

Úvod do hlubokého učení

Dominik Farhan (dominik.farhan@gmail.com)

10. března 2025

Proč se učit o hlubokých neuronových sítích?

Nejsilnější ML model, pokud máme dostatek dat.

Proč se učit o hlubokých neuronových sítích?

Nejsilnější ML model, pokud máme dostatek dat.

Hluboké neuronové sítě v kostce

- Modelují libovolnou rozumnou funkci.
- Jsou masivně paralelizovatelné.
- Jsou mimořádně silné pokud pracujeme s obrázky, videi, zvukem, či textem.

Co nás dnes čeká?

1. Motivace
2. Opakování z minula
3. Problém linearity
4. Neuron
5. Neuronová síť
6. Trénování
7. Praktické tipy a použití

Klasifikace vs regrese

- klasifikace: predikujeme typ/třídu
- regrese: predikujeme spojitou hodnotu

Lineární regrese

Pro $\mathbf{x} \in \mathbb{R}^d$ je lineární regrese

$$y(\mathbf{x}; \mathbf{w}, b) = b + \sum_{i=1}^d w_i x_i.$$

Lineární regrese

Pro $\mathbf{x} \in \mathbb{R}^d$ je lineární regrese

$$y(\mathbf{x}; \mathbf{w}, b) = b + \sum_{i=1}^d w_i x_i.$$

Chybová funkce

$$MSE(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^n (y(\mathbf{x}_i; \mathbf{w}, b) - y_i)^2$$

Kontrola znalostí – LR

Obchod se snaží odhadnout denní *zisk* na základě dvou informací

- počtu návštěvníků kamenné prodejny (x_1) a
- počtu prokliků na eshop (x_2).

Natrénovaný model

$$\hat{y} = 50x_1 + 20x_2 - 300$$

Kontrola znalostí – LR

Obchod se snaží odhadnout denní *zisk* na základě dvou informací

- počtu návštěvníků kamenné prodejny (x_1) a
- počtu prokliků na eshop (x_2).

