ВАРИАНТЫ ГРАДИЕНТНОГО СПУСКА

Сергей Николенко

СПбГУ — Санкт-Петербург

17 сентября 2020 г.

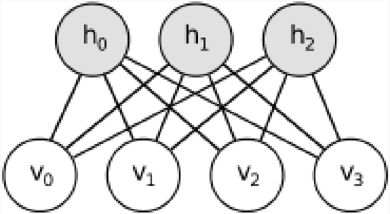
Random facts:

- 17 сентября 1683 г. Антони ван Левенгук написал письмо в Королевское общество, в котором описал «animalcules» («малых животных»), увиденных им в дождевой воде через свежеизобретённый микроскоп
- 17 сентября 1787 г. в Филадельфии была подписана конституция Соединённых Штатов
- 17 сентября 1916 г. во время первой мировой войны близ Камбре свой первый воздушный бой выиграл Манфред фон Рихтгофен
- 17 сентября 1922 г. в Берлине состоялся первый в мире публичный показ звукового фильма; демонстрация системы Tri-Ergon состоялась во время премьеры фильма «Поджигатель» (Der Brandstifter)
- 17 сентября 1991 г. Линус Торвальдс опубликовал исходный код ядра Linux, версия 0.01
- 17 сентября 2013 г. состоялся релиз игры Grand Theft Auto V; за первый день игра заработала больше полумиллиарда долларов

Инициализация весов

Предобучение без учителя

- Революция глубокого обучения началась с *предобучением без учителя* (unsupervised pretraining).
- Главная идея: добраться до хорошей области пространства весов, затем уже сделать fine-tuning градиентным спуском.
- Ограниченные машины Больцмана (restricted Boltzmann machines):



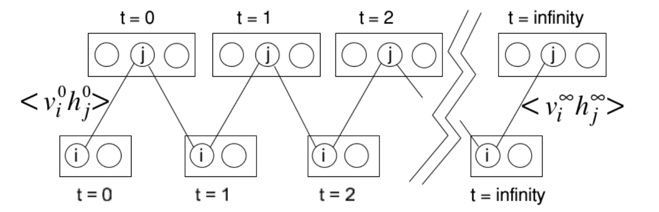
Предобучение без учителя

- Это ненаправленная графическая модель, задающая распределение

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})},\ \text{где}$$

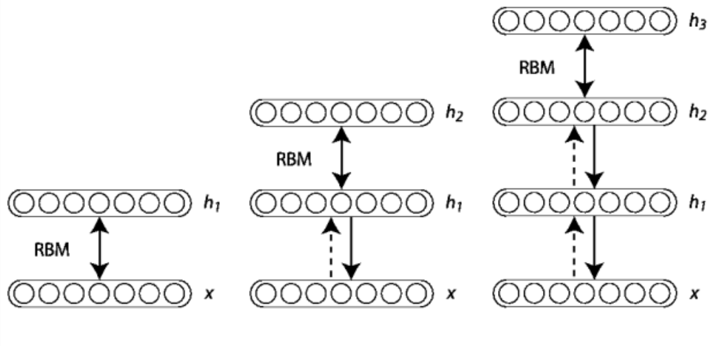
$$E(\mathbf{v},\mathbf{h}) = -\mathbf{b}^\top\mathbf{v} - \mathbf{c}^\top\mathbf{h} - \mathbf{h}^\top W\mathbf{v}.$$

- Обучают алгоритмом Contrastive Divergence (приближение к сэмплированию по Гиббсу).



