AGML, Sommersemester 2020 Vorlesung 12 (5 Seiten)

Overfitting

Thomas Kesselheim **Vorschau** Letzte Aktualisierung: 9. Juni 2020

In den letzten Vorlesungen haben wir vor allem diskutiert, wie wir eine Hypothese finden, die die Trainingsdaten möglichst gut beschreibt. Der Sinn einer solchen Hypothese ist es, Vorhersagen bei neuen Datenpunkten zu machen. Konkret also: Uns sind viele E-Mails gegeben, die jeweils als „Spam“ oder „kein Spam“ markiert sind. Auf dieser Basis wollen wir neu ankommende E-Mails möglichst gut klassifizieren.

1 Beispiel: Klassifikation

In Abbildung 1 sind Punkte in $\mathbb{R}$ mit binären Labels -1 und $+1$ gegeben, dargestellt als blaue und rote Punkte. Es wäre nun möglich, mittels durch Auswahl einiger Intervalle eine Hypothese zu finden, die keinen Trainingsfehler hat. Erstellt wurden die Daten jedoch wie folgt. Zunächst wurde $x_i \sim \text{Uniform}[0,1]$ gezogen wurde. Anschließend wurden die Labels bestimmt über

$$y_i = \begin{cases} +1 & \text{falls } x_i + \nu_i \geq \frac{1}{2} \\ -1 & \text{sonst} \end{cases},$$

wobei $\nu_i \sim \text{Uniform}[-0.3, 0.3]$ ein Rauschen mit Erwartungswert 0 ist. Das Rauschen lässt sich nicht vorhersagen. Entsprechend sollte einfach nur

$$h(x) = \begin{cases} +1 & \text{falls } x \geq \frac{1}{2} \\ -1 & \text{sonst} \end{cases}$$

als Hypothese verwendet werden. Diese hat zwar einen Trainingsfehler, ist aber die bestmögliche Vorhersage für neue Punkte.

Abbildung 1: Datenpunkte mit Rauschen.

2 Beispiel: Regression

Ein ähnliches Problem tritt auch bei *Regression* auf. Hier sind nun die Labels nicht mehr -1 oder $+1$ sondern beliebige reelle Zahlen.

Abbildung 2 zeigt ein Beispiel von acht Paaren von Datenpunkten mit ihren Labels (x_i, y_i), wobei $x_i \in [0,1]$ und $y_i \in \mathbb{R}$. Es wäre nun sehr verführerisch, eine Funktion h zu wählen, die die Werte in allen gegebenen Punkten genau trifft. Beispielsweise ein Polynom von Grad sieben. In diesem Fall ist es gegeben durch

$$h(x) = 5940.33x^7 - 20262.6x^6 + 27659.7x^5 - 19294.7x^4 + 7302.01x^3 - 1476.7x^2 + 148.067x - 5.53035 \ .$$

Dies entspricht dem roten Funktionsgraph in der Abbildung.

In diesem Fall wurden die Daten wie folgt generiert: Zunächst wurde $x_i \sim \text{Uniform}[0,1]$ gezogen. Anschließend wurde das Label für x_i bestimmt als $y_i = x_i + \nu_i$, wobei $\nu_i \sim \text{Normal}(0, 0.0025)$. Das heißt, ν_i ist ein zufälliges Rauschen aus einer Normalverteilung mit Erwartungswert 0 und Varianz 0.0025.

Auch in diesem Fall können wir das Rauschen nicht vorhersagen. Deshalb ist die beste Hypothese h in diesem Fall gegeben durch $h(x) = x$, eingetragen als die blaue Gerade.

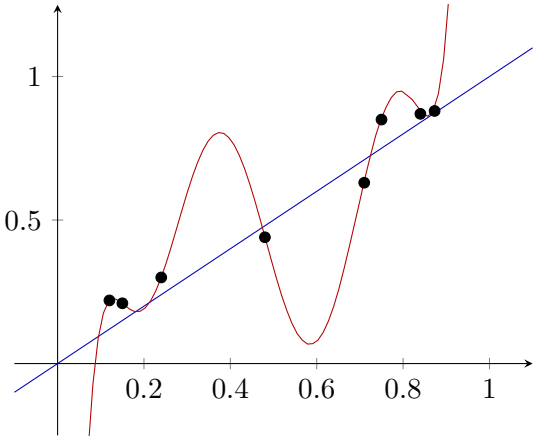

Abbildung 2: Die rote Kurve ist ein Polynom vom Grad sieben, das genau durch die acht gegebenen Punkte geht. Die blaue Gerade minimiert den tatsächlichen Fehler.

