Увод в машинното обучение

ПЕТЪР ХРИСТОВ

21/09/2023

УЧИТЕЛСКА ПРОГРАМА, CERN

Ресурси за машинно обучение (МО)

- Не губете време да изобретявате собствени методи, използвайте готови пакети
  - Повечето библиотеки са достъпни на Python

- Препоръчани пакети (Python)
  - **TensorFlow**. Разработван от Google, TensorFlow е платформа с отворен код за машинно обучение. Има богата и гъвкава екосистема от инструменти, библиотеки и ресурси за най-съвременно МО. Позволява лесно създаване и използване на приложения за МО. Пълна поддръжка на GPU, включително Mac M1-3 Metal GPU.
  - **PyTorch**. Разработван от Facebook, Pytorch е система с отворен код за МО. Ускорява прехода от изследователски прототип към пълноценен работещ продукт. Основен конкурент на Tensorflow
  - **Scikit-Learn**. Прост и ефективен инструмент за предсказващ анализ на данни. Достъпен за всички, с възможност за многократна употреба в най-разнообразен контекст. Изграден върху NumPy, SciPy и matplotlib. С отворен код и възможност за комерсиална употреба (BSD license). Покрива много традиционни методи, в това число линейна регресия и анализ на главните компоненти.

Машинно обучение (МО)

МО дава на компютрите способността да учат, без да бъдат изрично програмирани.

— Arthur Samuel (1959).

Една компютърна програма се учи от опит E по отношение на някаква задача T и някаква мярка за изпълнение P, ако нейните резултати върху T, измерени чрез P, се подобряват с натрупания опит E.

—Tom Mitchell (1997).

МО е част от по-общата област Изкуствен Интелект (AI).

Примери (Andrew Ng):

Работа с глеми бази данни

- Големи масиви данни от мрежата:
- посещения на Уеб сайтове, медицински записи, биология, инженерство

Приложения без „ръчно“ програмиране

- автономно пилотиране, разпознаване на ръкопис, обработка на естествени езици (NLP), компютърно зрение.

Самонастройващи се програми

- системи за препоръчване на продукти и съдържание (Amazon, Netflix,…), моделиране на човешкото обучение (мозък, мислене).

Още примери за приложение на МО

Общи задачи (J. Brownlee)

1. Оцветяване на черно-бели снимки.
2. Добавяне на звук към неми филми.
3. Автоматичен машинен превод.
4. Класифициране на обекти он снимки.
5. Генериране на ръкопис.
6. Генериране на текст (ChatGPT, Bing, Bard).
7. Генериране на описания към снимки.
8. Автоматични игри (шах, табла, го).

Физика (see Carleo Giuseppe et al.: Machine learning and the physical sciences)

1. Предсказване свойствата на материалите
2. Класификация на фазите на материята
3. Представяне на квантовите вълнови функции

Физика на високите енергии

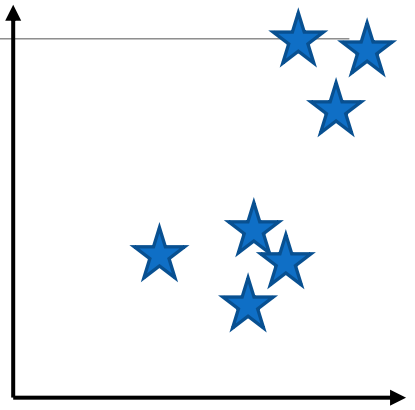
1. Идентификация на частици
2. Отбор на събития
3. Идентификация на струи
4. Детектиране на неутринни взаимодействия

Примерна класификация на методите за МО

Unsupervised – без учител, неподдържано, неконтролирано
Supervised – с учител, контролирано, поддържано
Semi-supervised – частично контролирано, частично поддържано
Reinforcement – с утвърждение, със затвърдяване

Patrick R. Nicolas

Примерна класификация на МО без учител/Unsupervised ML



МО без учител

- Клъстеризация
  - Клъстеризация с усредняване по k значения
    k-Means clustering
  - Спектрална клъстеризация
    Spectral clustering
  - Йерархична клъстеризация
    Hierarchical clustering
  - Максимизиране на очакването
    Expectation-maximisation (EM)
- Редуциране на размерността
  - **Метод на главните компоненти**
    Principal Component Analysis (PCA)
  - **Линеен дискриминантен анализ/**
    Метод на опорните вектори
    Linear Discriminant Analysis (LDA)

