Averaging Weights Leads to Wider Optima and Better Generalization

Безрукова Анастасия, 193 группа

Вспомнить все

SGD в нейронных сетях

$$\Delta\theta(t) = -\eta\nabla_\theta J(\theta(t)),$$

$$\theta(t+1) = \theta(t) + \Delta\theta(t) = \theta(t) - \eta\nabla_\theta J(\theta(t)),$$

- Batch gradient descent

  При этом подходе градиент функционала обычно вычисляется как сумма градиентов, учитывая каждый элемент обучения сразу.

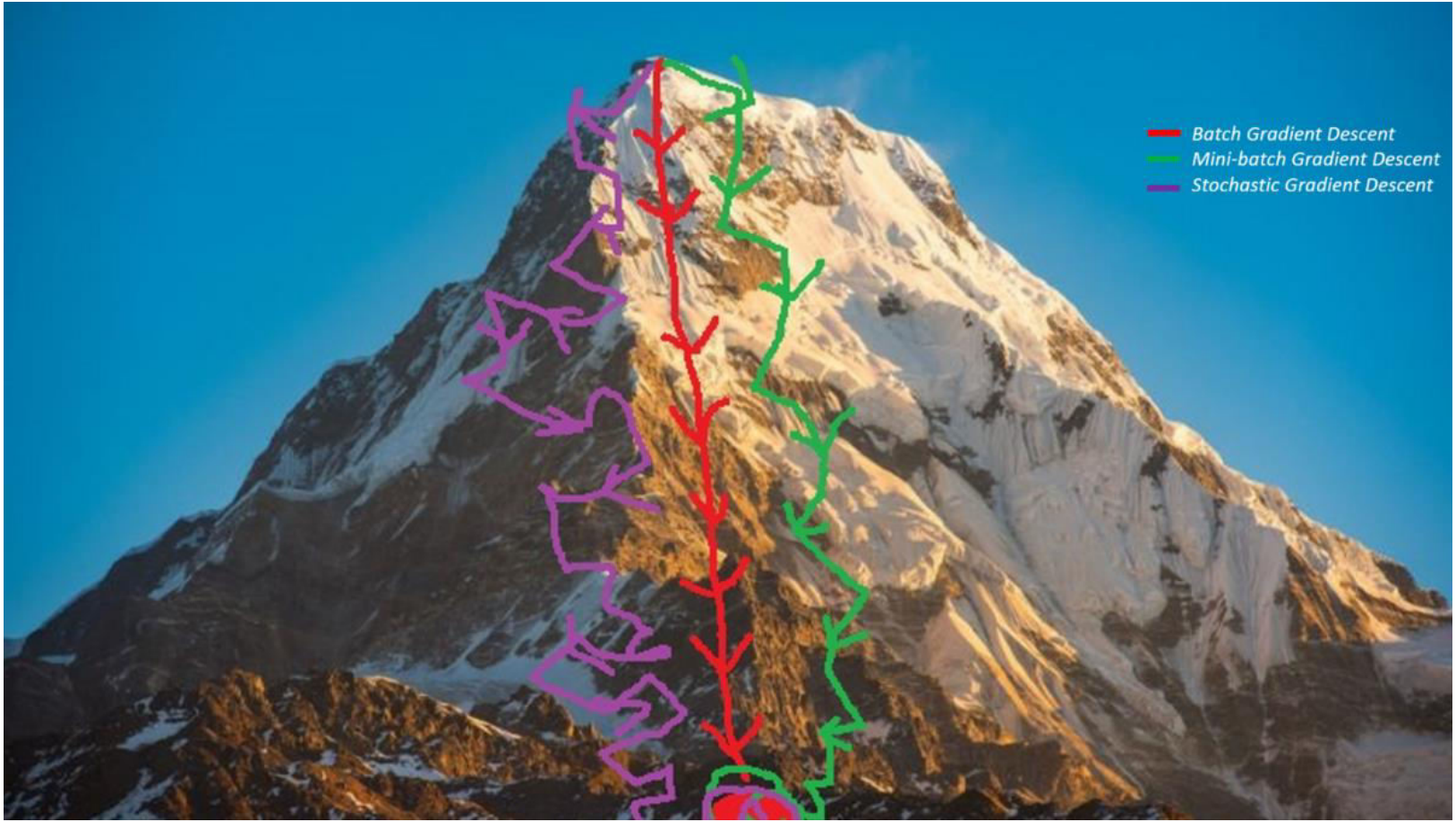
- Stochastic gradient descent

  Этот подход подразумевает корректировку весов нейронной сети, используя аппроксимацию градиента функционала, вычисленную только на одном случайном обучающем примере из выборки.

