دانشکده مهندسی برق و کامپیوتر

استاد درس:

دستورکار آزمایشگاه هوش محاسباتی

جلسه 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 کد های خود را اشکال‌زدایی کنید.
- چگونه نتایج را بصورت بصری نمایش دهید.
- برتری‌ها و کاستی‌های این الگوریتم ها را متوجه شوید؟
- اثرِ داده های پرت را بر روی توابع هزینه ی MSE و MAE بررسی کنید.

در این جلسه از دیتاستِ `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 فایل های این جلسه قرار داده شده اند. لطفا قبل از شروع، این فایل ها را مطالعه نمایید.

۱ محاسبه تابع هزینه

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

$$y_n \approx f(x_{n1}) = w_0 + w_1 x_{n1} \tag{۱}$$

ما از `height` به عنوان ورودیِ x_{n1} و از `weight` به عنوان خروجیِ y_n استفاده خواهیم کرد. ضرایبِ w_0 و w_1 به عنوان *پارامتر های مدل* هم شناخته می‌شوند. ما از یک تابع کمینه مربع خطا *(MSE)* استفاده خواهیم کرد که به صورت زیر تعریف می‌شود:

$$\mathcal{L}(w_0, w_1) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - f(x_{n1})\right)^2 = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - w_0 - w_1 x_{n1}\right)^2 \tag{۲}$$

هدف ما پیدا کردنِ w_0^* و w_1^* ای است که هزینه ما را کمینه کند.

برای شروع، سراغ array data type در کتابخانه *NumPy* می‌رویم. تمامی جفت های (y_n, x_{n1}) را در یک بردار و یک ماتریس، مانند زیر، ذخیره‌سازی کرده ایم:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \tilde{X} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{21} \\ \vdots & \vdots \\ 1 & x_{N1} \end{bmatrix} \tag{۳}$$

تمرین اول

برای یادگیری ساختار اطلاعات، به سوالات زیر پاسخ دهید:

- هر ستونِ $\tilde{X}$ چه چیزی را نشان می‌دهد؟
- هر سطر $\tilde{X}$ چه چیزی را نشان می‌دهد؟
- چرا مقادیرِ عددیِ ۱ در $\tilde{X}$ داریم؟
- اگر وزن و قدِ سه نفر را داشته باشیم، سایز y و $\tilde{X}$ چه می‌شد؟ عنصرِ $\tilde{X}_{32}$ چه چیزی را نشان می‌داد؟
- در فایلِ `helpers.py` یک کد برای فرم دادن به آرایه ها 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 پارامتر های $\boldsymbol{w} = [w_0, w_1]^\top$ تعریف شود، **اثبات کنید** که فرمول MSE را می‌توان با استفاده از برداری که تعریف کردیم نیز بدست آورد :

$$\mathcal{L} = \dots \tag{۴}$$

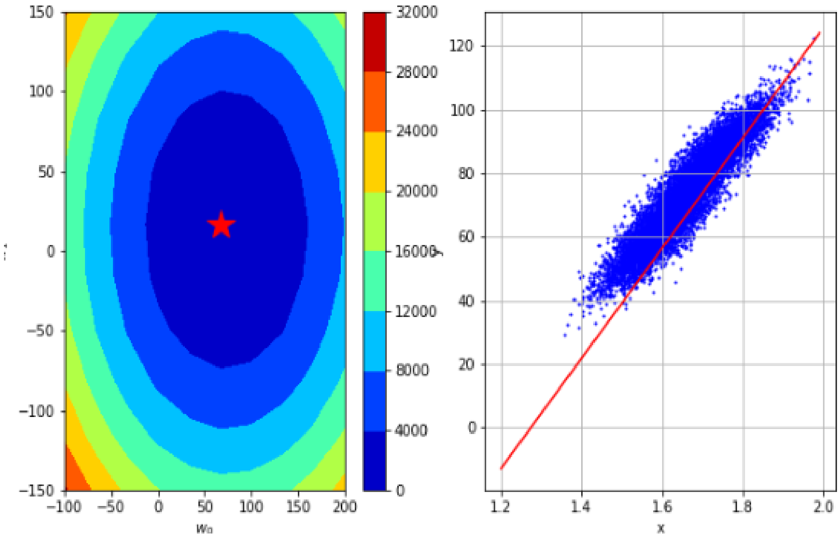
۲. کد تابع `compute_loss(y,tx,w)` را تکمیل کنید. برای تست می توانید از مقادیرِ $\boldsymbol{w} = [1, 2]^\top$ استفاده کنید.