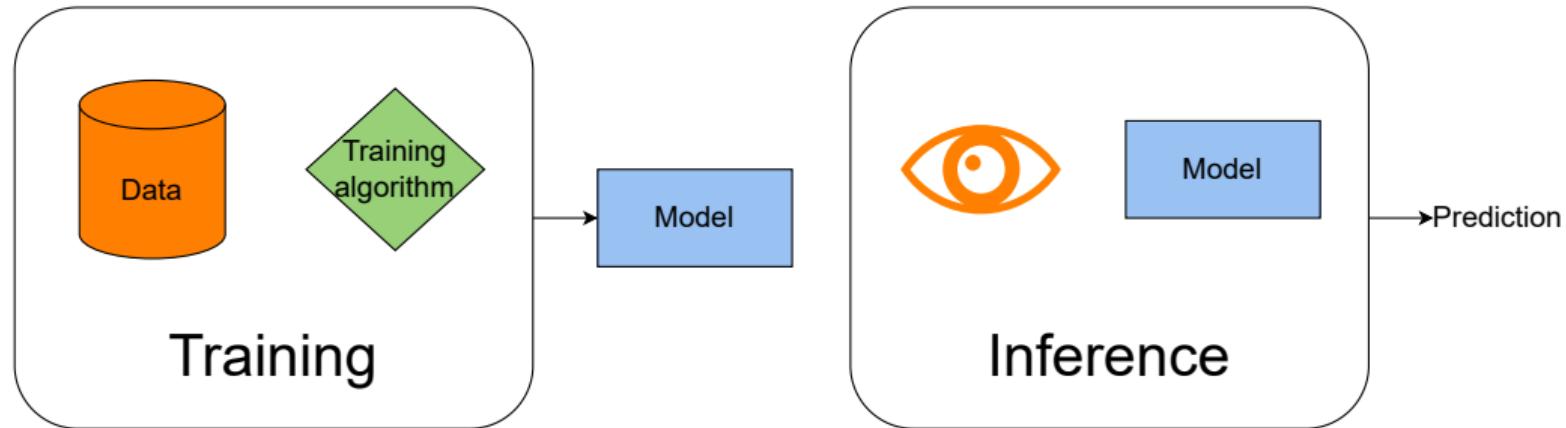
Natrénovaný model

$$\hat{y} = 50x_1 + 20x_2 - 300$$

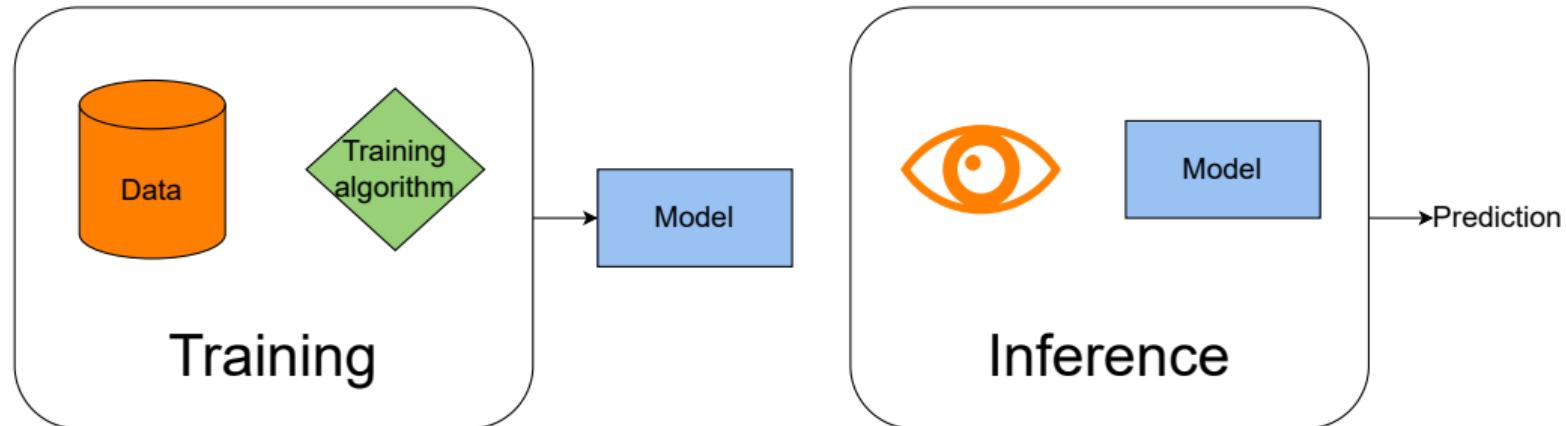
Otázky

1. Který typ potenciálního zákazníka přináší vyšší zisk?
2. Pokud nikdo do prodejny ani eshopu nepřijde, prodělává obchod?

Dvě fáze ML



Dvě fáze ML



Trénovací vs testovací

- Na trénovacích datech učíme model.
- Na testovacích chceme nejvyšší kvalitu.

Kontrola znalostí – gradient

Gradient funkce

Určete gradient funkce

$$L(w_1, w_2) = w_1^2 + w_2^3 - w_1 w_2$$

v bodě $(2, 1)$.

Gradient descent (gradientní sestup)

Gradient Descent

- 1: **Input:** Learning rate α , initial parameters \mathbf{w}_0 , loss function $L(\mathbf{w})$, dataset \mathbf{X}
 - 2: Initialize $\mathbf{w} \leftarrow \mathbf{w}_0$
 - 3: **while** not converged **do**
 - 4: Compute gradient: $\nabla L(\mathbf{w}; \mathbf{X})$
 - 5: Update parameters: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla L(\mathbf{w})$
 - 6: **end while**
 - 7: **Output:** Optimized parameters \mathbf{w}
-

Minibatch Stochastic Gradient Descent

Minibatch Stochastic Gradient Descent

- 1: **Input:** Learning rate α , initial parameters \mathbf{w}_0 , loss function $L(\mathbf{w})$, batch size B , dataset \mathbf{X}
 - 2: Initialize $\mathbf{w} \leftarrow \mathbf{w}_0$
 - 3: **while** not converged **do**
 - 4: Shuffle dataset \mathbf{X}
 - 5: **for** each minibatch $\mathbf{M} \subseteq \mathbf{X}$ of size B **do**
 - 6: Compute gradient: $\mathbf{g} \leftarrow \nabla L(\mathbf{w}; \mathbf{M})$ ▷ Gradient over minibatch
 - 7: Update parameters: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{g}$
 - 8: **end for**
 - 9: **end while**
 - 10: **Output:** Optimized parameters \mathbf{w}
-

Obsah

1. Motivace
2. Opakování z minula
- 3. Problém linearity**
4. Neuron
5. Neuronová síť
6. Trénování
7. Praktické tipy a použití

Je svět lineární?