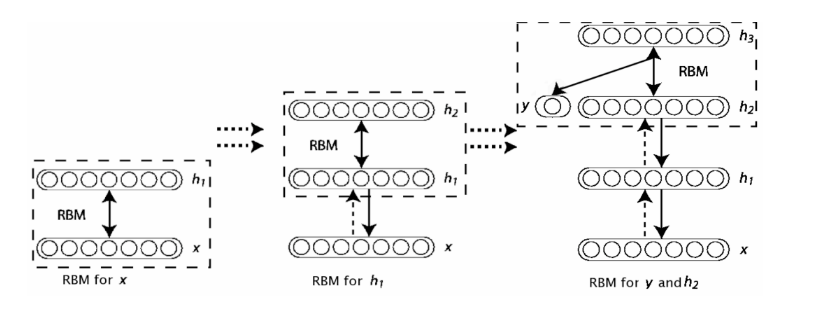
Предобучение без учителя

- Из RBM можно сделать глубокие сети, поставив одну на другую:



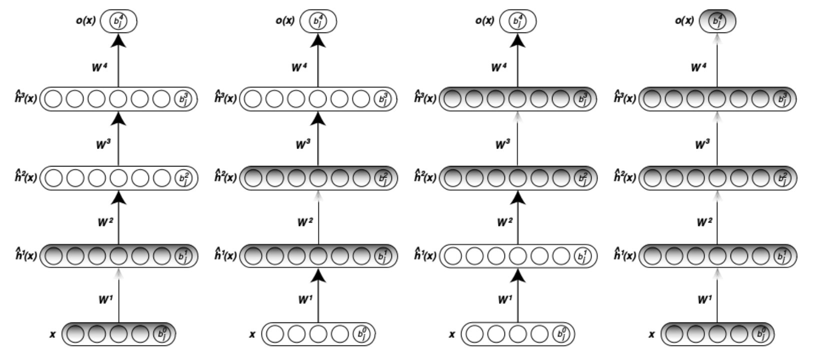
Предобучение без учителя

- И вывод можно вести последовательно, уровень за уровнем:



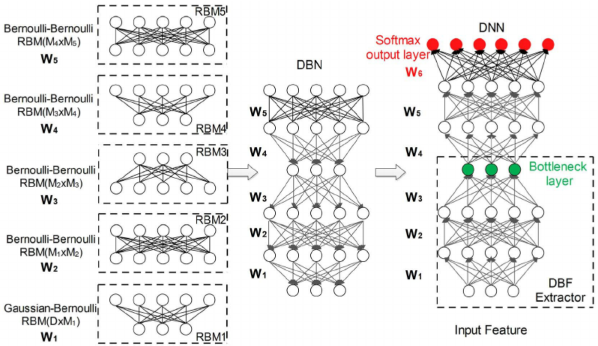
Предобучение без учителя

- А потом уже дообучать градиентным спуском (fine-tuning).



Предобучение без учителя

- Этот подход привёл к прорыву в распознавании речи.



Предобучение без учителя

- Но обучать глубокие сети из RBM довольно сложно, они хрупкие, и вычислительно тоже нелегко.
- И сейчас уже не очень-то и нужны сложные модели вроде RBM для того, чтобы попасть в хорошую начальную область.
- Инициализация весов — важная часть этого.

Инициализация Ксавье

- *Xavier initialization* (Glorot, Bengio, 2010).
- Рассмотрим простой линейный нейрон:

$$y = \mathbf{w}^\top \mathbf{x} + b = \sum_i w_i x_i + b.$$

- Его дисперсия равна

$$\begin{aligned}\mathrm{Var}\left[y_i\right] = \mathrm{Var}\left[w_i x_i\right] = \mathbb{E}\left[w_i^2 x_i^2\right] - \left(\mathbb{E}\left[w_i x_i\right]\right)^2 = \\ = \mathbb{E}\left[x_i\right]^2 \mathrm{Var}\left[w_i\right] + \mathbb{E}\left[w_i\right]^2 \mathrm{Var}\left[x_i\right] + \mathrm{Var}\left[w_i\right] \mathrm{Var}\left[x_i\right].\end{aligned}$$

- Его дисперсия равна

$$\begin{aligned}\mathrm{Var}\left[y_i\right] = \mathrm{Var}\left[w_i x_i\right] = \mathbb{E}\left[w_i^2 x_i^2\right] - \left(\mathbb{E}\left[w_i x_i\right]\right)^2 = \\ = \mathbb{E}\left[x_i\right]^2 \mathrm{Var}\left[w_i\right] + \mathbb{E}\left[w_i\right]^2 \mathrm{Var}\left[x_i\right] + \mathrm{Var}\left[w_i\right] \mathrm{Var}\left[x_i\right].\end{aligned}$$

