2020 年 1 月 7 日
慶應義塾大学理工学部物理学科
岡崎健人

MLP の訓練のための誤差逆伝播の例

6.5.7　MLP の訓練のための誤差逆伝播の例

隠れ層が一つの MLP を用いてバックプロパゲーションの例を見てみる。学習にはミニバッチ勾配降下法を用いる。つまり訓練集合から複数の訓練データをランダムに選んで、それらのコスト関数を用いて最小点を求める。ランダムに選ばれた訓練データの集合をミニバッチと言うのだった。

まずミニバッチを用いて作られた計画行列 X と、各訓練データに対応したラベル $\boldsymbol{y}$ を用意する。例えば手書き数字の認識を目的にするならば、訓練データ $\boldsymbol{x}^{(i)}$ は、28×28 pixels の画素の黒さを並べた $28\times 28=784$ 成分のベクトルであり、$y^{(i)}$ はその画像に書かれた数字（答え）を表している。ミニバッチに選ばれた訓練データを $\left\{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(m')}\right\}$, $\boldsymbol{y}=\left[y^{(1)},\dots,y^{(m')}\right]^{\mathsf{T}}$ とすれば、この場合の計画行列は

$$X=\begin{bmatrix}\boldsymbol{x}^{(1)\mathsf{T}}\\ \vdots\\ \boldsymbol{x}^{(m')\mathsf{T}}\end{bmatrix}=\begin{bmatrix}x_1^{(1)} & \cdots & x_{784}^{(1)}\\ \vdots & & \vdots\\ x_1^{(m')} & \cdots & x_{784}^{(m')}\end{bmatrix}$$

のようになる。より一般には、スカラーの基底関数を $\phi_j(\boldsymbol{x})$, $j=1,\dots,k$ とすれば、計画行列 X の要素は $(X)_{ij}=\phi_j\left(\boldsymbol{x}^{(i)}\right)$ である。k はパラメータの数に対応している。

計画行列とパラメータを並べた行列 $W^{(1)}$ の行列積 $XW^{(1)}=U^{(1)}$ を計算し、各成分についての活性化関数 φ を施した行列を、隠れ層 $H=\varphi\left(XW^{(1)}\right)$ とする。教科書の例では活性化関数を ReLU としている。$W^{(1)}$ の行が訓練データの成分に対応していて、列が隠れ層内のノードに対応している。簡便さのためにバイアスは考えないとしているが、訓練データとパラメータのベクトルに $x_0^{(i)}=1$, $w_0^{(i)}=b_i$ のような第 0 成分を加えることで問題を解決できる。

隠れ層 H とパラメータ $W^{(2)}$ との行列積 $HW^{(2)}=U^{(2)}$ の各成分が、規格化されていない対数尤度である。これから交差エントロピー J_{MLE} を計算し、これをコスト関数とする[*1]。ただしより実用に近くするために、重さ減衰を加えたもの

$$J=J_{\mathrm{MLE}}+\lambda\left(\sum_{i,j}\left(W_{i,j}^{(1)}\right)^2+\sum_{i,j}\left(W_{i,j}^{(2)}\right)^2\right)$$

を最終的なコスト関数とする。ここまでの計算の流れをダイアグラムにしたのが図 1（教科書の図 6.11）である。交差エントロピーの計算経路を紫色で、重み減衰のそれを青色で色付けした。

コスト関数が最小となるようなパラメータ $W^{(1)}$, $W^{(2)}$ の値を求めることが、ここでの目的である。ミニバッチ勾配降下法を用いるので、コスト関数の勾配 $\nabla_{W^{(1)}}J$, $\nabla_{W^{(2)}}J$ を求める必要がある。図 1 に青色で示した、重み減衰の勾配を求めるのは簡単で、それは $2\lambda W^{(i)}$, $i=1,2$ である。