Patrick R. Nicolas

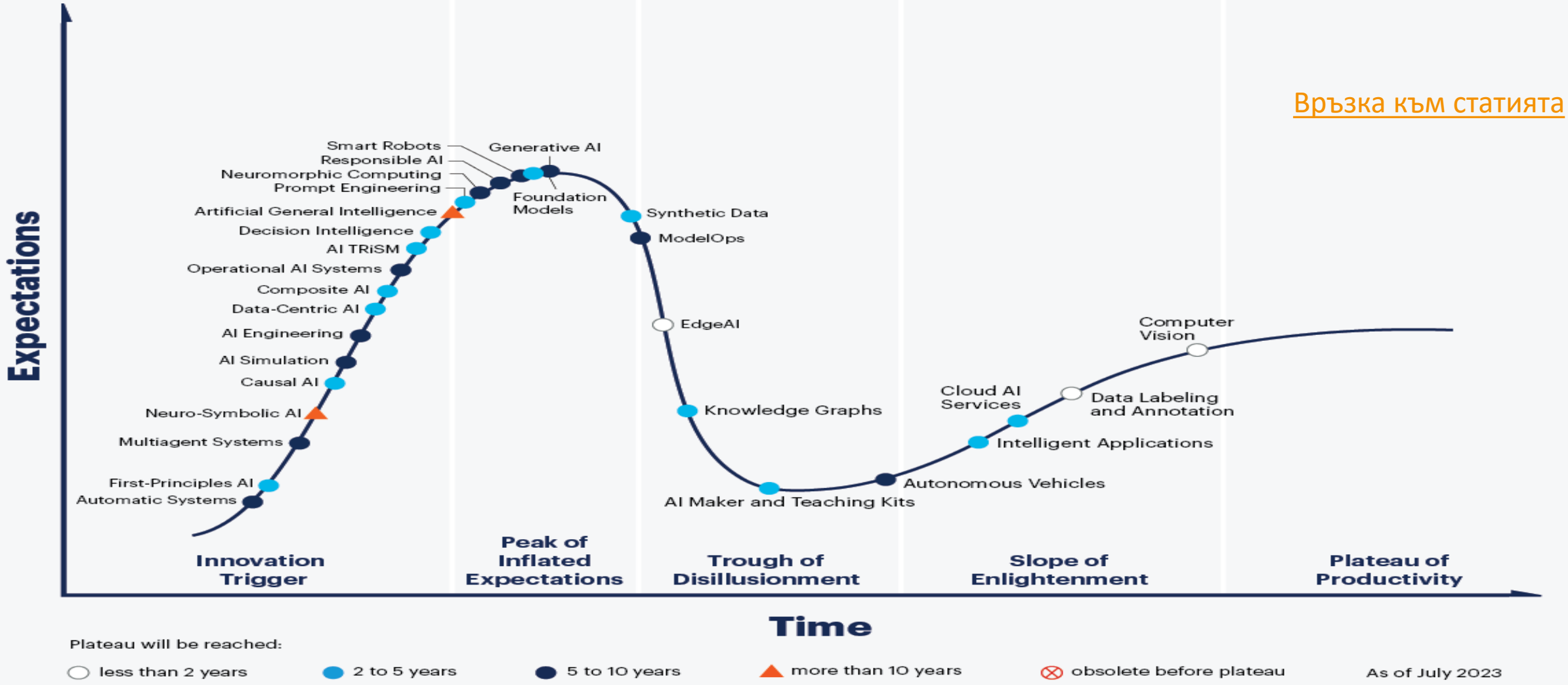
Примерна класификация на МО с учител/Supervised ML

Дадено е множество данни за обучение $T_D = \{y, \mathbf{x}\} = (y,\mathbf{x})_1 \dots (y,\mathbf{x})_N$, множество функции {f } и ограничения върху тези функции. Задача – да се научи машината да асоциира y = f(**x**)

Patrick R. Nicolas

Hype Cycle for Artificial Intelligence, 2023

Връзка към статията



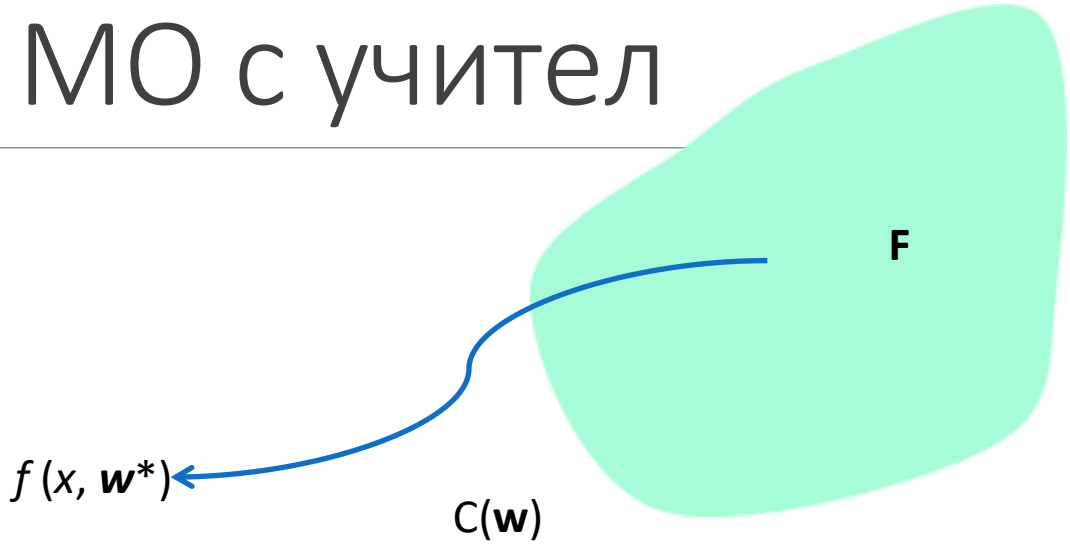
gartner.com

Gartner

Глобален подход при МО с учител

Избираме

- Функционално пространство ***F*** **= {** $f(x, \boldsymbol{w})$ **}**
- Ограничения ***C***
- Функция на загубите ***L*** в случай на неправилен избор



