

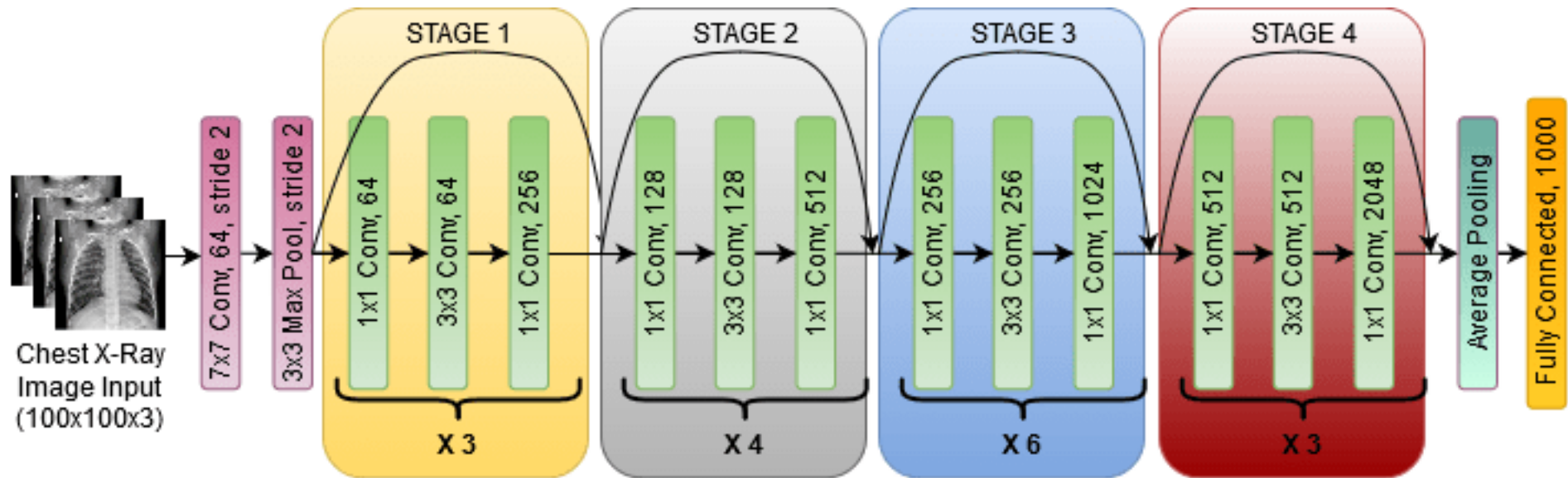
Лекция 4

Перенос обучения

Разработка нейросетевых систем

Канев Антон Игоревич

Предобученные модели



```
model = torch.hub.load("pytorch-cifar-models", "cifar100_mobilenetv2_x0_5", pretrained=True)
```

