

Lekce 2

Describe maximum likelihood estimation, as minimizing NLL, cross-entropy and KL divergence. [10]

Self information

- $I(x) = -\log P(x)$
- Jak moc jsme překvapeni, když dostaneme $x \sim P$
- Pro nezávislé jevy se počítá, pro jevy s pností 1 je rovna 0

Entropie

- $H(P) = \mathbb{E}_{x \sim P}[I(x)] = -\mathbb{E}_{x \sim P}[\log P(x)]$
- Množství překvapení v distribuci P

Cross-entropy

- $H(P, Q) = -\mathbb{E}_{x \sim P}[\log Q] = \mathbb{E}_{x \sim P}[I_Q(x)]$
- V podmíněném případě pak $H(Y|X) = H(X, Y) - H(X)$, tedy jak moc jsem překvapen když se dozvím obojí oproti tomu, když se dozvím jen X
- Měří, jak moc budu překvapený, když budu tahat x z distribuce P , ale své překvapení budu měřit na základě distribuce Q

Kullback-Leibler Divergence

- $D_{KL}(P||Q) = H(P, Q) - H(P) = -\mathbb{E}_{x \sim P}[\log P - \log Q]$
- Nemí symetrická
- V zásadě říká, jak “špatná” je moje distribuce Q . Konkrétně zjišťuje o jak moc více budu překvapen, když tahám x z P , ale překvapení měřím skrze Q
 - Čím jsou si Q a P podobnější, tím menší tohle “překvapení navíc” bude

MLE

- Samo o sobě je to takové hledání parametrů modelu, aby

$$\theta_{\text{ML}} = \arg \max_{\theta} p_{\text{model}}(\mathcal{X}; \theta)$$

Což se dá dále upravovat, až se dostaneme NLL, binární crossentropii, a KL divergenci

Define mean squared error and show how it can be derived using MLE. [5]

$$MSE = \mathbb{E}[(\hat{y} - y)^2] = \frac{1}{m} \sum_{i=1}^m ((f(x_i; \theta) - y_i)^2)$$

Pokud se nám při regresi nechce odhadovat celá distribuce, můžeme si usnadnit práci, predikovat pouze její střední hodnotu a říct, že ta distribuce je normální s nějakým rozptylem (a s tou naší střední hodnotou).

$$\begin{aligned}
\theta_{\text{ML}} &= \arg \max_{\theta} p_{\text{model}}(\mathbb{Y}|\mathbb{X}; \theta) \\
&= \arg \min_{\theta} \sum_{i=1}^m -\log p_{\text{model}}(y^{(i)}|x^{(i)}; \theta) \quad \leftarrow \text{NLL} \\
&= \arg \min_{\theta} \mathbb{E}_{\mathbf{x}, y \sim \hat{p}_{\text{data}}} [-\log p_{\text{model}}(y|\mathbf{x}; \theta)] \\
&= \arg \min_{\theta} H(\hat{p}_{\text{data}}, p_{\text{model}}(y|\mathbf{x}; \theta)) \quad \leftarrow \text{binary crossentropy loss} \\
&= \arg \min_{\theta} D_{\text{KL}}(\hat{p}_{\text{data}} \| p_{\text{model}}(y|\mathbf{x}; \theta)) + H(\hat{p}_{\text{data}}) \quad \leftarrow \text{KL-divergence}
\end{aligned}$$

Figure 1: IMG_C7542DD4DE41-1

Dává to smysl, protože normální rozdělení má mezi rozděleními se stejnou střední hodnotou a rozptylem maximální entropii, tedy nejméně navíc vnesené informace.