Метод

Намираме $f(x)$ като минимизираме емпиричния риск $R(\boldsymbol{w})$ и отчитаме ограниченията C(**w**)

$$R[f_w] = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i, \boldsymbol{w}))$$

Бръснач на Окам – избираме най-простата функция => използваме например регуляризация или пък сканираме от по-прости към по-сложни функции.

w са параметрите на функцията
Регуларизация – допълнителен „наказателен“ член към емпиричния риск, например $\boldsymbol{w}^2$

Fitting = апроксимиране

Примерна методика за обучение на моделите (за всички типове)

РАЗДЕЛЕНИ ДАННИ/SPLIT DATA

Пример

- 60% за обучение;
- 20% за валидиране на външните параметри (дизайн на невронна мрежа, параметър за регуляризация и т.н.);
- 20% за тест на производителността.

Забележка: използвайте произволни разделяния, за да избегнете артефакти на данни (подреждане, групиране и т.н.)

КРЪСТОСАНО ВАЛИДИРАНЕ/CROSS-VALIDATION

Пример: разбиваме масива данни на 5 части

- Използваме части 1-3 за обучение, 4 за валидиране, 5 за тест на производителността;
- Използваме 2-4 за обучение, 5 за валидиране, 1 за тест;
- Използваме 3-5 за обучение, 1 за валидиране, 2 за тест и т.н.
- Усредняваме резултатите от итерациите и изчисляваме емпиричните грешки

Предимство: по-добро използване на наличния набор от данни, оценка на грешката.

Примери за функции на загубите (на емпирическия риск)

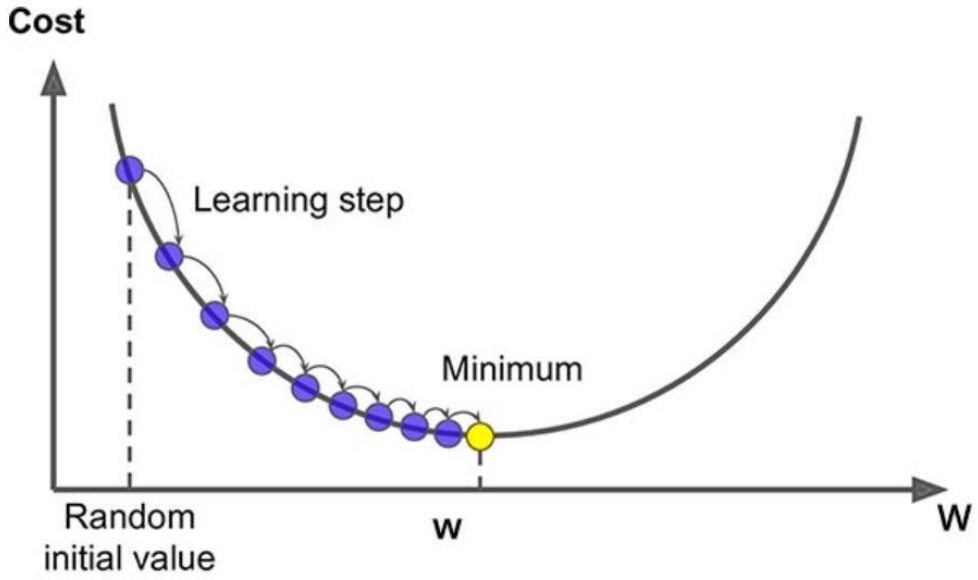
Регресия

- Средно квадратична грешка: усреднен квадрат на отклонението между предсказаното $\hat{y}_i$ и истинското y_i значения: $\boldsymbol{\Sigma}\|\hat{y}_i - y_i\|^2/N$
- Логаритмична функция на правдоподобието: $-\ln(p(\hat{y}_i|y_i))$, където p е вероятността да предскажем $\hat{y}_i$ ако истинското значение е y_i
- Huber Loss (1964) за изхвърляне на груби грешки: $\begin{cases} \frac{1}{2}(\hat{y}_i - y_i)^2, if\ |\hat{y}_i - y_i| \leq \delta \\ \delta\left(|\hat{y}_i - y_i| - \frac{1}{2}\delta\right)\ otherwise \end{cases}$

Класификация: значението y_i показва вероятността за принадлежност към клас *i*

- Логистична регресия – двоична кръстосана ентропия при един клас: $-y\ln\hat{y} - (1-y)\ln(1-\hat{y})$
- Кръстосана ентропия за *j* класове: $-\sum_j y_j \ln \hat{y}_j$

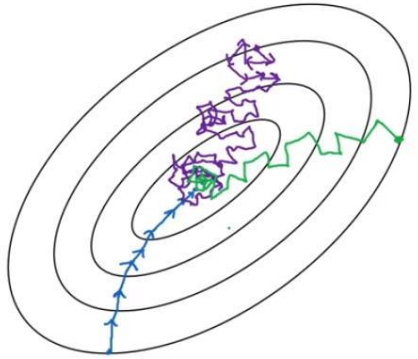
Минимизиране: градиентно спускане (GD)



Итеративна процедура, $\Delta w = -\eta \frac{\partial R}{\partial w} = -\eta \nabla_w R$,

η – стъпка на обучение,

R – емпиричен риск, ограниченията С са включени чрез множители на Лагранж



Използване на данните за обучение:

Стохастичен GD: параметрите се обновяват за всеки пример. Бърз, но може да има проблеми със сходимостта.