- Mini-batch gradient descent

  Гибрид двух подходов SGD и BatchGD, в этом варианте изменение параметров происходит, беря в расчет случайное подмножество примеров обучающей выборки.

В общем случае, при работе с нейронными сетями, веса оптимизируют стохастическим градиентным спуском или его вариацией.


Batch Gradient Descent
Mini-batch Gradient Descent
Stochastic Gradient Descent

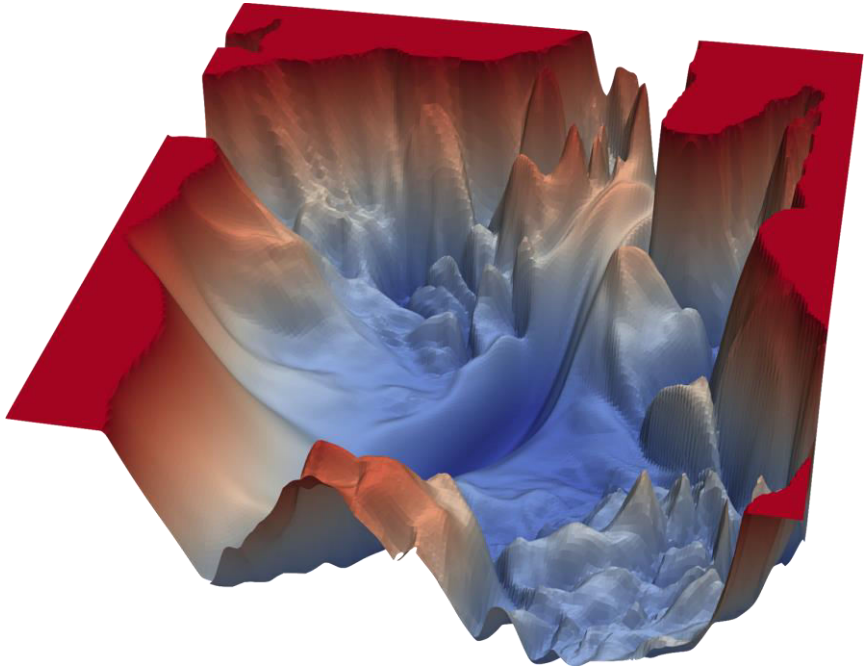
Поверхность потерь

Поверхность потерь можно понимать как множество значений функции потерь при всех возможных значениях весов.

Каждой точке в весовом пространстве (точка в весовом пространстве = некоторый вектор весов) сопоставляется значение функционала ошибки

Цель обучения - найти локальный оптимум, но тут есть две особенности:

1. локальных оптимумов существует некоторое множество
2. не все локальные оптимумы достаточно хороши для обучения



Fast Geometric Ensembling

Сначала мы инициализируем копию сети с весами *w_set*, равными весам обученной сети *w*. Теперь, чтобы заставить нас отойти от *w* без существенного снижения точности прогнозирования, мы обучаем модель, циклически меняя *learning rate* вот так:

$$\alpha(i) = \begin{cases} (1-2t(i))\alpha_1 + 2t(i)\alpha_2 & 0 < t(i) \leq \frac{1}{2} \\ (2-2t(i))\alpha_2 + (2t(i)-1)\alpha_1 & \frac{1}{2} < t(i) \leq 1 \end{cases}$$

В середине каждого цикла скорости обучения, когда скорость обучения достигает минимального значения, полученные веса запоминаются. Когда обучение заканчивается, собранные модели объединяются в ансамбль.

Algorithm 1 Fast Geometric Ensembling

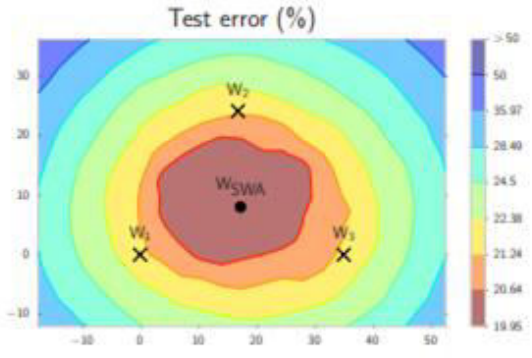
Require:
weights $\hat{w}$, LR bounds α_1, α_2,
cycle length c (even), number of iterations n
Ensure: ensemble
$w \leftarrow \hat{w}$ {Initialize weight with $\hat{w}$}
ensemble $\leftarrow [\,]$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\mathrm{mod}(i, c) = c/2$ **then**
    ensemble $\leftarrow$ ensemble $+ [w]$ {Collect weights}
  end if
end for

