Процесс решения задач с применением нейронных сетей в библиотеке Keras

Обобщенный процесс решения задач машинного обучения

1. **Определение задачи и создание набора данных**

Определение того, какой вид имеют входные данные.

На основе этой информации определяется, какая задача решается (что необходимо предсказать). На данном этапе определяется, какой тип обучения используется:

- Контролируемое обучение (обучение с учителем)
- Неконтролируемое обучение (обучения без учителя)
- Самоконтролируемое обучение
- Обучение с подкреплением

На данном этапе необходимо помнить о двух гипотезах, которые должны быть справедливы к данным:

- гипотеза о том, что выходные данные можно предсказать по входным данным
- гипотеза о том, что доступные данные достаточно информативны для изучения отношений между входными и выходными данными

2. **Выбор меры успеха**

На данном этапе определяется, что означает успешная модель. То есть то, как определяется хорошая и правильно обученная модель от плохой. Это может быть: близость, точность и полнота, счет, и.т.д. На основе этой информации выбирается функция, которую необходимо будет оптимизировать.

3. **Выбор протокола оценки**

После определения цели необходимо также выяснить, как измерять движение к ней. Существует 3 распространённых протокола оценки:

- ***выделение из общей выборки отдельного проверочного набора данных***

Смысл состоит в следующем. Некоторая часть данных выделяется в контрольный набор. Обучение производится на оставшихся данных, а оценка качества — на контрольных. Как уже говорилось, для предотвращения утечек информации модель не должна настраиваться по результатам прогнозирования на контрольных данных, поэтому требуется также зарезервировать отдельный проверочный набор.

```
num_validation_samples = 10000
np.random.shuffle(data)
validation_data  = data[:num_validation_samples]
data = data[num_validation_samples:]

training_data =  data[:]
model =  get_model()
model.train(training_data)
validation_score = model.evaluate(validation_data)
```

Это самый простой протокол оценки, страдающий одним существенным недостатком: при небольшом объеме доступных данных проверочный и контрольный наборы могут содержать слишком мало образцов, чтобы считаться статистически репрезентативными.

- ***Перекрестная проверка по K блокам***

При использовании этого подхода данные разбиваются на K блоков равного размера. Для каждого блока i производится обучение модели на остальных K—1 блоках и оценка на блоке i . Окончательная оценка рассчитывается как среднее K промежуточных оценок. Этот метод может пригодиться, когда качество модели слишком сильно зависит от деления данных на тренировочный/контрольный наборы. Подобно проверке с простым расщеплением выборки, этот метод не избавляет от необходимости использовать отдельный проверочный набор для калибровки модели.

```
k = 4
num_validation_samples = len(data) // k
np.random.shuffle(data)
validation_scores = []
for fold in range(k):
    validation_data = data[num_validation_samples * fold: num_validation_samples *
(fold + 1)]
    training_data = data[:num_validation_samples * fold] + data[num_validation_samples
* (fold + 1):]
    model = get_model()
    qmodel.train(training_data)
    validation_score = model.evaluate(validation_data)
    validation_scores.append(validation_score)
```

- ***Итерационная проверка по K блокам с перемешиванием***

Этот метод подходит для ситуаций, когда имеется относительно небольшой набор данных и требуется оценить модель максимально точно. Суть его заключается в многократном применении перекрестной проверки по K блокам с перемешиванием данных перед каждым разделением на K блоков. Конечная оценка — среднее по оценкам, полученным в прогонах перекрестной проверки по K блокам. Обратите внимание: в конечном счете обучению и оценке подвергается P × K моделей (где P — количество итераций), что может быть очень затратным.

4. Предварительная подготовка данных

После того как определена задача и исходные данные для обучения, цель для оптимизации и порядок оценки предпринятого подхода, у вас есть практически все, чтобы начать обучение модели. Но прежде необходимо преобразовать исходные данные в формат, в котором их можно передать в модель машинного обучения — в данном случае в глубокую нейронную сеть:

- данные должны быть представлены в виде тензора
- значения, помещаемые в тензоры, обычно требуют масштабирования и приведения к меньшим величинам: например, в диапазоне [–1, 1] или [0, 1]
- если значения разных признаков находятся в разных диапазонах (разнородные данные), их следует нормализовать
- возможно, понадобится выполнить конструирование признаков,
- особенно при небольшом объеме исходных данных. То есть из исходных признаков сформировать новые в пространстве меньшей размерности. Например, можно использовать метод главных компонент.

