Вариационные Автоенкодеры с мультиномиальным Prior'ом.

Денис Мазур

Ноябрь 2018

Аннотация

Метод вариационных автоэнкодеров является, вероятно, самым эффективным среди методов глубинного обучения без учителя...

Содержание

1 Вступление

Так как прочтение данной работы не предполагает ознакомленность с методами машинного обучения, будет дан поверхностый разбор некоторых ключевых принципов и методов, необходимых для понимания вариационных автоенкодеров.

От читателя предполагается понимание математического анализа, линейной алгебры и теории вероятности.

2 Обучение нейронных сетей

Для базового понимания автоенкодеров не требуется понимание принципа работы нейронных сетей, по этой причине объяснение устройства нейронных сетей было решено опустить и уделить больше внимания принципу их обучения. Для интересующихся, литература на тему нейронных сетей будет приложина в библиографии.

Для простоты устройсто нейронной сети можно редуцировать до некоторой функции, имеющей параметы W, принимающей значения X и выдающей значения $\widehat{X}$. Физический смысл X - данные, подаваемые на вход сети, $\widehat{X}$ - выход нейронной сети. Каждому элементу $x \in X$ соответсвует элемент $y \in Y$, и задачей обучения нейронной сети является подобрать параметы W нейронной сети так, чтобы они минимально или не отличались от Y. Также можно сформулировать задачу следущим образом:

$$d(f(x_n), y_n) \to \min$$

Также, можно поставить задачу как поиск оптмальных параметров W:

$$\underset{W}{\mathbf{argmin}}\ d(f(x_n), y_n)$$

Где $f(x)$ - нейронная сеть, а $d(x, y)$ определенная нами функция расстояния (отличия) между выходом нейронной сети и настоящим ответом, называемая функцией потерь. Важно понимать следующие свойства функций $f(x)$ и $d(x, y)$, они обе дифференцируемые и определены на всех X, а это значит, что мы к параметрам нейронной сети W можем применять любые методы оптимизации функций, например, градиентный метод.

2.1 Градиентный метод

Градиентый метод является методом решения задач вида $\min\limits_{a \in \mathbb{R}^n} g(x)$. Алгоритм градиентного метода начинается с выбора параметоров a, это может делать как и из какого либо представления того какими примерно эти параметры должны быть, так и случайным образом. Далее идет так называемый "шаг"градиентного метода. Из параметров a вычитаеся значение градента функции $g(a)$ в точке текущего значения a:

$$a_{n+1} = a_n - \nabla g(a_n)$$

Чтобы не "перескочить"через минимум функции во время итерации метода, градиент $\nabla f(x)$ обычно умножается на какую-то константу α:

$$a_{n+1} = a_n - \alpha \nabla g(a_n)$$

2.2 Градиентный метод в обучении нейронных сетей

Теперь о том как градиентный метод может применяться в обучении нейронных сетей.

Функцию потерь нейронной сети $d(f(x_n), y_n)$ можно записать в виде, $d(f(x_n, W), y_n)$, до этого, для простоты, записи парамерты нейронной сети W не записывались как аргумент, хотя, очевидно, им являются. Поясню, что X и Y являются константными значениями, так как являются выборкой наших данных и известны нам заранее. Таким образом, единственный

аргумент, который мы можем изменять, минимизируя функцию потерь - W.

На практике, оптимизировать функцию потерь градиентным спуском "в лоб"не получится. У этого есть несколько причин, одна из главных - риск переобучения модели (ситуация при которой модель показывает идеальных результат на обучающей выборке, но плохой на валидации). Но помимо этого, выборка данных чаще всего слишком велика и не помещается в память компьютера, по этой причине на каждой итерации градиентного метода выбирается случайная подвыборка тренировочных данных и над ней совершается шаг граиентого метода. Данный метод называется стохастическим градиентным спуском.

3 Вариационные автоэнкодеры

3.1 Модели с латентными переменными

Предположим, что мы хотим моделировать какую-либо систему с помощью вероятностого распределения её состояний $p(\mathbf{x})$, где $\mathbf{x} \in \mathcal{R}^D$. Такая система может быть достаточно сложной и мы можем не знать явный вид $p(\mathbf{x})$. Для решения этой проблемы можем ввести новую переменную $\mathbf{z} \in \mathcal{R}^d$, которая описывает состояния $\mathbf{x}$, например если $\mathbf{x}$ - картинки котиков, то $\mathbf{z}$ может описывать их цвет, породу и положение на картинке. Подобные переменные называются “латентными”. Также новая переменная позволяет нам выразить $p(\mathbf{x})$ как:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) \, d\mathbf{z} \tag{1}$$

Имея подобную систему нам бы хотелось знать плотность распределения самих латентных переменных $\mathbf{z}$, при том, что мы знаем соответсвующий им $\mathbf{x}$. Если быть точнее, то мы хотим знать апостериорное распределение $p(\mathbf{z} \mid \mathbf{x})$. Но зависимость между $\mathbf{x}$ и $\mathbf{x}$ может быть крайне нелинейной, а размерности D и d достаточно большими, а так как и частное распределение $p(\mathbf{x})$ и апостериорное распределение $p(\mathbf{z} \mid \mathbf{x})$ требуют вычисления интеграла, они трудновычислимы.