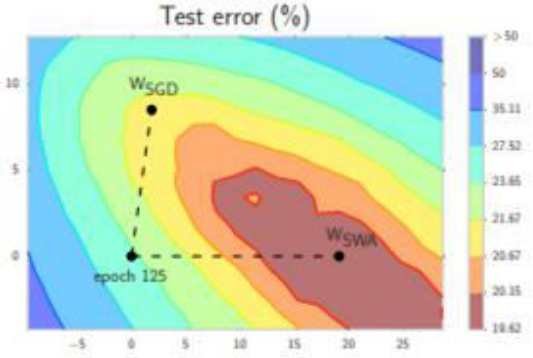
SWA Algorithm

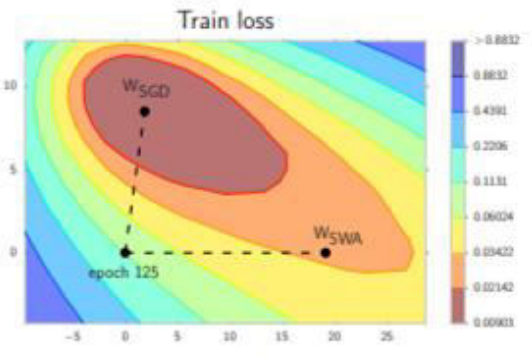
Предпосылки

- В экспериментах было выявлено, что веса сетевых ансамблей, созданных *FGE*, обычно находятся на периферии наиболее желательных решений, в то время как лучшее решение находится в центре этой области
- В то время как ансамбль *FGE* может быть обучены за то же время, что и одна модель, вычисление прогноза для ансамбля из *k* моделей требуют в *k* раз больше вычислений.
- *SGD* обычно находит острый оптимум (*sharp minimum*), в то время как для повышение обобщающей способности модели хотелось бы находить широкий оптимум (*flat minimum*)

Предпосылки







Преимущества SWA

Мы покажем, что

- полученные с помощью SGD модели никогда не достигнут оптимальных с точки зрения функции ошибки точек, в то время как при усреднении веса попадание в оптимальную область становится возможным
- SWA аппроксимирует FGE, однако экономит вычислительные мощности
- SWA находит более широкие оптимумы, чем SGD и это связано с асимметрией функции потерь на направлениях, соединяющих SWA и SGD
- SWA обеспечивает заметное улучшение при обучении широкому спектру архитектур по нескольким последовательным критериям.
- SWA прост в реализации и не требует больших вычислительных затрат

Анализ траекторий SGD

Проведем эксперимент: запустим SGD с константным значением и с циклически меняющимся значением *learning rate:*

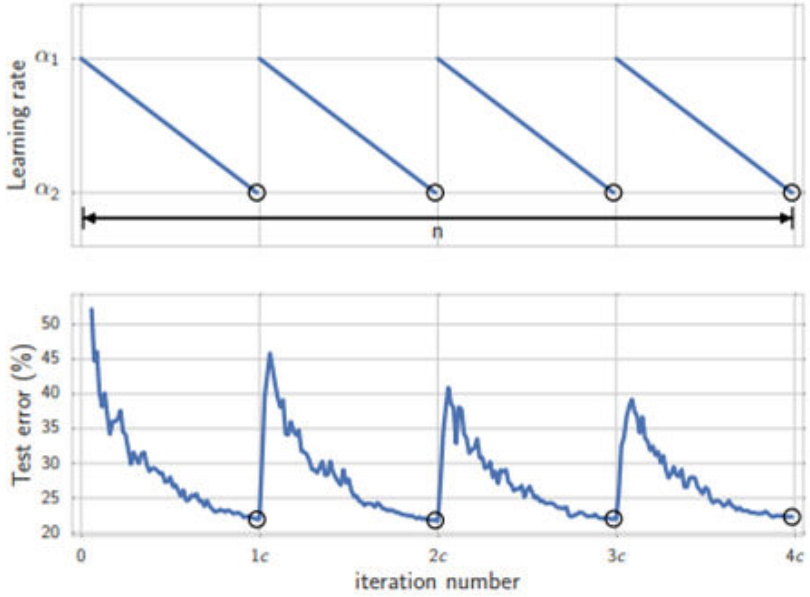
1. циклическое изменение learning rate определяется тремя параметрами: длиной ***c***, начальным и конечным значениями цикла ***a1*** и ***a2.*** В течение цикла learning rate меняется по формуле

$$\alpha(i) = (1 - t(i))\alpha_1 + t(i)\alpha_2,$$
$$t(i) = \frac{1}{c}\left(\mathrm{mod}(i-1, c) + 1\right).$$

1. константную learning rate возьмем как ***a1***

Анализ траекторий SGD

Figure 2: **Top**: cyclical learning rate as a function of iteration. **Bottom**: test error as a function of iteration for cyclical learning rate schedule with Preactivation-ResNet-164 on CIFAR-100. Circles indicate iterations corresponding to the minimum learning rates.