۲ Grid Search

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

تمرین دوم

آ. تابعِ `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 درکنار مقادیر پیدا شده‌ی w_0^* و w_1^* چاپ کند. همچنین باید یک contour-plot و نمودارِ برازش را همانطورکه در شکل زیر نشان داده شده است، نشان دهد.



شکل ۱: نمایش مصور نتایجِ grid search

ب. بنظر شما، آیا این یک تخمینِ خوب است؟ در صورت منفی بودنِ پاسخ، به نظر شما مشکل کار کجاست؟ چرا MSE هموار و نَرم 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 مدلسازی دقیق، داشتنِ grid دُرُشت [۱] بهتر است یا ریز [۲] ؟
- مقادیر مختلفِ فاصله‌ی grid را امتحان کنید. چه چیزی مشاهده می‌کنید؟
- افزایش مقادیر متغیر ها، بر سرعت محاسبات چه تاثیری دارد؟

[۱] coarse
[۲] fine

۳ Gradient Descent

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

$$\frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_0} = -\frac{1}{N}\sum_{n=1}^{N}(y_n - w_0 - w_1 x_{n1}) = -\frac{1}{N}\sum_{n=1}^{N} e_n \tag{۵}$$

$$\frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_1} = -\frac{1}{N}\sum_{i=1}^{N}(y_n - w_0 - w_1 x_{n1})\, x_{n1} = -\frac{1}{N}\sum_{n=1}^{N} e_n x_{n1} \tag{۶}$$

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

$$\nabla\mathcal{L}(\boldsymbol{w}) := \left[\frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_0} \frac{\partial \mathcal{L}(w_0, w_1)}{\partial w_1}\right] = -\frac{1}{N}\begin{bmatrix} \sum_{n=1}^{N} e_n \\ \sum_{n=1}^{N} e_n x_{n1} \end{bmatrix} = -\frac{1}{N}\tilde{\boldsymbol{X}}^{\top}\boldsymbol{e} \tag{۷}$$

تمرین سوم

آ . حال تابعی را پیاده سازی کنید که مقدار گرادیان ها را حساب کند. تابعِ `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 این‌کار، بصورت دستی با مقادیر از پیش تعیین شده‌ی y و $\tilde{\boldsymbol{X}}$ و $\boldsymbol{w}$ 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 این‌کار، گرادیان را برای مقادیرِ

- $w_1 = 20$ و $w_0 = 100$
- $w_1 = 10$ و $w_0 = 50$

محاسبه کنید.

مقادیر گرادیان ها، چه چیزی را نشان می‌دهند؟ برای مثال، به نُرمِ 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 گامِ $\mathbf{t}$ اُم، از قانون زیر بهره گرفته می‌شود:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \gamma\nabla\mathcal{L}\left(\boldsymbol{w}^{(t)}\right) \tag{۸}$$

که $\gamma > 0$ همان *اندازه‌گام* [۳] است و $\nabla\mathcal{L} \in \boldsymbol{R}^2$ ، بردار گرادیان است.

ج . تابعِ `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 مقادیرِ $\mathcal{L}$ و $w_1^{(t)}$ $w_0^{(t)}$ را نشان می‌دهد، دقت کنید. با توجه به نمودار های خروجی،

- آیا هزینه، کمینه شده است؟
- آیا الگوریتم در حال همگرایی است؟ درمورد سرعت همگرایی چه نظری می‌توان داد؟
- مقادیر نهاییِ w_0 و w_1 تا چه اندازه قابل قبول هستند؟

د . حال میخواهیم در مورد مقدارِ اندازه‌گام مقادیر پارامترها تجاربی را بدست آوریم و ببینیم که تاثیر آنها بر همگرایی، چگونه است. در تئوری، نزول گرادیانی، در صورت انتخابِ درستِ اندازه‌گام در یک تابعِ محدب به مقدار بهینه خود، همگرا می‌شود.