Задача до

Предобученная модель распознает виды животных

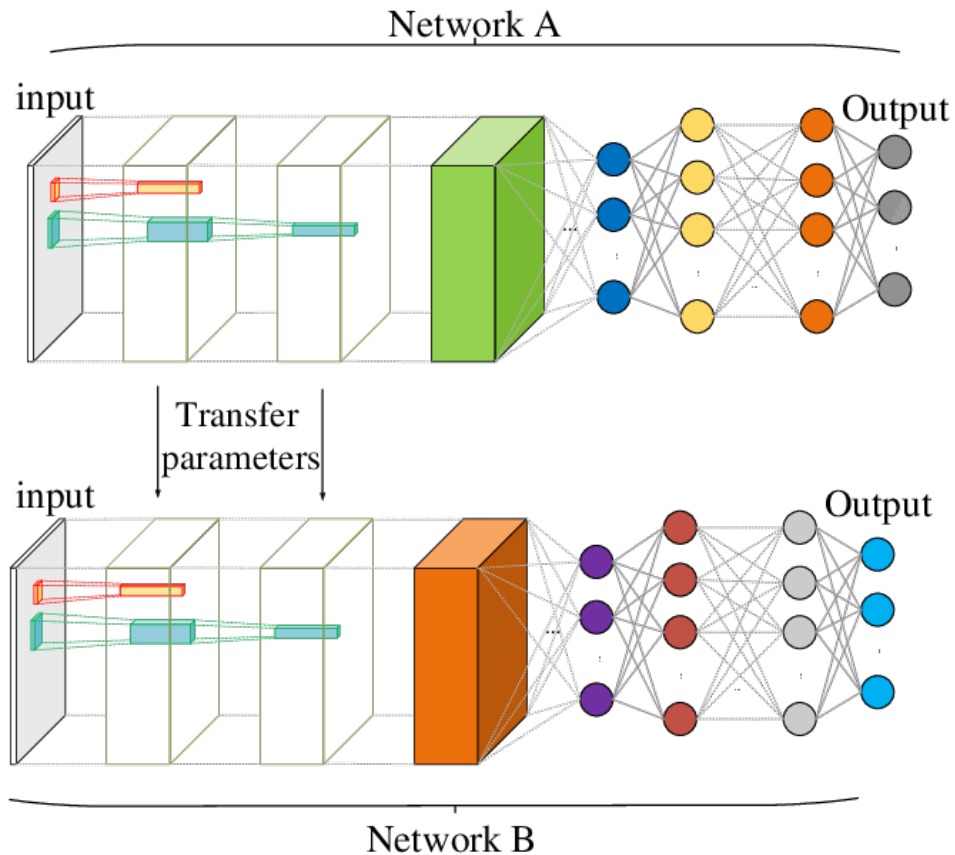


Задача после

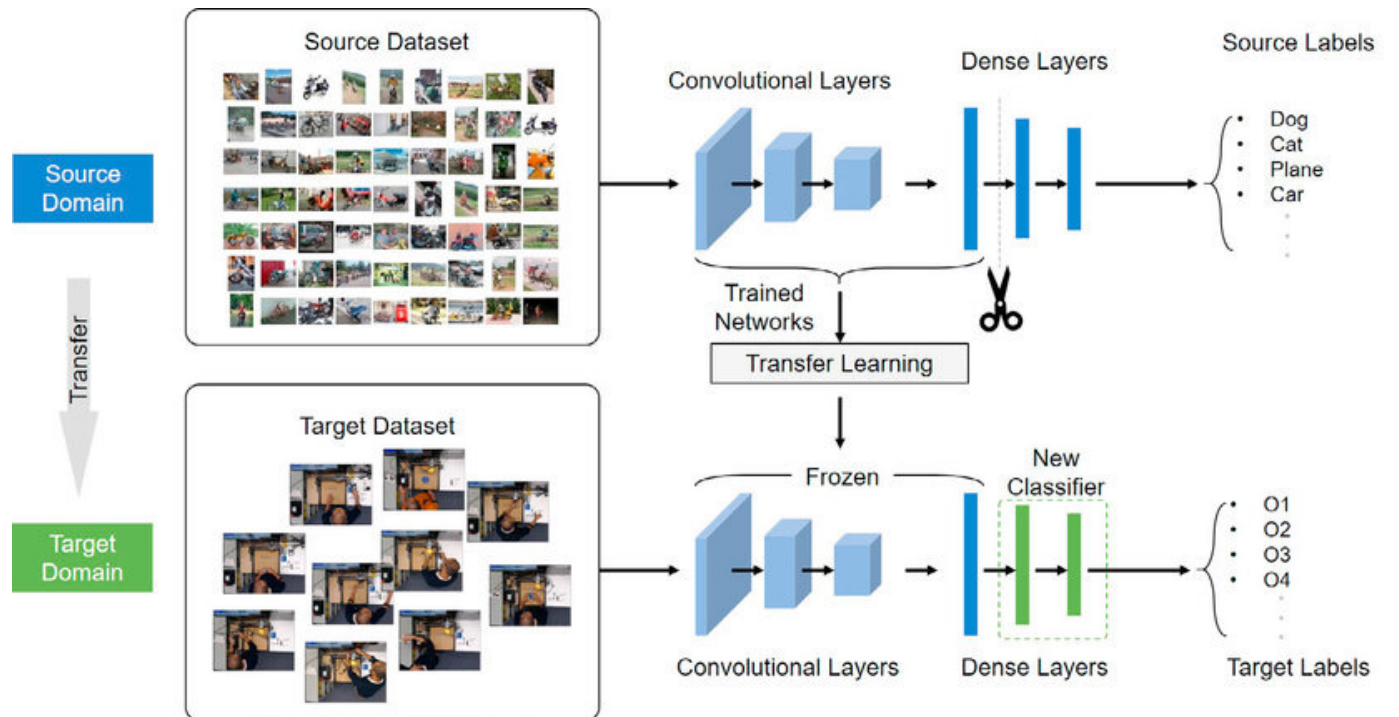
Дообученная модель распознает породы собак



