Image classification

Anders and Vedrana Dahl
DTU Compute

02506 Advanced Image Analysis
March 2022

Learning objectives – continuation from last week

- Become able to explain the principles of neural networks
- Be able to implement a feed forward neural network (multilayer perceptron)
- Understand the effect of different parameters and parameter choices
- Implement and understand effects of optimization methods for neural networks
- Get experience with classification

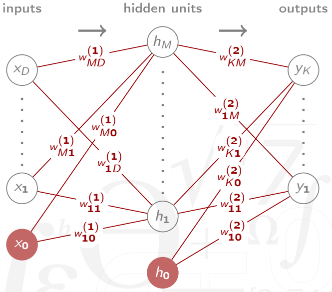
Summary of the last week

- Forward

$$
\begin{aligned}
z_i &= \sum_{d=0}^{D} w_{id}^{(1)} x_d \\
h_i &= a(z_i) = \max\{0, z_i\} \\
\hat{y}_j &= \sum_{m=0}^{M} w_{jm}^{(2)} h_m \\
y_j &= \frac{\exp(\hat{y}_j)}{\sum_{k=1}^{K} \exp(\hat{y}_k)}
\end{aligned}
$$

- Loss

$$
\begin{aligned}
L &= -\sum_{k=1}^{K} t_k \log y_k \ , \\
t_k &= \begin{cases} 1 & \text{if class label is } k \\ 0 & \text{otherwise} \end{cases} .
\end{aligned}
$$

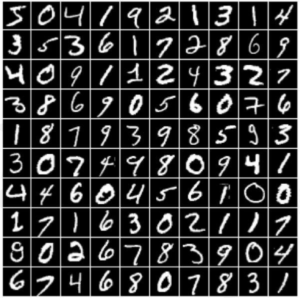


- Backward

$$
\begin{aligned}
\frac{\partial L}{\partial w_{ij}^{(l)}} &= \delta_i^{(l)} h_j^{(l-1)} \\
\delta_i^{(l^*)} &= y_i - t_i \\
\delta_i^{(l)} &= a'\left(z_i^{(l)}\right) \sum_k w_{ki}^{(l+1)} \delta_k^{(l+1)}
\end{aligned}
$$

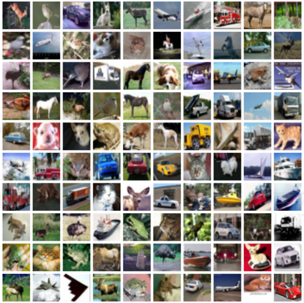
MNIST data

- MNIST problem – classifying handwritten digits
- MNIST data set – 70000 handwritten digits (10000 for test)
- 28×28 pixels – can be treated as 784×1 vectors
- Learn a network to classify
- Later compare to convolutional neural networks



CIFAR-10 data

- CIFAR-10 problem – classifying small RGB images
- CIFAR-10 data set – 60000 RGB images (10000 for test)
- $32 \times 32 \times 3$ pixels – can be treated as 3072×1 vectors
- Learn a network to classify
- Later compare to convolutional neural networks



Challenges with images

- Large size – only 28×28, but 784 dimensions for MNIST (and only 32×32, but 3072 dimensions for CIFAR-10)
- Network for point set:
  - Input – three neurons (incl. bias)
  - Hidden layer – five neurons (incl. bias)
  - Output layer – two neurons
  - In all: $3 \cdot 4 + 5 \cdot 2 = 22$ unknowns
- Network for MNIST:
  - Input – 785 neurons (incl. bias)
  - Hidden layers – e.g. $[785, 1000, 300]$ neurons (incl. bias)
  - Output layer – ten neurons
  - In all: $785 \cdot 999 + 1000 \cdot 299 + 300 \cdot 10 = 1086215$ unknowns

Dataset augmentation (Goodfellow 7.4)

- More data – the best way to make machile learning model generalize
- Create fake data: translate, rotate, scale – as long as you stay within the class
- Sometimes dubious whether an approach (e.g. noise injection) is considered data augmentation or part of the machine learning algorithm, which is important when evaluating methods. Hint: is it generally applicable?
- Especially relevant for object recognition

Batch and minibatch algorithms (Goodfellow 8.1.3)

- Normal (stochastic, online) feed forward network
  1. For each data point compute forward pass
  2. Compute gradients
  3. Backpropagate
- Batch method: process all data points, then update weights with averaged updates
- Minibatches
  1. Divide data into randomly selected minibatches
  2. For each data point in minibatch compute forward pass
  3. For each data point in minibatch compute gradients
  4. Average gradients and update once per minibatch

(Consider parallelization!)