- Для нулевого среднего весов

$$\mathrm{Var}\left[y_i\right] = \mathrm{Var}\left[w_i\right] \mathrm{Var}\left[x_i\right].$$

- И если w_i и x_i инициализированы независимо из одного и того же распределения,

$$\mathrm{Var}\left[y\right] = \mathrm{Var}\left[\sum_{i=1}^{n_{\mathrm{out}}} y_i\right] = \sum_{i=1}^{n_{\mathrm{out}}} \mathrm{Var}\left[w_i x_i\right] = n_{\mathrm{out}} \mathrm{Var}\left[w_i\right] \mathrm{Var}\left[x_i\right].$$

- Иначе говоря, дисперсия на выходе пропорциональна дисперсии на входе с коэффициентом $n_{\mathrm{out}} \mathrm{Var}\left[w_i\right]$.

Инициализация Ксавье

- До (Glorot, Bengio, 2010) стандартным способом инициализации было

$$w_i \sim U\left[-\frac{1}{\sqrt{n_{\text{out}}}}, \frac{1}{\sqrt{n_{\text{out}}}}\right].$$

- См., например, *Neural Networks: Tricks of the Trade.*
- Так что с дисперсиями получается

$$\mathrm{Var}\left[w_i\right] = \frac{1}{12}\left(\frac{1}{\sqrt{n_{\text{out}}}} + \frac{1}{\sqrt{n_{\text{out}}}}\right)^2 = \frac{1}{3n_{\text{out}}}, \text{ и}$$

$$n_{\text{out}}\mathrm{Var}\left[w_i\right] = \frac{1}{3},$$

и после нескольких уровней сигнал совсем умирает; аналогичный эффект происходит и в backprop.

Инициализация Ксавье

- Инициализация Ксавье — давайте попробуем уменьшить изменение дисперсии, т.е. взять

$$\mathrm{Var}\left[w_i\right] = \frac{2}{n_{\mathrm{in}} + n_{\mathrm{out}}};$$

  для равномерного распределения это

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{\mathrm{in}} + n_{\mathrm{out}}}}, \frac{\sqrt{6}}{\sqrt{n_{\mathrm{in}} + n_{\mathrm{out}}}}\right].$$

- Но это работает только для симметричных активаций, т.е. не для ReLU...

- ...до работы (He et al., 2015). Вернёмся к

$$\mathrm{Var}\left[w_i x_i\right] = \mathbb{E}\left[x_i\right]^2 \mathrm{Var}\left[w_i\right] + \mathbb{E}\left[w_i\right]^2 \mathrm{Var}\left[x_i\right] + \mathrm{Var}\left[w_i\right] \mathrm{Var}\left[x_i\right]$$

- Мы теперь можем обнулить только второе слагаемое:

$$\mathrm{Var}\left[w_i x_i\right] = \mathbb{E}\left[x_i\right]^2 \mathrm{Var}\left[w_i\right] + \mathrm{Var}\left[w_i\right] \mathrm{Var}\left[x_i\right] = \mathrm{Var}\left[w_i\right] \mathbb{E}\left[x_i^2\right], \text{ и}$$

$$\mathrm{Var}\left[y^{(l)}\right] = n_{\mathrm{in}}^{(l)} \mathrm{Var}\left[w^{(l)}\right] \mathbb{E}\left[\left(x^{(l)}\right)^2\right].$$

- Мы теперь можем обнулить только второе слагаемое:

$$\mathrm{Var}\left[y^{(l)}\right] = n_{\mathrm{in}}^{(l)} \mathrm{Var}\left[w^{(l)}\right] \mathbb{E}\left[\left(x^{(l)}\right)^2\right].$$

