Оптимизация сети

<https://habr.com/ru/articles/318970/>

 Обучение сети заключается в подборе параметров сети таким образом, чтобы минимизировать значение функции потерь (функции ошибок, целевой функции). Для этого обычно используется метод градиентного спуска.

 Пусть $F\left(w\right)$ – целевая функция, зависящая от параметров нейронной сети (весовые коэффициенты и смещения нейронов сети). Рассмотрим разложение целевой функции в ряд Тэйлора вблизи оптимальных значений параметров $w$:

$$F\left(w+p\right)=F\left(w\right)+\left[g\left(w\right)\right]^{T}p+\frac{1}{2}p^{T}Hp+…$$

где

$$g\left(w\right)=∇F\left(w\right)=\left[\frac{∂F}{∂w\_{1}},\frac{∂F}{∂w\_{2}}, …\right]^{T}$$

– вектор градиента,

$$H\left(w\right)=\left[\begin{matrix}\frac{∂^{2}F}{∂w\_{1}∂w\_{1}}&\cdots &\frac{∂^{2}F}{∂w\_{1}∂w\_{n}}\\\vdots &\ddots &\vdots \\\frac{∂^{2}F}{∂w\_{n}∂w\_{1}}&\cdots &\frac{∂^{2}F}{∂w\_{n}∂w\_{n}}\end{matrix}\right]$$

– матрица вторых производных, называемая гессианом.

 Ограничимся первыми производными в разложении. В этом случае вектор градиента указывает направление изменения параметров сети:

$$F\left(w^{new}\right)-F\left(w^{old}\right)=\left[g\left(w^{old}\right)\right]^{T}∆w$$

$$⟹w^{new}=w^{old}-λg\left(w^{old}\right)$$

$λ$ – параметр обучения. Малое значение параметра обучения – медленное обучение, Большое значение – «перепрыгиваем» через точку минимума.

 Способы выбора пакетов из обучающего набора:

1. Batch – целевая функция $F\left(w\right)$ вычисляется на всех примерах обучающей последовательности.
2. Mini-batch - обучающей последовательность разбивается на группы, $F\left(w\right)$ вычисляется на всех примерах группы.
3. Стохастический – $F\left(w\right)$ вычисляется на одном примере, примеры из обучающей последовательности выбираются случайным образом.



***Оптимизация метода градиентного спуска.***

 Зачем нужна оптимизация? Обычный метод градиентного спуска выглядит следующим образом:

$$w\_{i+1}=w\_{i}-λg\left(w\_{i}\right)$$

$i$ *–* номер итерации.



Проблемы:

1. Застревание в локальных минимумах или седловых точках целевой функции, для функции большого числа переменных может быть очень много.
2. Сложный ландшафт целевой функции: плато чередуются с регионами сильной нелинейности. Производная на плато практически равна нулю, а внезапный обрыв приводит к резкому изменению градиента.
3. Слишком маленькая скорость обучения заставляет алгоритм сходиться очень долго и застревать в локальных минимумах, слишком большая — «пролетать» узкие глобальные минимумы или вовсе расходиться

**Идея оптимизации:** учитывать не только текущее состояние, но и предыдущие значения параметров сети.

 Пусть $s\_{i}$ среднее значение некоторого параметра за предыдущие итерации. Для упрощения вычислений можно использовать экспоненциальное скользящее среднее:

$$s\_{i}=γs\_{i-1}+(1-γ)x$$

$x$ – текущее значение используемого параметра, например $w\_{i}$.

Коэффициент $0<γ\leq 1$ задаёт размер интервала усреднения, чем меньше значение $γ$ – тем короче интервал усреднения.

 Если накапливать градиент целевой функции, то модифицированный метод градиентного спуска примет вид (**Nesterov Accelerated Gradient**):

$$s\_{i}=γs\_{i-1}+(1-γ)g\left(w\_{i-1}\right)$$

$$w\_{i}=w\_{i-1}-λs\_{i}$$

Напомним, что $g\left(w\right)=∇F\left(w\right)$ – градиент целевой функции в точке $w$. Возможны два варианта: посчитать значение параметра на «старом» значении градиента затем найти градиент или наоборот.

 Метод оптимизации **RMSprop** (root mean square propagation):

В этом методе используется усреднённый квадрат градиента:

$$E\left[g^{2}\right]\_{i}=γE\left[g^{2}\right]\_{i-1}+(1-γ)g^{2}\left(w\_{i-1}\right)$$

$$w\_{i}=w\_{i-1}-\frac{λ}{\sqrt{E\left[g^{2}\right]\_{i}+ε}}g\left(w\_{i-1}\right)$$

 *Идея ясна. Разница в выборе параметров для усреднения и начальных значениях при старте итераций.*