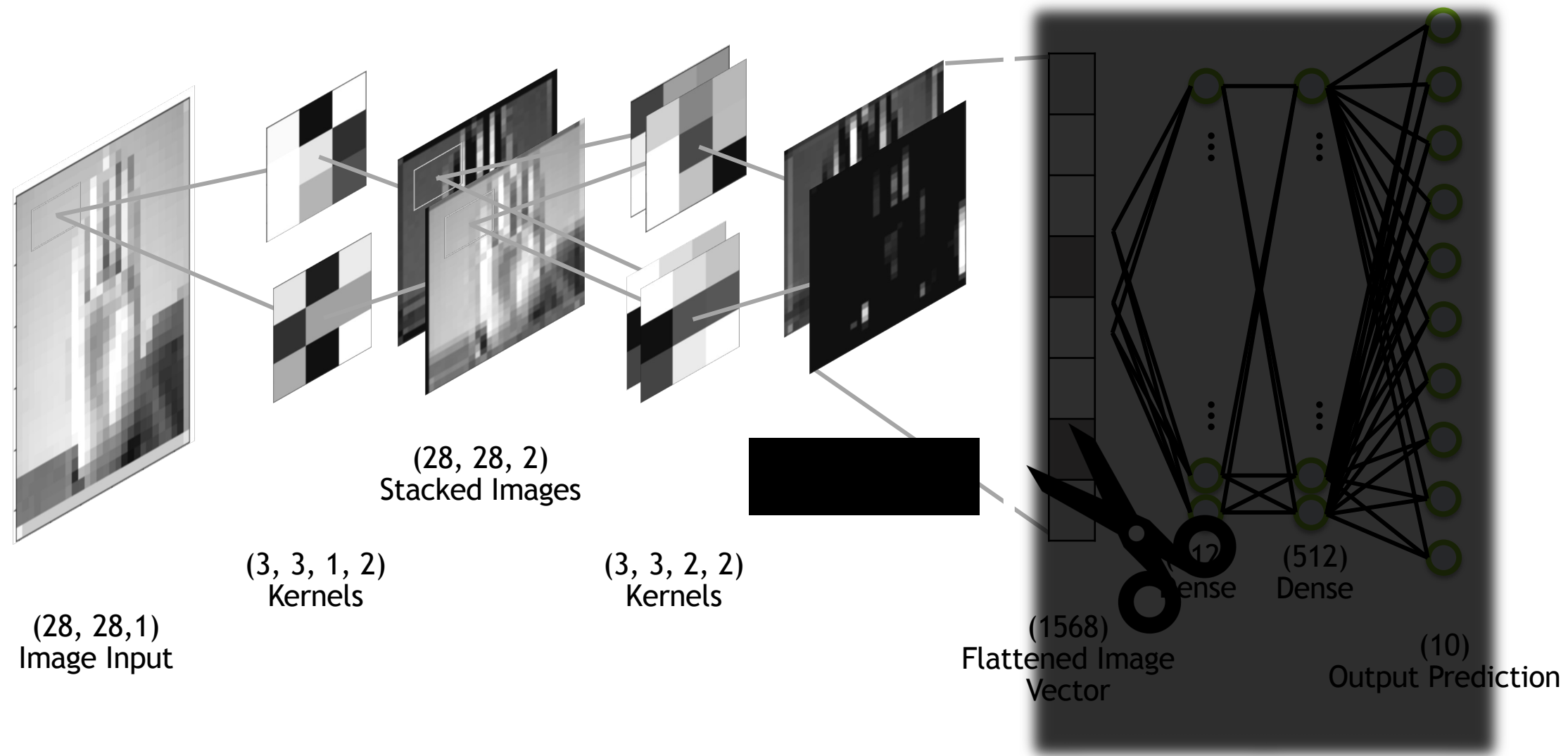
Дообучение - transfer learning



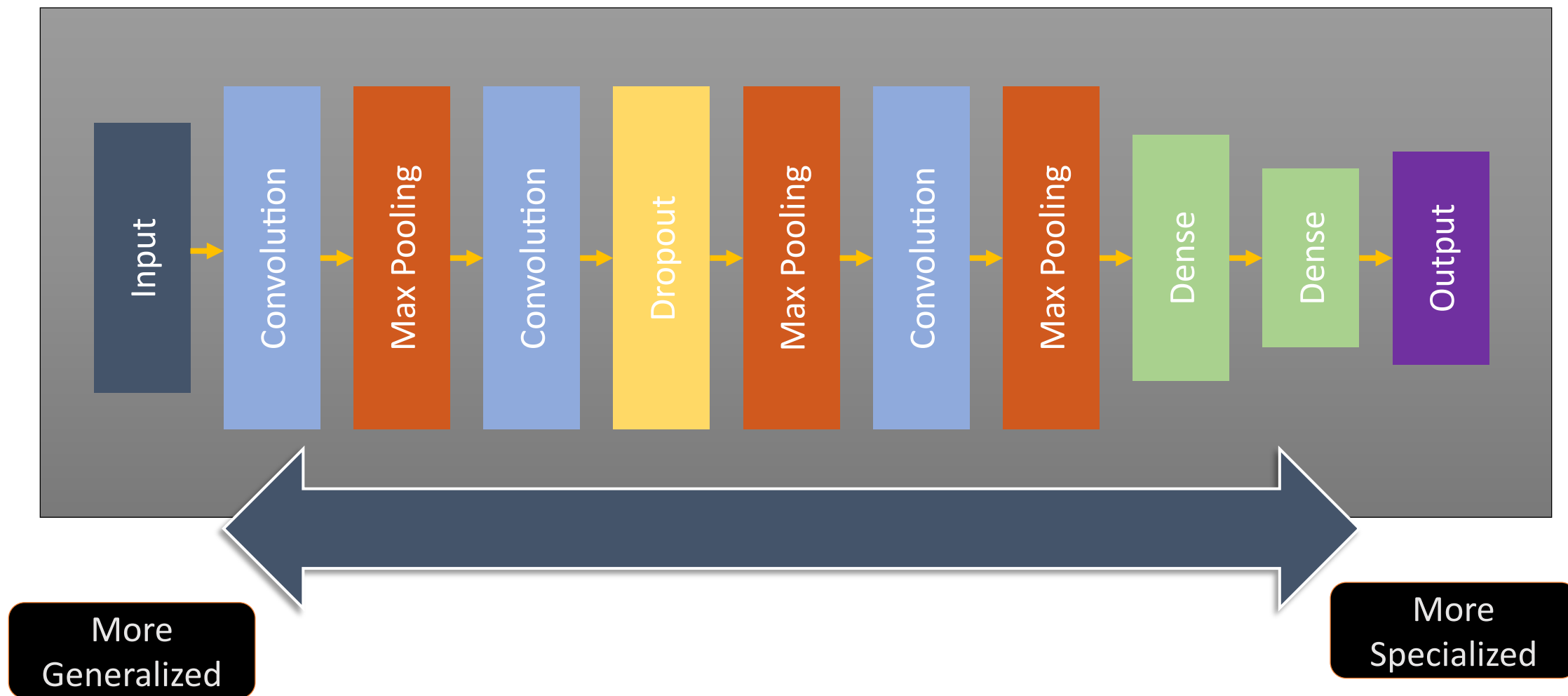
- Использование готовых моделей
- Для близких задач на новых класса можно применить transfer learning



Перенос обучения



Transfer Learning



Заморозка модели



```
for i, (name, param) in enumerate(new_model.named_parameters()):  
    param.requires_grad = False
```


Fine Tuning



- После заморозки весов преобученной модели наступает этап обучения новых слоев
- После обучения остальные веса можно разморозить и продолжить обучение

Стохастический градиентный спуск

$L(f(\mathbf{x}(i); \boldsymbol{\theta}), \mathbf{y}(i))$ – значение функции потерь

$f(\mathbf{x}(i); \boldsymbol{\theta})$ – результат вычисления нейронной сети от входа $\mathbf{x}(i)$ и параметров (весов) $\boldsymbol{\theta}$

Обновление на k -ой итерации стохастического градиентного спуска (СГС)

Require: скорость обучения ϵ_k

Require: Начальные значения параметров $\boldsymbol{\theta}$

while критерий остановки не выполнен **do**

Выбрать из обучающего набора мини-батч m примеров $\{\mathbf{x}(1), \dots, \mathbf{x}(m)\}$ и соответствующие им метки $\mathbf{y}(i)$.