- Предположим, что $x^{(l)} = \max(0, y^{(l-1)})$, и у $y^{(l-1)}$ симметричное распределение вокруг нуля. Тогда

$$\mathbb{E}\left[\left(x^{(l)}\right)^2\right] = \frac{1}{2}\mathrm{Var}\left[y^{(l-1)}\right], \quad \mathrm{Var}\left[y^{(l)}\right] = \frac{n_{\mathrm{in}}^{(l)}}{2}\mathrm{Var}\left[w^{(l)}\right]\mathrm{Var}\left[y^{(l-1)}\right].$$

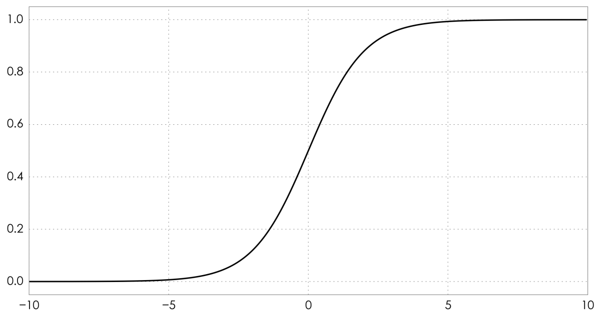
- И это приводит к формуле для дисперсии активации ReLU; теперь нет никакого n_{out}:

$$\mathrm{Var}\left[w_i\right] = 2/n_{\mathrm{in}}^{(l)}.$$

- Кстати, равномерную инициализацию делать не обязательно, можно и нормальное распределение:

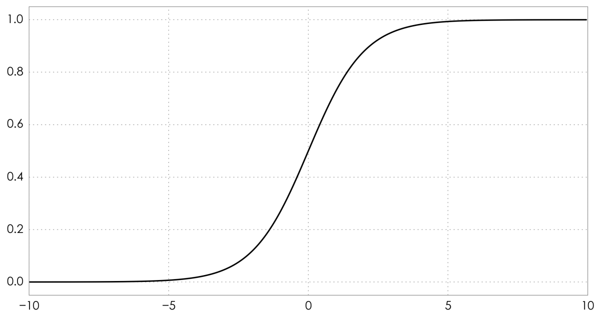
$$w_i \sim N\left(0, \sqrt{2/n_{\mathrm{in}}^{(l)}}\right).$$

- Кстати, о (Glorot, Bengio, 2010) – ещё одна важная идея.
- Эксперименты показали, что $\sigma(x) = \frac{1}{1+e^{-x}}$ работает в глубоких сетях довольно плохо.



О сигмоидах

- *Насыщение*: если $\sigma(x)$ уже «обучилась», т.е. даёт большие по модулю значения, то её производная близка к нулю и «поменять мнение» трудно.



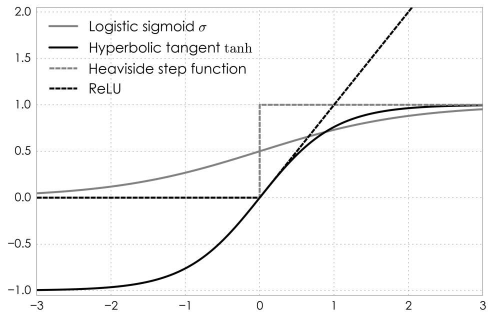
- Но ведь другие тоже насыщаются? В чём разница?

О сигмоидах

- Рассмотрим последний слой сети $h(W\mathbf{a} + \mathbf{b})$, где $\mathbf{a}$ — выходы предыдущего слоя, $\mathbf{b}$ — свободные члены, h — функция активации последнего уровня, обычно $\mathrm{softmax}$.
- Когда мы начинаем оптимизировать сложную функцию потерь, поначалу выходы $\mathbf{h}$ не несут полезной информации о входах, ведь первые уровни ещё не обучены.
- Тогда неплохим приближением будет константная функция, выдающая средние значения выходов.
- Это значит, что $h(W\mathbf{a} + \mathbf{b})$ подберёт подходящие свободные члены $\mathbf{b}$ и постарается обнулить слагаемое $W\mathbf{h}$, которое поначалу скорее шум, чем сигнал.

