دانشکده مهندسی برق و کامپیوتر

استاد درس:

دستورکار آزمایشگاه هوش محاسباتی

جلسه ۱

برنامه نویسی کارآمد پایتون و NumPy

مقدمه

برای بازدهی محاسباتیِ اَعمال معمولی در کاربردهای ماشین لرنینگ، استفاده از آرایه‌های NumPy همراه با دستورات برداری، بجای استفاده مستقیم از حلقه‌های for بسیار سودمندتر است. دستورات برداری، نسبت به حالات ساده، بهینه‌تر هستند و کارایی کدهای پایتون را به زبان‌های سطح پایین‌تری مانند C نزدیک‌تر می‌کنند. در این جلسه، از شما انتظار می‌رود که به سه تمرین که در حوزه یادگیری ماشین کاربرد دارند، پاسخ دهید.
برای این جلسه، سه فایل ژوپیتر برای شما آماده شده است که شما باید موارد خواسته شده را در این فایل‌ها پیاده‌سازی کنید. همچنین برای آشنا شدن بیشتر با اعمال و توابع ماتریسی در *NumPy* ، فایلِ `npprimer.ipynb` در کنار دیگر فایل‌ها قرار داده شده است که پیشنهاد می‌شود این فایل را نیز مطالعه بفرمایید.
نکته: سه تمرین پیش‌رو قابلیت پیاده‌سازی در بستر حلقه‌های for را نیز دارا هستند اما هدف این جلسه، آشنا شدن شما با روش‌های بهینه‌تر با استفاده از دستوراتِ برداری می‌باشد.

دستوراتِ مفید

در این بخش، قصد داریم نگاهی به دستوراتِ مفید برای نوشتن کدهای برداری بیندازیم. برای دیدن مستندات کامل می‌توانید از دستورِ `help(func)` استفاده کنید.
ابتدا باید کتابخانه‌ی NumPy را فراخوانی کنیم:

```
import numpy as np
```

سپس می‌توانیم از دستورات زیر، استفاده کنیم:

- a * b, a / b : ضرب و تقسیمِ درایه‌ایِ دو ماتریسِ a و b
- a.dot(b) : ضربِ ماتریسیِ دو ماتریسِ a و b
- a.max(0) : پیدا کردنِ بزرگترین درایه در هر ستون از ماتریسِ a (نکته: در NumPy ، اولین عنصرِ، در مکان صفرم قرار دارد، در حالی که در متلب، در خانه یکم قرار میگیرد.)
- a.max(1) : پیدا کردنِ بزرگترین درایه در هر سطر از ماتریسِ a
- np.mean(a) ، np.std(a) : محاسبه‌ی میانگین و انحراف معیارِ تمامی درایه‌های a
- a.shape() : ابعاد ماتریسِ a را نشان می‌دهد.
- a.shape[k] : اندازه‌ی آرایه a را در بعدِ k-ام ، نشان می‌دهد
- np.sum(a, axis=k) : مجموع درایه‌های ماتریسِ a در بُعدِ k -ام را برمی‌گرداند.
- linalg.inv(a) : معکوسِ ماتریسِ مربعیِ a را برمی‌گرداند.

تمرین اول: استانداردسازیِ ماتریس

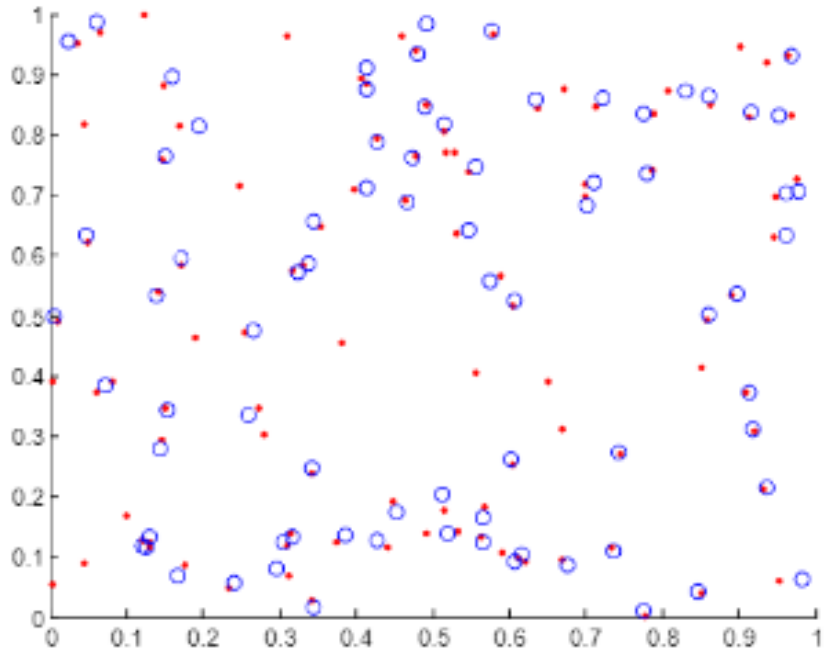
ابعاد و یا ویژگی‌های مختلفِ یک نمونه‌ی داده، اغلب واریانس‌های متفاوتی نسبت به یکدیگر دارند. بخاطر یکسری عملیات که متعاقبا به آن‌ها نیاز پیدا خواهیم کرد، بهتر است که داده‌ی خود را استاندارد کنیم. یعنی برای هر بُعد، داده را از میانگین کم کرده و نتیجه را بر انحراف معیار تقسیم کنیم. بعد از این پردازش، میانگینِ هر بُعد، برابر با صفر و واریانس آن برابر واحد خواهد بود. توجه داشته باشید که این عملیات، مساوی با عملیاتِ سفیدسازیِ داده [۱] ، که علاوه بر این، ابعاد را (با استفاده از چرخش مختصات) از هم، غیرمرتبط می‌سازد، نیست.
تابعی بنویسید که داده‌ی ماتریسیِ $\boldsymbol{x} \in \mathbb{R}^{n \times d}$ را بعنوان ورودی بگیرد و همان داده را بصورت نرمال‌سازی شده، برگرداند. n تعداد نمونه‌ها و d ابعاد ماتریس است. سطرها حاویِ نمونه‌ها و ستون ها، ویژگی‌های نمونه‌ها می‌باشد.

