Развертывание таких приложений сильно отличается от традиционных систем, так как нет серверных приложений для запуска. Приложение использует так называемую FaaS (Function as a Service) модель использования облачного провайдера.

По сути, FaaS – это запуск бэкэнд-кода без управления собственными серверами или собственными серверными приложениями. Предложения FaaS не требуют кодирования в конкретную структуру или библиотеку. Функции FaaS являются регулярными приложениями, когда дело доходит до языка и окружающей среды. Например, функции AWS Lambda могут быть реализованы «первым классом» на Javascript, Python, Go, любом языке JVM (Java, Clojure, Scala и т.д.) или на любом языке .NET.

В среде FaaS мы загружаем код для нашей функции поставщику FaaS, а поставщик делает все остальное, необходимое для предоставления ресурсов, диспетчеризации запросов, управления процессами и т.д.

Конечно, такая архитектура имеет и свои недостатки.

Контроль над определённым функционалом приложения отдаётся стороннему поставщику, может приводить к простоям системы, неожиданные ограничения, изменения затрат, потеря функциональности, принудительное обновление API и многое другое.

Проблемы с разграничением доступа между клиентскими процессами, работающими на сервере провайдера (например: один клиент может получит доступ к данным другого, или ошибка в программном обеспечении одного клиента, вызывающая сбой в программном обеспечении другого).

Если вы хотите поменять поставщика определённой услуги, то вам почти наверняка нужно будет изучить новые инструменты развертывания, мониторинга. С большой вероятностью придется изменить код или даже архитектуру подсистемы, для использования другого API. Среди поставщиков существует стратегия «привязки» клиента к своему сервису, которая может заключаться в разработке специфического API, мешающего применять универсальные подходы к доступу даже сходных по задачам сервисов разных поставщиков.

Список использованных источников

1. Roberts, M. Serverless Architectures [Электронный ресурс]. – Режим доступа: https://martinfowler.com/articles/serverless.html. – Дата доступа: 04.05.2021.
2. Knorr, E. What is cloud computing? Everything you need to know. – Режим доступа: https://www.infoworld.com/article/2683784/what-is-cloud-computing.html. – Дата доступа: 24.04.2021.
3. Материалы из раздела serverless сайта amazon.com. – Режим доступа: https://aws.amazon.com/ru/blogs/rus/category/serverless/. – Дата доступа: 15.04.2021.
4. Обзор облачных сервисов для разработки бэкенда мобильных приложений. – Режим доступа: https://habr.com/ru/company/surfstudio/blog/463435/. – Дата доступа: 04.05.2021.

УДК 004.8

МНОГОСЛОЙНЫЙ ПЕРСЕПТРОН И АЛГОРИТМ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Лапко М.Л., студ., Дунина Е.Б., к.ф.-м.н., доц., Корниенко А.А., д.ф.-м.н., проф.

Витебский государственный технологический университет, г. Витебск, Республика Беларусь

Реферат. *В статье рассмотрены теоретические основы моделирования алгоритма обратного распространения ошибки.*

Ключевые слова: многослойный персептрон, алгоритм, нейронная сеть, активационная функция, весовые коэффициенты, метод обучения с учителем.

Алгоритм обратного распространения ошибки – популярный алгоритм обучения нейронных сетей (многослойных персептронов). В основу метода положен алгоритм

градиентного спуска по поверхности ошибок. Он относится к методам обучения с учителем. Поэтому обучающая выборка состоит из пар векторов: входного и целевого вектора. Целевой вектор – желаемый отклик сети при подаче на ее входы данного входного вектора. В процессе обучения на выходах сети вычисляются текущие сигналы, называемые фактическим откликом сети. Фактический отклик сети сравнивается с желаемым откликом, и вычисляются ошибки для коррекции весовых коэффициентов. Целью обучения является получение набора весовых коэффициентов, для которых достигается минимум ошибки между фактическим и желаемым откликами сети.

Рассмотрим трехслойную нейронную сеть, состоящую из входного слоя (в нем 3 нейрона), 1-го скрытого слоя (в нем тоже 3 нейрона) и выходного слоя (здесь 2 нейрона) (рис. 1). Каждый элемент некоторого слоя связан с каждым элементом следующего слоя, но ни один элемент не связан с самим собой или с другим нейроном в том же слое, что и он сам. Связи в такой сети являются однонаправленными. По одной связи сигнал может проходить только в одном направлении, и он идет от входного слоя к выходному.