5. Разработка модели, более совершенной, чем базовый случай

Цель на этом этапе — достичь статистической мощности, то есть разработать небольшую модель, способную выдать более качественный результат по сравнению с базовым случаем. Стоит обратить внимание на то, что не всегда удается достичь статистической мощности. Если модель не в состоянии дать более высокую точность, чем простой случайный выбор (базовый случай) после опробования нескольких разумных архитектур, вполне возможно, что во входных данных отсутствует ответ на вопрос, который вы пытаетесь задать. Если все идет как надо, вам нужно сделать три ключевых выбора для создания первой рабочей модели:

- Функция активации для последнего уровня — устанавливает эффективные ограничения на результат сети.
- Функция потерь — должна соответствовать типу решаемой задачи.
- Конфигурация оптимизации — какой оптимизатор использовать

Выбирая функцию потерь, имейте в виду, что не всегда можно напрямую оптимизировать показатель успеха решения задачи. Иногда нет простого способа преобразовать показатель успеха в функцию потерь; функции потерь, в конце концов, должны быть вычислимыми на мини-пакетах данных (в идеале функция потерь должна быть вычислимой на очень маленьких объемах данных, вплоть до одного экземпляра) и дифференцируемыми (иначе не получится использовать обратное распространение ошибки для обучения сети).

Выбор функции активации и функции потерь для типовых задач представлен в таблице 1.

Таблица 1

Задача	Функция активации	Функции потерь
Бинарная классификация	sigmoid	binary_crossentropy
Многоклассовая, однозначная классификация	sigmoid	binary_crossentropy
Многоклассовая, многозначная классификация	softmax	categorical_crossentropy
Регрессия по произвольным значениям	Нет	mse
Регрессия по значениям между 0 и 1	sigmoid	mse или binary_crossentropy

6. Масштабирование по вертикали: разработка модели с переобучением

После получения модели, обладающей статистической мощностью, встает вопрос о достаточной мощности модели. Достаточно ли слоев и параметров, чтобы правильно смоделировать задачу?

Не забывайте о распространенной проблеме машинного обучения — противоречии между оптимизацией и общностью; идеальной считается модель, которая стоит непосредственно на границе между недообучением и переобучением, между недостаточной и избыточной емкостью. Чтобы понять, где пролегает эта граница, ее сначала нужно пересечь. Чтобы понять, насколько большой должна быть модель, сначала нужно сконструировать модель, обладающую эффектом переобучения. Сделать это просто:

- Добавьте слои.
- Задайте большое количество параметров в слоях.
- Обучите модель на большом количестве эпох.

7. Регуляризация модели и настройка гиперпараметров

Этот шаг занимает больше всего времени: вам придется многократно изменять свою модель, обучать ее, оценивать качество на проверочных данных (контрольные данные не должны принимать никакого участия в этом этапе), снова изменять ее и повторять этот цикл, пока качество модели не достигнет желаемого уровня. Вот кое-что из того, что вы должны попробовать:

- добавить прореживание
- опробовать разные архитектуры: добавлять и удалять слои
- добавить L1- и (или) L2-регуляризацию
- опробовать разные гиперпараметры (например, число нейронов на слой или шаг обучения оптимизатора), чтобы найти оптимальные настройки
- дополнительно можно выполнить цикл конструирования признаков: добавить новые признаки или удалить имеющиеся, которые не кажутся информативными

Получив удовлетворительную конфигурацию, можно обучить окончательный вариант модели на всех доступных данных (тренировочных и проверочных) и оценить ее качество на контрольном наборе. Если качество модели на контрольных данных окажется значительно хуже, чем на проверочных, это может означать, что ваша процедура проверки была ненадежной или в процессе настройки параметров модели проявился эффект переобучения на проверочных данных. В этом случае можно попробовать переключиться на использование другого, более надежного протокола оценки (такого, как итерационная проверка по K блокам с перемешиванием).