[۱] data whitening

تمرین دوم: فواصلِ جفتی در صفحه

یکی از کاربردهای یادگیری ماشین در بینایی کامپیوتر، ردیابی نقطه‌ی مورد علاقه است. مختصات گوشه‌های یک تصویر، در طول زنجیره ای از فریم های یک سیگنالِ ویدیویی، رهگیری می‌شود. (بعنوان یک مثال ساختگی، شکل یک را ببینید.) در این زمینه، اغلب به فاصله جفتی همه نقاط در فریم اول نسبت به همه نقاط در فریم دوم، توجه می‌شود. تطبیق نقاط با توجه به حداقل فاصله، یک روش ابتکاری ساده است که در صورت یافتن نقاط مورد علاقه در هر دو فریم و وجودِ آشفتگیِ کم، به خوبی کار می کند.

تابعی بنویسید که دو ماتریسِ $\mathbf{P} \in \mathbb{R}^{p\times 2}, \mathbf{Q} \in \mathbb{R}^{q\times 2}$ را به عنوان ورودی دریافت می‌کند. هر سطر در این ماتریس ها، حاوی مختصاتِ یک نقطه‌ی علاقه [۲] می‌باشد. به یاد داشته باشید که تعداد نقاط (p و q) لزوما یکسان نیستند. به عنوان خروجی، شما باید فاصله‌ی جفتیِ تمام نقاط داخلِ $\mathbf{P}$ را نسبت به تمام نقاط داخلِ $\mathbf{Q}$ محاسبه کنید و نتیجه را داخل ماتریسِ $\mathbf{D}$ ذخیره کنید. درایه‌ی $D_{i,j}$ ، فاصله‌ی اقلیدسی نقطه‌ی i –ام در $\mathbf{P}$ و نقطه‌ی j –ام در $\mathbf{Q}$ می باشد.



شکل ۱: دو مجموعه نقاط در صفحه، دایره‌ها، زیرمجموعه‌ای از نقطه ها هستند و بصورت تصادفی، آشفته‌سازی شده اند.

[۲] interest point

تمرین سوم: احتمالِ دو نمونه‌ی داده

در این تمرین، شما نیاز نیست که مفاهیم یادگیری ماشین و آماریِ تشریح شده را متوجه شوید. هدف فقط پیاده‌سازیِ نسبت‌دادنِ داده‌ها به دو توزیعِ داده شده، در پایتون، می‌باشد.

یک عمل فرعی در بسیاری از الگوریتم های یادگیری ماشین، محاسبه‌ی احتمالِ $p(\boldsymbol{x}_n \mid \boldsymbol{\theta})$ برای یک نمونه‌ی $\boldsymbol{x}_n$ برای مدل مقصد با پارامتر های θ می‌باشد. با داشتنِ k مدل، اکنون هدف ما، نسبت‌دادنِ $\boldsymbol{x}_n$ به مدل به نحوی که احتمال آن بیشینه شود، است:

$$a_n = \text{argmax}_m p(\boldsymbol{x}_n \mid \boldsymbol{\theta}_m) \text{ where } m = 1, \ldots, k.$$

در اینجا، $\boldsymbol{\theta}_m = (\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ پارامتر های m -اُمین مدلِ مقصد هستند. ($\mu_m \in \mathbb{R}^d$ میانگین است و $\boldsymbol{\sigma}_m$ ، کوواریانسِ ماتریس نامیده می‌شود.) شما باید مرحله‌ی نسبت دادنِ دو نمونه مدل را، در صورت وجود، پیاده‌سازی نمایید. یعنی در اینجا $k = 2$ است. بعنوان ورودی، تابعِ شما، یک مجموعه از مثالهای داده‌ایِ $\boldsymbol{x}_m \in \mathbb{R}^d$ (که بصورتِ $1 \leq n \leq N$ شماره‌گذاری شده اند) را همراه با دو مجموعه از پارامترهای $\boldsymbol{\theta}_1 = (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ و $\boldsymbol{\theta}_2 = (\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ مربوط به دو دو توزیعِ چندمتغیره‌ی گاوسی داده شده‌ی:

$$p(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu})\right).$$

را دریافت می‌کند. $|\boldsymbol{\Sigma}|$ دترمینانِ $\boldsymbol{\Sigma}$ و $\boldsymbol{\Sigma}^{-1}$ وارون آن است. تابع شما باید «مشابه ترین» تبدیلِ $a_n \in \{1, 2\}$ را برای هر نقطه‌ی ورودیِ n برگرداند. هنگامی که $a_n = 1$ است به این معنی است که x_n به مدلِ یک نسبت‌داده شده است.به بیان دیگر در حالتی که $a_n = 1$ است، نشان می‌دهد که : $p(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) > p(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$