Initialization and Input Transformation

Nan Ye

School of Mathematics and Physics
The University of Queensland

Recall: Avoiding exploding/vanishing gradients

Good initialization

- Starting from large weights is bad — gradient can easily explode.
- Starting from 0 weights are bad — the hidden neurons will not be different from each other.
- Small random weights are often used in practice.

Recall: we experimented with different initialization strategies in Assignment 2...

Controlling initial activations and gradients

- What can we do to control the initial activations and gradient?
- Recall: a stochastic gradient g over a random mini-batch S

$$g = \frac{1}{|S|} \sum_{(\mathbf{x},y)\in S} \nabla_{\mathbf{w}} L((\mathbf{x}, y), \mathbf{w})$$

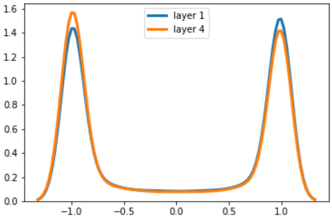
  g depends both on the training examples and the weights – the same for activation values.
- Two strategies to control the initial activations and gradients
  - (Weight) initialization: careful choice of $\mathbf{w}$
  - Input transformation: transform $\mathbf{x}$ (e.g. scale features)

Symmetry is Bad

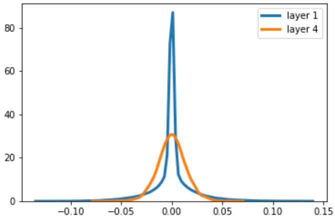
- If two units in the same hidden layer are initialised to have the same bias and incoming and outgoing weights, they will have the same gradients.
- Using gradient-based training, the two units will learn exactly the same features.
- In particular, initializing all parameters to a constant (like 0) is a bad choice.
- We break symmetry using random initial weights.

Naive Sampling

- Simply sampling from a distribution like $U[-1,1]$ (the uniform distribution on $[-1,1]$) does not work well.
- Example: distribution of activation and gradient values for neurons in the same layer for an MLP with tanh activation when random input examples with 0 mean and unit variance are given



(a) activation



(b) gradient

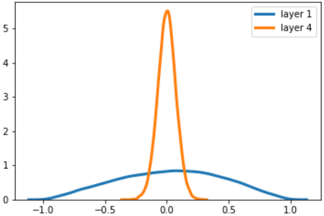
- Activation values in the deeper layers are close to the saturation values.
    - This becomes worse when depth increases — initial values are at a plateau (numerically, no gradient, no learning).
- Even when learning happens, it often converge to local minima with poor generalization performance.

Standard Uniform Initialization

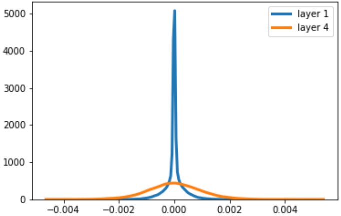
- A better initialization strategy need to take into account that if a hidden unit has a large fan-in n_{in} (number of inputs), its weighted input sum can blow up easily.
- A standard random initialization strategy is to sample the weights from

$$U\left(-\frac{1}{\sqrt{n_{\text{in}}}}, \frac{1}{\sqrt{n_{\text{in}}}}\right).$$

- The standard uniform initialization avoids saturating deeper neurons



(a) activation



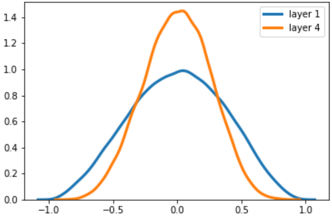
(b) gradient

- However, activation values in the same layer tend to have small variance for deeper layers, and gradient values show a reverse trend.
- Hard to set a good learning rate
  - Large learning rate $\Rightarrow$ weights with large gradients can overshoot.
  - Small learning rate $\Rightarrow$ weights with small gradients get stuck.

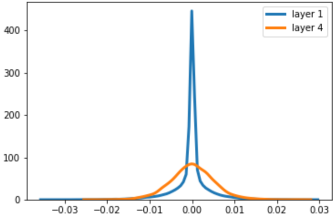
Xavier Initialization

- Xavier initialization makes a small change to the sampling interval to avoid either blowing up the activations or the gradients.
- For weights in a layer with n_{in} inputs and n_{out} outputs, each weight is sampled from

$$U\left(-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}\right).$$



(a) activation



(b) gradient

Glorot and Bengio, Understanding the difficulty of training deep feedforward neural networks, 2010

- As compared to standard initialization, Xavier initialization shows less drastic changes in the activation and gradient values across layers.

Explaining Xavier Initialization

Variance of products

- For two independent random variables W and X, we have

$$\mathrm{var}(WX) = \mathrm{var}(X)\,\mathrm{var}(W) + (\mathbb{E}\,X)^2\,\mathrm{var}(W) + (\mathbb{E}\,W)^2\,\mathrm{var}(X).$$

- If W and X also have zero mean, then

$$\mathrm{var}(WX) = \mathrm{var}(X)\,\mathrm{var}(W).$$

- In addition, if W_i's are i.i.d. copies of W and X_i's are uncorrelated copies of X, then

$$\mathrm{var}\left(\sum_{i=1}^{n} W_i X_i\right) = n\,\mathrm{var}(X)\,\mathrm{var}(W).$$

Assumptions

- Input variables are i.i.d with mean 0.
- All the weights are independently sampled from a distribution with mean 0.
- MLP with identity activation (or in a linear region of the activation function).