- مقادیرِ $0.001, 0.01, 0.5, 1, 2, 2.5$ را برای step size امتحان کنید. چه چیزی مشاهده می‌شود؟ آیا فرایند به همگرایی ختم می‌شود؟

[۳] step size

- برای مقدار ثابت $\gamma = 0.1$ مقدار دهی های اولیه‌ی
  - $w_0 = 0$ ، $w_1 = 0$
  - $w_0 = 100$ ، $w_1 = 10$
  - $w_0 = -1000$ ، $w_1 = 1000$

  را امتحان کنید.
  چه چیزی مشاهده می‌کنید؟ آیا فرایند به همگرایی ختم می‌شود؟

۴ Stochastic Gradient Descent

تمرین چهارم:

حال می‌خواهیم stochastic gradient descent را پیاده سازی کنیم. در درس خواندیم که قانون بروزرسانی برای تابع هدفِ $\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n(\boldsymbol{w})$ در مرحله‌ی t اُم، برابرِ:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \gamma\nabla\mathcal{L}_n\left(\boldsymbol{w}^{(t)}\right) \tag{۹}$$

می‌باشد.

راهنمایی: شما می‌توانید از تابعِ `batch_iter()` که در فایلِ `helpers.py` وجود دارد، برای ایجادِ داده‌ی mini-batch برای SGD استفاده کنید.

۵ تاثیر داده های پَرت بر روی تابع هزینه‌ی MAE

در درس، درمورد داده های پَرت صحبت کردیم، داده‌های پرت ممکن است نتیجه‌ی خطای اندازه‌گیری باشند. برای مثال در داده های مربوط به قد/وزن یک خطای اندازه‌گیری، ممکن است وزن را به جای کیلوگرم، به پوند محاسبه و ثبت کند. داده‌های پَرتِ اینچنینی، ممکن است تاثیر بسیار زیادی بر روی پارامتر های مدل (model parameters) داشته باشند. برای مثال، MSE (تابعی که در تمرین های بالا آن را پیاده‌سازی کردید) به حساس بودن به داده های پَرت معروف است.

تمرین پنجم

حال وجود دو داده‌ی پَرت و تاثیر آنها بر روی تابع هزینه‌ی MSE را شبیه‌سازی میکنیم.

- بوسیله‌ی تابعِ `load_data()` و با مقدار دهیِ `True` به متغیرِ 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 قبلا در مورد کل دیتاست مشاهده کرده بودید، مشاهده کنید.
- مانند گذشته، مقادیرِ w_0و w_1 را برای برازش کردنِ 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 داده های خود اضافه کنیم. فرض میکنیم این اشتباه در اثر وارد کردن جرم بر حسب پوند بجای کیلوگرم، اتفاق افتاده است. این فرایند می‌تواند با مقداردهیِ `add_outlier=`$True$ در تابع `load_data()` صورت بپذیرد. می‌توانید داده های پرت بیشتری هم اضافه کنید.
- با دیتاست جدید، مدل خود را دوباره برازش کنید. آیا این مدل شبیهِ یک مدلسازی خوب است؟

یک راه برای جلوگیری از تاثیر داده‌های پَرت، استفاده از یک تابع هزینه‌ی *قدرتمندتر*، مانند میانگین خطای مطلق (MAE) است.

۶ Subgradient Descent

تمرین ششم

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

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

ب. subgradient descent را برای تابع هزینه‌ی MAE پیاده سازی کنید.
برای این‌کار، یک تابع جدید به اسمِ `compute_subgradient(y,tx,w)` برای MAE بنویسید و آن را طوری تغییر دهید که اگر ورودیِ $\boldsymbol{w}$ مشتق‌ناپذیر بود، یک subgradient را برگرداند.
مدل خروجی را بر روی نمودار، همراه با دو منحنی مربوط به تمرین قبل، با هم نشان دهید.

- آیا خروجی ای که با استفاده از MAE بدست آمده، از خروجی ای که با استفاده از MSE بدست آوردیم، بهتر است؟
- آیا الگوریتمِ بهینه‌سازی شما، به نقطه‌ی مشتق‌ناپذیری برخورد کرد؟

ج. stochastic subgradient descent (SGD) را برای تابع هزینه‌ی MAE پیاده سازی کنید.
درهنگام مقایسه، تصویر نتیجه‌ی دو الگوریتمِ MSE و MAE چه تفاوت هایی 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, grid_search.py, gradient_descent.py` و `stochastic_gradient_descent.py` آماده کرده‌ایم که می‌توانید از آن‌ها استفاده کنید.