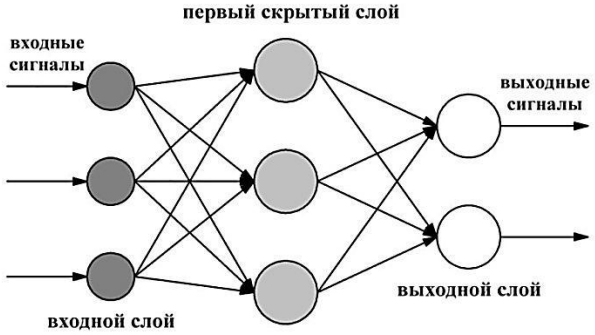

Рисунок 1 – Трехслойная сеть

В качестве активационной функции чаще всего используется сигмоидальная функция

$$f_\sigma(S) = \frac{1}{1+exp(-S)}. \quad (1)$$

Данная функция является дифференцируемой, и ее производная выражается через значения самой функции

$$f'_\sigma(S) = f_\sigma(S)(1-f(S)). \quad (2)$$

Следует отметить, что в многослойном персептроне активационная функция должна быть нелинейной. Если она будет линейной, то многослойная сеть станет эквивалентна однослойной и ее возможности будут ограничены.

Б.Уидроу и М.Е.Хофф предложили минимизировать квадратичную ошибку, определяемую формулой:

$$\varepsilon = \frac{1}{2}\sum_{i=1}(d_i - y_i)^2, \quad (3)$$

где d_i – желаемый выход i-го нейрона, y_i – значение, которое получилось в результате вычисления персептрона. Тогда

$$\frac{\partial \varepsilon}{\partial y_i} = -(d_i - y_i). \quad (4)$$

Квадратичная ошибка обучения персептрона будет зависеть от весовых коэффициентов. $\varepsilon = \varepsilon(\omega_{ij})$. Необходимым условием оптимизации является выполнение равенства

$$\nabla\varepsilon(\omega)=0, \tag{5}$$

где ∇ – оператор градиента, который представляет собой вектор, проекциями которого на оси координат являются производные от функции ε по этим координатам $\frac{\partial\varepsilon}{\partial\omega_{ij}}$. Градиент функции всегда направлен в сторону ее наибольшего возрастания. Наша задача состоит в отыскании минимума функции, т.е. мы двигаемся в сторону противоположную градиенту. Отсюда и название – метод градиентного спуска. Поэтому на каждой итерации к координатам текущей точки ω_{ij} будем добавлять величину, прямо пропорциональную производной по координате, взятую с противоположным знаком:

$$\Delta\omega_{ij}=-\alpha\frac{\partial\varepsilon}{\partial\omega_{ij}}, \tag{6}$$

где α – коэффициент скорости обучения, обычно задаваемый в пределах от 0,05 до 1.

По правилу дифференцирования сложной функции $\varepsilon=\varepsilon(y_i(\omega_{ij}))$ мы можем записать

$$\frac{\partial\varepsilon}{\partial\omega_{ij}}=\frac{\partial\varepsilon}{\partial y_i}\frac{\partial y_i}{\partial\omega_{ij}}. \tag{7}$$

Выходные сигналы нейронов y_i вычисляются с помощью сигмоидальной активационной функции (1)

$$y_i=f_\sigma(S), S_i=\sum_{j=1}\omega_{ij}x_j, \frac{\partial y_i}{\partial\omega_{ij}}=f'_\sigma(S_i)x_j. \tag{8}$$

Учитывая (2) и подставляя (8) и (4) в (7), мы получаем итерационную формулу для обучения персептрона

$$\omega_{ij}(t+1)=\omega_{ij}(t)+\Delta\omega_{ij}, \tag{9}$$

где

$$\begin{aligned}&\Delta\omega_{ij}=\alpha\delta_i x_j,\\&\delta_i=(d_i-y_i)y_i(1-y_i).\end{aligned} \tag{10}$$

При вычислении неизвестной нейронной ошибки для скрытого слоя можно использовать суммарные нейронные ошибки с выходного слоя, помноженные на силы соответствующих синоптических связей, т.е.

$$d_i-y_i=\sum_i\delta_i\omega_{ij}. \tag{11}$$

Таким образом, чтобы найти поправки к весам, мы вначале входной сигнал прогоняем от входного слоя к выходному, а затем прогоняем ошибки от выходного слоя к входному.