GD с мини-групи: параметрите се обновяват за всяка мини-група. Броят на примерите в мини-групите е по избор.

GD върху група: параметрите се обновяват върху всичките данни за обучение. Добра сходимост, но може да е много бавен.

Бързо минимизиране: модифицирано GD

Импулсна оптимизация: „памет“ за предишни градиенти чрез „импулсен“ вектор m.

$$\mathrm{m} \leftarrow \beta m - \eta \nabla_w R, w \leftarrow w + m$$

Ускорен градиент на Нестеров: градиентът на емпирическия риск се изчислява не в локалната точка w а леко напред в посока на „импулса“ m

$$\mathrm{m} \leftarrow \beta m - \eta \nabla_w R(w + \beta m), w \leftarrow w + m$$

AdaGrad: намаляване мащаба на най-стръмните компоненти от вектора на градиента

RMSProp: стъпката на обучение се мащабира с усреднените модули на градиентите от последните итерации, като теглата на най-старите итерации намаляват експоненциално

Adam: като в импулсната оптимизация се използва усреднен градиент с експоненциално намаляващи тегла за старите итерации; като в RMSProp се мащабира стъпката на обучение

Nadam: Adam плюс ускорен градиент на Нестеров

Регуларизация

Автоматически избор на „най-простата“ функция от даден клас или предпочитане на функции с необходими свойства. Обикновено се използва допълнителен „наказателен“ член към емпирическия риск

- Регуляризация на Тихонов: допълнителен член $-\lambda\|w\|_2^2$ избира малки значения на параметрите
- LASSO (least absolute shrinkage and selection operator) използва $-\lambda\|w\|_1$ което води до "разредност" на параметрите, някои от тях стават нула
- Отпадане (dropout): случаен избор на параметри, които не се актуализират по време на текущата итерация („памет“ за предишната итерация, опростяване на модела)
- Групова нормировка: стандартизиране на входа. За всяка размерност j се използват центрирани и нормирани значения $x_j \leftarrow \frac{x_j - \langle x_j \rangle}{\sigma_j}$

Бейсов подход: регуляризацията предоставя априорно знание за функцията на регресията

Линейна алгебра: регуларизацията отмества малките собствени значения на матрицата, която се инвертира и така намалява влиянието на флуктуациите

Ползи: **по-стабилни резултати** от минимизирането, избягване на прекалената апроксимация

Оценка на производителността – таблица на объркването (confusion matrix)

Таблица на объркването		Истински P+N	
		Положителни P=TP+FN	Отрицателни N=FP+TN
Предсказани	Положителни	Истински положителни TP	Фалшиви положителни Грешка от I род FP
	Отрицателни	Фалшиви отрицателни Грешка от II род FN	Истински отрицателни TN

Точност(Accuracy) = (TP+TN)/(P+N)

Селективност(Selectivity) = TN/N

Прецизност(Precision) или Ефективност(Efficiency)= TP/(TP+FP)

Чувствителност(Recall, Sensitivity) = TP/P

Пример: диагностика на рак

Ако диагностицираме само когато сме твърдо убедени – голяма прецизност, малка чувствителност

Ако не искаме да изтървем възможни случаи – малка прецизност, голяма чувствителност

Компромисна мярка F_1 = 2 x Прецизност x Чувствителност / (Прецизност + Чувствителност)

Дървета за решения и ансамблово обучение

Лекции за физици (физика на частиците):

Yann Coadou, ESIPAP 2019.

- Machine learning
- Decision trees

Katherine Woodruff, FNAL 2017.

- Introduction to Boosted Decision Trees and Hands-On Tutorial

Книги

Jason Brownlee, XGBoost With Python.

- Data
- Examples

Дървета за решения

Един от първите методи за МО (края на 50-те години), известен в статистиката като CART – Classification and regression tree, (Breiman et al., 1984)

Всяко разделяне във възел е избрано да максимизира информационната печалбата или да минимизира ентропията

- Информационната печалба е разликата в ентропията преди и след потенциалното разделяне
- Ентропията е max за 50/50 разделяне и min за 100/0 разделяне