Forward propagation

- For forward propagation through a layer with fan-in n_{in}, if $X_1, \ldots, X_n$ are the input variables, and Y is the output, then

$$\mathrm{var}(Y) = n_{\text{in}} \, \mathrm{var}(W) \, \mathrm{var}(X).$$

- Thus a layer blows up the activation variance if $n_{\text{in}} \, \mathrm{var}(W) > 1$, and decreases it if $n_{\text{in}} \, \mathrm{var}(W) < 1$.
- Good weights: $\mathrm{var}(W) = 1/n_{\text{in}}$.

Backward propagation

- For backward propagation through a layer with fan-out n_{out}, the variance of the input derivatives $G_1, \ldots, G_{n_{\text{out}}}$, and the variance of the output derivative H are related by

$$\text{var}(H) = n_{\text{out}} \, \text{var}(W) \, \text{var}(G)$$

- Thus a layer blows up the previous layer's gradient variance if $n_{\text{out}} \, \text{var}(W) > 1$, and decreases it if $n_{\text{out}} \, \text{var}(W) < 1$.
- Good weights: $\text{var}(W) = 1/n_{\text{out}}$.

Middle ground

- To keep the variance of the activation and gradient unchanged across layers, take the middle ground between $1/n_{\text{in}}$ and $1/n_{\text{out}}$, and set

$$\text{var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}.$$

- If W follows a uniform distribution $U[-b, b]$, then $\text{var}(W) = \frac{b^2}{3}$, thus $b = \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}$.
- If W follows a normal distribution $N(0, \sigma^2)$, then $\sigma = \sqrt{\frac{2}{n_{\text{in}}+n_{\text{out}}}}$.

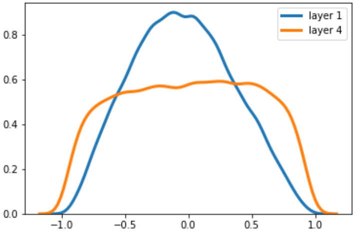
He Initialization

- A key assumption is Xavier initialization is that we are working in the linear region of the activation function.
- This is not satisfied if we are using ReLU.
- Using a similar argument as for Xavier initialization, a good initialization strategy is to sample weights from

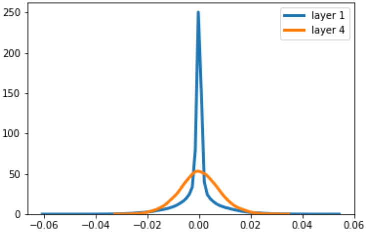
$$U\left(-\sqrt{\frac{6}{n_{\text{in}}}}, \sqrt{\frac{6}{n_{\text{in}}}}\right).$$

- This is used in ResNet.

He, Zhang, Ren, and Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015



(a) activation



(b) gradient

Good Initialization

- Diversity
  - Different initial weights will allow neurons to evolve into different features.
- Stability
  - The activation variance shouldn't change much across layers.
  - The gradient variance shouldn't change much across layers.

Input Transformations

Centering inputs

- It usually helps to center each component of the input vector to 0 over the whole training set.
- When using an activation function symmetric around 0, this can help to keep the activations around 0.
- It makes training more robust to poor choice of initial weights.
  - e.g., even if we only choose positive weights, we won't have a very large weighted input sum at deep layers.

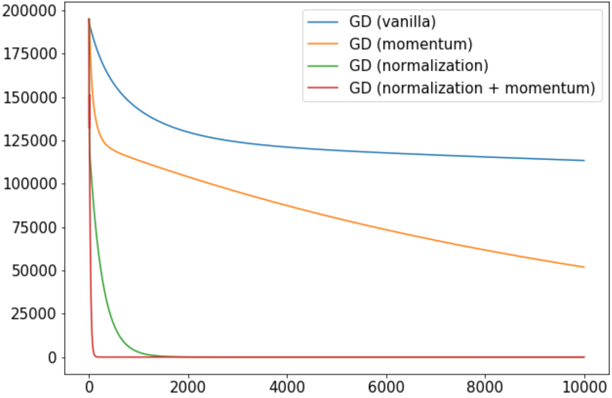
Scaling the inputs

- It usually helps to scale each component of the input vector to have unit variance over the whole training set.
- This makes the error surface less elongated but more circular, and thus easier to navigate.

- We often normalize each component of the input vector to have 0 mean and unit variance over the training set.
- This can make vanilla SGD work very well, even when momentum/Nesterov SGD converge slowly on the original dataset.

Case study: OLS on the diabetes dataset

- We have seen in the tutorial that vanilla gradient descent converges very slowly on the diabetes dataset.
- Standard momentum has faster convergence rate.
- Normalization makes convergence much faster.
- Even better: normalization + momentum.



- x-axis: number of iterations
- y-axis: MSE for the estimated parameters

Decorrelation

- Sometimes input variables are highly correlated, and simple normalization does not work very well.
- If we can decorrelate the input components, that can significantly speed up convergence.
- This is not very practical for high-dimensional data though.

Good Inputs

- Zero mean per dimension
- Unit variance per dimension
- Uncorrelated dimensions

Your Turn

Which of the following statement is correct? (Multiple choice)

(a) A desirable property of an initialization strategy is that the variance of the activation values do not converge to 0 at deeper layers.

(b) Any initialization strategy that initializes weights by sampling from an interval of the form $U[-\frac{c}{\sqrt{n_{in}}}, \frac{c}{\sqrt{n_{in}}}]$, where $c > 0$ is a user-specified constant, is guarantteed to keep the gradient variance constant across layers.

(c) Input normalization can potentially make learning a model significantly easier.

What You Need to Know...

- Good initialization
  - Diversity and stability are important
  - Xavier initialization, He initialization
- Good inputs
  - "Easy data": each dimension has zero mean and unit variance, and uncorrelated with other dimension
  - Making data easy: centering, unit variance, decorrelation.