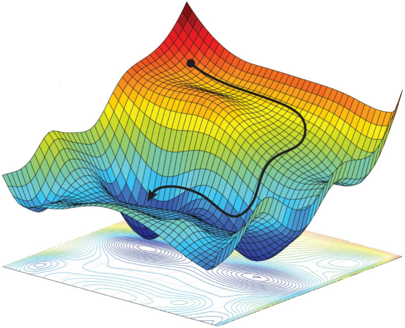
Особенности настройки глубоких нейросетей

Виктор Китов

v.v.kitov@yandex.ru



Мотивация для глубокого обучения

- Машинное обучение опирается на хорошие признаки.
    - приходится подбирать вручную
- Глубокое обучение - использование нейросетей с большим числом слоев.
- Нейроны глубоких слоев могут сами выучивать сложные признаки.
    - важно для сложных областей, таких как изображения, видео, звук и т.д.
    - изображения: извлекаем границы, углы, простейшие фигуры, объекты (глаза, брови), определяем класс (например, человека)
    - работает лучше, чем вручную подобранные признаки
        - при достаточной обучающей выборке
- Общее число нейронов глубокой сети может быть меньше за счет переиспользования промежуточных нейронов (признаков)

Содержание

Расширение обучающей выборки

- Модели глубокого обучения имеют много параметров.
  - полносвязный слой $A \to B$ нейронов содержит AB связей!
- Методы борьбы с переобучением:
  - предобучить начальные слои по большим доступным выборкам схожей задачи
    - например, по ImageNet (>14 миллионов размеченных объектов) для изображений
    - domain adaptation: адаптация модели одной доменной области к другой
  - использовать расширение (augmentation) обучающей выборки: $(x, y) \to \{(\phi_1(x), y), (\phi_2(x), y), (\phi_3(x), y), ...\}$
    - $\phi_1, \phi_2, \phi_3, ...$ - инвариантные к отклику преобразования
    - например, изменения яркости/контрастности, поворот, обрезка.

Регуляризация

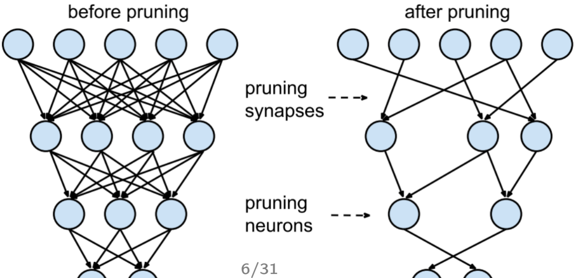
- Бороться с переобучением можно упрощая модель:
  - использовать меньше нейронов/связей
  - задавать ограничения на веса при настройке
    - пример: свертка, где веса общие и связи только для близких нейронов
  - использовать регуляризацию
    - L_1, L_2 (можно со своим весом на каждом слое)
    - ранняя остановка (early stopping)
    - DropOut

Уменьшение вычислительной сложности

Для уменьшения вычислительной сложности:

- можно обучить большую сеть, а потом ее прореживать:
  - отбросить связи с низким весом
  - отбросить связи, слабо влияющие на ф-цию потерь (optimal brain damage)
  - отбросить нейроны с низкими входными весами
  - отбросить нейроны со слабыми активациями

Прореживание связей и нейронов



Optimal brain damage: связи, слабо влияющие на $L(w)$

$$L(w) - L(w^*) = \nabla L(w^*)^T (w - w^*)$$
$$+\frac{1}{2}(w - w^*)^T \nabla^2 L(w^*) (w - w^*) + O\left(\|w - w^*\|^3\right)$$
$$\approx \frac{1}{2}(w - w^*)^T \nabla^2 L(w^*) (w - w^*) \approx \frac{1}{2}\sum_i \frac{\partial L(w^*)}{\partial w_i^2}(w_i - w_i^*)^2$$

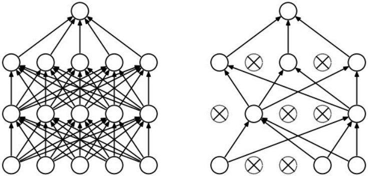
Приближения:

- $O\left(\|w - w^*\|^3\right) \approx 0$
- в оптимуме $\nabla L(w^*) = 0$
- $\nabla^2 L(w^*)$ диагональна-игнорируем смешанные производные

Алгоритм optimal brain damage:

- повторять нужно число раз
    - обучить нейросеть
    - отбросить связи с $\frac{\partial L(w^*)}{\partial w_i^2}(0 - w_i^*)^2 < threshold$

DropOut: обучение


(a) Standard Neural Net (b) After applying dropout.