[*1] 詳しくは別の PDF に計算の流れを書いた。

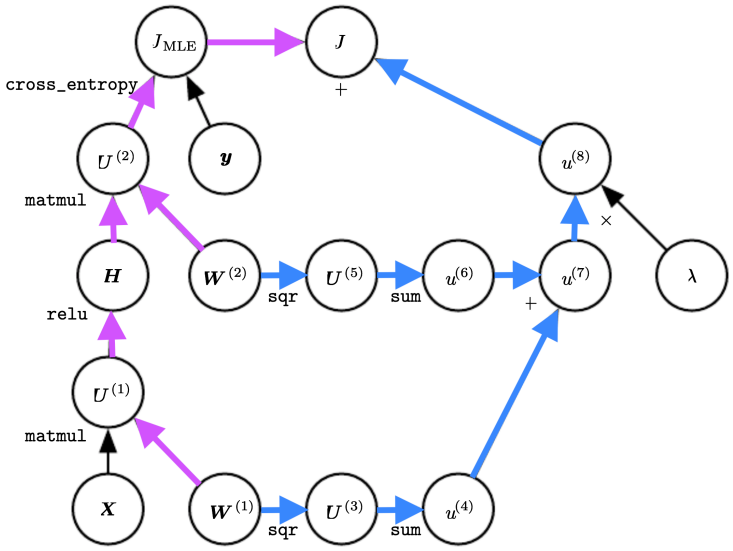

図 1　隠れ層 1 層の MLP の訓練をするときのグラフ

難しいのは、紫色で示した交差エントロピーの経路の勾配である。$G = \nabla_{U^{(2)}} J_{\mathrm{MLE}}$ とおく。まず $\nabla_{W^{(2)}} J_{\mathrm{MLE}}$ は

$$
\begin{aligned}
\left(\nabla_{W^{(2)}} J_{\mathrm{MLE}}\right)_{ij} &= \frac{\partial J_{\mathrm{MLE}}}{\partial W^{(2)}_{ij}} = \frac{\partial U^{(2)}_{kl}}{\partial W^{(2)}_{ij}} \underbrace{\frac{\partial J_{\mathrm{MLE}}}{\partial U^{(2)}_{kl}}}_{G_{kl}} = \frac{\partial \left(H_{km} W^{(2)}_{ml}\right)}{\partial W^{(2)}_{ij}} G_{kl} \\
&= H_{km} \underbrace{\frac{\partial W^{(2)}_{ml}}{\partial W^{(2)}_{ij}}}_{\delta_{mi}\delta_{lj}} G_{kl} = H_{ki} G_{kj} = \left(H^{\mathsf{T}}\right)_{ik} G_{kj} = \left(H^{\mathsf{T}} G\right)_{ij}
\end{aligned}
$$

という計算により $\nabla_{W^{(2)}} J_{\mathrm{MLE}} = H^{\mathsf{T}} G$ とわかる。ただし式変形にはアインシュタインの縮約を用いた。次に $\nabla_{W^{(1)}} J_{\mathrm{MLE}}$ について考える。$G' = \nabla_{U^{(1)}} J_{\mathrm{MLE}}$ とすれば、

$$
\begin{aligned}
\left(\nabla_{W^{(1)}} J_{\mathrm{MLE}}\right)_{ij} &= \frac{\partial J_{\mathrm{MLE}}}{\partial W^{(1)}_{ij}} = \frac{\partial U^{(1)}_{kl}}{\partial W^{(1)}_{ij}} \underbrace{\frac{\partial J_{\mathrm{MLE}}}{\partial U^{(1)}_{kl}}}_{G'_{kl}} = \frac{\partial \left(X_{km} W^{(1)}_{ml}\right)}{\partial W^{(1)}_{ij}} G'_{kl} \\
&= X_{km} \underbrace{\frac{\partial W^{(1)}_{ml}}{\partial W^{(1)}_{ij}}}_{\delta_{mi}\delta_{lj}} G'_{kl} = X_{ki} G'_{kj} = \left(X^{\mathsf{T}}\right)_{ik} G'_{kj} = \left(X^{\mathsf{T}} G'\right)_{ij}
\end{aligned}
$$

より $\nabla_{W^{(1)}} J_{\mathrm{MLE}} = X^{\mathsf{T}} G'$ とわかる。なお G' の ij 成分を計算すると、$\nabla_H J_{\mathrm{MLE}} = G {W^{(2)}}^{\mathsf{T}}$ であることは