NN terminology: one epoch = one forward pass and one backward pass of all the training examples

Momentum (Goodfellow 8.3.2)

- Attempts to accelerates learning
- Remembers past gradients and keeps a momentum
- Should help with very elongated basins of the loss function (poor conditioning of the Hessian)

1. Compute gradient estimate

$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \delta_i^{(l)} h_j^{(l-1)}$$

2. Compute velocity update

$$v_{ji}^{(t+1)} = \alpha v_{ji}^{(t)} - \eta \frac{\partial L}{\partial w_{ji}}$$

3. Apply update

$$w_{ji}^{(t+1)} = w_{ji}^{(t)} + v_{ji}^{(t+1)}$$

(Note that for $\alpha = 0$ this reduces to SGD.)

Parameter initialization strategies (Goodfellow 8.4)

- Initialization can influence: whether algorithm converges at all, how quickly it converges, whether it converges to minima with higher or lower cost, and whether minima has desirable generalization error
- Strategies are simple and heuristic, little understanding on how initialization affects optimization
- One certain property: initialization needs to break the symmetry – this motivates random initialization
- Gaussian or uniform distribution typically used, some heuristics on parameters, e.g. for layer with m inputs use $\frac{1}{\sqrt{m}}$ to set initial scale.

Adaptive learning rates – AdaGrad (Goodfellow 8.5)

- Adapt the learning rate for all gradients
- Weigh by a parameter related to the inverse of gradient size
- Updates per mini-batch

1. Choose a parameter $\delta = 10^{-}7$, learning rate η
2. Gradient g and accumulation variable r
3. Update according to
$$r = r + g \odot g$$
($\odot$ is element-wise multiplication)
4. Compute update
$$\Delta\theta = -\frac{\eta}{\delta\sqrt{r}} \odot g$$
(element-wise multiplication)
5. Apply update
$$\theta = \theta + \Delta\theta$$

Adaptive learning rates – RMSProp (Goodfellow 8.5)

- Similar to AdaGrad but with decay parameter ρ
- Updates per mini-batch

1. Choose a parameter $\delta = 10^{-}6$
2. Gradient g and accumulation variable r (inital $r = 0$)
3. Update according to
$$r = \rho r + (1 - \rho) g \odot g$$
($\odot$ is element-wise multiplication)
4. Compute update
$$\Delta\theta = -\frac{\eta}{\delta\sqrt{r}} \odot g$$
(element-wise multiplication)
5. Apply update
$$\theta = \theta + \Delta\theta$$

Adaptive learning rate – Adam (Goodfellow 8.5)

- Adaptive learning rate with first and second moment (two parameters $\rho_1 = 0.9$ and $\rho_2 = 0.999$ (in $[0, 1)$]))
- Step size $\eta = 0.001$
- Updates per mini-batch
- Choose a parameter $\delta = 10^-8$

1. Initialize $t = 0$ and in each iteration $t = t + 1$
2. $s = \rho_1 s + (1 - \rho_1) g, \quad \hat{s} = \frac{s}{1-\rho_1^t}$
3. $r = \rho_2 r + (1 - \rho_2) g \odot g, \quad \hat{r} = \frac{r}{1-\rho_2^t}$
4. Compute update
$$\Delta\theta = -\eta \frac{\hat{s}}{\sqrt{\hat{r}+\delta}}$$
(element-wise multiplication)
5. Apply update
$$\theta = \theta + \Delta\theta$$

Other approaches

- Adaptive learning rate (Goodfellow 8.5), simple strategies described in (Goodfellow 8.3.1)
- Noise robustness (Goodfellow 7.5)
- Dropout (Goodfellow 7.12)

MNIST competition

- Pre-trained NN for MNIST and CIFAR-10 classification based on your own implementation
- Assessment of your networks will be done by Sophia, Billy, and Andreas
- Strategy: first get working code and then optimize
- Begin training with a small subset, run training with validation, then run all training data
- Competition – winning team will be highlighted
- Deadline: Tuesday the 19th of April