Přidání features

x_1	y
1	3
5	51
3	19
9	163
11	243
10	201

x_1	x_2	y
1	1	3
5	25	51
3	9	19
9	81	163
11	121	243
10	100	201

Přidání features

x_1	y
1	3
5	51
3	19
9	163
11	243
10	201

x_1	x_2	y
1	1	3
5	25	51
3	9	19
9	81	163
11	121	243
10	100	201

$$0x_1 + 2x_2 + 1$$

Obsah

1. Motivace
2. Opakování z minula
3. Problém linearity
- 4. Neuron**
5. Neuronová síť
6. Trénování
7. Praktické tipy a použití

Neuron

Neuron je jen fancy název pro

$$a \left(b + \sum_{i=1}^d w_i x_i \right),$$

kde a je **vhodně** zvolená funkce.

Volba aktivační funkce a

Který model je silnější? (a je identita)

- a** $y_1(x; w, b) = wx + b$
- b** $y_2(x; w_1, w_2, b_1, b_2) = w_2(w_1x + b_1) + b_2$

Volba aktivační funkce a

Který model je silnější? (a je identita)

- a** $y_1(x; w, b) = wx + b$
- b** $y_2(x; w_1, w_2, b_1, b_2) = w_2(w_1x + b_1) + b_2$

Nelinearity

a nesmí být identita. a nesmí být ani lineární funkce!

Nejběžnější volba a je ReLU

rectified linear unit: $\text{ReLU}(x) = \max(0, x)$

Nejběžnější volba a je ReLU

rectified linear unit: $\text{ReLU}(x) = \max(0, x)$

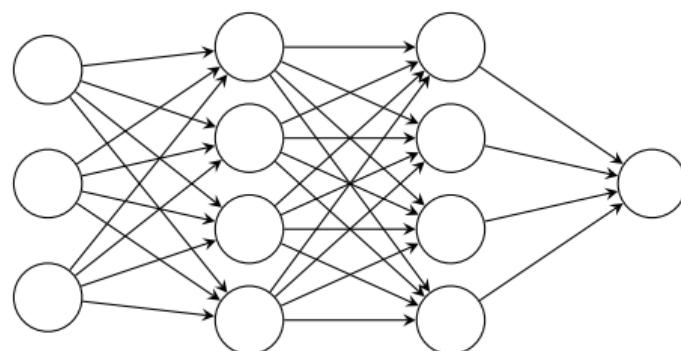
Kreslení

- Jak vypadá graf?
- Jak vypadá derivace?

Obsah

1. Motivace
2. Opakování z minula
3. Problém linearity
4. Neuron
- 5. Neuronová síť**
6. Trénování
7. Praktické tipy a použití

Skládáme neurony do sítí



Příklad výpočtu

Proč je vrstevnatá síť dobrý nápad?

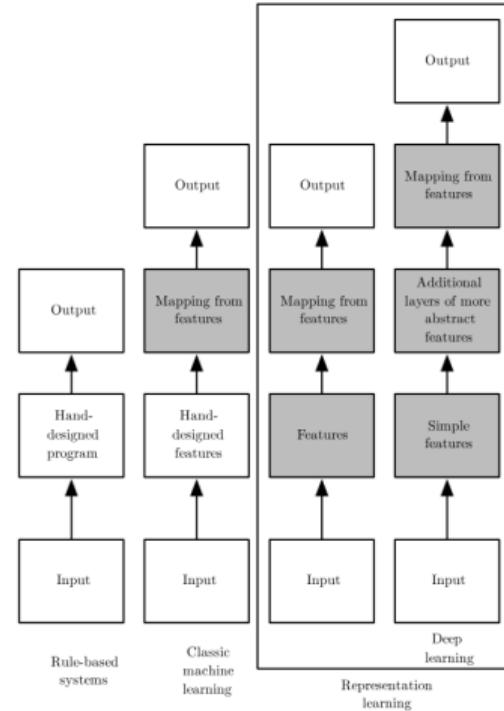


Obrázek: Pierre-Selim, CC BY-SA 2.0, <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Aproximace

Pro každou rozumnou funkci existuje neuronová síť, která ji approximuje libovolně přesně.