- Для каждого минибатча каждый нейрон, кроме выходных и входных отбрасывается с вероятностью $(1-p)$ и оставляется с вероятностью p.
    - независимо для каждого нейрона
- Это уменьшает переобучение, препятствуя со-настройке нейронов
- По сути, мы учим ансамбль прореженных моделей.
- В среднем нейрон получает на вход $\sum_i (pw_i x_i + (1-p)w_i 0) = \sum_i pw_i x_i$.

DropOut: применение

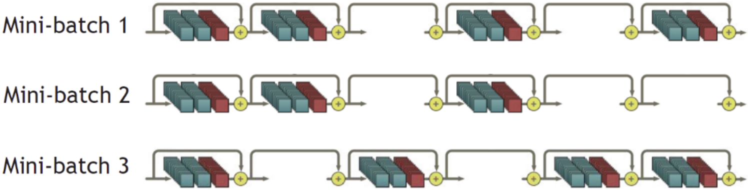
- Применение модели: оставляются все нейроны.
- Для компенсации изменений, выход каждого нейрона домножается на p.
  - для уменьшения вычислений можно:
    - обучение: выход x_i/p, когда x_i оставлен.
    - прогноз: выход x_i без изменений.

Стохастическая глубина в сетях[1]

Deep Networks with Stochastic Depth - используется случайное отбрасывание слоёв в классификации изображений.

Полная архитектура (используется на тесте):

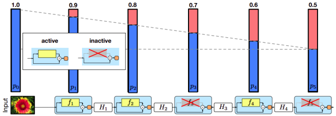
Обучение: отбрасывается преобразования блока l с вероятностью $1 - p_l$



[1] https://arxiv.org/abs/1603.09382

Stochastic Depth

Рекомендуемая зависимость вероятности оставления блока p_l от слоя l:



- Мотивация p_l: более ранние слои извлекают более базовые признаки, используемые всеми последующими слоями.
- Применение: используются сеть со всеми блоками, домноженными на p_l.
- Мотивация архитектуры:
  - быстрее обучение (менее глубокая сеть на каждом минибатче)
  - борьба с переобучением (когда всё зависит от взаимосвязей между отдельными блоками)
  - по сути обучается ансамбль моделей разной глубины.

Содержание

Особенности оптимизации нейросетей

- Зависимость $\widehat{y}(x)$ в общем случае невыпукла.
- $\mathcal{L}(\widehat{y}, y)$ - невыпукла => много локальных минимумов.
- На найденный минимум влияют:
  - начальное приближение
  - объекты минибатчей
  - метод обучения и динамика ε_t
- Можно настраивать разными способами, а потом
  - выбрать наилучшее решение по валидации
  - усреднить несколько решений (ансамбль)

Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки
    - не применим для динамических данных

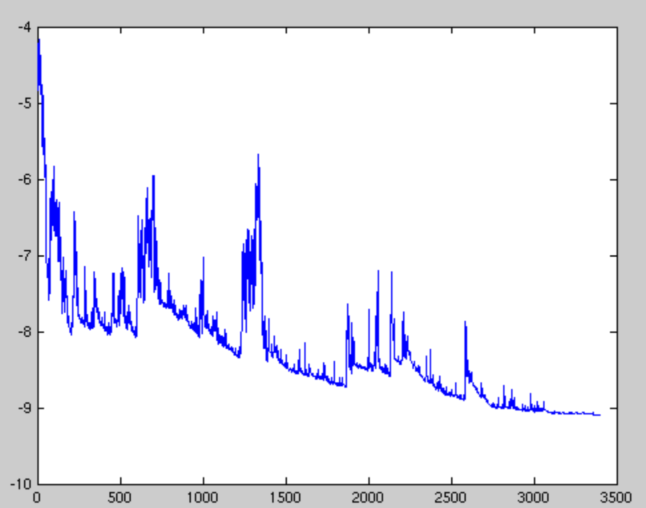
$$w_{t+1} := w_t - \eta \nabla L(\theta; X, Y)$$

- Stochastic gradient descent: стохастический спуск с сэмплированием по 1 объекту

$$w_{t+1} := w_t - \eta \nabla L(\theta; x_i, y_i)$$

  - не производит лишних вычислений по повторяющимся объектам
  - неустойчивая оценка градиента

Пример сходимости SGD



Базовые градиентные методы

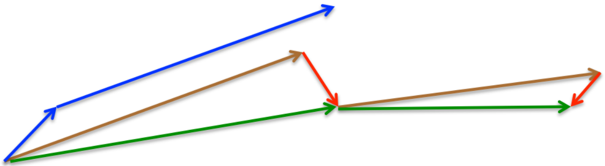
- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta L(\theta; x_{i+1:i+K}, y_{i+1:i+K})$$

