group of machine

Modele generatywne – wprowadzenie

October 11, 2023

Grupa uczenia maszynowego GMUM `gmum.net`

GMUM – sumarycznie około 25 osób, około 15 doktorantów

Tematyka:

- głębokie sieci neuronowe (deep learning)
- przetwarzanie obrazów
- zastosowania (chemia, biologia)

Zainteresowanych współpracą zapraszam:
jacek.tabor(at)uj.edu.pl

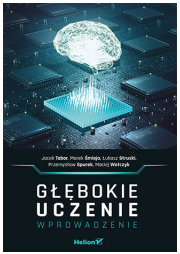


Figure: Wprowadzenie do głębokiego uczenia

Model generatywny

group of machine gmum learning research

Zadanie

- zbiór danych X (domyślnie zdjęcia)
- zadaniem jest nauczenie się rozkładu danych
- nauczony model ma pozwalać na generowanie nowych punktów z rozkładu
- optymalnie jeżeli można manipulować uzyskanymi zdjęciami

Jak to zrobić źle: losowo

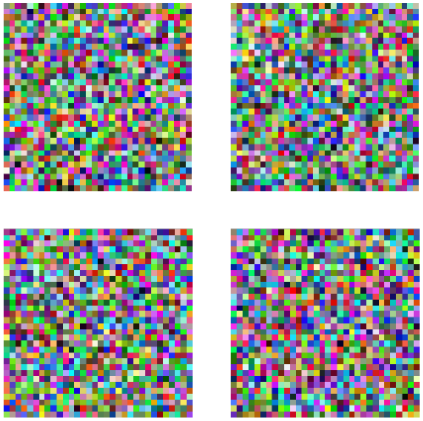


Figure: Obrazy wygenerowane z losowego rozkładu

Jak to zrobić źle: losowo

- Obrazy wylosowane z rozkładu równomiernego nie odpowiadają niczemu, co możemy spotkać w przestrzeni rzeczywistych zdjęć
- Rozkład równomierny błędnie modeluje prawdziwe dane
- Znaczące – rzeczywiste obrazy zajmują tylko bardzo małą część całej przestrzeni możliwych obrazów -– mówimy, że leżą na niskowymiarowej rozmaitości

Jak to zrobić poprawnie?

- klasyczne modele – GMM, kernelowa estymacja gęstość
- modele generatywne bazujące na AutoEnkoderze - VAE, WAE
- Generative adversarial networks (GANy),
- flow-based generative models,
- diffusion models

Generowanie danych - GANy

Generowanie sztuki https://neuromorphic.art/

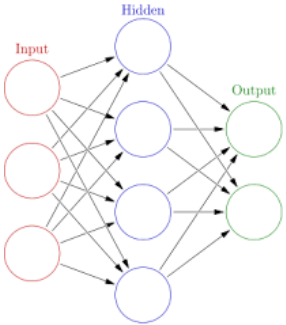
group of machine gmum learning research

Romuald Janik (GMUM), zaburzanie wag BigGAN arxiv.org/pdf/2109.08103

Generowanie obrazów za pomocą opisu

Tekst: By the rivers of Babylon, there we sat down Ye-eah we wept, when we remembered Zion. By the rivers of Babylon, there we sat down Ye-eah we wept, when we remembered Zion. (model dyfuzyjny)

(Sztuczne) sieci neuronowe

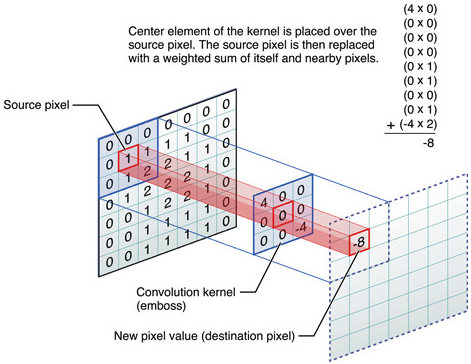


Idea: budujemy reprezentację za pomocą sieci, w której łatwo dokonać klasyfikacji za pomocą prostego modelu liniowego.

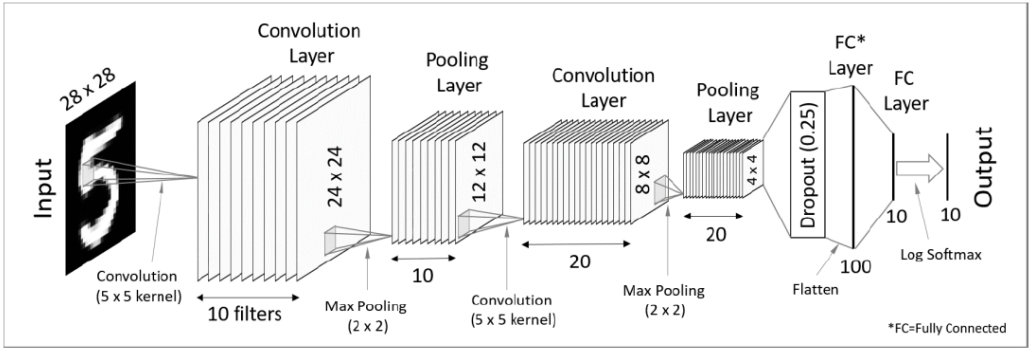
Funkcja aktywacji: ReLU = Rectified Linear Union

$$ReLU(x) = \max(0, x).$$

Filtry konwolucyjne



Architektura: t ypowa sieć konwolucyjna w klasyfikacji



Funkcja kosztu w modelu regresyjnym

Jak zmodyfikować wagi sieci by dostać lepsze wyniki?