Tradiční strojové učení vs hluboké učení



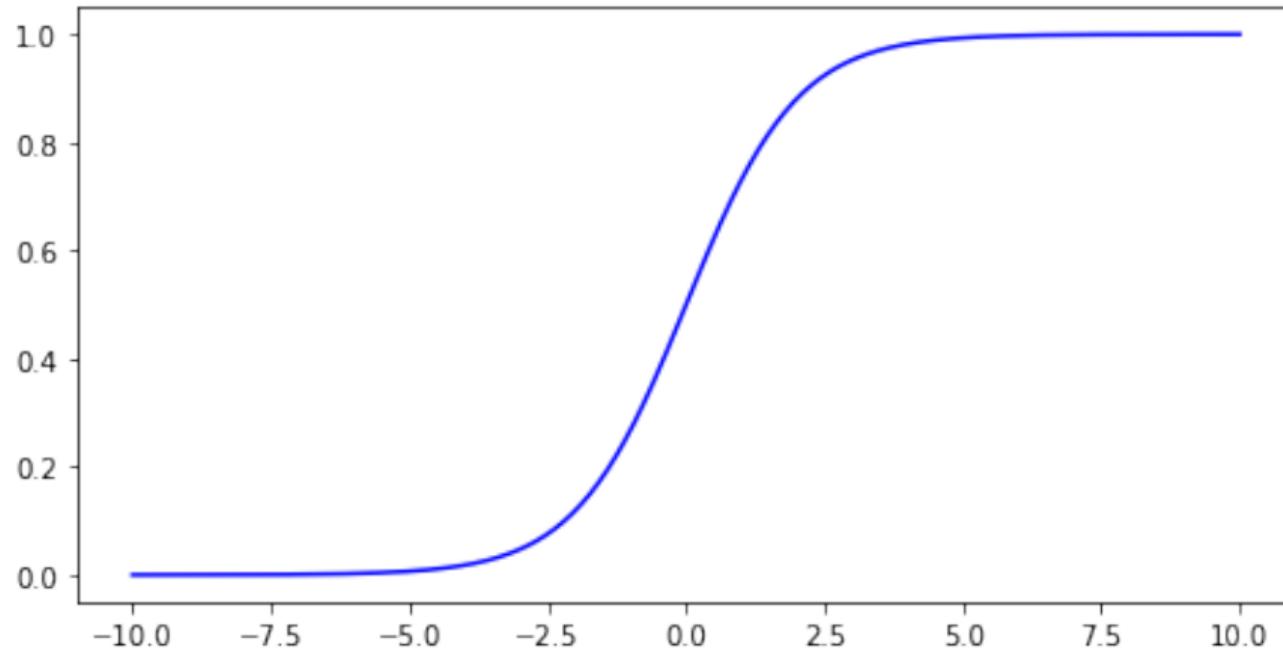
Obrázek: Aaron Courville, Ian Goodfellow, and Yoshua Bengio; "Deep Learning" book,
<https://www.deeplearningbook.org>

Poslední vrstva

Typicky jiná aktivace než ReLU. Záleží na problému.

- nic
- $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$
- $\text{softmax}_k(\mathbf{x}) = \frac{e^{x_k}}{\sum_{i=1}^d e^{x_i}}$

Funkce sigmoid



Obrázek: Funkce sigmoid

Obsah

1. Motivace
2. Opakování z minula
3. Problém linearity
4. Neuron
5. Neuronová síť
6. Trénování
7. Praktické tipy a použití

Připomenutí minibatch SGD

Minibatch Stochastic Gradient Descent

- 1: **Input:** Learning rate α , initial parameters \mathbf{w}_0 , loss function $L(\mathbf{w})$, batch size B , dataset \mathbf{X}
 - 2: Initialize $\mathbf{w} \leftarrow \mathbf{w}_0$
 - 3: **while** not converged **do**
 - 4: Shuffle dataset \mathbf{X}
 - 5: **for** each minibatch $\mathbf{M} \subseteq \mathbf{X}$ of size B **do**
 - 6: Compute gradient: $\mathbf{g} \leftarrow \nabla L(\mathbf{w}; \mathbf{M})$ ▷ Gradient over minibatch
 - 7: Update parameters: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{g}$
 - 8: **end for**
 - 9: **end while**
 - 10: **Output:** Optimized parameters \mathbf{w}
-

Jak počítat gradient u neuronové sítě?