3 Problemstellung

Wir nehmen an, dass wir Datenpunkte aus einer Menge X erhalten und Labels für derartige Datenpunkte vorhersagen sollen. Diese Menge möglicher Labels heißt nun Y. Ein Fall ist binäre Klassifikation, also $Y = \{-1, +1\}$. Ein anderer Fall ist Regression mit $Y = \mathbb{R}$.

Wie im agnostischen Fall des PAC-Learning nehmen wir an, dass es eine Wahrscheinlichkeitsverteilung $\mathcal{D}$ über Paare $z = (x, y) \in X \times Y$ gibt, sodass y das korrekte Label ist für x. Uns ist eine Trainingsmenge $S = \{z_1, \ldots, z_m\}$, $z_i = (x_i, y_i) \in X \times Y$ aus m Samples gegeben, die aus $\mathcal{D}$ gezogen ist. Auf Basis von S berechnen wir eine Hypothese $h_S \colon X \to Y$, die ein Label $h_S(x)$ für jeden Punkt x vorhersagt.

Wir haben bereits Begriffe wie den Trainingsfehler und den tatsächlichen Fehler kennengelernt. Diese werden wir nun erweitern.

4 Loss-Funktionen und Fehlerbegriffe

Allgemein schreiben wir $\ell(h, z)$ für den Loss von Hypothese h auf $z = (x, y)$. Im Fall von binärer Klassifizierung ist die einfachste Wahl für ℓ der *0/1 Loss*, definiert durch

$$\ell^{0-1}(h, z) = \begin{cases} 0 & \text{falls } h(x) = y \\ 1 & \text{sonst} \end{cases} .$$

Wir haben bereits den *tatsächlichen Fehler* kennengelernt. Diesen verallgemeinern wir zum erwarteten Loss einer Hypothese h auf einem Datenpunkt-/Label-Paar gezogen aus $\mathcal{D}$, das heißt

$$L_{\mathcal{D}}(h) = \mathbf{E}_{z \sim \mathcal{D}} \left[\ell(h, z) \right] .$$

Auch können wir den Trainingsfehler verallgemeinern. Dieser ist für eine Menge S von m Datenpunkt-/Label-Paaren definiert als

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i) .$$

Anstatt des 0/1 Loss können wir auch andere Funktionen einsetzen. Wir haben bereits den *Hinge Loss* im Kontext von Support Vector Machines kennengelernt. Diesen hatten wir nur für lineare Klassifikatoren $h_{\mathbf{w}}$ definiert als

$$\ell^{\text{hinge}}(h_{\mathbf{w}}, \mathbf{z}) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\} \ .$$

Der Vorteil des Hinge Loss ist, dass die Funktion stetig und konvex ist. Es gilt $\ell^{0-1}(h_{\mathbf{w}}, \mathbf{z}) \leq \ell^{\text{hinge}}(h_{\mathbf{w}}, \mathbf{z})$ für alle $\mathbf{w}$ und $\mathbf{z}$. Das heißt, Fehler werden im Vergleich zum 0/1 Loss im Normalfall überschätzt. Der Soft-SVM-Ansatz lässt sich auch so interpretieren, dass die Funktion ℓ^{0-1} durch eine stetige, konvexe Funktion ersetzt wird, die leichtere Optimierung ermöglicht.

Bei Regression versucht man im Normalfall, Fehlerquadrate zu minimieren. Die Loss-Funktion ist in diesem Fall

$$\ell^{\text{squared}}(h, z) = (h(x) - y)^2 \ .$$

5 Verallgemeinerungsfehler und Overfitting

Unser Ziel ist es, eine Hypothese zu finden, deren tatsächlicher Fehler $L_{\mathcal{D}}(h)$ möglichst klein ist. Dafür betrachten wir einen Lernalgorithmus, der eine Hypothese h_S auf Basis einer Trainingsmenge S berechnet. Diese Menge S besteht aus m Paaren $z_i = (x_i, y_i)$, die jeweils aus der Verteilung $\mathcal{D}$ gezogen werden.

Den tatsächlichen Fehler $L_{\mathcal{D}}(h_S)$ der berechneten Hypothese können wir uns nun wie folgt vorstellen: Einerseits ist h_S womöglich auf S schon nicht perfekt. Dies beschreibt der Trainingsfehler $L_S(h_S)$. Andererseits repräsentiert das Sample S die Verteilung $\mathcal{D}$ möglicherweise nicht perfekt. Deshalb bezeichnen wir nun

$$L_{\mathcal{D}}(h_S) - L_S(h_S)$$

als den *Verallgemeinerungsfehler*.

Als *Overfitting* versteht man nun das Phänomen, dass bei gewissen Lernalgorithmen der Trainingsfehler klein wird, der Verallgemeinerungsfehler aber groß. Insbesondere problematisch ist es, wenn größere Trainingsmengen über einen größeren Verallgemeinerungsfehler zu einem größeren tatsächlichen Fehler führen.

