



دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

دستورکار آزمایشگاه هوش محاسباتی

جلسه ۲

رگرسیون خطی و نزول گرادیان

استاد درس: دکتر مهران صفایانی

فصل ۲

رگرسیون خطی و نزول گرادیان

اهداف این جلسه

شما در این جلسه یاد خواهید گرفت که :

- چگونه `grid search` ، `Gradien descent` و `stochastic gradient descent` را پیاده سازی کنید
- چگونه کدهای خود را اشکال زدایی کنید.
- چگونه نتایج را بصورت بصری نمایش دهید.
- برتری‌ها و کاستی‌های این الگوریتم‌ها را متوجه شوید؟
- اثر داده‌های پرت را بر روی توابع هزینه `MSE` و `MAE` بررسی کنید.

در این جلسه از دیتاست `height-weight-genders.csv` استفاده خواهیم کرد همچنین کدهای نمونه و کمکی برای شما آماده شده است. در طول این جلسه، شما بر روی فایل `ex02.ipynb` کار خواهید کرد. وظیفه شما نوشتن توابع مورد نیاز در این فایل می‌باشد. برای کمک به شما دو فایل `helpers.py` و `plots.py` در کنار فایل‌های این جلسه قرار داده شده‌اند. لطفا قبل از شروع، این فایل‌ها را مطالعه نمایید.

۱.۲ محاسبه تابع هزینه

در این تمرین، ما تمرکز خود را بر روی رگرسیون خطی ساده خواهیم گذاشت، این رابطه به شکل زیر است:

$$y_n \approx f(x_{n1}) = w_0 + w_1 x_{n1} \quad (1.2)$$

ما از `height` به عنوان ورودی `xn1` و از `weight` به عنوان خروجی `yn` استفاده خواهیم کرد. ضرایب `w0` و `w1` به عنوان پارامترهای مدل هم شناخته می‌شوند. ما از یک تابع کمینه مربع خطا (`MSE`) استفاده خواهیم کرد که به صورت زیر تعریف می‌شود:

$$\mathcal{L}(w_0, w_1) = \frac{1}{2N} \sum_{n=1}^N (y_n - f(x_{n1}))^2 = \frac{1}{2N} \sum_{n=1}^N (y_n - w_0 - w_1 x_{n1})^2 \quad (2.2)$$

هدف ما پیدا کردن `w0*` و `w1*` ای است که هزینه ما را کمینه کند.

برای شروع، سراغ `array data type` در کتابخانه `NumPy` می‌رویم. تمامی جفت‌های `(yn, xn1)` را در یک بردار و یک ماتریس، مانند زیر، ذخیره‌سازی کرده ایم:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \tilde{X} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{21} \\ \vdots & \vdots \\ 1 & x_{N1} \end{bmatrix} \quad (3.2)$$

تمرین اول

برای یادگیری ساختار اطلاعات، به سوالات زیر پاسخ دهید:

- هر ستون `\tilde{X}` چه چیزی را نشان می‌دهد؟
- هر سطر `\tilde{X}` چه چیزی را نشان می‌دهد؟
- چرا مقادیر عددی ۱ در `\tilde{X}` داریم؟
- اگر وزن و قد سه نفر را داشته باشیم، سائز `y` و `\tilde{X}` چه می‌شد؟ عنصر `\tilde{X}_{32}` چه چیزی را نشان می‌داد؟
- در فایل `helpers.py` یک کد برای فرم دادن به آرایه‌ها برای `y` و `\tilde{X}` آماده شده است. کد را به خوبی مشاهده کرده و مطمئن شوید که نحوه ساخت آن‌ها را متوجه شده اید.
- ۱. می‌خواهیم `MSE` را محاسبه کنیم، اگر بردار `e` بصورت `$e = y - \tilde{X}w$` ، با پارامترهای `$w = [w_0, w_1]^T$` تعریف شود، اثبات کنید که فرمول `MSE` را می‌توان با استفاده از برداری که تعریف کردیم نیز بدست آورد:

