Задание № Б-3.1-2. Метод главных компонент

Введение

В этом упражнении вы используете метод главных компонент (PCA) для нахождения представления изображений лиц в пространстве малой размерности.

1. Метод главных компонент

В этом упражнении вы примените метод главных компонент (Principal Component Analysis, PCA) для решения задачи уменьшения размерности пространства признаков. Сначала вы поэкспериментируете с методом на небольшом двухмерном датасете, чтобы получить интуитивное представление о том, как работает метод, а затем примените его на большой выборке, состоящий из 5000 изображений лиц.

Скрипт `ex7_pca.m` поможет вам с выполнением второй части упражнения.

1.1 Учебная выборка

Для того чтобы глубже познакомиться с тем, как работает метод главных компонент, мы начнем с игрушечной учебной выборки, содержащей всего два измерения, в которой наблюдается большая вариация вдоль одного из направлений и небольшая вдоль другого. Скрипт ex7_pca.m визуализирует выборку (см. Рисунок 4). На данном шаге упражнения вы продемонстрируете, что происходит, когда мы с помощью PCA уменьшаем размерность данных с двух измерений до одного. Конечно, на практике имеет смысл понижать размерность с 256 измерений до, скажем, 50, но, при использовании подобных игрушечных примеров легче понять, что делает PCA с данными.

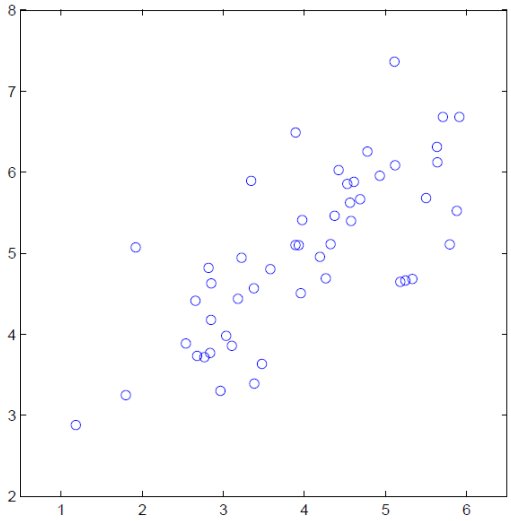

Рисунок 4: Учебная выборка №1

1.2 Реализация PCA

Для начала вам необходимо реализовать PCA. Алгоритм состоит из двух вычислительных шагов: сначала вы вычисляет ковариационную матрицу данных. Затем вы используете функцию SVD (или аналогичную для других платформ), которая вычисляет собственные вектора $U_1, U_2, \ldots, U_n$. Они будут соответствовать главным компонентам (направлениям) данных, в которых наблюдается максимальная вариация (направлению с максимальной вариацией соответствует максимальное собственное число λ_i и так далее в порядке убывания).

Прежде чем применять PCA, важно сначала нормализовать данные. Для этого необходимо из каждого примера вычесть среднее значение по каждому измерению, и разделить его на величину

разброса, чтобы привести все величины к одному масштабу. В скрипте `ex7_pca.m` этот шаг уже выполнен с помощью функции featureNormalize. После нормализации можно запускать сам алгоритм PCA. Ваша задача – завершить код в файле `pca.m`. Первое, что вы должны сделать, это вычислить ковариационную матрицу данных по формуле

$$\Sigma = \frac{1}{m} X^T X,$$

где X – матрица данных, в которой каждая строка – это отдельный пример из обучающей выборки, а m – количество элементов данной выборки. Обратите внимание, что Σ – это матрица $n \times n$, а не оператор суммирования.

После вычисления ковариационной матрицы вы можете запустить на ней алгоритм SVD для вычисления главных компонент. В среде Octave/Matlab это можно сделать, вызвав команду

```
[U, S, V] = svd(Sigma);
```

после которой в U будет находиться список главных компонент, а S будет являться диагональной матрицей.

После того как вы завершите код `pca.m` скрипт `ex7_pca.m` запустит его на учебной выборке и нарисует график с найденными главными компонентами (см. Рисунок 5). Скрипт также напечатает первый главный компонент (собственный вектор). Вы должны увидеть что-то около [-0.707, -0.707]. (Вполне возможно, что Octave выведет и [0.707, 0.707], потому что направление вектора в данном случае роли не играет.)

