<http://www.machinelearning.ru/wiki/images/1/1e/Sem07_ann.pdf>

**Обучение глубоких сетей**

Классический алгоритм обратного распространения ошибки хорошо работает на двухслойных и трехслойных нейронных сетях, но при дальнейшем увеличении глубины начинает испытывать проблемы. Одна из причин — так называемое затухание градиентов. По мере распространения ошибки от выходного слоя к входному на каждом слое происходит домножение текущего результата на производную функции активации. Производная у традиционной сигмоидной функции активации меньше единицы на всей области определения, поэтому после нескольких слоев ошибка станет близкой к нулю. Если же, наоборот, функция активации имеет неограниченную производную (как, например, гиперболический тангенс), то может произойти взрывное увеличение ошибки по мере распространения, что приведет к неустойчивости процедуры обучения.

В последние годы были достигнуты существенные успехи в области методов настройки глубоких нейросетей. В данном разделе мы рассмотрим некоторые наиболее интересные приемы.

**ReLU**. Известно, что нейронные сети способны приблизить сколь угодно сложную функцию, если в них достаточно слоев и функция активации является нелинейной. Функции активации вроде сигмоидной или тангенциальной являются нелинейными, но приводят к проблемам с затуханием или увеличением градиентов. Однако можно использовать и гораздо более простой вариант — выпрямленную линейную функцию активации (*rectified linear unit*, **ReLU**):

Ее производная равна либо единице, либо нулю, и поэтому не может произойти разрастания или затухания градиентов. Более того, использование данной функции приводит к прореживанию весов.

**Dropout**. Глубокие нейронные сети сильно подвержены переобучению из-за большого числа параметров. Одним из способов борьбы с ним является dropout-регуляризация. Обучение нейронной сети обычно производят стохастическим градиентным спуском, случайно выбирая по одному объекту из выборки. Dropout-регуляризация заключается в том, что при выборе очередного объекта изменяется структура сети: каждая вершина выбрасывается с некоторой вероятностью . По такой прореженной сети делается обратное распространение ошибки, для оставшихся весов делается градиентный шаг, после чего все выброшенные вершины возвращаются в нейросеть. Таким образом, на каждом шаге стохастического градиента мы настраиваем одну из возможных архитектур сети, где под архитектурой мы понимаем структуру связей между вершинами, а через *N* обозначаем суммарное число вершин. При применении нейросети вершины уже не выбрасываются, но выход каждой вершины домножается на — благодаря этому на выходе вершины мы будем получать матожидание ее ответа по всем архитектурам. Таким образом, обученную с помощью dropout-регуляризации нейросеть можно рассматривать как результат усреднения сетей.

**Эвристики**

**Optimal Brain Damage (LeCun, 1990)**

Ускорение стадии вывода (inference) сети. После попадания в локальный минимум удаляем ребра с наименьшей по модулю второй производной и продолжаем оптимизацию.

**Dropout**Простая эвристика: на каждой итерации backpropagation «выкидываем» половину нейронов скрытых слоев, вместе с их входящими и исходящими весами, а после завершения итерации – возвращаем. После окончания обучения делим все веса пополам. Hinton et al., 2012

**Обучение**

**Batch stochastic gradient descent** (**стохастический градиентный спуск**).

??? (**пакетный градиентный спуск**).

**Backpropagation** (**обратное распространение ошибки**)

**Selective search** (**селективный поиск**).