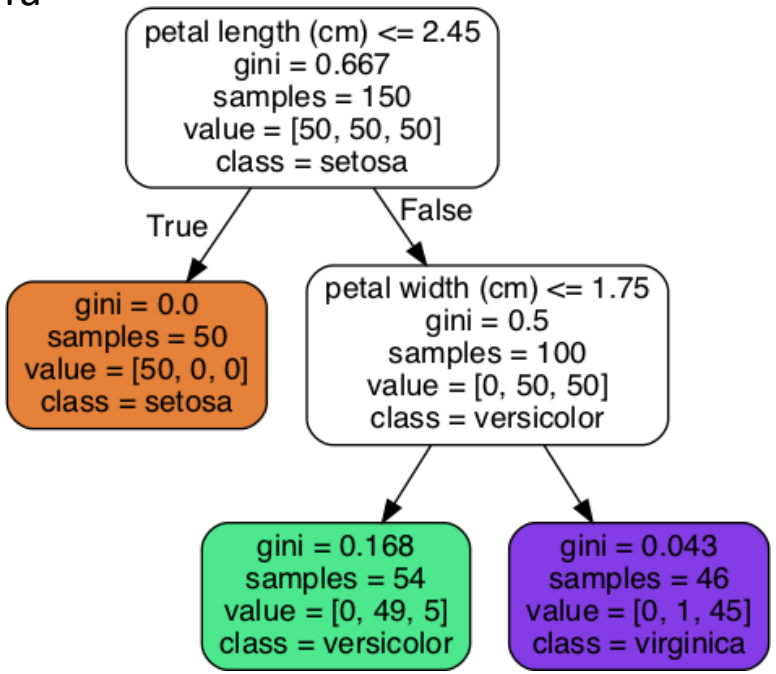
Разделянето е рекурсивно

- Повтаряме процедурата за всеки дъщерен възел, ако критериите за спиране не са удовлетворени: максимална дълбочина, липса на информационна печалба, перфектно разделяне и т.н.

Пример: класификация на цветовете от перуника по дължината и ширината на венчелистчетата

Възел: “най-добрият” атрибут и праг

„Клон“: Резултати от теста



„Лист“: Класификация или вероятност. Непрекъснат атрибут води до регресионно дърво

Оптимално разделяне

Функция за оценка на примесите:

- Максимална при еднакво количество примери от клас S и B
- Симетрична по отношение на вероятностите p_S и p_B
- Минимална ако примерите са само от един клас
- Вдлъбната: предпочита разделяне на по-голямо и по-малко подмножества

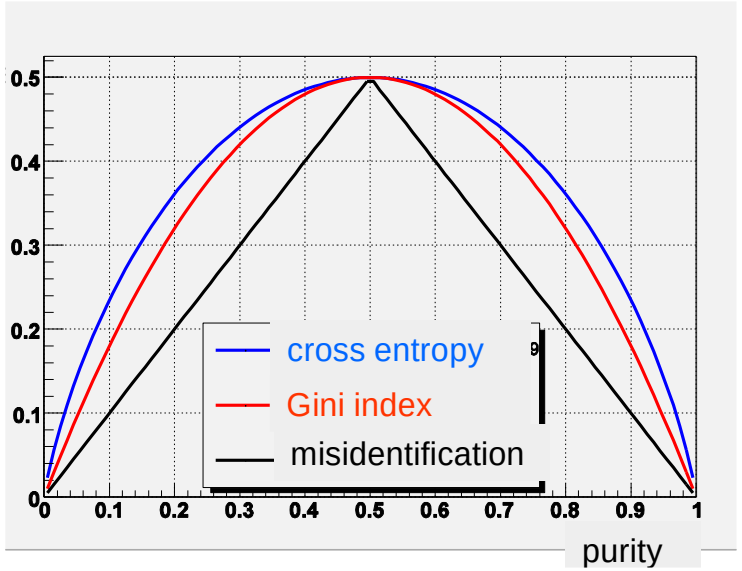
Намалява примесите за разделяне s на възел t към дъщерни възли t_P и t_F:

$$\Delta i(s, t) = i(t) - p_P \cdot i(t_P) - p_F \cdot i(t_F)$$

Избираме разделяне $\Delta i(s*,t) = \max \Delta i(s,t)$

Примери за функции

- Грешка на класификацията = 1 − max(p, 1 − p)
- Ентропия = − [p log(p) + (1-p) log(1-p)]/2
- Индекс на Джини Gini = 2p(1-p)



Окастряне на дърветата за решения

Дърветата за решения могат да станат прекалено сложни и прекалено нагодени към тренировъчните данни.

- Екстремален случай – всеки лист съдържа един пример. Прецизността и чувствителността за тренировъчното множество са максимални, но за тестовото множество са лоши.

Предварително окастряне – изискване за минимален брой примери във всеки лист.

Окастряне: премахване на части от дървото които не съдържат достатъчно примери или са прекалено специфични за тренировъчното множество.

- Започваме от листата и вървим към корена. Възлите заедно с всичките им деца се превръщат последователно в листа.

Регуляризация на база брой възли, клони и листа.

Окастрените дървета са по-малки и се интерпретират по-лесно

Предимства и недостатъци

ПРЕДИМСТВА

Бързо обучение.

Лесна интерпретация като списък от критерии.

Работи едновременно с непрекъснати и дискретни атрибути.

Не изисква трансформация на атрибутите.

Нечувствителни към неуместни атрибути.

Работят добре с голям брой атрибути.

НЕДОСТАТЪЦИ

Важни атрибути могат да бъдат маскирани.

Нестабилна структура, изменя се силно при смяна на тренировъчните данни.

Не могат да екстраполират.