Но решить эту проблему можно с помощью параметрического распределения, параметры которого задаются нейронной сетью с параметрами $\theta \in \Theta$. Таким образом мы можем выучить оптимальные параметры с помощью метода максимального правдоподобия:

$$\theta^* = \arg\max_{\theta \in \Theta} p_\theta(\mathbf{x}) \tag{2}$$

Но мы всё ещё не можем максимизировать выражение (1), значение которого мы не можем вычислить. Для решения этой проблемы мы можем применить метод выборки по значимости (importance sampling). Когда

нам нужно оценить величину из какого либо первичного вероятностого распределения, выборка по значимости позволяет нам выбирать величины из другого (смещённого) распределения и взвешивать новые велечины через функцию плотности вероятности первичного распределения. Пусть $q_\phi(\mathbf{z} \mid \mathbf{x})$ будет смещённым распределением, заданным нейронной сетью с параметрами $\phi \in \Phi$. Теперь мы можем записать $p(\mathbf{x})$ как:

$$p(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{x}) \; d\mathbf{z} =$$

$$\mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x} \mid \mathbf{z}) \right] = \mathbb{E}_{p(\mathbf{z})} \left[\frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})} p_\theta(\mathbf{x} \mid \mathbf{z}) \right] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \right] \tag{3}$$

В методе выборки по значимости оптимальные переменные пропорциональны функции плотности первичного распределения, в нашем случае это распределение $p_\theta(\mathbf{x} \mid \mathbf{z})$, применив теорему Байеса мы получим $p_\theta(\mathbf{x} \mid \mathbf{z}) = \frac{p_\theta(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x})}{p_\theta(\mathbf{z})}$. Как мы можем наблюдать, оптимальные переменные пропорциональны апостериорному распределению $p_\theta(\mathbf{z} \mid \mathbf{x})$, которое, конечно, невычислимо.

3.2 Вариационные автоэнкодеры

К счастью, мы можем решить эту проблему. Пытаясь аппроксимировать апосетриорное распределение выученным смещённым распределением ($q_\phi(\mathbf{z} \mid \mathbf{x})$), мы можем неплохо аппроксимировать $p_\theta(\mathbf{x})$. Подобная система похожа на модель, называемой автоенкодером, в которой есть некоторый энкододер $q_\phi(\mathbf{z} \mid \mathbf{x})$ и декодер $p_\theta(\mathbf{x} \mid \mathbf{z})$.

Разберём подробнее, что делают энкододер и декодер. Декодер - генеративная модель $p_\theta(\mathbf{x}, \mathbf{z})$, состоящая из самого декодера $p_\theta(\mathbf{x} \mid \mathbf{z})$ и приорного распределения латентных переменных $p(\mathbf{z})$. По-сути декодер преобразует латентные переменные $\mathbf{z}$ в моделируемые переменные $\mathbf{x}$. Енкодер $q_\phi(\mathbf{z} \mid \mathbf{x})$ осуществляет обратную операцию преобразования моделируемых переменных $\mathbf{x}$ в латентные переменные $\mathbf{z}$.

Так как и энкододер и декодер в нашей системе заданы как нейронные сети, мы можем обучать их, оптимизируя параметры градиентным методом, а значит нам нужна функция потерь. В данном случае мы хотим сделать распределения $p_\theta(\mathbf{z} \mid \mathbf{x})$ и $q_\phi(\mathbf{z} \mid \mathbf{x})$ максимально похожими на друг-друга. Для этого мы можем минимизировать какую-либо функцию расстояния между ними, возьмём, например, девергенцию Кульбака-Лейблера:

$$KL(q_\phi(\mathbf{z} \mid \mathbf{x}) || p_\theta(\mathbf{z} \mid \mathbf{x})) = \int q_\phi(\mathbf{z} \mid \mathbf{x}) \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z} \mid \mathbf{x})} \; d\mathbf{x} =$$

$$\mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})} \left[\log q_\phi(\mathbf{z} \mid \mathbf{x}) - p_\theta(\mathbf{z} \mid \mathbf{x}) \right]$$

Разумеется, мы не можем напрямую посчитать это “расстояние” по причине того, что нам неизвесты оба апостериорных распределения, но решить

эту проблему можно решая двойственную задачу, называемой оцениванием нижней границы (estimate lower bound):