Řetízkové pravidlo

Derivaci složené funkce:

$$L(w) = f_2(f_1(w))$$

vypočteme jako

$$\frac{dL}{dw} = \frac{df_1}{df_2} \frac{df_2}{dw}$$

Řetízkové pravidlo

Derivaci složené funkce:

$$L(w) = f_2(f_1(w))$$

vypočteme jako

$$\frac{dL}{dw} = \frac{df_1}{df_2} \frac{df_2}{dw}$$

$$\frac{df_n}{dw} = \frac{df_n}{df_{n-1}} \cdot \frac{df_{n-1}}{df_{n-2}} \cdot \frac{df_{n-2}}{df_{n-3}} \cdot \dots \cdot \frac{df_3}{df_2} \cdot \frac{df_2}{df_1} \cdot \frac{df_1}{dw}$$

Backpropagation

$$\frac{df_n}{dw} = \frac{df_n}{df_{n-1}} \cdot \left(\frac{df_{n-1}}{df_{n-2}} \cdot \left(\frac{df_{n-2}}{df_{n-3}} \cdot \dots \cdot \left(\frac{df_3}{df_2} \cdot \left(\frac{df_2}{df_1} \cdot \frac{df_1}{dw} \right) \right) \dots \right) \right)$$

$$\frac{df_n}{dw} = \left(\left(\dots \left(\left(\frac{df_n}{df_{n-1}} \cdot \frac{df_{n-1}}{df_{n-2}} \right) \cdot \frac{df_{n-2}}{df_{n-3}} \right) \dots \cdot \frac{df_3}{df_2} \cdot \frac{df_2}{df_1} \right) \cdot \frac{df_1}{dw} \right)$$

Backpropagation

$$\frac{df_n}{dw} = \frac{df_n}{df_{n-1}} \cdot \left(\frac{df_{n-1}}{df_{n-2}} \cdot \left(\frac{df_{n-2}}{df_{n-3}} \cdot \dots \cdot \left(\frac{df_3}{df_2} \cdot \left(\frac{df_2}{df_1} \cdot \frac{df_1}{dw} \right) \right) \dots \right) \right)$$

$$\frac{df_n}{dw} = \left(\left(\dots \left(\left(\frac{df_n}{df_{n-1}} \cdot \frac{df_{n-1}}{df_{n-2}} \right) \cdot \frac{df_{n-2}}{df_{n-3}} \right) \dots \cdot \frac{df_3}{df_2} \cdot \frac{df_2}{df_1} \right) \cdot \frac{df_1}{dw} \right)$$

Shrnutí backpropagation

- Obecný přístup
- *Ozdadu* aplikujeme řetízkové pravidlo

Obsah

1. Motivace
2. Opakování z minula
3. Problém linearity
4. Neuron
5. Neuronová síť
6. Trénování
7. Praktické tipy a použití

Škálování

Vstupy by měly být normalizované.