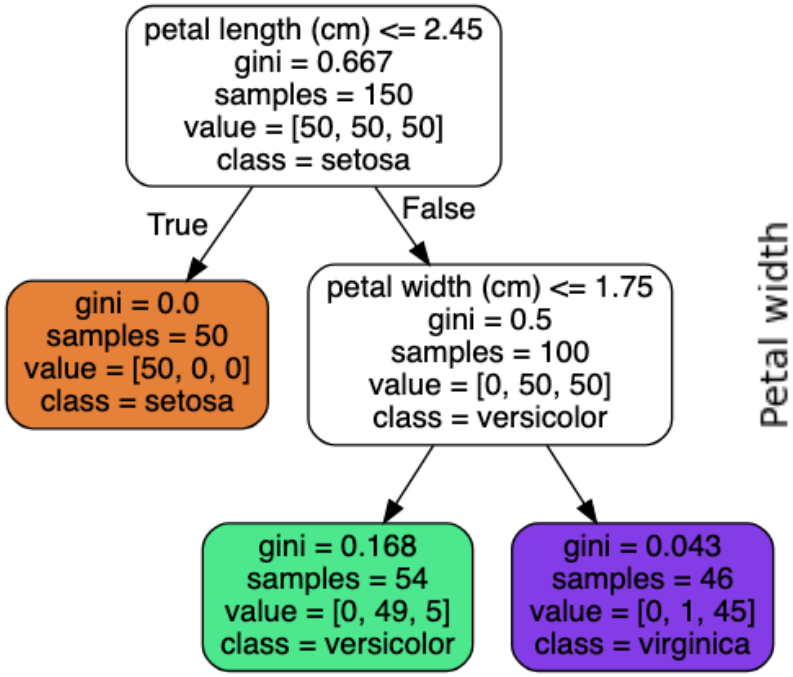
Имат нужда от много тренировъчни примери.

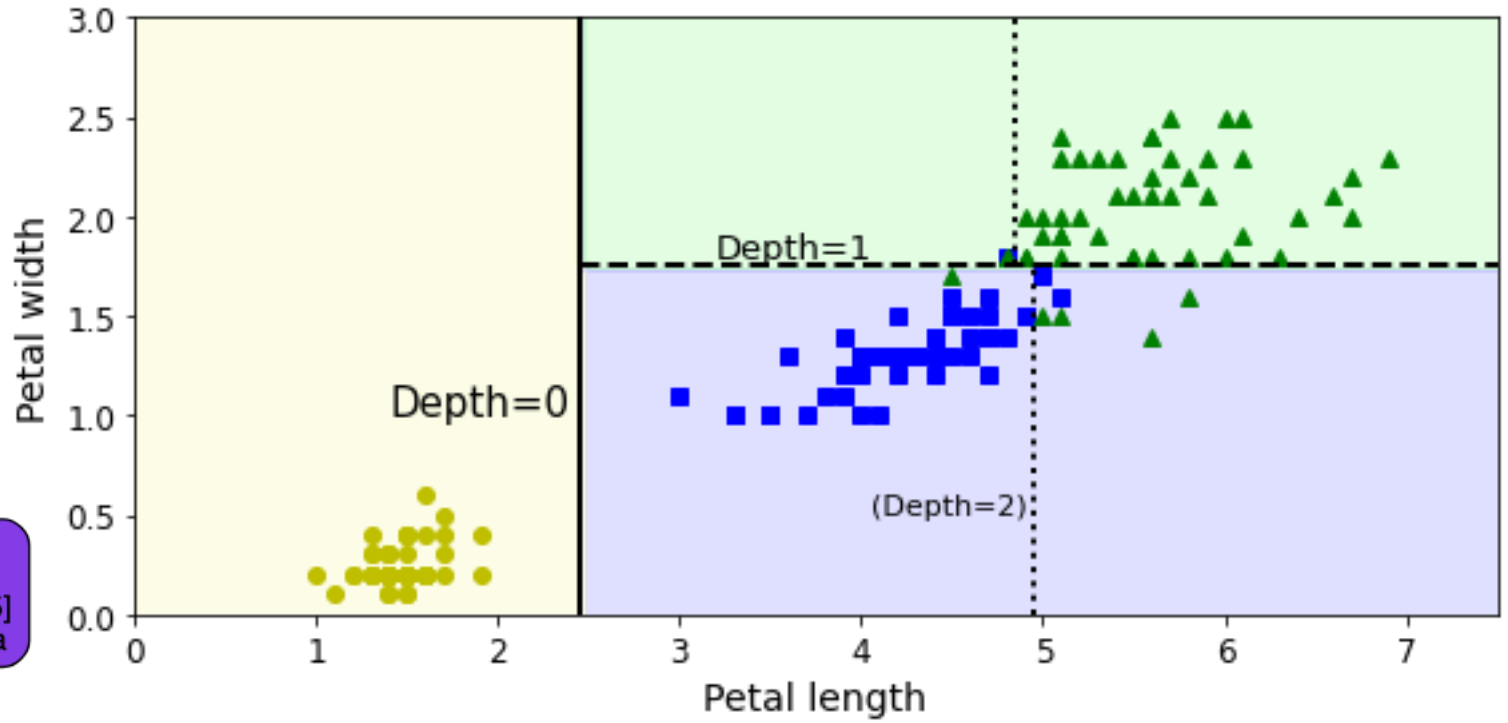
Не дават непрекъсната интерполация.

