به‌نام خدا
تئوری یادگیری ماشین
دکتر سیدصالحی
جلسه هفتم
دانشکده ریاضی و علوم کامپیوتر
اسفند ماه ۱۴۰۲

۱ شرح مسئله

در بحث رگرسیون خطی، ما به مسائلی مانند پیش‌بینی قیمت یک خانه (که با y نشان داده می‌شود) از مساحت خانه (که با x نشان داده می‌شود) پرداختیم و یک تابع خطی از x را بر روی داده‌های آموزشی فیت کردیم. اگر قیمت y بتواند به صورت دقیق‌تری به عنوان یک تابع غیرخطی از x نمایش داده شود، در این صورت به خانواده‌ای از مدل‌ها نیاز داریم که از مدل‌های خطی بیانگرتر باشند. در ادامه، به معرفی و بحث درباره‌ی ویژگی‌ها و نحوه‌ی ساخت چنین مدل‌هایی می‌پردازیم.

۲ متغیرهای ویژگی (Feature maps)

ما با در نظر گرفتن منطبق کردن توابع مکعبی $y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$ شروع می‌کنیم. واضح است که ما می‌توانیم تابع مکعبی را به عنوان یک تابع خطی بر روی یک مجموعه‌ی متفاوت از متغیرهای ویژگی (که در زیر تعریف شده‌اند) در نظر بگیریم. به طور مشخص، فرض کنید که تابع $\phi : \mathbb{R} \to \mathbb{R}^4$ به صورت زیر تعریف شده است:

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix} \in \mathbb{R}^4.$$

فرض کنید $\theta \in \mathbb{R}^4$ برداری باشد که شامل $\theta = \{\theta_0, \theta_1, \theta_2, \theta_3\}$ است. پس می‌توانیم تابع مکعبی y را به صورت زیر بازنویسی کنیم:

$$\theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0 = \theta^T \phi(x)$$

بنابراین، یک تابع مکعبی از متغیر x می‌تواند به عنوان یک تابع خطی بر روی متغیرهای $\phi(x)$ دیده شود. برای تفکیک این دو دسته از متغیرها، ما مقدار ورودی یک مسئله را **ورودی اصلی** می‌نامیم (در این مورد

x یا همان مساحت خانه) و زمانی که ورودی اصلی به یک مجموعه‌ی جدید از مقادیر $\phi(x)$ نگاشته می‌شود، ما آن مقادیر جدید را **متغیرهای ویژگی** می‌نامیم.

۳ کمینه میانگین مربعات (LMS)

ما از الگوریتم Gradient Descent برای منطبق کردن مدل $\theta^T\phi(x)$ استفاده خواهیم کرد. ابتدا یادآوری کنیم که در مسئله‌ی کمینه مربعات معمولی که ما باید $\theta^T x$ را منطبق می‌کردیم، Batch Gradient Descent دسته‌ای به صورت زیر است:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x^{(i)}$$

فرض کنید $\phi : \mathbb{R}^d \to \mathbb{R}^p$ یک نگاشت ویژگی باشد که ویژگی x در $\mathbb{R}^d$ را به ویژگی‌های $\phi(x)$ در $\mathbb{R}^p$ نگاشت می‌کند. حال هدف ما منطبق کردن تابع $\theta^T\phi(x)$ با θ به عنوان یک بردار در $\mathbb{R}^p$ به جای $\mathbb{R}^d$ است. ما می‌توانیم تمام وقوع‌های $x^{(i)}$ را در الگوریتم بالا با $\phi(x^{(i)})$ جایگزین کنیم تا بروزرسانی جدیدی به دست آوریم:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T\phi(x^{(i)})\right) \phi(x^{(i)})$$

به طور مشابه، قانون Stochastic Gradient Descent به صورت زیر است:

$$\theta := \theta + \alpha \left(y^{(i)} - \theta^T\phi(x^{(i)})\right) \phi(x^{(i)})$$

۴ LMS با ترفند کرنل

بروزرسانی Gradient Descent یا بروزرسانی Stochastic Gradient Descent فوق زمانی که ویژگی‌های $\phi(x)$ چندبعدی باشند، از لحاظ محاسباتی سنگین می‌شود. به عنوان مثال، در نظر بگیرید که توسعه مستقیم نگاشت ویژگی در معادله‌ی اولیه به ورودی چندبعدی x (فرض کنید $x \in \mathbb{R}^d$) و $\phi(x)$ برداری باشد که شامل تمام چندجمله‌ای‌های x با درجه‌ی ≤ 3 است.

بُعد ویژگی‌های $\phi(x)$ در حدود d^3 است. این یک بردار به شدت طولانی برای اهداف محاسباتی است؛ زمانی که $d = 1000$ شود، هر به‌روزرسانی حداقل نیاز به محاسبه و ذخیره‌سازی یک بردار با ابعاد $d^3 = 10^9$ دارد، که $d^2 = 10^6$ بار کندتر از قانون به‌روزرسانی برای به‌روزرسانی‌های معمولی کمینه مربعات است.