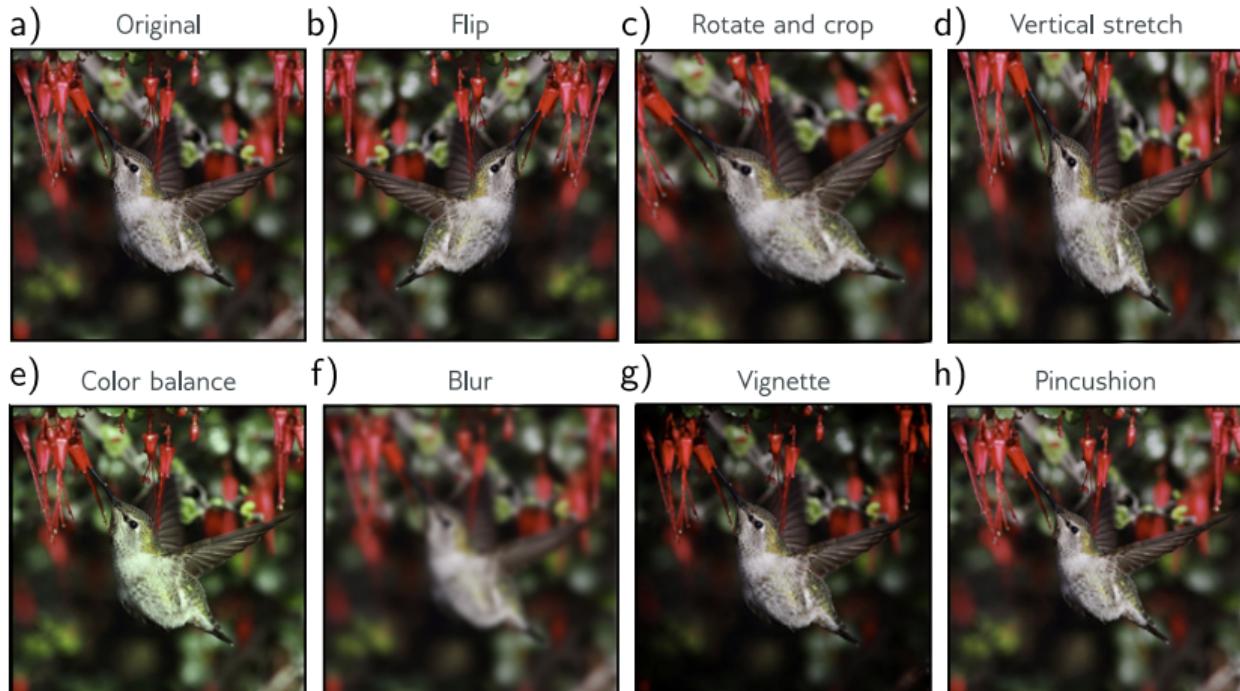
Nejběžnější možnost:

$$\frac{X_{*j} - \mu_j}{\sigma_j}$$

Ve zkratce: preference pro dobře zobecňují váhy.

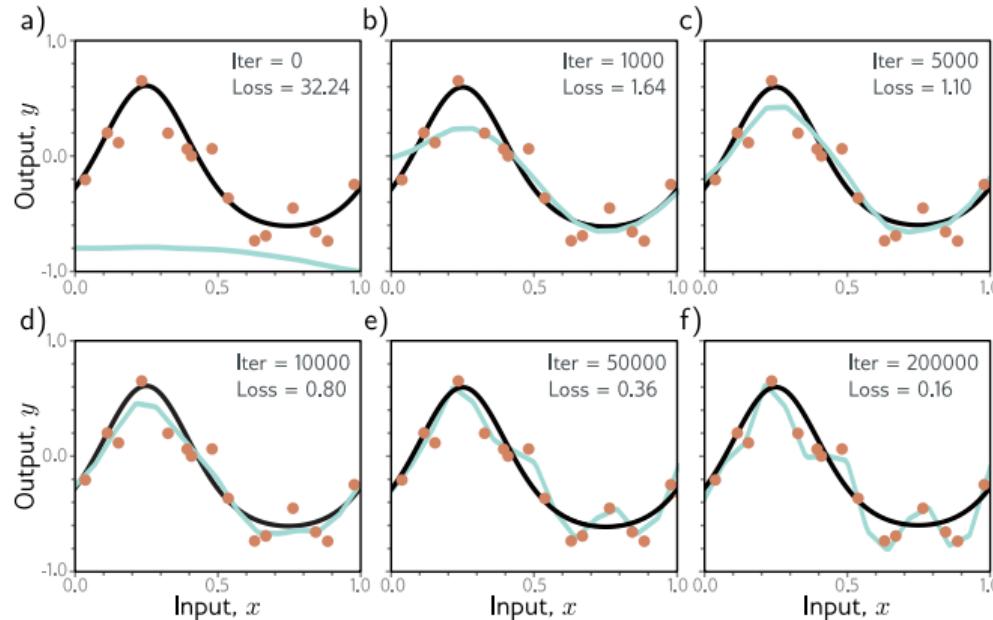
- L2 (AdamW)
- data augmentation, šum
- early stopping
- ensemble
- dropout
- batch normalization