Поддават се на прекалена апроксимация.

Demo: Jupyter notebook (Aurélien Géron)

https://github.com/ageron/handson-ml2/blob/master/06_decision_trees.ipynb





Ансамблови методи

ЕДНО ДЪРВО

Съдържа само една информация за пример (един лист).

Практически не може да обобщи за вариации извън тренировъчното множество.

МНОГО ДЪРВЕТА

Разпределено представяне: броят на пресичащи се листа е експоненциален по отношение на броя дървета.

Много листа съдържат всеки пример ⇒ по-богато описание на тренировъчното множество.

Колекция от „слаби ученици/дървета“ след усредняване се превръща в ефективен метод.

Популярни ансамблови методи

Bagging (bootstrap aggregation)

– Всяко дърво се обучава върху **случайна извадка** от тренировъчното множество

Random Forest/ случайна гора

– Bagging със **случайни дървета**
– Случайно подмножество от атрибути за всяко разделяне

Boosting/подсилване

– Всяко дърво се обучава върху пълното тренировъчно множество, но примерите имат **различни тегла**. Може да се използва в различни методи за МО.

Агрегиране на прогнозите: **гласуване** или **осредняване**

Като цяло Boosting>Random Forests>Bagging>Single Tree.

Методи за подсилване

AdaBoost :“Adaptive Boosting” – един от първите алгоритми
- Неправилно класифицираните примери имат по-голямо тегло в следващите дървета
- Класификация с мажоритарен вот от всички дървета
- Freund and Schapire, 1996

Gradient Boosting/Градиентно подсилване
- Използва градиентно спускане за създаване на нови „ученици/дървета“
- Функцията на загубите е диференцируема
- Friedman, 1999

XGBoost “eXtreme Gradient Boosting”. https://github.com/dmlc/xgboost
- Нов метод за градиентно подсилване: добавянето на ново дърво става част от минимизацията
- Много популярен и успешен в състезанията на Kaggle
- Chen and Guestrin, 2016

CatBoost: високо производителна библиотека за градиентно подсилване на дървета от Yandex
- По-добра производителност в сравнение с LightGBM, XGBoost, H2O
- По-бързо обучение

Предимства и недостатъци на подсилените дървета за решения

ПРЕДИМСТВА

Бързи
- И тренировката, и предсказанията са бързи

Лесни за настройка

Нечувствителни към скалата
- Атрибутите могат да са категории или променливи

Добра производителност

Обучението върху неправилно класифицираните примери повишава силно точността

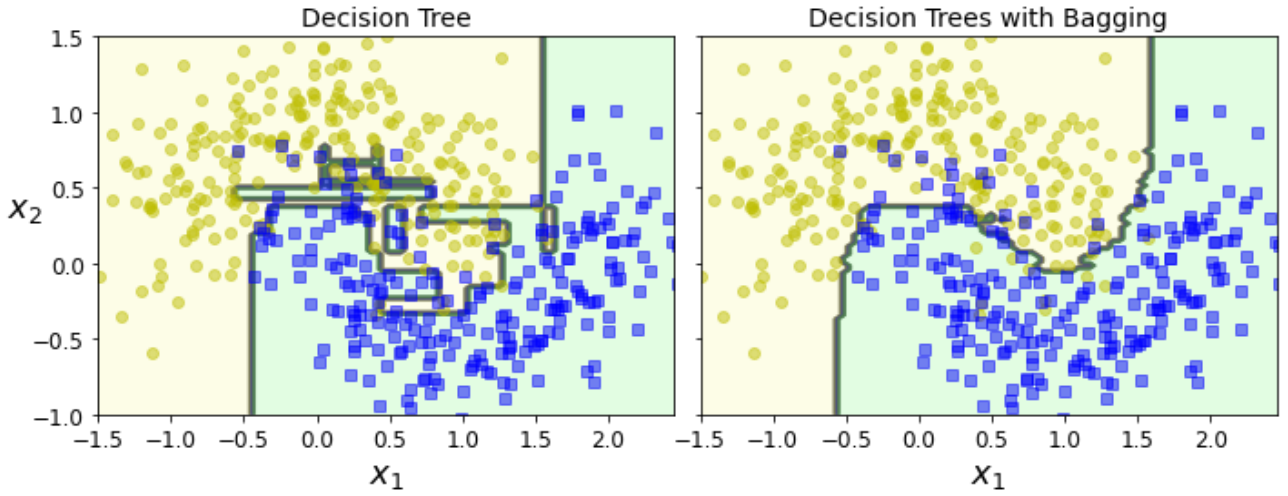
Достъпен и разнообразен софтуер
- Алгоритмите се използват много често
- Добра поддръжка и тестиране

НЕДОСТАТЪЦИ

Алгоритмите са чувствителни към прекалена апроксимация (нагаждане) и шум (груби грешки)
- Винаги е нужно валидиране
- Модерният софтуер има средства за избягване на прекаленото нагаждане