در ابتدا ممکن است به نظر برسد که چنین زمان اجرای d^3 در هر به‌روزرسانی و استفاده از حافظه

اجتناب‌ناپذیر است، زیرا بردار θ خود از بُعد $p \approx d^3$ است، و ممکن است لازم باشد هر ورودی از θ را بروزرسانی و ذخیره کنیم. با این حال، ما ترفند کرنل را معرفی خواهیم کرد که با آن نیازی به ذخیره‌سازی θ به صورت صریح نخواهیم داشت، و زمان اجرا به طور قابل توجهی بهبود می‌یابد.

برای سادگی ما فرض می‌کنیم که مقدار اولیه‌ی $\theta = 0$ است و ما بر روی فرمول بروزرسانی تمرکز داریم. متوجه می‌شویم که در هر زمان θ را می‌توان به عنوان ترکیب خطی از بردارهای $\phi(x^{(1)}), \ldots, \phi(x^{(n)})$ نمایش داد. در واقع ما می‌توانیم این را به صورت استقرایی نشان دهیم؛ در ابتدا $\theta = 0 = \sum_{i=1}^n 0 \cdot \phi(x^{(i)})$. فرض کنید در یک نقطه‌ای θ را بتوان برای برخی $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ به صورت:

$$\theta = \sum_{i=1}^n \alpha_i \phi(x^{(i)})$$

نمایش داد. سپس ما ادعا می‌کنیم که در دور بعدی، θ همچنان یک ترکیب خطی از $\phi(x^{(1)}), \ldots, \phi(x^{(n)})$ باقی می‌ماند:

$$\begin{aligned}
\theta := & \theta + \alpha \sum_{i=1}^n \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)}) \\
= & \sum_{i=1}^n \alpha_i \phi(x^{(i)}) + \alpha \sum_{i=1}^n \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)}) \\
= & \sum_{i=1}^n \left(\alpha_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)\right) \phi(x^{(i)})
\end{aligned}$$

شما ممکن است متوجه شوید که استراتژی کلی ما نمایش ضمنی بردار p-بعدی θ با استفاده از مجموعه‌ای از ضرایب $\alpha_1, \ldots, \alpha_n$ است. به این منظور، ما قانون بروزرسانی ضرایب $\alpha_1, \ldots, \alpha_n$ را استخراج می‌کنیم. با استفاده از معادله‌ی بالا، می‌بینیم که α_i جدید بستگی به α_i قدیمی دارد:

$$\alpha_i := \alpha_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)$$

در اینجا ما هنوز هم θ قدیمی را در سمت راست معادله داریم. با جایگزینی θ با $\sum_{j=1}^n \alpha_j \phi(x^{(j)})$ ما داریم:

$$\forall i \in \{1, \ldots, n\} : \alpha_i := \alpha_i + \alpha \left(y^{(i)} - \sum_{j=1}^n \alpha_j \phi(x^{(j)})^T \phi(x^{(i)})\right)$$

ما اغلب $\phi(x^{(j)})^T \phi(x^{(i)})$ را به صورت $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$ بازنویسی می‌کنیم تا تأکید کنیم که این یک

ضرب داخلی از دو بردار ویژگی است. با دیدن α_i ها به عنوان نمایش جدیدی از θ ، ما با موفقیت الگوریتم Batch Gradient Descent را به یک الگوریتم که همواره ارزش α را بروز می‌کند، تبدیل کردیم. ممکن است به نظر برسد که در هر تکرار، هنوز نیاز به محاسبه‌ی ارزش‌های $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$ برای همه جفت‌های i, j داریم که هر کدام ممکن است عملیاتی در حدود $O(p)$ داشته باشند. با این حال، دو ویژگی مهم به نجات ما می‌آیند:

- ما می‌توانیم ضرب‌های داخلی $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$ را برای همه‌ی جفت‌های i, j پیش‌پردازش کنیم تا حین شروع حلقه مقادیر تمام آن‌ها را داشته باشیم.
- برای نگاشت ویژگی ϕ تعریف شده در مسئله‌ی اولیه، محاسبه‌ی $\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle$ می‌تواند به طور مؤثر انجام شود و لزوماً نیازی به محاسبه‌ی صریح $\phi(x^{(i)})$ ندارد. این به این دلیل است که:

$$
\begin{aligned}
\langle \phi(x), \phi(z) \rangle =& 1 + \sum_{i=1}^{d} x_i z_i + \sum_{i,j \in \{1,\ldots,d\}} x_i x_j z_i z_j + \sum_{i,j,k \in \{1,\ldots,d\}} x_i x_j x_k z_i z_j z_k \\
=& 1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3
\end{aligned}
$$

بنابراین، برای محاسبه‌ی $\langle \phi(x), \phi(z) \rangle$ ما می‌توانیم ابتدا $\langle x, z \rangle$ را با مرتبه زمانی $O(d)$ محاسبه کنیم و سپس تعدادی عملیات دیگر برای محاسبه‌ی $1 + \langle x, z \rangle + \langle x, z \rangle^2 + \langle x, z \rangle^3$ انجام دهیم.