Phi} & \mathbf{1} \end{pmatrix} \\
&= \begin{pmatrix} \frac{1}{N}\sum_{i=1}^N \phi\left(\mathbf{x}^{(i)}\right)\phi\left(\mathbf{x}^{(i)}\right)^\top & \frac{1}{N}\sum_{i=1}^N \phi\left(\mathbf{x}^{(i)}\right) \\ \frac{1}{N}\sum_{i=1}^N \phi\left(\mathbf{x}^{(i)}\right)^\top & 1 \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^\top & \boldsymbol{\mu} \\ \boldsymbol{\mu}^\top & 1 \end{pmatrix}
\end{aligned} \tag{6}$$

Where μ and Σ denote the empirical feature mean and covariance, respectively. Based on equation 5, one of the best way to analyze gradient descent convergence speed is to measure the condition number of curvature matrix $\mathbf{A}$. By definition we know that:

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

Where $\sigma_{max}(\mathbf{A})$ and $\sigma_{min}(\mathbf{A})$ are maximal and minimal singular values of $\mathbf{A}$ respectively. Let's think about few particular choices of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and the effect of each on convergence:

- When $\mu = 0$ and $\Sigma = I$ we can see that the the curvature matrix $\mathbf{A}$ will $A = I$ due to equation 6. This is the ideal case for optimization because the condition number is $\kappa = 1$. Gradient descent heads directly for the optimal solution.

- In the case of $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma}$ is a diagonal matrix whose entries are in the interval $\left[\sigma_{\min}^2, \sigma_{\max}^2\right]$, with $\sigma_{\min}^2 \leq 1 \leq \sigma_{\max}^2$, $\mathbf{A}$ is a diagonal matrix whose diagonal entries are the diagonal entries of Σ, plus a 1 for the homogeneous coordinate. So in this case the condition number is $\kappa = \sigma_{\max}^2/\sigma_{\min}^2$. As we know when the condition number is high, convergence can become slow. So so it is beneficial for the speed of optimization that the variances of different dimensions be close together.

- Suppose $\boldsymbol{\Sigma}$ is a diagonal matrix but now let $\boldsymbol{\mu} \neq \mathbf{0}$. Let's decompose equation 6 as follow:

  $$\mathbf{A} = \begin{pmatrix} \boldsymbol{\Sigma} & \\ & 0 \end{pmatrix} + \begin{pmatrix} \boldsymbol{\mu} \\ 1 \end{pmatrix}\begin{pmatrix} \boldsymbol{\mu}^\top & 1 \end{pmatrix}$$

  The first term is a diagonal matrix whose non zero eigenvalues are bounded between σ_{min}^2 and σ_{max}^2. But the second term is a rank-1 matrix whose nonzero eigenvalue is $\|\boldsymbol{\mu}\| + 1$. Although In general, there is no direct relationship between the eigendecomposition of a sum and the eigendecomposition of the components but approximately we can say if the nonzero eigenvalue of the second term is much larger than the eigenvalues of the first term, then A will have a large outlier eigenvalue greater than or equal to $\|\boldsymbol{\mu}\| + 1$. Hence, the condition number bounded as $\kappa \geq \left(\|\boldsymbol{\mu}\|^2 + 1\right)/\sigma_{\max}^2$. As we increase the feature dimension D, the $\|\boldsymbol{\mu}\| + 1$ grows proportionally to D and consequently our model will be learned more and more slowly.

In order to normalize, we transform feature representations to have zero mean and unit variance as follow:

$$\tilde{\phi}_j(\mathbf{x}) = \frac{\phi_j(\mathbf{x}) - \mu_j}{\sigma_j}$$

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} \phi_j\left(\mathbf{x}^{(i)}\right)$$

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\phi_j\left(\mathbf{x}^{(i)}\right) - \mu_j\right)^2$$

In fact, by normalizing data, we are centered the data and reduce the condition number for better convergence.

Let's back to our problem. due to the convexity of the cross-entropy loss

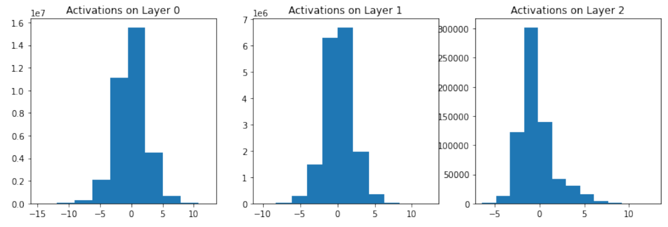

Figure 1: Distribution of activations before normalization

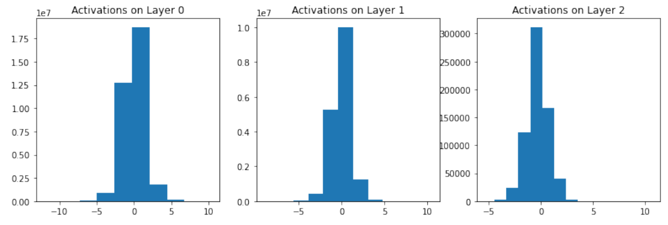

Figure 2: Distribution of activations after normalization

function, we can use the same insight as above. In Figures.1,2 we illustrated the distribution of activation on each layer (before ReLU) for normalized and unnormalized data after one feed-forward (initialization step). We observed that after normalization the activations are much closer to the standard normal distribution for each layer. On the other hand, for unnormalized data, the distribution of activations get far from standard normal and this difference gets more significant in the last layers. In this case, the condition number of the curvature matrix of last activation layer for unnormalized data is equal to 483.7 and for normalized data is 54.3 which is significantly less than unnormalized case. This difference can justify the much faster convergence of the normal model.

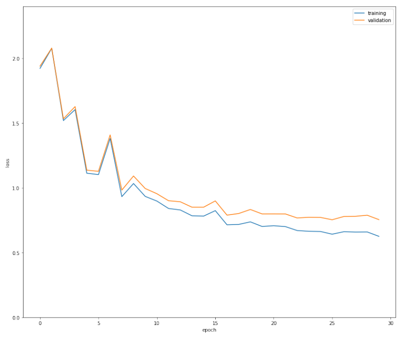

Figure 3: accuracy

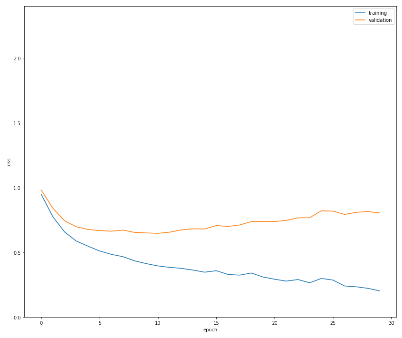

Figure 4: loss

Another interesting phenomenon we observed in Figure.4. is that our model simply overfits when we train with normalized data. In my point of view, this the difference between optimization and learning. An optimal solution for a model will not imply better generalization. With normalizing the training data we can achieve better convergence during gradient descent. But can worsen model generalization to out-of-domain data.