
# دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

# دستورکار آزمایشگاه هوش محاسباتی

جلسه ۶
ماشینِ بردارِ پشتیبان

استاد درس: دکتر مهران صفایانی

# فصل ۶

# ماشینِ بردارِ پشتیبان

## اهداف این جلسه

شما در این جلسه یاد خواهید گرفت که :

- با استفاده از نزول گرادیانِ تصادفی [۱] و نزولِ هم‌پایه [۲] ، ماشینِ بردارِ پشتیبان [۳] را پیاده‌سازی و اشکال‌زدایی کنید.
- به‌روزرسانی‌هایی را برای الگوریتمِ نزولِ هم‌پایه برای مسئله‌ی بهینه‌سازی دوگانه [۴] برای SVM استخراج کنید.
- الگوریتمِ نزولِ هم‌پایه را پیاده‌سازی و اشکال‌زدایی کنید.
- آن را با پاسخ اولیه مقایسه کنید.

در این جلسه از مجموعه دادهِ اسباب بازی که در scikit-learn موجود است، استفاده خواهیم کرد. همچنین نمونه کدهای آماده که حاوی قطعات مفیدِ کد، که در این تمرین به آنها نیاز پیدا خواهید کرد، هستند نیز برای شما آماده شده است.

---

[۱] Stochastic Gradient Descent (SGD)
[۲] coordinate descent
[۳] Support Vector Machine(SVM)
[۴] dual optimization

## ۱.۶ SVM با استفاده از SGD

تا اینجا، برای عملِ طبقه‌بندی، رگرسیون خطی و لاجیستیک را پیاده‌سازی کرده‌ایم. در این تمرین قصد داریم از SVM برای عملِ طبقه‌بندیَ استفاده کنیم. همانطور که می‌دانید، مسئله‌ی بهینه‌سازیِ اصلی برای SVM توسط رابطه‌ی

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} \sum_{n=1}^{N} \ell\left(y_n \boldsymbol{x}_n^\top \boldsymbol{w}\right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{۱.۶}$$

تعریف می‌شود که $\ell : \mathbb{R} \to \mathbb{R}, \ell(z) := \max\{0, 1-z\}$ تابعِ hinge loss می‌باشد. در اینجا برای هر $n$ ، $1 \le n \le N$ ، بردارِ $\boldsymbol{x_n} \in \mathbb{R}^{\boldsymbol{D}}$ ، $n$ ـاُمین نمونه‌ی داده است و $y_n \in \pm 1$ برچسبِ متناظرِ آن است.

### تمرین اول: پیاده‌سازیِ SVM با استفاده از SGD

الگوریتمِ نزولِ گرادیانِ تصادفی را برای فرمولاسیونِ اصلیِ SVM ( عبارتِ (۱) ) پیاده سازی کنید. شما باید در هر تکرار، یک نمونه‌ی داده‌ی $n \in [N]$ را بصورت تصادفیِ یکنواخت، انتخاب کنید و یک به‌روزرسانی بر روی پارامترِ $\boldsymbol{w}$ ، بر اساسِ (زیر) گرادیانِ $n$ ـاُمین جمع‌وندِ [۱] عبارتِ (۱) ، انجام دهید. سپس برای $n$ بعدی، تکرار کنید.

- در فایل ژوپیترِ مربوط به این جلسه، تابعِ `calculate_accuracy(y, X, w)` که دقت را برای داده‌های آموزشی/آزمون به ازای هر $\boldsymbol{w}$ محاسبه می‌کند و همچنین تابعِ `calculate_primal_objective(y, X, w, lamda_)` را که مجموع عبارت اولیه (عبارت (۱)) را محاسبه می‌کند، پیاده‌سازی نمایید.
- به‌روزرسانی های SGD را برای فرمولاسیونِ اصلیِ SVM استخراج کرده و تابعِ `calculate_stochastic_gradient()` را تکمیل کنید. این تابع باید نزولِ گرادیانی تصادفیِ تابع هزینه‌ی مجموع را با توجه به $\boldsymbol{w}$ برگرداند. در نهایت برای امتحان کردن، از تابعِ `sgd_for_svm_demo()` که در فایل این جلسه قرار دارد، استفاده کنید.