$$\mathcal{L} = \dots \quad (4.2)$$

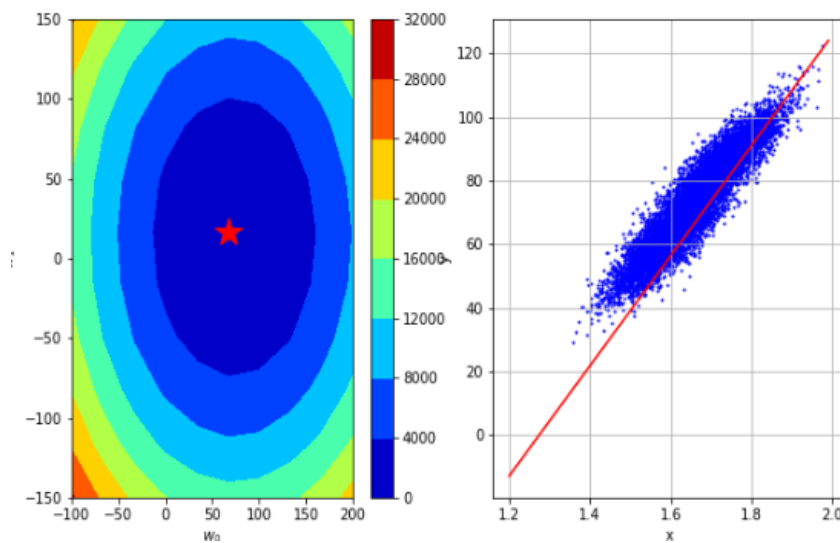
۲. کد تابع `compute_loss(y, tx, w)` را تکمیل کنید. برای تست می‌توانید از مقادیر `$w = [1, 2]^T$` استفاده کنید.

۲.۲ Grid Search

حال، ما برای پیاده‌سازی اولین الگوریتم بهینه سازی خود آماده هستیم.

تمرین دوم

آ. تابع `grid_search(y, tx, w0, w1)` را تکمیل کنید. برای این کار شما باید یک `for-loop` به ازای هر بُعد تشکیل دهید و تابع هزینه را برای هر مجموعه‌ی W_0 و W_1 محاسبه کنید. هنگامی که تمامی مقادیر تابع هزینه را در متغیر `loss` داشتید، کد مقدار کمینه‌ی تقریبی را پیدا خواهد کرد. کد باید مقدار کمینه‌ی پیدا شده برای تابع هزینه را در کنار مقادیر پیدا شده‌ی w_0^* و w_1^* چاپ کند. همچنین باید یک `contour-plot` و نمودار برازش را همانطور که در شکل زیر نشان داده شده است، نشان دهد.



شکل ۱.۲: نمایش مصور نتایج `grid search`

ب. بنظر شما، آیا این یک تخمین خوب است؟ در صورت منفی بودن پاسخ، به نظر شما مشکل کار کجاست؟ چرا MSE هموار و نرم نیست؟

تمرین بالا را با تغییر فاصله‌ی `grid` از ۵۰ به ۱۰، تکرار کنید. خروجی را با حالت قبل مقایسه کنید.

ج. به سوال های زیر پاسخ دهید:

- برای رسیدن به یک پاسخ و مدل سازی دقیق، داشتن `grid` درشت^۱ بهتر است یا ریز^۲؟
- مقادیر مختلف فاصله‌ی `grid` را امتحان کنید. چه چیزی مشاهده می‌کنید؟
- افزایش مقادیر متغیرها، بر سرعت محاسبات چه تاثیری دارد؟

^۱coarse
^۲fine

۳.۲ Gradient Descent

در درس، عبارات زیر را در مورد گرادیان MSE (بردار مشتقات جزئی) برای رگرسیون خطی بدست آوردیم:

$$\frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_0} = -\frac{1}{N} \sum_{n=1}^N (y_n - w_0 - w_1 x_{n1}) = -\frac{1}{N} \sum_{n=1}^N e_n \quad (5.2)$$

$$\frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_1} = -\frac{1}{N} \sum_{n=1}^N (y_n - w_0 - w_1 x_{n1}) x_{n1} = -\frac{1}{N} \sum_{n=1}^N e_n x_{n1} \quad (6.2)$$

اگر گرادیان را با $\nabla \mathcal{L}(\mathbf{w})$ نشان دهیم، می‌توانیم عملیات زیر را در فرم برداری، بنویسیم:

$$\nabla \mathcal{L}(\mathbf{w}) := \left[\frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_0} \quad \frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_1} \right] = -\frac{1}{N} \begin{bmatrix} \sum_{n=1}^N e_n \\ \sum_{n=1}^N e_n x_{n1} \end{bmatrix} = -\frac{1}{N} \tilde{\mathbf{X}}^\top \mathbf{e} \quad (7.2)$$

تمرین سوم

آ. حال تابعی را پیاده سازی کنید که مقدار گرادیان ها را حساب کند. تابع `compute_gradient(y, tx, w)` را با استفاده از معادله‌ی شماره ۷ پیاده سازی کنید. مطمئن شوید که مقادیر خروجی تابع شما، معتبر هستند. برای این کار، بصورت دستی با مقادیر از پیش تعیین شده‌ی y و $\tilde{\mathbf{X}}$ و w مقادیر گرادیان را محاسبه و با خروجی تابع مقایسه کنید.