MLE potom vyjde

$$\begin{aligned}
\arg \max_{\theta} p(y | x; \theta) &= \arg \min_{\theta} \sum_{i=1}^m -\log p(y^{(i)} | x^{(i)}; \theta) \\
&= \arg \min_{\theta} -\sum_{i=1}^m \log \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \hat{y}(x^{(i)}; \theta))^2}{2\sigma^2}\right) \\
&= \arg \min_{\theta} -m \log (2\pi\sigma^2)^{-1/2} - \sum_{i=1}^m -\frac{(y^{(i)} - \hat{y}(x^{(i)}; \theta))^2}{2\sigma^2} \\
&= \arg \min_{\theta} \sum_{i=1}^m \frac{(y^{(i)} - \hat{y}(x^{(i)}; \theta))^2}{2\sigma^2} = \arg \min_{\theta} \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \hat{y}(x^{(i)}; \theta))^2
\end{aligned}$$

To $1/m$ jsme si nakonec přimysleli, protože můžeme. MSE tedy dává smysl jako loss funkce, ale **pouze pokud má náš estimátor pevný rozptyl (tj. vlastně chybu) σ^2 .**

Describe gradient descent and compare it to stochastic (i.e., online) gradient descent and minibatch stochastic gradient descent. [5]

Pokud máme nějaký loss L a nějaká trénovací data, chceme při tréninku minimalizovat

$$J(\theta) = \mathbb{E}_{(x,y) \sim \hat{p}_{\text{data}}} L(f(x; \theta), y)$$

Což můžeme udělat v krocích pomocí tzv. **gradient descent** s learning rate α

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} J(\theta)$$

Druhy

- V běžném GD počítáme J a jeho gradient ze všech trénovacích dat

- V **online (stochastic) gradient descent** nasamplujeme pouze jedno dato
- V **minibatch SGD** vybereme m samplů, ze kterých poté odhadujeme střední hodnotu J

Formulate conditions on the sequence of learning rates used in SGD to converge to optimum almost surely. [5]

SGD skoro jistě konverguje k optimu, pokud je naše loss spojitá a konvexní a zároveň pro learning raty platí

$$\forall i : \alpha_i > 0, \quad \sum_i \alpha_i = \infty, \quad \sum_i \alpha_i^2 < \infty$$

Tedy můžeme složením krůčků dojít kamkoli, ale zároveň musí platit $\alpha \rightarrow 0$. Konkrétně to *na druhou* se tam vyskytuje za MSE, říká v podstatě že “nabraná chyba bude konečná”.

Write down the backpropagation algorithm. [5]

Chceme spočítat derivaci posledního vrcholu (u_n) vzhledem ke všem předešlým vrcholům. Tím získáme derivaci loss vůči parametrům, což je to, co potřebujeme do SGD.

1. Spustíme forward propagation, kterým spočteme hodnoty všech vrcholů
2. Nastavíme $g_n = 1$
3. Od konce počítáme g_i jako $\sum_{j:i \in P(u^{(j)})} g^{(j)} \frac{\partial u^{(j)}}{\partial u^{(i)}}$, využití chain rule

Write down the mini-batch SGD algorithm with momentum. Then, formulate SGD with Nesterov momentum and show the difference between them. [5]

$$\begin{aligned} g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)}) \\ v &\leftarrow \beta v - \alpha g \\ \theta &\leftarrow \theta + v \end{aligned}$$

Navíc oproti SGD tam je to v , které zajišťuje roli “kudy jsme šli minule”. V Nestor momentum je to pak jen trochu pozměněno, *momentum krok* se dělá *před* výpočtem gradientu, tak, aby ten samotný gradient byl přesnější.

$$\begin{aligned} \theta &\leftarrow \theta + \beta v \\ g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)}) \\ v &\leftarrow \beta v - \alpha g \\ \theta &\leftarrow \theta - \alpha g \end{aligned}$$

Write down the AdaGrad algorithm and show that it tends to internally decay learning rate by a factor of $1/t$ in step t . Then write down the RMSProp algorithm and explain how it solves the problem with the involuntary learning rate decay. [10]

$$\begin{aligned}
g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)}) \\
r &\leftarrow r + g^2 \\
\theta &\leftarrow \theta - \frac{\alpha}{\sqrt{r + \varepsilon}} g
\end{aligned}$$

Velikosti gradientu normalizujeme, aby se parametry s různými rozptyly měnily zhruba stejně. To, co máme uloženo v r , si můžeme představit zhruba jako σ^2 jednotlivých složek, takže vydělení learning ratu $\sqrt{r + \varepsilon}$ provede normalizaci.