Анализ траекторий SGD

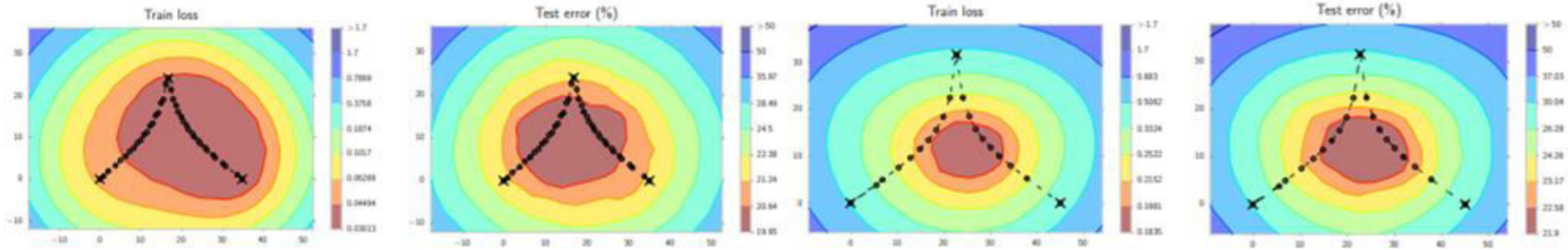


Figure 3: The L_2-regularized cross-entropy train loss and test error surfaces of a Preactivation ResNet-164 on CIFAR-100 in the plane containing the first, middle and last points (indicated by black crosses) in the trajectories with (**left two**) cyclical and (**right two**) constant learning rate schedules.

Анализ траекторий SGD: выводы

1. оба метода исследуют точки, близкие к периферии набора высокопроизводительных сетей.
2. оба метода имеют высокую точность
3. основное различие между двумя подходами заключается в том, что индивидуальные предложения SGD с циклическим графиком скорости обучения в целом намного точнее, чем предложения SGD с фиксированной скоростью обучения.
4. имеет место смещение - поверхности потерь схожи, но не выровнены

ИТОГ: более надежные центральные точки в наборе высокопроизводительных сетей могут привести к лучшему обобщению

Алгоритм SWA

Зафиксируем ***B*** (budget) - количество эпох, необходимых для обучения данного DNN с помощью обычной процедуры обучения

Мы стартуем с предварительной модели *w*, обученной на весь бюджет или некоторую его долю (0.75*B*). Дальше - 2 варианта:

1. при использовании циклической скорости обучения мы фиксируем модели, соответствующие минимальным значениям скорости обучения
2. при постоянной скорости обучения мы фиксируем модели в каждую эпоху

В конце мы усредняем веса всех фиксированных сетей, чтобы получить нашу окончательную модель *w_swa*

Algorithm 1 Stochastic Weight Averaging

Require:
weights $\hat{w}$, LR bounds α_1, α_2,
cycle length c (for constant learning rate $c = 1$), number of iterations n

Ensure: w_{SWA}

$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\mathrm{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
{Compute BatchNorm statistics for w_{SWA} weights}

Вычислительная сложность

Вместо множества моделей нам требуется только две:

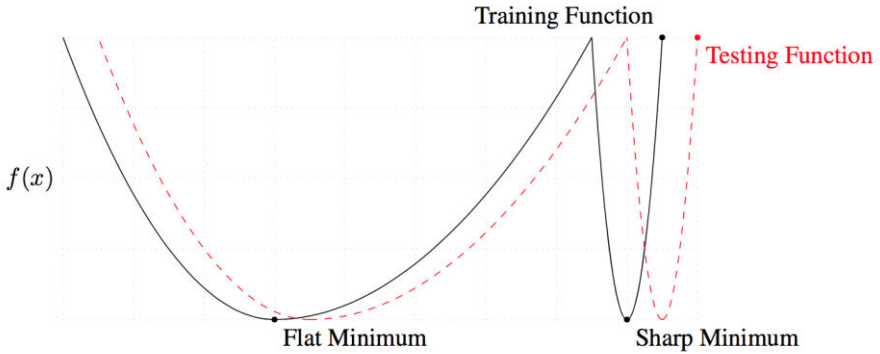
- первая модель, хранящая среднее скользящее значение весов модели. Она будет окончательной моделью, которая будет использоваться для прогнозов.
- вторая модель исследует пространство весов с помощью цикличных повторений, пересекая его