ب. هنگامی که از معتبر بودن خروجی های تابع خود مطمئن شدید، لازم است که نسبت به مقادیر گرادیان شهود پیدا کنید، برای این کار، گرادیان را برای مقادیر

$$\bullet w_1 = 20 \text{ و } w_0 = 100$$

$$\bullet w_1 = 10 \text{ و } w_0 = 50$$

محاسبه کنید.

مقادیر گرادیان ها، چه چیزی را نشان می‌دهند؟ برای مثال، به نرم این بردار توجه کنید. در چه حالتی، بزرگتر است؟ از این مشاهده، چه نتیجه ای می‌توان گرفت؟
راهنمایی یک: یک تابع درجه دوم را تصور کنید و گرادیان آن را نزدیک کمینه و دور از کمینه آن، در نظر بگیرید.
راهنمایی دو: همانطور که می‌دانید، برای بروزرسانی در نزول گرادیانی، در گام t ام، از قانون زیر بهره گرفته می‌شود:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}(\mathbf{w}^{(t)}) \quad (8.2)$$

که $\gamma > 0$ همان اندازه گام^۱ است و $\nabla \mathcal{L} \in \mathbf{R}^2$ ، بردار گرادیان است.

ج. تابع `gradient_descent(y, tx, initial_w, ...)` را تکمیل کنید. کد را اجرا و فرایند هر مرحله را بصورت مصور در بیاورید. همچنین، به پیام های چاپ شده ای که مقادیر \mathcal{L} و $w_0^{(t)}$ و $w_1^{(t)}$ را نشان می‌دهد، دقت کنید. با توجه به نمودار های خروجی،

¹step size

- آیا هزینه، کمینه شده است؟
 - آیا الگوریتم در حال همگرایی است؟ درمورد سرعت همگرایی چه نظری می توان داد؟
 - مقادیر نهایی w_0 و w_1 تا چه اندازه قابل قبول هستند؟
- د. حال می خواهیم در مورد مقدار اندازه گام مقادیر پارامترها تجاری را بدست آوریم و ببینیم که تاثیر آنها بر همگرایی، چگونه است. در تئوری، نزول گرادیانی، در صورت انتخاب درست اندازه گام در یک تابع محدب به مقدار بهینه خود، همگرا می شود.

- مقادیر 0.001, 0.01, 0.5, 1, 2, 2.5 را برای step size امتحان کنید. چه چیزی مشاهده می شود؟ آیا فرایند به همگرایی ختم می شود؟
- برای مقدار ثابت $\gamma = 0.1$ مقدار دهی های اولیه

$$w_1 = 0, w_0 = 0 -$$

$$w_1 = 10, w_0 = 100 -$$

$$w_1 = 1000, w_0 = -1000 -$$

را امتحان کنید.

چه چیزی مشاهده می کنید؟ آیا فرایند به همگرایی ختم می شود؟

۴.۲ Stochastic Gradient Descent

تمرین چهارم:

حال می خواهیم stochastic gradient descent را پیاده سازی کنیم. در درس خواندیم که قانون بروزرسانی برای تابع هدف $\mathcal{L}(w) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}_n(w)$ در مرحله t ام، برابر:

$$w^{(t+1)} = w^{(t)} - \gamma \nabla \mathcal{L}_n(w^{(t)}) \quad (9.2)$$

می باشد.

راهنمایی: شما می توانید از تابع `batch_iter()` که در فایل `helpers.py` وجود دارد، برای ایجاد داده های mini-batch برای SGD استفاده کنید.

۵.۲ تاثیر داده های پرت بر روی تابع هزینه MAE

در درس، درمورد داده های پرت صحبت کردیم، داده های پرت ممکن است نتیجه ی خطای اندازه گیری باشند. برای مثال در داده های مربوط به قد/وزن یک خطای اندازه گیری، ممکن است وزن را به جای کیلوگرم، به پوند محاسبه و ثبت کند. داده های پرت اینچنینی، ممکن است تاثیر بسیار زیادی بر روی پارامترهای مدل (model parameters) داشته باشند. برای مثال، MSE (تابعی که در تمرین های بالا آن را پیاده سازی کردید) به حساس بودن به داده های پرت معروف است.