О сигмоидах

- Иначе говоря, в процессе обучения мы постараемся привести выходы предыдущего слоя к нулю.
- Здесь и проявляется разница: у $\sigma(x) = \frac{1}{1+e^{-x}}$ область значений $(0, 1)$ при среднем $\frac{1}{2}$, и при $\sigma(x) \to 0$ будет и $\sigma'(x) \to 0$.
- А у $\tanh$ наоборот: когда $\tanh(x) \to 0$, $\tanh'(x)$ максимальна.



Варианты градиентного спуска

Градиентный спуск

- «Ванильный» градиентный спуск:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha \nabla f(\mathbf{x}_k).$$

- Всё зависит от скорости обучения α.
- Первая мысль — пусть α уменьшается со временем:
  - линейно (linear decay):

$$\alpha = \alpha_0 \left(1 - \frac{t}{T}\right);$$

  - или экспоненциально (exponential decay):

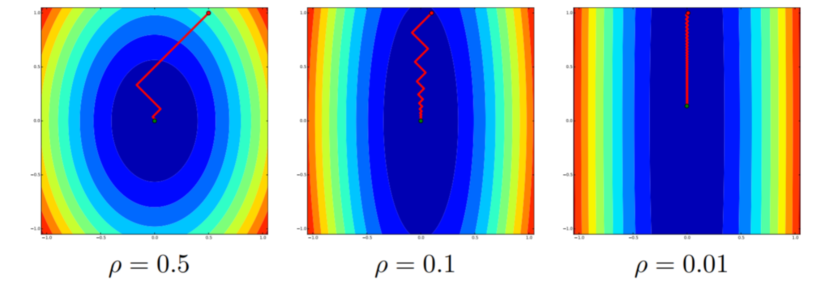
$$\alpha = \alpha_0 e^{-\frac{t}{T}}.$$

Градиентный спуск

- Об этом есть большая наука. Например, условия Вольфе (Wolfe conditions): если мы решаем задачу минимизации $\min_{\mathbf{x}} f(\mathbf{x})$, и на шаге k уже нашли направление p_k, в котором двигаться (например, $p_k = \nabla_{\mathbf{x}} f(\mathbf{x}_k)$), т.е. надо решить $\min_\alpha f(\mathbf{x}_k + \alpha p_k)$, то:
  - для $\phi_k(\alpha) = f(\mathbf{x}_k + \alpha p_k)$ будет $\phi'_k(\alpha) = \nabla f(\mathbf{x}_k + \alpha p_k)^\top p_k$, и если p_k – направление спуска, то $\phi'_k(0) < 0$;
  - шаг α должен удовлетворять условиям Армихо (Armijo rule):
  $$\phi_k(\alpha) \le \phi_k(0) + c_1 \alpha \phi'_k(0) \text{ для некоторого } c_1 \in (0, \frac{1}{2});$$
  - или даже более сильным условиям Вульфа (Wolfe rule): Армихо плюс
  $$|\phi'_k(\alpha)| \le c_2 |\phi'_k(0)|,$$
  т.е. мы хотим уменьшить проекцию градиента.
- Останавливаем когда $\|\nabla_{\mathbf{x}} f(\mathbf{x}_k)\|^2 \le \epsilon$ или $\|\nabla_{\mathbf{x}} f(\mathbf{x}_k)\|^2 \le \epsilon \|\nabla_{\mathbf{x}} f(\mathbf{x}_0)\|^2$ (а почему квадрат, кстати?).

Градиентный спуск

- Давайте посмотрим, что происходит, если масштаб разный: для функции $f(x, y) = \frac{1}{2}x^2 + \frac{\rho}{2}y^2 \to \min_{x,y}$



- Для вытянутых «долин» (переменных с разным масштабированием) мы сразу получаем кучу лишних итераций, очень медленно.
- Лучше быть *адаптивным*; как это сделать?