[۱] summand

## ۲.۶ SVM با استفاده از Coordinate Descent

همانطوز که می‌دانید، یکی دیگر از راه‌های آموزشِ SVM ها، استفاده از مسئله‌ی بهینه‌سازی دوگانه که توسط عبارت:

$$\max_{\alpha \in \mathbb{R}^N} \alpha^\top 1 - \frac{1}{2\lambda} \alpha^\top \boldsymbol{Y}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{Y}\alpha \quad \text{that such} \quad 0 \le \alpha_n \le 1 \forall n \in [N] \tag{۲.۶}$$

تعریف می‌شود، است. که در آن $\mathbf{1}$ یک بردار با اندازه‌ی N است که با اعدادِ یک پر شده است، $\boldsymbol{Y} := \text{diag}(y)$ است و $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ ، حاوی تمامیِ N نمونه‌ی داده‌ها، به همراه سطرهای آنها است. در این روش، ما بهینه‌سازی را بر روی متغیرهای دوگانه‌ی $\boldsymbol{\alpha}$ انجام می‌دهیم و سپس پاسخ را به بردار اولیه‌ی $\boldsymbol{w}$ ، با استفاده از رابطه‌ی $\boldsymbol{w}(\boldsymbol{\alpha}) = \frac{1}{\lambda}\boldsymbol{X}^T\boldsymbol{Y}$ ربط می‌دهیم.

### استفاده از coordinate descent برای SVM

در این تمرین قصد داریم از الگوریتمِ coordinate descent (یا در این مثال خاص بهتر است بگوییم coordinate ascent) برای حل کردن دوگانِ فرمولاسیونِ SVM (طبق عبارتِ (۲) )، بصورت تصادفی یکنواخت، استفاده کنیم. برای این‌کار، در هر تکرار، بصورتِ تصادفیِ یکنواخت، یک هم‌پایه‌ی $n \in [N]$ انتخاب می‌کنیم و صرفا با توجه به هم‌پایه، عبارت (۲) را برای این کار، بهینه‌سازی می‌کنیم. از این رو، یک گام از coordinate ascent شاملِ حلِ مساله‌ی تک بُعدیِ

$$\max_{\gamma \in \mathbb{R}} f\left(\alpha + \gamma e_n\right) \quad \text{that such} \quad 0 \le \alpha_n + \gamma \le 1 \tag{۳.۶}$$

که در آن، $f(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \boldsymbol{Y}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{Y}\boldsymbol{\alpha}$ و $e_n = [0, \dots, 1, \dots]^\top$ (تمامی درایه ها برابر با صفر ، بجز $n$ـاُمین درایه)، برای یک هم‌پایه‌ی داده شده‌ی $n \in [N]$ و بردارِ کنونیِ ما، $\boldsymbol{\alpha} \in [0,1]^N$ ، می‌باشد. ما به $\gamma^*$ مقدارِ $\gamma$ ای را نسبت می‌دهیم که مسئله‌ی ۳ را بیشینه می‌کند و درنتیجه، عبارتِ به‌روزرسانی کردنِ هم‌پایه، برابر: $\boldsymbol{\alpha}^{\text{new}} = \boldsymbol{\alpha} + \gamma^* e_n$ خواهد شد.

- مسئله‌ی ۳ را حل کنید و یک راه‌حل به فرم بسته برای عبارتِ به روزرساننده‌ی $\boldsymbol{\alpha}^{\text{new}} = \boldsymbol{\alpha} + \gamma^* e_n$ ارائه دهید. این عبارت باید فقط حاوی متغیرهای $\boldsymbol{\alpha}$ ، $\lambda$ ، $\boldsymbol{x}_n$ ، $y_n$ و $\boldsymbol{w}(\alpha)$ باشد. ( راهنمایی: توجه داشته باشید که $\gamma \mapsto f\left(\alpha + \gamma e_n\right)$ به فرم چندجمله‌ای است و **محدودیت ها را نیز فراموش نکنید.** ) توجه داشته باشید که این به‌روزرسانی می‌تواند در مرتبه زمانیِ $O(D)$ محاسبه شود.

- یک راهِ کارآمد برای به‌روزرسانیِ $\boldsymbol{w}(\alpha^{\text{new}})$ پیدا کنید. این راه باید از مرتبه زمانیِ $O(D)$ باشد.

- در فایل ژوپیترِ این جلسه، تابعِ `calculate_coordinate_update()` که باید به‌روزرسانیِ هم‌پایه را برای یک هم‌پایه‌ی دلخواه محاسبه کند، و همچنین تابعِ `calculate_dual_objective()` که باید هزینه هدف را برای مساله‌ی دوگانِ (۲) را برگرداند، پیاده سازی کنید.

- در نهایت مدل خود را با استفاده از نزولِ هم‌پایه (در اینجا، صعودِ هم‌پایه) و با استفاده از تابعِ `sgd_for_svm_demo()` که در نمونه‌ی داده شده موجود است، آموزش دهید. در مقایسه با پیاده‌سازیِ SGD، کدامیک سریع‌تر است؟ ( مقادیر هدفِ (۱) برای تکرارهای $\boldsymbol{w}$ که از هر روش بدست آوردید، مقایسه کنید. ) آیا شکاف به سمت صفر میل می‌کند؟