6 Stabilität von Lernalgorithmen

Wir wollen nun den *erwarteten* Verallgemeinerungsfehler eines Lernalgorithmus besser verstehen. Das heißt, uns interessiert

$$\mathbf{E}\left[L_{\mathcal{D}}(h_S) - L_S(h_S)\right] \ , \tag{1}$$

wobei der Erwartungswert über die Menge S geht. Dies wollen wir umschreiben.

Sei nun I eine Zufallsvariable, die unabhängig gleichverteilt aus $\{1, \ldots, m\}$ gezogen wir. Der erwartete Trainingsfehler ist nun

$$\mathbf{E}\left[L_S(h_S)\right] = \mathbf{E}\left[\frac{1}{m}\sum_{i=1}^{m} \ell(h_S, z_i)\right] = \mathbf{E}\left[\ell(h_S, z_I)\right] \ .$$

Der erwartete tatsächliche Fehler ist der erwartete Loss auf einem frisch gezogenen Datenpunkt-/Label-Paar z', das wiederum aus $\mathcal{D}$ gezogen wird

$$\mathbf{E}\left[L_{\mathcal{D}}(h_S)\right] = \mathbf{E}\left[\ell(h_S, z')\right] \ .$$

Auch dies können wir anders schreiben. Gegeben Samples $z_1, \dots, z_m$ und z', sei S^i die Menge $z_1, \dots, z_{i-1}, z', z_{i+1}, \dots, z_m$. Das heißt, wir ersetzen z_i durch z'. Da z_i und z' beide aus $\mathcal{D}$ gezogen werden, sind sie identisch verteilt und wir können ihre Rollen vertauschen. Deshalb gilt für alle i

$$\mathbf{E}\left[\ell(h_S, z')\right] = \mathbf{E}\left[\ell(h_{S^i}, z_i)\right] \ .$$

Weil diese Gleichung für alle i gilt, können wir auch die Zufallsvariable I von oben wieder verwenden. Damit gilt insgesamt

$$\mathbf{E}\left[L_{\mathcal{D}}(h_S)\right] = \mathbf{E}\left[\ell(h_{S^I}, z_I)\right] \ .$$

Und so kann der erwartete Verallgemeinerungsfehler aus (1) mittels Linearität des Erwartungswerts auch umgeschrieben werden zu

$$\begin{aligned}\mathbf{E}\left[L_{\mathcal{D}}(h_S) - L_S(h_S)\right] &= \mathbf{E}\left[L_{\mathcal{D}}(h_S)\right] - \mathbf{E}\left[L_S(h_S)\right] = \mathbf{E}\left[\ell(h_{S^I}, z_I)\right] - \mathbf{E}\left[\ell(h_S, z_I)\right] \\ &= \mathbf{E}\left[\ell(h_{S^I}, z_I) - \ell(h_S, z_I)\right] \ .\end{aligned}$$

Das heißt, der erwartete Verallgemeinerungsfehler kann nur groß sein, wenn es irgendwelche S^i und S gibt, die zu sehr unterschiedlichen Hypothesen führen. Dabei sollte man bedenken, dass S^i und S sich nur in einem einzigen Punkt unterscheiden. Ein Algorithmus, bei dem dies niemals geschieht, nennen wir stabil.

Definition 12.1. *Sei $\delta\colon \mathbb{N} \to \mathbb{R}$. Ein Lernalgorithmus ist* universell δ-austauschstabil*, wenn für alle $m \in \mathbb{N}$, alle Mengen S von m Datenpunkt-/Label-Paaren, alle $i \in \{1, \dots, m\}$ und alle weiteren Datenpunkt-/Label-Paare z' gilt*

$$\ell(h_{S^i}, z_i) - \ell(h_S, z_i) \leq \delta(m) \ .$$

Wir nennen ihn universell austauschstabil*, falls er universell δ-austauschstabil ist für eine Funktion δ mit $\delta(m) \to 0$ für $m \to \infty$.*

Wir sehen nun, dass wenn unser Lernalgorithmus universell δ-austauschstabil ist, dass

$$\mathbf{E}\left[L_{\mathcal{D}}(h_S) - L_S(h_S)\right] = \mathbf{E}\left[\ell(h_{S^I}, z_I) - \ell(h_S, z_I)\right] \leq \delta(m) \ .$$

Insbesondere, wenn $\delta(m) \to 0$ für $m \to \infty$, dann gibt es kein Overfitting.