Градиентный спуск

- Лучше всего, конечно, *метод Ньютона*: давайте отмасштабируем обратно при помощи гессиана

$$\mathbf{g}_k = \nabla_{\mathbf{x}} f(\mathbf{x}_k),\ H_k = \nabla^2_{\mathbf{x}} f(\mathbf{x}_k),\ \text{и } \mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k H_k^{-1} \mathbf{g}_k.$$

- Здесь тоже применимо условие Армихо:

$$\alpha_k : \quad f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - c_1 \alpha_k g_k^\top H_k^{-1} \mathbf{g}_k,\ c_1 \approx 10^{-4}.$$

- Было бы круто! Но H_k посчитать просто нереально.

Градиентный спуск

- Есть, правда, приближения.
- Метод сопряжённых градиентов, квази-ньютоновские методы...
- L-BFGS (limited memory Broyden–Fletcher–Goldfarb–Shanno):
  - строим аппроксимацию к H^{-1};
  - для этого сохраняем последовательно апдейты аргументов функции и градиентов и выражаем через них H^{-1}.
- Интересный открытый вопрос: можно ли заставить L-BFGS работать для deep learning?
- Но пока не получается, в основном потому, что всё-таки нужно уметь считать градиент.
- А ведь у нас обычно нет возможности даже градиент вычислить...

Стохастический градиентный спуск

- У нас обычно стохастический градиентный спуск:

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \alpha \nabla f(\mathbf{x}_t, \mathbf{x}_{t-1}, y_t).$$

- Да ещё и с мини-батчами; как это понять формально?
- Мы обычно решаем задачу *стохастической оптимизации*:

$$F(\mathbf{x}) = \mathbb{E}_{q(\mathbf{y})} f(\mathbf{x}, \mathbf{y}) \to \min_{\mathbf{x}} :$$

  - минимизация эмпирического риска

$$F(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}) = \mathbb{E}_{i\sim \mathrm{U}(1,\ldots,N)} f_i(\mathbf{x}) \to \min_{\mathbf{x}};$$

  - минимизация вариационной нижней оценки (ELBO)... но об этом позже.
- Что такое теперь, получается, мини-батчи?

Стохастический градиентный спуск

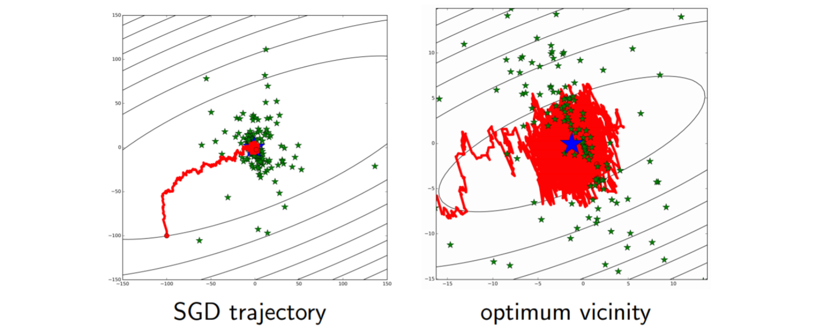
- Да просто эмпирические оценки общей функции по подвыборке:

$$\hat{F}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}, \mathbf{y}_i), \quad \hat{\mathbf{g}}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}_i).$$

- Это очень хорошие оценки: несмещённые, сходятся на бесконечности (правда, медленно), легко посчитать.
- В целом, так и мотивируется стохастический градиентный спуск (SGD): метод Монте-Карло по сути.
- Но есть проблемы...

Стохастический градиентный спуск

- Проблемы SGD:
  - никогда не идёт в правильном направлении,
  - шаг не равен нулю в оптимуме $F(\mathbf{x})$, т.е. не может сойтись с постоянной длиной шага,
  - мы не знаем ни $F(\mathbf{x})$, ни $\nabla F(\mathbf{x})$, т.е. не можем использовать правила Армихо и Вульфа.