上と同様の計算によってわかるので

$$G'_{ij} = \frac{\partial J_{\mathrm{MLE}}}{\partial U^{(1)}_{ij}} = \frac{\partial H_{kl}}{\partial U^{(1)}_{ij}} \underbrace{\frac{\partial J_{\mathrm{MLE}}}{\partial H_{kl}}}_{\left(GW^{(2)\mathsf{T}}\right)_{kl}} = \underbrace{\frac{\partial \varphi\left(U^{(1)}_{kl}\right)}{\partial U^{(1)}_{ij}}}_{\delta_{ki}\delta_{lj}\varphi'\left(U^{(1)}_{kl}\right)} \left(GW^{(2)\mathsf{T}}\right)_{kl}$$
$$= \varphi'\left(U^{(1)}_{ij}\right)\left(GW^{(2)\mathsf{T}}\right)_{ij} = \left[\varphi'\left(U^{(1)}\right) \odot \left(GW^{(2)\mathsf{T}}\right)\right]_{ij}$$

より $G' = \varphi'\left(U^{(1)}\right) \odot \left(GW^{(2)\mathsf{T}}\right)$ とわかる。活性化関数が ReLU の場合は $\varphi'(x) = \theta(x)$ であるから、$G' = \theta\left(U^{(1)}\right) \odot \left(GW^{(2)\mathsf{T}}\right)$ となる。ただし $\theta(x)$ はヘヴィサイドの階段関数で、正の引数に対して 1 を、負の引数に対して 0 を返す。引数がゼロに等しいときは定義されていない。

まとめると、コスト関数の勾配は

$$\nabla_{W^{(1)}} J = X^{\mathsf{T}} G' + 2\lambda W^{(1)},$$
$$\nabla_{W^{(2)}} J = H^{\mathsf{T}} G + 2\lambda W^{(2)}$$

となる。あとはこの勾配を用いて最小点を求めればよい。

MLP の計算コストにおいて支配的になるのは、行列積の計算コストである。重みの行列の数を w とすれば、順伝播、逆伝播ともに $O(w)$ くらいの積和演算が必要になる。上の例では、順伝播（訓練データからコスト関数を計算する向き）では重み行列との行列積を 2 回、逆伝播（コスト関数から勾配を計算する向き）でも $\nabla_{W^{(1)}} J$, $\nabla_{W^{(2)}} J$ を計算するのにそれぞれ 1 回ずつ、全体で 2 回の行列積を計算している。

また m' をミニバッチ内の訓練事例の数、n_h を隠れ層のユニット数とすれば、メモリのコストは $O(m'n_h)$ となる。

6.5.8　複雑さの要因

実際の実装では、上のシンプルな例よりも複雑になる。

まず我々が定義した演算（`op.bprop` など）は 1 つのテンソルを返す関数であったが、ソフトウェアの実装ではより多くのテンソルを返す関数が必要になる。

メモリの消費についてはあまり言及していなかった。バックプロパゲーションでは多くのテンソルの足し算の演算がよく行われる。第 1 段階でそれぞれの項を計算し、第 2 段階でそれらの総和を計算する、という素朴なやり方がある。しかしそれではメモリを過度に消費してボトルネックになり、全体の処理性能を低下させてしまう。このことは 1 つのバッファ（緩衝装置）を用意してやり、それぞれの項を計算しながらこれに足してゆくことで、回避できる。

実際のバックプロパゲーションの実装では、単精度、倍精度、整数型などの様々な数値表現を取り扱う必要がある。これらの数値表現をどう扱うかという方針については、注意して設計する。

勾配が定義できないような演算（例えば ReLU のゼロにおける勾配は未定義）があるので、そのような場合を追跡し、未定義なのかどうかを判断することが大事である。

現実では、専門的な事柄によって微分が複雑になるが、乗り越えられないわけではない。この章ではキーとなる手法を紹介したが、もっと多くの微妙な点があることを頭の片隅に入れておくことが重要である。