Вычислить оценку градиента: $\mathbf{g} \leftarrow + (1/m) \nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}(i); \boldsymbol{\theta}), \mathbf{y}(i))$.

Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \mathbf{g}$.

end while

Стохастический градиентный спуск

- Основной параметр алгоритма СГС – скорость обучения ϵ
- На практике же необходимо постепенно уменьшать скорость обучения со временем
- $\epsilon_k = (1 - \alpha) \epsilon_k + \alpha \epsilon_k$, где $\alpha = k / \tau$. После τ -й итерации ϵ остается постоянным.
- Если скорость изменяется линейно, то нужно задать параметры $\epsilon_0, \epsilon_\tau$ ит.

Импульсный метод

Стохастический градиентный спуск (СГС) с учетом импульса

Require: скорость обучения ε , параметр импульса α

Require: начальные значения параметров θ , начальная скорость v

while критерий остановки не выполнен **do**

Выбрать из обучающего набора мини-пакет m примеров $\{x(1), \dots, x(m)\}$ и соответствующие им метки $y(i)$.

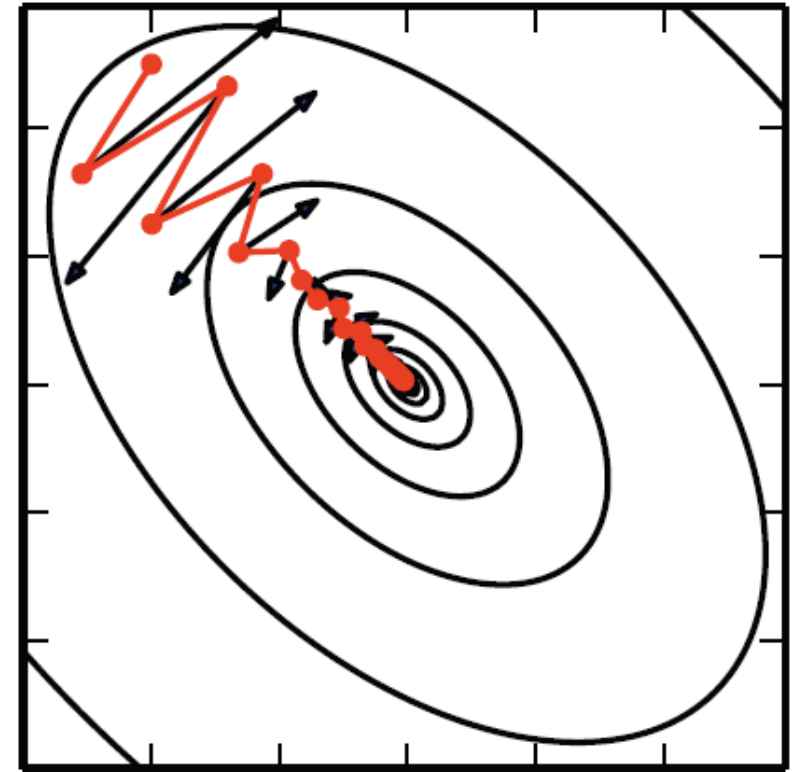
Вычислить оценку градиента:

$$g \leftarrow (1/m) \nabla_{\theta} \sum_i L(f(x(i); \theta), y(i)).$$

Вычислить обновление скорости: $v \leftarrow \alpha v - \varepsilon g$.

Применить обновление: $\theta \leftarrow \theta + v$.

end while



Импульсный алгоритм можно рассматривать как имитацию движения частицы, подчиняющейся динамике Ньютона.