В конце каждого цикла обучения текущие веса второй модели используются для обновления веса модели скользящего среднего, принимая взвешенное среднее между старыми весами скользящего среднего и новым набором весов из второй модели. Следуя этому подходу, нам нужно обучать только вторую модель.

$$w_{\mathrm{SWA}} \leftarrow \frac{w_{\mathrm{SWA}} \cdot n_{\mathrm{models}} + w}{n_{\mathrm{models}} + 1,},$$

Оптимумы и SWA

В работах Кескара и др.[2017] и Гарипова и др. [2018]показывается, что ширина локального оптимума связана с обобщением. Общее объяснение важности ширины состоит в том, что поверхности потерь при обучения и при тестировании смещены относительно друг друга, и поэтому желательно сходиться к режимам широких оптимумов, которые остаются приблизительно оптимальными при небольших возмущениях.



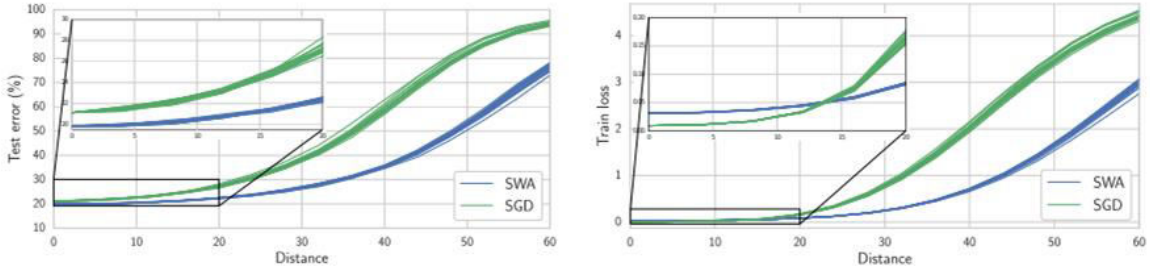


Figure 4: (**Left**) Test error and (**Right**) L_2-regularized cross-entropy train loss as a function of a point on a random ray starting at SWA (blue) and SGD (green) solutions for Preactivation ResNet-164 on CIFAR-100. Each line corresponds to a different random ray.

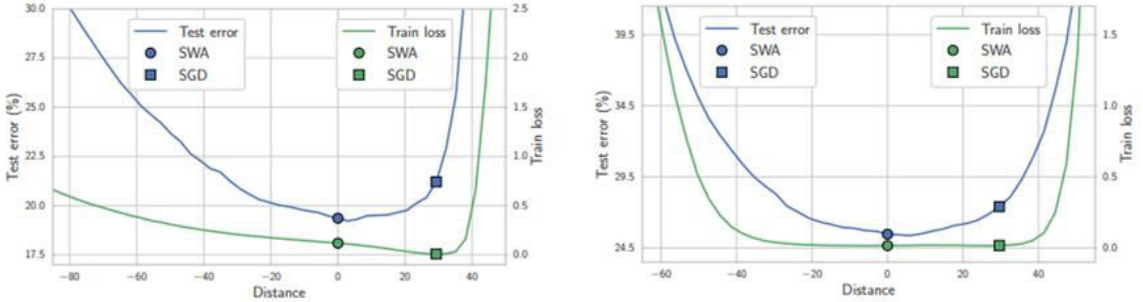


Figure 5: L_2-regularized cross-entropy train loss and test error as a function of a point on the line connecting SWA and SGD solutions on CIFAR-100. **Left**: Preactivation ResNet-164. **Right**: VGG-16.

Связь в FGE

Несмотря на то, что подходы FGE и SWA различны, оказывается, что найденные этими способами решения обладают одинаковыми свойствами

Пусть **f(·)** обозначает предсказания нейронной сети, параметризованные весами *w* (мы будем считать, что **f** является скалярной дважды непрерывно дифференцируемой функцией относительно *w),* тогда:

$$f(w_j) = f(w_{\text{SWA}}) + \langle \nabla f(w_{\text{SWA}}), \Delta_j \rangle + O(\|\Delta_j\|^2),$$

$$\bar{f} - f(w_{\text{SWA}}) = \frac{1}{n} \sum_{i=1}^{n} \left(\langle \nabla f(w_{\text{SWA}}), \Delta_i \rangle + O(\|\Delta_i\|^2) \right)$$

$$= \left\langle \nabla f(w_{\text{SWA}}), \frac{1}{n} \sum_{i=1}^{n} \Delta_i \right\rangle + O(\Delta^2) = O(\Delta^2),$$

$$f(w_i) - f(w_j) = \langle \nabla f(w_{\text{SWA}}), \Delta_i - \Delta_j \rangle + O(\Delta^2),$$