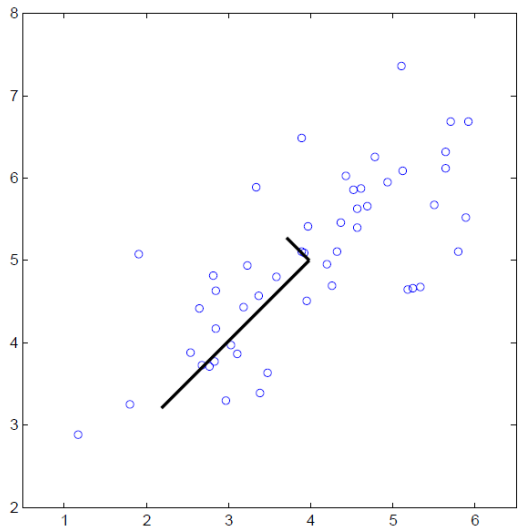

Рисунок 5: Собственные вектора выборки

1.3 Уменьшение размерности с помощью PCA

После вычисления главных компонент можно переходить к уменьшению размерности исходных данных. Делаем мы это путем проецирования данных в пространство меньшей размерности (в нашем примере из двухмерного пространства на линию). В этой части упражнения вы спроецируете примеры из учебной выборки в одномерное пространство, задаваемое первым собственным вектором из списка, найденного на предыдущем шаге.

На практике, если бы вы использовали какую-нибудь модель машинного обучения с учителем, как например, линейную регрессию или нейронную сеть, то вы бы смогли теперь использовать новую (спроецированную) учебную выборку для обучения модели. Так как она содержит меньше признаков, то процесс обучения будет проходить быстрее.

1.3.1 Проецирование данных на главные компоненты

Теперь вам необходимо завершить код в файле `projectData.m`. Вам дана матрица данных X, список главных компонент U и количество измерений K, которые необходимо оставить. Вам необходимо спроецировать каждый пример из X на первые K векторов из U. Обратите внимание, что первые K компонент идут в первых K столбцах U, т.е. U_reduce = U(:,1:K).

После того как завершите код в файле `projectData.m` скрипт `ex7_pca.m` спроецирует первый пример из учебной выборки на первую компоненту, и вы должны увидеть результат около 1.481 (возможно со знаком минус).

1.3.2 Реконструкция аппроксимации данных

После проецирования данных в пространство меньшей размерности вы можете построить аппроксимацию этих данных в пространстве исходной размерности[1]. Ваша задача – завершить код функции `revoverData.m`, который проецирует каждый пример из пространства меньшей размерности обратно в исходное пространство и возвращает результат в X_rec.

После этого запустите скрипт ex7_pca.m и он должен вывести значение около [-1.047, -1.047].

1.3.3 Визуализация проекций

Теперь скрипт `ex7_pca.m`, используя написанные вами функции projectData и recoverData, произведет сначала проекцию учебной выборки, а затем ее реконструкцию в исходном двумерном пространстве и отрисует все на графике (см. Рисунок 6). Исходные точки изображены синими кружками, а спроецированные – красными. Как видим, проецирование оставляет только информацию по направлению, заданному U_1.

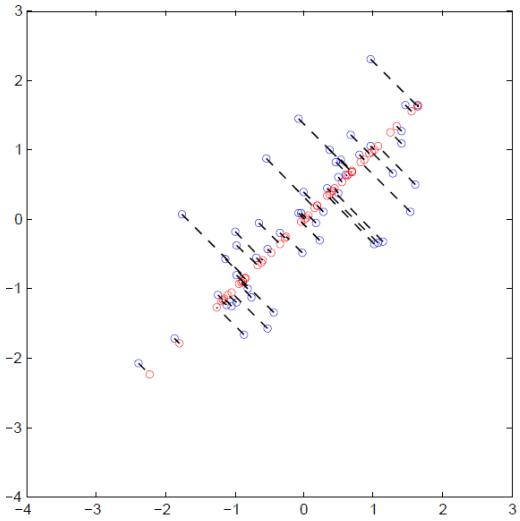

Рисунок 6: Исходные и спроецированные данные

1.4 Датасет с изображениями лиц

В этой части упражнения вы примените алгоритм PCA к реальному датасету, состоящему из изображений лиц, и посмотрите, как на практике используется этот алгоритм для снижения размерности пространства признаков. Файл `ex7faces.mat` содержит набор черно-белых изображений лиц размером 32×32 пикселя. После загрузки данных в X каждая строка матрицы будет содержать одно изображение (вектор длины 1024).

[1] Это полезно, в частности, для визуализации изменений, которые происходят в результате уменьшения размерности методом PCA.

На следующем шаге скрипт ex7_pca.m загрузит всю выборку и визуализирует первые 100 изображений (см. Рисунок 7).

Рисунок 7: Выборка изображений лиц

1.4.1 PCA на лицах

Прежде чем запустить PCA, мы, как и прежде, сначала нормализуем данные, вычитая среднее значение по каждому признаку. Скрипт ex7_pca.m сделает это за вас, а затем запустит написанный вами алгоритм PCA. В результате вы получите список главных компонент учебной выборки. Заметьте, что каждая главная компонента, записанная в строке матрицы U, представляет собой вектор длины n (для нашей выборки с изображениями лиц $n = 1024$). Оказывается, мы можем визуализировать главные компоненты, преобразовав их в матрицы 32×32 и выведя их как черно-белые изображения. Скрипт ex7_pca.m отрисует первые 36 главных компонент, которые соответствуют измерениям с наибольшей вариацией (см. Рисунок 8). Если хотите, вы можете отрисовать и другие главные компоненты, чтобы посмотреть, каким образом они улавливают все больше и больше деталей изображений.