- Тем не менее, можно попробовать проанализировать итерацию SGD для $F(\mathbf{x}) = \mathbb{E}_{q(\mathbf{y})} f(\mathbf{x}, \mathbf{y}) \to \min_{\mathbf{x}}$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \hat{\mathbf{g}}_k, \quad \mathbb{E}\hat{\mathbf{g}}_k = \mathbf{g}_k = \nabla F(\mathbf{x}_k).$$

- Давайте оценим невязку точки на очередной итерации:

$$\begin{aligned}\|\mathbf{x}_{k+1} - \mathbf{x}_{\mathrm{opt}}\|^2 &= \|\mathbf{x}_k - \alpha_k \hat{\mathbf{g}}_k - \mathbf{x}_{\mathrm{opt}}\|^2 = \\ &= \|\mathbf{x}_k - \mathbf{x}_{\mathrm{opt}}\|^2 - 2\alpha_k \hat{\mathbf{g}}_k^\top (\mathbf{x}_k - \mathbf{x}_{\mathrm{opt}}) + \alpha_k^2 \|\hat{\mathbf{g}}_k\|^2.\end{aligned}$$

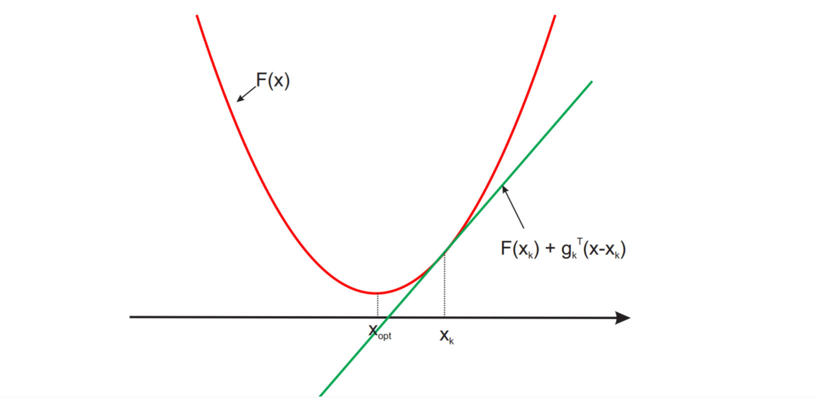
- Возьмём ожидание по $q(\mathbf{y})$ в момент времени k:

$$\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}_{\mathrm{opt}}\|^2 = \|\mathbf{x}_k - \mathbf{x}_{\mathrm{opt}}\|^2 - 2\alpha_k \mathbf{g}_k^\top (\mathbf{x}_k - \mathbf{x}_{\mathrm{opt}}) + \alpha_k^2 \mathbb{E}\|\hat{\mathbf{g}}_k\|^2.$$

Стохастический градиентный спуск

- Для простоты предположим, что F выпуклая:

$$F(\mathbf{x}_{\text{opt}}) \geq F(\mathbf{x}_k) + \mathbf{g}_k^\top(\mathbf{x}_k - \mathbf{x}_{\text{opt}})$$



Стохастический градиентный спуск

- У нас было

$$
\begin{aligned}
\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}_{\text{opt}}\|^2 = \|\mathbf{x}_k - \mathbf{x}_{\text{opt}}\|^2 - 2\alpha_k \mathbf{g}_k^\top (\mathbf{x}_k - \mathbf{x}_{\text{opt}}) + \alpha_k^2 \mathbb{E}\|\hat{\mathbf{g}}_k\|^2, \\
F(\mathbf{x}_{\text{opt}}) \geq F(\mathbf{x}_k) + \mathbf{g}_k^\top (\mathbf{x}_k - \mathbf{x}_{\text{opt}}).
\end{aligned}
$$

- Значит,

$$
\begin{aligned}
\alpha_k (F(\mathbf{x}_k) - F(\mathbf{x}_{\text{opt}})) &\leq \alpha_k \mathbf{g}_k^\top (\mathbf{x}_k - \mathbf{x}_{\text{opt}}) = \\
&= \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_{\text{opt}}\|^2 + \frac{1}{2}\alpha_k^2 \mathbb{E}\|\hat{\mathbf{g}}_k\|^2 - \frac{1}{2}\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}_{\text{opt}}\|^2.
\end{aligned}
$$