На основании теоремы, доказанной В.И.Арнольдом и А.Н.Колмагоровым, о том, что любая непрерывная функция любого количества переменных представляется в виде суперпозиции непрерывных функций одной и двух переменных, следует, что нейронные сети позволяют аппроксимировать любую непрерывную функцию с желаемой точностью. Однако нет четкого правила, позволяющего точно определить количество слоев нейронной сети, а так же количество нейронов в каждом слое, достаточных для приближения неизвестной функциональной зависимости. Поэтому архитектуру нейронной сети, скорость обучения для каждой задачи следует подбирать экспериментально.

Список использованных источников
1. Хливенко, Л. В. Практика нейросетевого моделирования: монография / Л. В. Хливенко. – Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2015. – 214 с.
2. Материалы сайта ai.lector.ru [Электронный ресурс]. – Режим доступа: http://ai.lector.ru/?go=lection02. – Дата доступа: 12.04.2021.

УДК 62-83

МОДЕРНИЗАЦИЯ МЕХАТРОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ – ВЫБОР ПЕРЕДАЧ К ИСПОЛНИТЕЛЬНОМУ ОРГАНУ

Белов А.А., доц., Поляков А.А. студ.

Витебский государственный университет,
г. Витебск, Республика Беларусь

Реферат. *В данной работе рассмотрены способы модернизации привода станка с ЧПУ путем замены коробки передач на регулируемый электропривод.*

Ключевые слова: электродвигатель, коробка передач, модернизация.

Модернизация станка в первую очередь подразумевает обновление систем ответственных за точное перемещение инструмента и детали. Проблем, связанных с механической обработкой и заменой изношенных механических узлов, мы касаться не будем, поговорим об электроприводе, так как без него ни один станок работать не будет.

Асинхронный электродвигатель с датчиком поворота (энкодером) на валу и преобразователем частоты с обратной связью – наиболее простой вариант замены. При выборе надо учитывать, что у асинхронного электродвигателя, при одинаковой мощности, крутящий момент меньше, чем у электродвигателя постоянного тока. Обычно выбирается двигатель на 20–30 % большей мощности. Современные асинхронные привода позволяют получить характеристику скорость/момент не хуже, чем у привода постоянного тока. Причем у них отсутствуют щетки, а все остальное в обслуживании практически не нуждается.

Асинхронный электродвигатель с преобразователем частоты без обратной связи – если требуется выбрать привод для вращения патрона токарного станка, не предназначенного для нарезания резьбы и прочих операций, требующих позиционирования патрона, то использование привода с обратной связью не всегда будет оправдано, в некоторых случаях можно обойтись обычным общепромышленным асинхронным электродвигателем и преобразователем частоты без обратной связи или с виртуальной обратной связью.

Бесколлекторный электродвигатель на постоянных магнитах с датчиком положения ротора (револьвером или абсолютным энкодером) и преобразователем частоты – это лучший вариант замены, но не самый экономичный. Бесколлекторный электродвигатель выбирается по требуемой частоте вращения вала и максимальному крутящему моменту. Выбирать такой мотор по мощности не стоит, так как их характеристики сильно отличаются как от двигателей постоянного тока, так и асинхронных. Эти электромоторы при меньших габаритах обеспечивают больший крутящий момент, что может являться решающим фактором при недостатке места для монтажа. В обслуживании они так же не нуждаются.

Бесколлекторный электродвигатель на постоянных магнитах с датчиком положения ротора, преобразователем частоты и безлюфтовым редуктором – очень хорошее решение для приводов, где не требуется большая скорость. За счет использования редуктора, можно увеличить крутящий момент, используя двигатель и преобразователь частоты меньшей мощности. Не стоит пугаться высокой стоимости безлюфтовых редукторов, стоимость маломощного двигателя с редуктором, скорее всего, окажется ниже, чем электродвигателя большой мощности. Вполне возможно, что двигатели, изначально стоявшие на Вашем станке, обеспечивали такие максимальные скорости перемещения, каким Вы никогда не пользовались и, переоценив необходимые скоростные характеристики, можно хорошо сэкономить на приводах, не потеряв в качестве работы станка.

Линейный электродвигатель прямого привода с преобразователем частоты – достаточно новое явление на станочном рынке. Конструктивно это тот же бесколлекторный