- точнее оценки градиента
    - быстрее: параллелизация вычислений по минибатчу
- Сложности:
    - нужен выбор динамики убывания η
    - одинаковый шаг для разных весов
        - логичнее веса брать меньше, где ф-ция резко меняется и отвечающие редко встречающимся признакам.
    - застревание в локальных оптимумах и точках перегиба

Модификации инерции и Нестерова

- Momentum
$$v_t := \gamma v_{t-1} + \eta \nabla_\theta L(\theta)$$
$$w_{t+1} := w_t - v_t$$
    - аналогия с мячом, скатывающимся с горы
- Nesterov Accelerated Gradient (Nesterov Momentum)
$$v_t := \gamma v_{t-1} + \eta \nabla_\theta L(\theta - \gamma v_{t-1})$$
$$w_{t+1} := w_t - v_t$$



Модификации SGD

- Обозначим $g_t = \nabla L(\theta_t)$; $\theta, g_t \in \mathbb{R}^K$. Операции над векторами поточечные.
- AdaGrad ($\varepsilon = 10^{-6}$)

$$G_t := G_t + g_t^2$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} g_t$$

- RMSprop

$$E\left[g^2\right]_t := \gamma E\left[g^2\right]_{t-1} + (1-\gamma)\, g_t^2$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \varepsilon}} g_t$$

Модификации SGD

- Adam=RMSprop+инерция
  ($\beta_1 = 0.9,\ \beta_2 = 0.999,\ \varepsilon = 10^{-8}$):

$$m_t := \beta_1 m_{t-1} + (1-\beta_1)\, g_1$$
$$v_t := \beta_2 v_{t-1} + (1-\beta_2)\, g_1^2$$
$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\widehat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \varepsilon}\widehat{m}_t$$

- Nadam: Adam+Nesterov Accelerated Gradient.

Модификации SGD

- AMSGrad: позволяет помнить про больше градиенты без экспоненциального забывания.
    - при этом перенормировка m_t, v_t не используется.

$$m_t := \beta_1 m_{t-1} + (1 - \beta_1) g_1$$
$$v_t := \beta_2 v_{t-1} + (1 - \beta_2) g_1^2$$
$$\widehat{v}_t = \max(\widehat{v}_{t-1}, v_t)$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \varepsilon} \widehat{m}_t$$

Дополнительные улучшения[2]

- Ранняя остановка (early stopping) - борьба с переобучением
- Добавление шума к градиенту - находим оптимум с большей окрестностью:

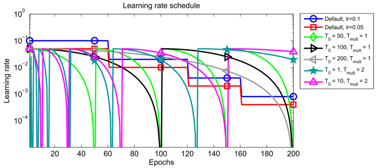
$$g_t := g_t + \mathcal{N}(0, \sigma_t^2)$$
$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

- Обучение по расписанию (curriculum learning)
    - сначала обучаемся на простых объектах, потом на сложных
- Параллелизация вычислений SGD.
- Ускорение обучения: batch normalization.

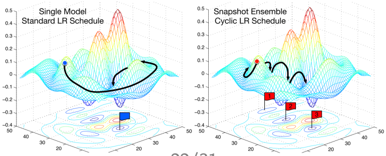
[2] Больше информации: https://ruder.io/deep-learning-optimization-2017/

Дополнительные улучшения

- LR schedule - закон изменения η. Подобный закон, приводит к к оптимуму с большей окрестностью:

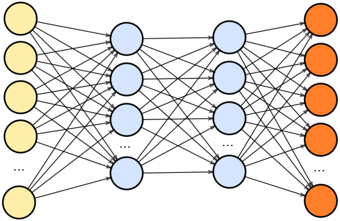


- Перед каждым увеличением η получили некоторую модель=>можем модели комбинировать в композицию:



Содержание

Перенормировка активаций: мотивация



- SGD $w := w - \varepsilon \nabla_w \mathcal{L}(x, y)$ обновляет все веса на всех слоях одновременно.
- Распределение выходов меняется, и поздние слои должны обучатся снова.
- Также вход может сдвинуться в область малых градиентов нелинейности.
- Перенормировка активаций (batch normalization) частично это решает.