Der große Vorteil davon, über Stabilität zu sprechen ist, dass es sich ausschließlich um eine Eigenschaft des Lernalgorithmus handelt. Wir müssen also keine Aussage über Wahrscheinlichkeitsverteilungen oder statistische Eigenschaften diskutieren, sondern lediglich Algorithmen entwickeln, deren Ausgabe sich nicht entscheidend ändert, wenn ein Datenpunkt ausgetauscht wird.

7 Beispiel

In unserem Einstiegsbeispiel haben wir anschaulich gesehen, dass es merkwürdige Effekte haben kann, Regression mittels einer Interpolation durch Polynome zu machen. Schon mit einem sehr einfach Beispiel können wir sehen, dass der Algorithmus, der den Trainingsfehler minimiert, nicht universell austauschstabil ist.

Der Merkmalsraum ist $X = \mathbb{R}$. Für unser Beispiel brauchen wir nur Polynome vom Grad 1, also Geraden bzw. Hypothesen der Form $h_{a,b}(x) = a \cdot x + b$ für $a, b \in \mathbb{R}$.

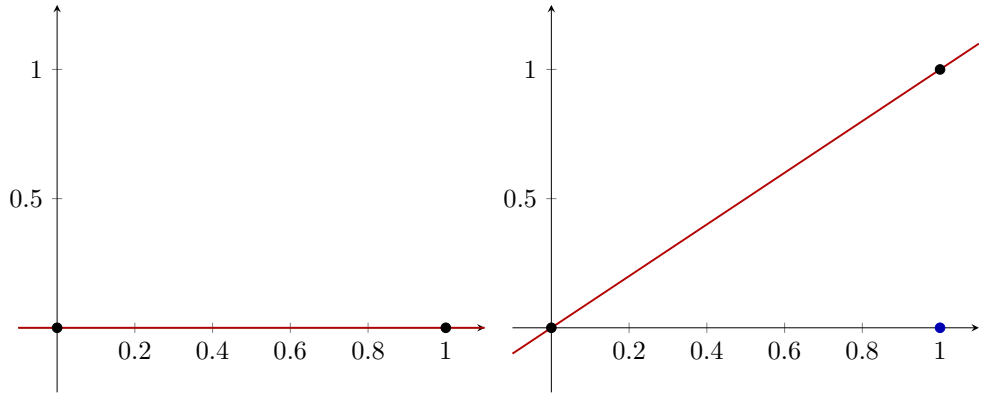

Abbildung 3: Die rote Gerade führt jeweils durch die schwarzen Punkte. Im rechten Bild ist ein Punkt ersetzt. Auf dem bisherigen Punkt (in blau) gibt es nun einen großen Fehler. Wie viele Punkte sich in $(0, 0)$ befinden, ist irrelevant.

Betrachten wir zunächst den Lernalgorithmus, der den Trainingsfehler minimiert. Das heißt $h_S = h_{a,b}$, wobei a und b so gewählt sind, dass $L_S^{\text{squared}}(h_{a,b}) = \frac{1}{m}\sum_{i=1}^{m}(ax_i + b - y_i)^2$ minimal ist. Dieser Algorithmus ist nicht universell austauschstabil. Seien dafür $(x_1, y_1) = (1, 0)$, $(x_2, y_2) = \ldots = (x_m, y_m) = (0, 0)$. Die Hypothese h, die den Trainingsfehler minimiert ist $h_{0,0}$. Betrachten wir $i = 1$, $(x', y') = (1, 1)$. Auf S^i wird der Trainingsfehler von $h_{1,0}$ minimiert (siehe Abbildung 3). Es gilt somit $\ell(h_{S^i}, z_i) - \ell(h_S, z_i) = 1$ und somit $\delta(m) \geq 1$. Auch bei Polynomen von höherem Grad tritt derselbe Effekt auf, denn weiterhin minimieren die Geraden den Trainingsfehler.

In der nächsten Vorlesung werden wir zeigen, dass dies nicht auftritt, wenn wir *Regularisierung* verwenden. In diesem konkreten Fall würden wir statt $L_S^{\text{squared}}(h_{a,b})$ nun $\lambda(a^2 + b^2) + L_S^{\text{squared}}(h_{a,b})$ minimieren, wobei λ ein Parameter ist. Der anschauliche Grund ist, dass der Einfluss eines Punktes (x_i, y_i) auf den Loss, also $\frac{1}{m}(ax_i + b - y_i)^2$, klein wird im Vergleich zu $\lambda(a^2 + b^2)$, sobald m groß wird. Wir erkaufen uns dies über einen höheren Trainingsfehler. Deshalb sollte λ nicht zu groß gewählt sein.

Referenzen

- Blog-Post von Moritz Hardt: `https://www.offconvex.org/2016/03/14/stability/`
- Understanding Machine Learning, Kapitel 13.2
- Foundations of Machine Learning, Kapitel 14.1–14.2 (etwas andere Aussage)