Adagrad

Алгоритм AdaGrad по отдельности адаптирует скорости обучения всех параметров модели. Для параметров, по которым частная производная функции потерь наибольшая, скорость обучения уменьшается быстро, а если частная производная мала, то и скорость обучения уменьшается медленнее. В итоге больший прогресс получается в направлениях пространства параметров со сравнительно пологими склонами

Алгоритм AdaGrad

Require: глобальная скорость обучения ϵ Require: начальные значения параметров θ

Require: небольшая константа δ , например 10^{-7} , для обеспечения численной устойчивости.

Инициализировать переменную для агрегирования градиента $\mathbf{r} = \mathbf{0}$

while критерий остановки не выполнен do

Выбрать из обучающего набора мини-пакет m примеров $\{x(1), \dots, x(m)\}$ и соответствующие им метки $y(i)$.

Вычислить градиент: $\mathbf{g} \leftarrow (1/m) \nabla_{\theta} \sum_i L(f(\mathbf{x}(i); \theta), y(i))$.

Агрегировать квадраты градиента: $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}$.

Вычислить обновление: $\Delta \theta \leftarrow -\epsilon / (\delta + \sqrt{\mathbf{r}}) \odot \mathbf{g}$

Применить обновление: $\theta \leftarrow \theta + \Delta \theta$. end while

RMSProp

AdaGrad уменьшает скорость обучения, принимая во внимание всю историю квадрата градиента, и может случиться так, что скорость станет слишком малой еще до достижения такой выпуклой структуры.

В алгоритме RMSProp используется экспоненциально затухающее среднее, т. е. далекое прошлое отбрасывается

Require: глобальная скорость обучения ϵ , скорость затухания ρ **Require:** начальные значения параметров θ

Require: небольшая константа δ , например 10^{-6} , для стабилизации деления на малые числа
Инициализировать переменную для агрегирования градиента $\mathbf{r} = 0$

while критерий остановки не выполнен **do**

Выбрать из обучающего набора мини-пакет m примеров $\{\mathbf{x}(1), \dots, \mathbf{x}(m)\}$ и соответствующие им метки \mathbf{y}_i .

Вычислить градиент: $\mathbf{g} \leftarrow (1/m) \nabla_{\theta} \sum_i L(f(\mathbf{x}(i); \theta), \mathbf{y}(i))$.

Агрегировать квадраты градиента: $\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}$.

Вычислить обновление параметров: $\Delta \theta \leftarrow -\epsilon / \sqrt{(\delta + \mathbf{r})} \odot \mathbf{g}$

Применить обновление: $\theta \leftarrow \theta + \Delta \theta$.

end while

Adam

«Adam» – сокращение от «adaptive moments» (адаптивные моменты). Его правильнее всего рассматривать как комбинацию RMSProp и импульсного метода

Require: величина шага ε (по умолчанию 0.001).

Require: коэффициенты экспоненциального затухания для оценок моментов ρ и ρ , принадлежащие диапазону $[0, 1)$ (по умолчанию 0.9 и 0.999 соответственно).

Require: небольшая константа δ для обеспечения численной устойчивости (по умолчанию 10^{-8}).

Require: начальные значения параметров θ .

Инициализировать переменные для первого и второго моментов $\mathbf{s} = \mathbf{0}, \mathbf{r} = \mathbf{0}$

Инициализировать шаг по времени $t = 0$

while критерий остановки не выполнен **do**

Выбрать из обучающего набора мини-пакет m примеров $\{\mathbf{x}(1), \dots, \mathbf{x}(m)\}$ и соответствующие им метки y_i .

Вычислить градиент: $\mathbf{g} \leftarrow (1/m) \nabla_{\theta} \sum_i L(f(\mathbf{x}(i); \theta), y(i))$.

$t \leftarrow t + 1$

Обновить смещенную оценку первого момента: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1) \mathbf{g}$

Обновить смещенную оценку второго момента: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}$

Скорректировать смещение первого момента: $\mathbf{s} \leftarrow \mathbf{s} / (1 - \rho_t)$

Скорректировать смещение второго момента: $\mathbf{r} \leftarrow \mathbf{r} / (1 - \rho_t)$

Вычислить обновление: $\Delta \theta = -\varepsilon \mathbf{s} / \sqrt{(\delta + \mathbf{r})}$

Применить обновление: $\theta \leftarrow \theta + \Delta \theta$.

end while

Другие оптимизаторы