تمرین پنجم

حال وجود دو داده ی پرت و تاثیر آنها بر روی تابع هزینه MSE را شبیه سازی میکنیم.

- بوسیله‌ی تابع `load_data()` و با مقدار دهی `True` به متغیر `sub_sample` داده را بازفراخوانی کنید. این کار باعث می‌شود تعداد کمی داده بارگذاری شود
- داده‌ها را رسم کنید. شما باید یک ابر از داده‌های مشابه ولی کمتر چگال، نسبت به چیزی که قبلاً در مورد کل دیتاست مشاهده کرده بودید، مشاهده کنید.
- مانند گذشته، مقادیر w_0 و w_1 را برای برازش کردن مدل خطی با استفاده از تابع هزینه‌ی `MSE` پیدا کنید. سپس نتیجه را در کنار داده‌ها بر روی نمودار نشان دهید.
- حال می‌خواهیم دو داده‌ی پرت را به مجموعه داده‌های خود اضافه کنیم. فرض میکنیم این اشتباه در اثر وارد کردن جرم بر حسب پوند بجای کیلوگرم، اتفاق افتاده است. این فرایند می‌تواند با مقداردهی `add_outlier=True` در تابع `load_data()` صورت بپذیرد. می‌توانید داده‌های پرت بیشتری هم اضافه کنید.
- با دیتاست جدید، مدل خود را دوباره برازش کنید. آیا این مدل شبیه یک مدلسازی خوب است؟
- یک راه برای جلوگیری از تاثیر داده‌های پرت، استفاده از یک تابع هزینه‌ی قدرتمندتر، مانند میانگین خطای مطلق (MAE) است.

۶.۲ Subgradient Descent

تمرین ششم

تابع `compute_loss(y, tx, w)` را به منظور استفاده از MAE تغییر دهید. متأسفانه، ما نمی‌توانیم به طور مستقیم از gradient descent استفاده کنیم زیرا تابع MAE ممکن است در چندین نقطه، مشتق ناپذیر باشد.

آ. یک subgradient از تابع هزینه‌ی MAE را برای هر بردار w دلخواه، محاسبه کنید. راهنمایی: از مشتق زنجیره‌ای برای محاسبه‌ی subgradient تابع قدر مطلق استفاده کنید. برای تابع $\mathcal{L}(w) = h(q(w))$ که q مشتق پذیر است، می‌توان subgradient را با استفاده از $\nabla q(w) \cdot \partial h(q(w))$ محاسبه نمود که هر $\partial \cdot$ ، مجموعه‌ی بردارهای subgradient را نشان می‌دهد.

ب. subgradient descent را برای تابع هزینه‌ی MAE پیاده سازی کنید. برای این کار، یک تابع جدید به اسم `compute_subgradient(y, tx, w)` برای MAE بنویسید و آن را طوری تغییر دهید که اگر ورودی w مشتق ناپذیر بود، یک subgradient را برگرداند. مدل خروجی را بر روی نمودار، همراه با دو منحنی مربوط به تمرین قبل، با هم نشان دهید.

- آیا خروجی‌ای که با استفاده از MAE بدست آمده، از خروجی‌ای که با استفاده از MSE بدست آوردیم، بهتر است؟

- آیا الگوریتم بهینه‌سازی شما، به نقطه‌ی مشتق ناپذیری برخورد کرد؟

ج. (SGD) stochastic subgradient descent را برای تابع هزینه‌ی MAE پیاده سازی کنید. در هنگام مقایسه، تصویر نتیجه‌ی دو الگوریتم MSE و MAE چه تفاوت‌هایی با هم دارند؟

جمع بندی

بعد از آنکه پیاده‌سازی تمرین‌های بالا را در فایل `ex02.ipynb` به اتمام رساندید، می‌توانید کدهای خود را بصورت جداگانه در فایل‌هایی با پسوند `.py` کپی کنید. برای این کار، ما قالب هایی را در فایل های: `cost.py` ، `Boxgrid_search.py` ، `gradient_descent.py` و `stochastic_gradient_descent.py` آماده کرده‌ایم که می‌توانید از آن‌ها استفاده کنید.