Рисунок 8: Главные компоненты выборки изображений лиц

1.4.2 Понижение размерности

Теперь, когда вы вычислили главные компоненты, вы можете использовать их для понижения пространства признаков выборки с лицами с 1024 измерений до, например, 100. Это позволит существенно ускорить процесс обучения, если вы используете эту выборку, скажем, для обучения нейронной сети.

На следующем шаге скрипт `ex7_pca.m` спроецирует датасет с лицами на первые 100 главных компонент. Каждая картинка теперь будет представляться вектором $z^{(i)} \in \mathbb{R}^{100}$.

Чтобы понять, что же был потеряно в результате понижения размерности, вы можете восстановить данные по новой выборке. Скрипт `ex7_pca.m` делает это и показывает на картинке исходные и восстановленные версии изображений (см. Рисунок 9). Вы можете заметить, что общая структура и внешний вид лиц сохраняются при уменьшении размерности, однако теряются мелкие детали, которые, собственно, и «живут» в маловариативных пространствах, от которых мы избавились. При этом мы добились существенного снижения объема данных – более чем в 10 раз. Теперь, если вы, например, осуществляете распознавание лиц, то вместо исходного большого изображения вы можете подавать на вход классификатору его существенно уменьшенную версию, которая, тем не менее, сохранила бóльшую часть информации.

Рисунок 9: Исходные и проецированные на первые 100 главных компонент изображения

1.5 Применение PCA для визуализации

В первой части упражнения вы использовали алгоритм K-средних для нахождения кластеров в трехмерном RGB-пространстве. В заключительной части скрипта `ex7_pca.m` мы предоставили код, который визуализирует полученные кластеры в виде трехмерного точечного графика. Каждая точка покрашена в цвет своего кластера (центроида). Вы можете повращать мышкой изображение, чтобы рассмотреть его с разных ракурсов.

Очевидно, что визуализация трехмерных данных, не говоря уже о данных большей размерности, вызывает определенные трудности. Поэтому иногда бывает полезно изобразить данные в двухмерном пространстве, даже если при этом теряется часть информации. На практике алгоритм PCA часто используется для снижения размерности произвольной выборки до двух-трех измерений с целью дальнейшей их визуализации.

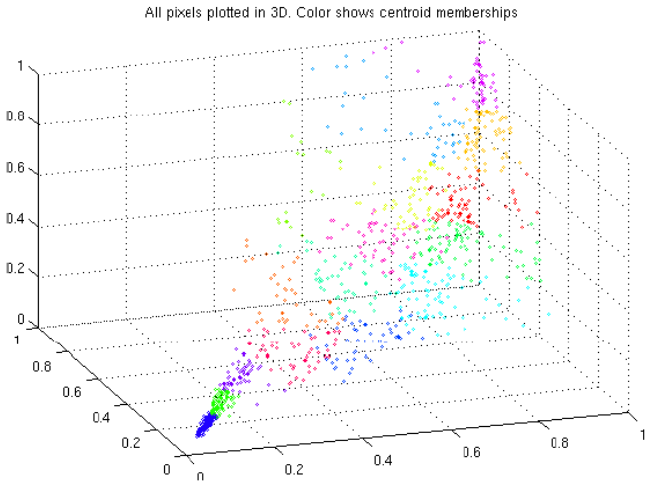

Рисунок 10: Исходные данные в 3D

Скрипт `ex7_pca.m` применит реализованный вами алгоритм PCA для снижения размерности данных из первой части упражнения с трех до двух. Как вы можете видеть из результата (Рисунок 11), проекцию, находимую PCA, можно рассматривать как «поворот» данных, позволяющий обеспечить «наилучший угол обзора», с которого видна максимальная вариация данных.

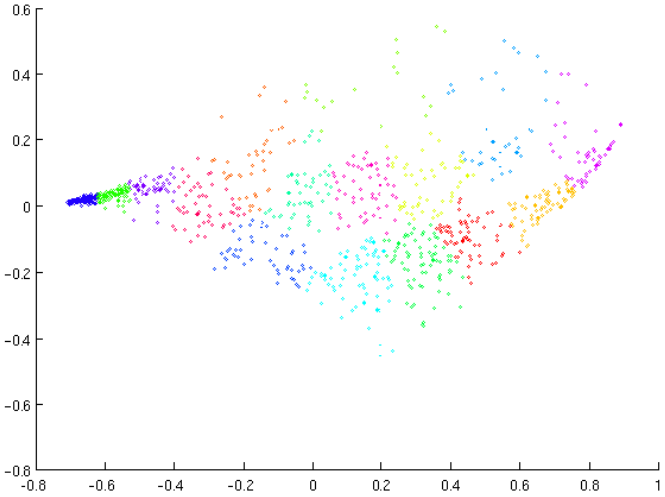

Рисунок 11: 2D-визуализация с помощью PCA