Funkcja kosztu. Aby ocenić, co to znaczy lepiej, potrzebujemy pewnej obiektywnej miary. Rozważmy problem regresyjny, to znaczy mamy dane $X = (x_i)_{i=1..k}$, i dla każdego x_i mamy wartość $y_i \in \mathbb{R}$.

Naszym zadaniem teraz jest dotunować sieć neuronową, czyli funkcję F_ω (funkcja F zwraca wartości w zależności od parametrów wag ω) tak by

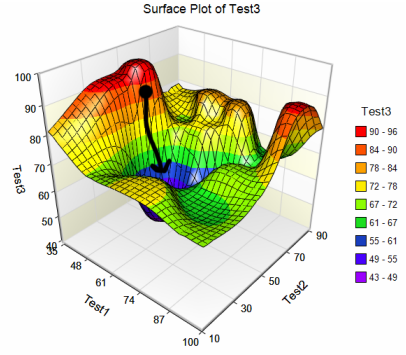
$$F_\omega(x_i) \approx y_i.$$

W konsekwencji, naszym celem jest minimalizacja funkcji straty względem paramatrów wag ω danej jako średni kwadratowy błąd popełniony przy regresji

$$\text{loss} = C(\omega) = \frac{1}{K} \sum_{i=1}^{K} |y_i - F_\omega(x_i)|^2.$$

Gradient

Minimalizujemy funkcję kosztu idąc w dół za pomocą gradientu (formalnie w kierunku przeciwnym do gradientu)::



Gradient to kierunek, gdzie funkcja idze maksymalnie ostro w górę.

Gradient

Jeżeli mamy funkcję G, to gradient G w punkcie dziedziny x oznaczamy przez

$$\nabla G(x).$$

Gradient to kierunek najszybszego wzrostu naszej funkcji.

Ponieważ chcemy iść w dół, metoda gradientowa polega więc na wybraniu dowolnego startowego punktu x_0, i poprawianiu

$$x_{n+1} = x_n - h \cdot \nabla G(x_n),$$

gdzie h to *learning rate* czyli wielkość naszego kroku. Im mniejsze, tym potencjalnie dokładniej idziemy, ale kosztem tego, że musimy wykonać więcej kroków by dojść do tego samego miejsca.

Liczbę iteracji przejścia przez cały zbiór danych które wykonujemy nazywamy liczbą *epok*. Na szczęście umiemy automatycznie liczyć gradient (backpropagation).

Stochastic gradient descent (SGD)

Problem: aby wyliczyć gradient $\nabla C(\omega)$ funkcji kosztu

$$C(\omega) = \sum_{i=1}^{K} |F_\theta(x_i) - y_i|$$

musimy przeprowadzić obliczenia względem całego zbioru danych $X = (x_i)_{i=1..K}$ co w przypadku dużych zbiorów danych jest po prostu niemożliwe!

Stochastic gradient descent (SGD)

Problem: aby wyliczyć gradient $\nabla C(\omega)$ funkcji kosztu

$$C(\omega) = \sum_{i=1}^{K} |F_\theta(x_i) - y_i|$$

musimy przeprowadzić obliczenia względem całego zbioru danych $X = (x_i)_{i=1..K}$ co w przypadku dużych zbiorów danych jest po prostu niemożliwe!

Dlatego upraszczamy sobie życie i zamiast liczyć gradient na całym zbiorze danych, liczymy po losowo wybranym małym fragmencie z danych o nazwie *mini-batch*. Okazuje się, że nie tylko mamy mniej obliczeń, ale stosując SGD (stochastic gradient descent) uzyskujemy lepsze wyniki. Typowo wybieramy wielkość batcha 64, ale zdarzają się sytuacje gdy jest 4 albo 1024 (czy więcej).

Ocena modelu – walidacja krzyżowa

Aby ocenić czy model dobrze zadziałał, musimy go testować na danych na których go nie uczyliśmy! W przeciwnym razie ryzykujemy overfitting, czyli nadmierne dopasowanie do danych (nauczenie się na pamięć).

Zbiór testowy.

Walidacja krzyżowa: zbiór na 5 części, uczymy na 4, sprawdzamy na pozostałej (uśredniamy wyniki z 5 możliwych).