Pokud zůstávají gradienty dlouho stejné, tj $g \approx g_0$, tak po t krocích algoritmu je $r \approx t \cdot g_0^2$, a proto

$$\frac{\alpha}{\sqrt{r + \varepsilon}} \approx \frac{\alpha/\sqrt{t}}{\sqrt{g_0^2 + \varepsilon/t}},$$

jinými slovy, jako kdybychom learning rate škálovali $1/\sqrt{t}$, což zpravidla nechceme, protože to může být moc rychlé.

RMSPProp funguje podobně, ale r počítáme tak, aby zhruba odpovídalo střední hodnotě poslechných g^2 — počítáme exponenciální průměr posledních několika hodnot.

$$\begin{aligned}
g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)}) \\
r &\leftarrow \beta r + (1 - \beta) g^2 \\
\theta &\leftarrow \theta - \frac{\alpha}{\sqrt{r + \varepsilon}} g
\end{aligned}$$

Jak tento exponenciální průměr funguje je ukázáno na následujícím obrázku.

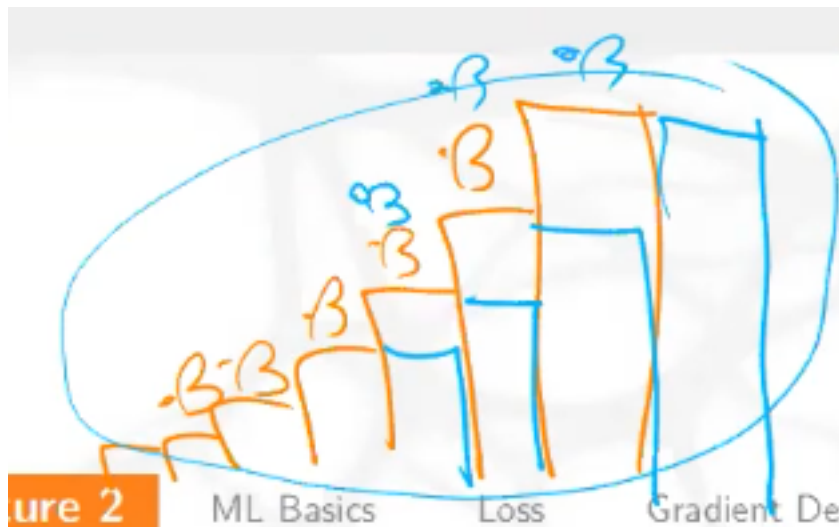


Figure 2: exp_prumer

Write down the Adam algorithm. Then show why the bias-correction terms $(1 - \beta^t)$ make the estimation of the first and second moment unbiased. [10]

Adam je spojením momentum a RMSProp.

$$\begin{aligned} g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)}) \\ t &\leftarrow t + 1 \\ s &\leftarrow \beta_1 s + (1 - \beta_1) g \text{ (biased first moment estimate)} \\ r &\leftarrow \beta_2 r + (1 - \beta_2) g^2 \text{ (biased second moment estimate)} \\ \hat{s} &\leftarrow s / (1 - \beta_1^t), \hat{r} \leftarrow r / (1 - \beta_2^t) \\ \theta &\leftarrow \theta - \frac{\alpha}{\sqrt{\hat{r} + \varepsilon}} \hat{s} \end{aligned}$$

První moment odpovídá momentum, druhý používáme kvůli normalizaci LR, stříškové verze složí jako korekce biasů. Po t krocích totiž r vypadá jako

$$r_t = (1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} g_i^2$$

Tedy jako bych dělal vážený průměr nějakých prvků s celkovou vahou

$$(1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} = (1 - \beta_2) \frac{1 - \beta_2^t}{1 - \beta_2} = 1 - \beta_2^t.$$

Jinými slovy,

$$\mathbb{E}[r_t] \approx \mathbb{E}[g^2] \cdot (1 - \beta_2^t)$$

A biasu se tedy zbavím vydělením.