SWA: обоснование

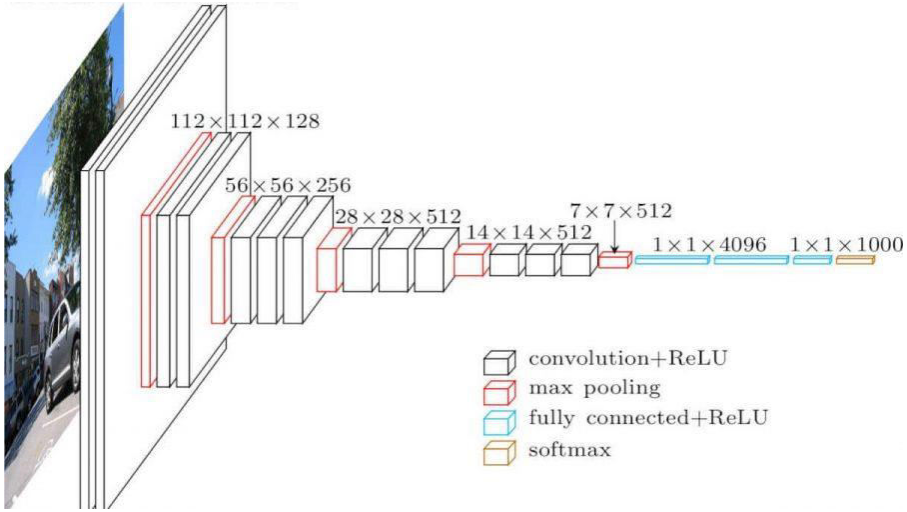
Известно, что существует набор точек, которые позволяют достичь низких потерь при обучении. Запустив SGD с высоким постоянным или циклическим графиком, мы пересекаем поверхность этого набора. Затем, усредняя соответствующие итерации, мы получаем возможность перемещаться внутри набора. Это наблюдение объясняет как скорость сходимости, так и обобщение.

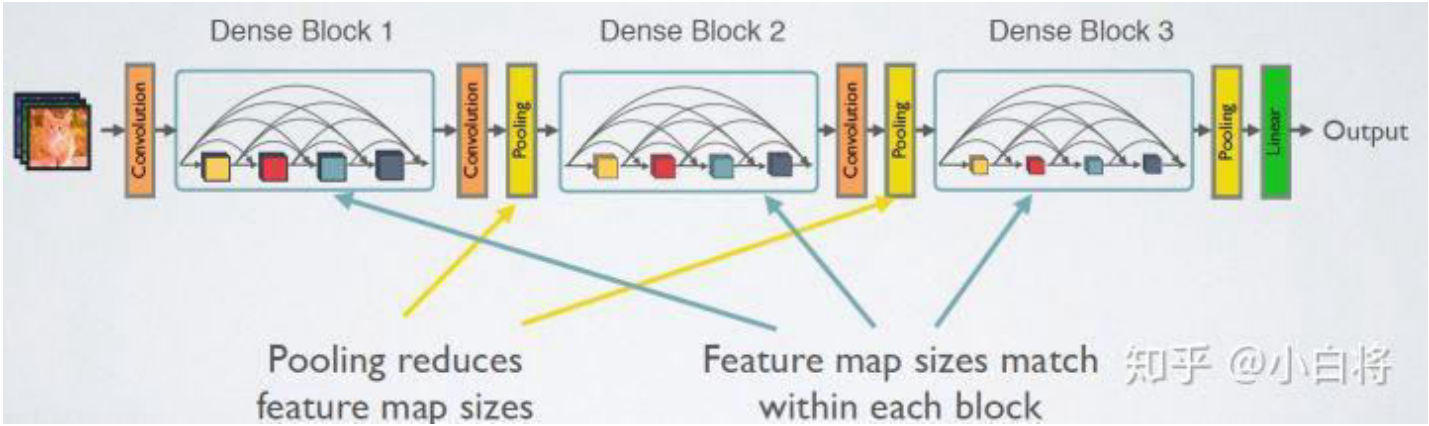
Усреднение может переместиться в более центральную точку. Напротив, обычный SGD с затухающим графиком будет сходиться к точке на периферии этого набора.