- Возьмём ожидание от левой части и просуммируем:

$$
\begin{aligned}
&\sum_{i=0}^{k} \alpha_i (\mathbb{E} F(\mathbf{x}_i) - F(\mathbf{x}_{\mathrm{opt}})) \le \\
&\qquad \le \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}_{\mathrm{opt}}\|^2 + \frac{1}{2}\sum_{i=0}^{k} \alpha_i^2 \mathbb{E}\|\hat{\mathbf{g}}_i\|^2 - \frac{1}{2}\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}_{\mathrm{opt}}\|^2 \le \\
&\qquad\qquad \le \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}_{\mathrm{opt}}\|^2 + \frac{1}{2}\sum_{i=0}^{k} \alpha_i^2 \mathbb{E}\|\hat{\mathbf{g}}_i\|^2.
\end{aligned}
$$

- Получилась сумма значений функции в разных точках с весами α_i. Что делать?

- Воспользуемся выпуклостью:

$$\mathbb{E}F\left(\frac{\sum_i \alpha_i \mathbf{x}_i}{\sum_i \alpha_i}\right) - F(\mathbf{x}_{\mathrm{opt}}) \leq$$

$$\leq \frac{\sum_i \alpha_i(\mathbb{E}F(\mathbf{x}_i) - F(\mathbf{x}_{\mathrm{opt}})}{\sum_i \alpha_i} \leq \frac{\frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}_{\mathrm{opt}}\|^2 + \frac{1}{2}\sum_{i=0}^k \alpha_i^2 \mathbb{E}\|\hat{\mathbf{g}}_i\|^2}{\sum_i \alpha_i}.$$

- Т.е. оценка получилась на значение в линейной комбинации точек (поэтому в статьях часто берут среднее/ожидание или линейную комбинацию, а на практике нет разницы или лучше брать последнюю точку)
- Если $\|\mathbf{x}_0 - \mathbf{x}_{\mathrm{opt}}\| \leq R$ и $\mathbb{E}\|\hat{\mathbf{g}}_k\|^2 \leq G^2$, то

$$\mathbb{E}F(\hat{\mathbf{x}}_k) - F(\mathbf{x}_{\mathrm{opt}}) \leq \frac{R^2 + G^2 \sum_{i=0}^k \alpha_i^2}{2\sum_{i=0}^k \alpha_i}.$$

- Это самая главная оценка про SGD:

$$\mathbb{E}F(\hat{\mathbf{x}}_k) - F(\mathbf{x}_{\text{opt}}) \leq \frac{R^2 + G^2 \sum_{i=0}^{k} \alpha_i^2}{2 \sum_{i=0}^{k} \alpha_i}.$$

- R – оценка начальной невязки, а G – оценка чего-то вроде дисперсии стохастического градиента.
- Например, для постоянного шага $\alpha_i = h$

$$\mathbb{E}F(\hat{\mathbf{x}}_k) - F(\mathbf{x}_{\text{opt}}) \leq \frac{R^2}{2h(k+1)} + \frac{G^2 h}{2} \to_{k\to\infty} \frac{G^2 h}{2}.$$

Стохастический градиентный спуск

- Итоги про SGD:
  - SGD приходит в «регион неопределённости» радиуса $\frac{1}{2}G^2h$, и этот радиус пропорционален длине шага;
  - чем быстрее идём, тем быстрее придём, но регион неопределённости будет больше, т.е. по идее надо уменьшать со временем скорость обучения;
  - SGD сходится медленно: полный GD для выпуклых функций сходится за $O(1/k)$, а SGD – за $O(1/\sqrt{k})$;
  - но далеко от региона неопределённости у нас скорость тоже $O(1/k)$ получилась для постоянной скорости обучения, т.е. замедляется только уже близко к оптимуму, и вообще цель наша – достичь региона неопределённости;
  - но всё равно всё зависит от G, и это будет особенно важно потом в нейробайесовских методах.

Спасибо!

Спасибо за внимание!