Regularizace – data augmentation, šum



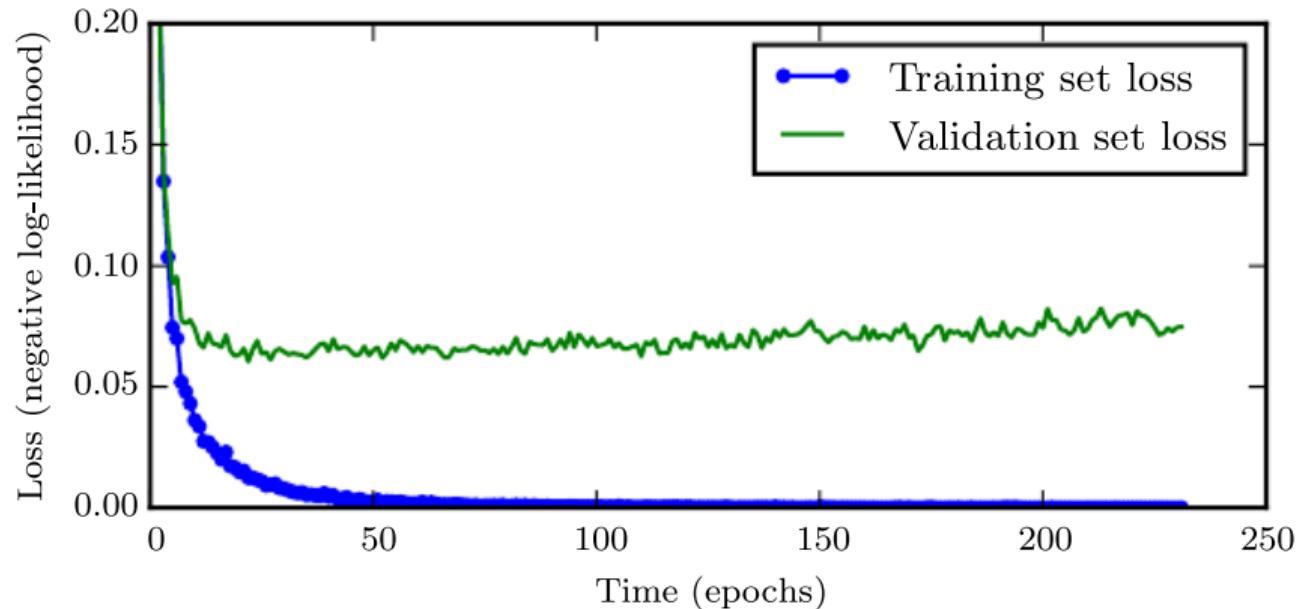
Obrázek: Simon J.D. Prince, Understanding DeepLearning, <https://udlbook.github.io/udlbook/>

Regularizace – early stopping



Obrázek: Simon J.D. Prince, Understanding DeepLearning, <https://udlbook.github.io/udlbook/>

Regularizace – early stopping 2



Obrázek: Aaron Courville, Ian Goodfellow, and Yoshua Bengio; "Deep Learning" book,
<https://www.deeplearningbook.org>

Regularizace – ensemble



Kuchařka pro trénování neuronových sítí

1. Pochopte data
2. Baseline (nemusí být ML model)
3. Přeucťte model
4. Regularizujte
5. Experimentujte s architekturou modelu
6. Opakujte body 4 a 5 dle potřeby

Více zde <https://karpathy.github.io/2019/04/25/recipe/>.

Hluboké neuronové sítě jsou populární, protože

- řeší problémy, se kterými jsme si dřív nevěděli rady (obraz, text),
- zvládají zpracovat gigantické množství dat,
- jsou všeestranné a znovupoužitelné,
- a zvládnou reprezentovat libovolnou rozumnou funkci.

Neváhejte se na mě obrátit s dotazy (bud' ted' nebo později na
dominik.farhan@gmail.com).

Doporučená literatura

- Understanding Deep Learning – skvělá kniha, která se věnuje i nejmodernějším tématům (GPT, generování obrázků).
- Deep Learning Book – obsáhlá teoretická kniha; ideální pokud chcete matematickou přesnost a obecně teoretičtější pohled.
- PyTorch – pokud chcete zkusit neuronovou síť trénovat, podívejte se na v současnosti nejpopulárnější balíček do Pythonu.
- 3Blue1Brown – videa s krásnými animacemi. Může pomoci s intuicí.
- Milan Straka: Deep Learning – rozsáhlý kurz přednášek z MFF UK s volně přístupnými materiály.