Эксперименты

Краткая методичка

- CIFAR 10 - это набор данных компьютерного зрения для универсального распознавания объектов, содержит 60 000 цветных изображений 32 X 32 RGB с 10 категориями.
- CIFAR 100 - это набор данных компьютерного зрения для универсального распознавания объектов, содержит 100 классов (20 суперклассов), по 600 изображений в каждом.
- ImageNet – открытый набор данных, предоставляемый в рамках конкурса ILSVRC
- VGG16, Preactivation-ResNet, Wide ResNet28-10, PyramidNet272, Shake-Shake2x64d, ResNet-50, ResNet152 and DenseNet-161 - нейросети


112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
1×1×1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax


Dense Block 1
Dense Block 2
Dense Block 3
Convolution
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output
Pooling reduces feature map sizes
Feature map sizes match within each block
知乎 @小白将

Эксперименты с CIFAR Datasets

Для каждой модели мы определяем бюджет как количество эпох, необходимых для обучения модели до сходимости с обычным обучением SGD, так что мы не видим улучшения с SGD за пределами этого бюджета. Для VGG, Preactivation ResNet and Wide ResNet - одинаковый бюджет, для Shake-Shake и PyramidNet - определяем отдельно

1. Для моделей VGG, Wide ResNet и Preactivation-ResNet мы сначала проводим стандартное обучение SGD за 75% бюджета обучения, а затем запускаем SWA с бюджетом 0,25, 0,5 и 0,75, чтобы завершить обучение в рамках бюджетов 1, 1,25 и 1,5 соответственно
2. Для архитектур Shake-Shake и Pyramid Net мы используем полный бюджет на этапе инициализации, а затем тренируемся с циклическим графиком скорости обучения для бюджетов 0,25 и 0,5.
3. Всего проводится 3 запуска, по их итогам считается среднее и стандартное отклонение *test accuracy*

Эксперименты с CIFAR Datasets

DNN (Budget)	SGD	FGE (1 Budget)	SWA		
			1 Budget	1.25 Budgets	1.5 Budgets
		CIFAR-100			
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
ResNet-164 (150)	78.49 ± 0.36	79.84	79.77 ± 0.17	80.18 ± 0.23	80.35 ± 0.16
WRN-28-10 (200)	80.82 ± 0.23	82.27	81.46 ± 0.23	81.91 ± 0.27	82.15 ± 0.27
PyramidNet-272 (300)	83.41 ± 0.21	–	–	83.93 ± 0.18	84.16 ± 0.15
		CIFAR-10			
VGG-16 (200)	93.25 ± 0.16	93.52	93.59 ± 0.16	93.70 ± 0.22	93.64 ± 0.18
ResNet-164 (150)	95.28 ± 0.10	95.45	95.56 ± 0.11	95.77 ± 0.04	95.83 ± 0.03
WRN-28-10 (200)	96.18 ± 0.11	96.36	96.45 ± 0.11	96.64 ± 0.08	96.79 ± 0.05
ShakeShake-2x64d (1800)	96.93 ± 0.10	–	–	97.16 ± 0.10	97.12 ± 0.06

Эксперименты с ImageNet

В ImageNet мы экспериментировали с ResNet-50, ResNet 152 и DenseNet-161. Для этих архитектур мы использовали предварительно обученные модели из PyTorch.torchvision. Для каждой из моделей мы провели SWA в течение 5/10 эпох с циклическим графиком скорости обучения с одинаковыми параметрами для всех моделей и сообщили о среднем значении и стандартном отклонении ошибки теста, усредненной за 3 прогона.

		SWA	
DNN	SGD	5 epochs	10 epochs
ResNet-50	76.15	76.83 ± 0.01	76.97 ± 0.05
ResNet-152	78.31	78.82 ± 0.01	78.94 ± 0.07
DenseNet-161	77.65	78.26 ± 0.09	78.44 ± 0.06

График обучения: **α1 = 0.001, α2 = 10−5 and c = 1.**

Эксперименты с Fashion-MNIST

Vetrov, Isaev, Mirieva [2019]

Сравним стохастическое усреднение весов с другими методами оптимизации. В качестве данных, на которых будем сравнивать методы оптимизации, возьмем базу данных «Fashion-MNIST» – базу классификации с 10 классами различных видов одежды. В качестве базовой нейронной сети возьмем двуслойную нейронную сеть с 256 нейронами на каждом из слоев. Сравнивать будем два показателя: точность и время обучения.

Таблица 1.