BatchNorm: идея

- Нормализуем выходы на промежуточных слоях:

$$\tilde{x}_k = \frac{x_k - \mu_k}{\sigma_k}, \quad \mu_k = \mathbb{E}x_k, \sigma_k = \sqrt{Var(x_k)}$$

  - гарантируем $\mathbb{E}\tilde{x}_k = 0$, $\mathrm{Var}\,\tilde{x}_k = 1$ после обновления весов на предыдущих слоях.
    - обучение быстрее для поздних слоёв
- **Обучение:**
  - проблема: не знаем μ_k, σ_k
    - изменяются динамически с обновлением весов
  - решение: оценим по текущему минибатчу (должен быть достаточного размера)
- **Применение:**
  - распределение x_k фиксировано, так что можем оценить μ_k, σ_k по всей обучающей выборке.
  - более эффективно: оценки μ_k, σ_k с последовательности последних минибатчей.

BatchNorm: основной алгоритм

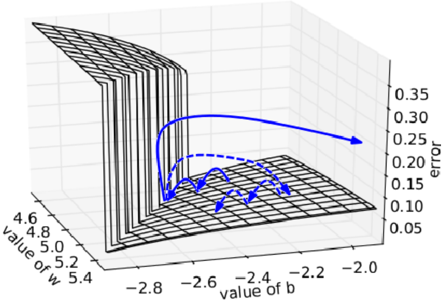
$$\tilde{x}_k = \alpha_k \frac{x_k - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} + \beta_k, \quad \mu_k = \bar{x}_k, \ \sigma_k = \sqrt{Var(x_k)}, \ \varepsilon = 10^{-6}.$$

- **Обучение:**
    - μ_k, σ_k по минибатчу
    - α_k, β_k выходное стд. отклонение и среднее.
        - обучаются в процессе настройки сети
    - мотивация:
        - можем отменить нормализацию (например в задаче предсказания времени суток по фото)
        - возможность лучше подстроиться под нелинейность (не обнулять вход в половине случаев для ReLU)
- **Применение:**
    - μ_k, σ_k оценены по широкому классу объектов.
    - α_k, β_k фиксированы.

Содержание

Проблема большого градиента

Нестабильный по величине градиент мешает сходимости.



Решения - ограничение градиента (gradient clipping) или адаптивное ограничение градиента (adaptive gradient clipping) во время обучения.

Ограничение градиента

Ограничение градиента по порогу (gradient clipping[3]): гарантия $|\Delta\theta| \le \varepsilon\lambda$ (помогает в настройке рекуррентных нейросетей)

- если $\|\nabla_\theta \mathcal{L}(\widehat{y}_i, y_i)\| < \lambda$

$$\theta \to \theta - \varepsilon \nabla_\theta \mathcal{L}(\widehat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i})$$

- иначе

$$\theta \to \theta - \varepsilon \frac{\lambda}{\|\nabla_\theta \mathcal{L}(\widehat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i})\|} \nabla_\theta \mathcal{L}(\widehat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i})$$

[3] https://arxiv.org/pdf/1211.5063.pdf

Адаптивное ограничение градиента

Адаптивное ограничение градиента по порогу (adaptive gradient clipping[4]): гарантия $\|\Delta\theta_i\| \leq \varepsilon\lambda \|\theta_i\|$

- помогло настроить SOTA модель классификации ImageNet без батч-нормализации
- Пусть θ_i - отдельный параметр (др. вариант применения: θ_i - все пар-ры i-го слоя, тогда $\|\cdot\|$-норма Фробениуса, $\|\theta_i\|^* = \max(\|\theta_i\|, \nu), \nu > 0$ - мало).

Для каждого i:

- если $\frac{\|\nabla_{\theta_i}\mathcal{L}(\widehat{y}_i, y_i)\|}{\|\theta_i\|^*} < \lambda$

$$\theta_i \to \theta_i - \varepsilon\nabla_{\theta_i}\mathcal{L}(\widehat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i})$$

- иначе

$$\theta_i \to \theta_i - \varepsilon\frac{\lambda\|\theta_i\|^*}{\|\nabla_{\theta_i}\mathcal{L}(\widehat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i})\|}\nabla_{\theta_i}\mathcal{L}(\widehat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i})$$

[4] https://arxiv.org/pdf/2102.06171.pdf

Заключение

- Глубокие сети способны сами настраивать сложные признаки.
    - работают лучше, но нужны большие обучающие выборки
- Борьба с переобучением нейросетей:
    - расширение выборки (pretraining, data augmentation)
    - сокращение #нейронов/связей
    - ранняя остановка, L_1/L_2 регуляризация, DropOut.
- Функция потерь невыпукла, возможны локальные оптимумы.
- Идеи улучшений SGD: инерция, ускоренный градиент Нестерова, индивидуальные настраиваемые веса для каждого параметра.
- BatchNorm ускоряет и упрощает сходимость.