Сравнение методов оптимизации

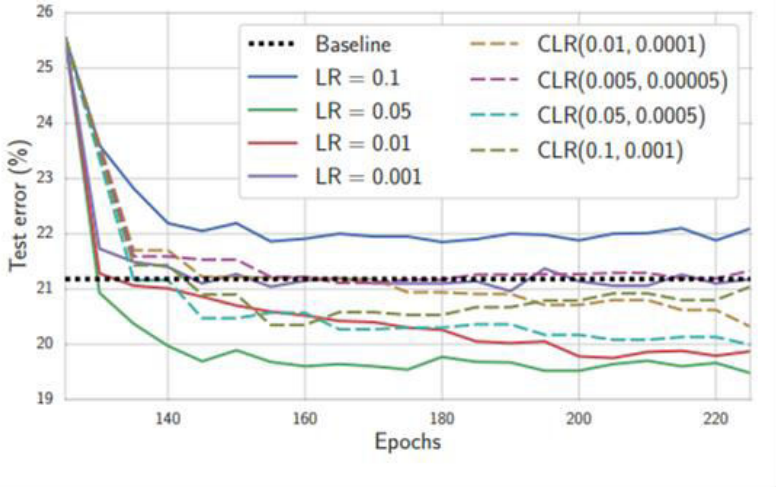
	Accuracy (точность)	Время
SGD	0,871	3min 46s
Adam	0,898	4min 27s
SnapShot	0,899	3min 35s
SWA	0,901	4min 5s

Эксперименты с learning rate

Второй тип эксперимента - влияние *learning rate* на SWA

1. Мы проводим эксперименты с ResNet-164 на CIFAR-100.
2. Для всех сетей мы используем одну и ту же инициализацию из модели, обученной 125 эпохам с использованием обычного обучения SGD. В качестве baseline мы используем полностью обученную модель, обученную с помощью обычного SGD в течение 150 эпох

ИТОГ: более агрессивный график постоянной скорости обучения приводит к более быстрой конвергенции SWA. В экспериментах обнаружено , что установка скорости обучения на некоторое промежуточное значение между наибольшей и наименьшей скоростью обучения дает наилучшие результаты.

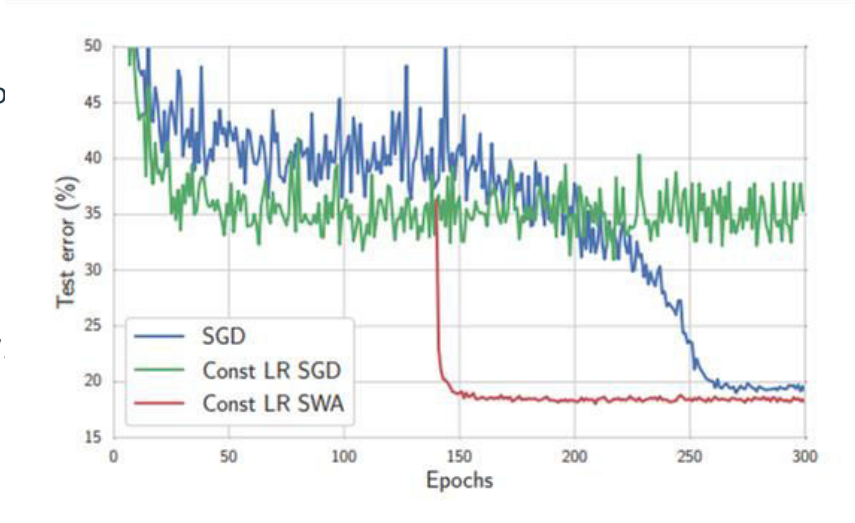


Эксперименты с DNN

Покажем, что можно обучать DNN с нуля с фиксированной скоростью обучения с использованием SWA.

1. Мы запустили SGD с фиксированной скоростью обучения a1 = 0,05 в широкой сети ResNet-28-10 в течение 300 эпох после случайной инициализации на CIFAR100.
2. Затем мы усреднили веса в конце каждой эпохи, начиная с эпохи 140 и до конца тренировки. Окончательная точность тестирования этой модели SWA составила 81,7

ИТОГ: есть возможность обучать DNN методом SWA с константным learning rate, в то время как у метода SGD нет такой опции



Заключение

- Используя метод, описанный выше, можно получать хорошо обобщенную модель – модель, которая показывает хорошую точность не только на обучающей выборке, но и на тестовой.
- Стохастическое усреднение веса может применяться к любой архитектуре и набору данных.
- SWA позволяет исследовать более широкое пространство параметров модели, в сравнении с другими методами оптимизации.
- В конце обучения мы получаем одну модель, но ее производительность приближается к FGE

Спасибо за внимание!