Demo: Jupyter notebook (Aurélien Géron)

https://github.com/ageron/handson-ml2/blob/master/07_ensemble_learning_and_random_forests.ipynb



Voting classifiers
Bagging examples
Random forests
Out-of-bag evaluation
Feature importance
AdaBoost
Gradient Boosting
Gradient Boosting with Early stopping
Using XGBoost

Hands-on XGboost

https://github.com/k-woodruff/bdt-tutorial

Hands-on boosted decision tree tutorial (using XGBoost) for September 2017 Fermilab Machine Learning Group Meeting

Primary Author: Katherine Woodruff

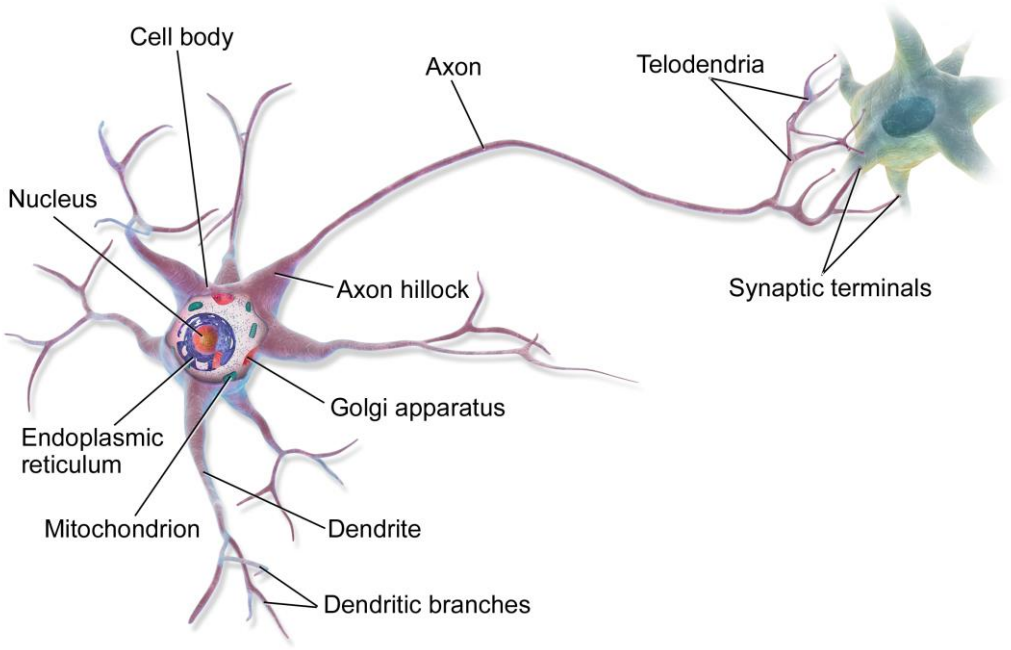
https://github.com/k-woodruff/bdt-tutorial/blob/master/bdt_tutorial.ipynb

Невронни мрежи

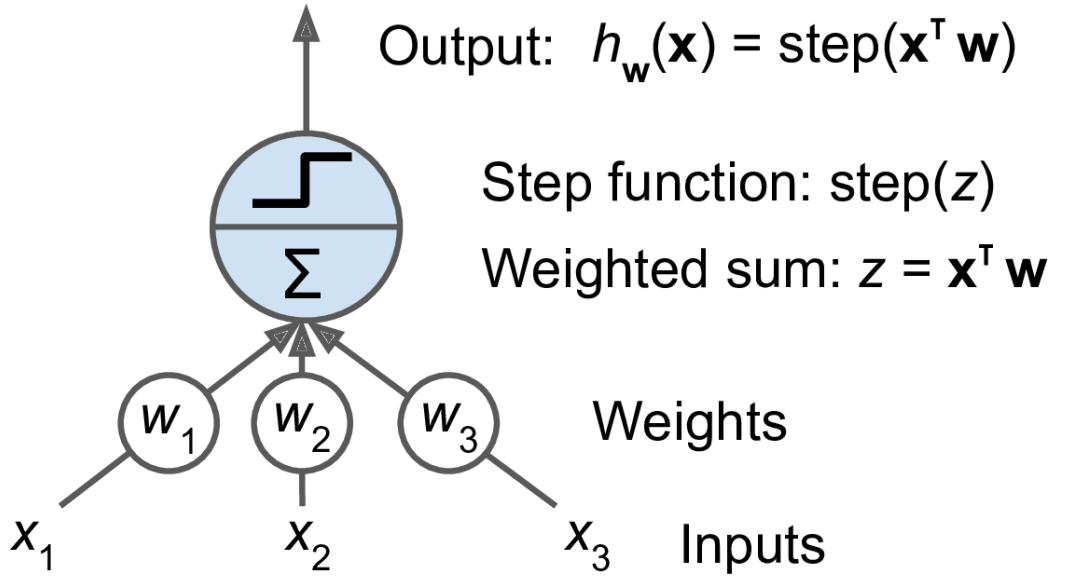
Използван е учебникът на A. Géron

Неврон и перцептрон

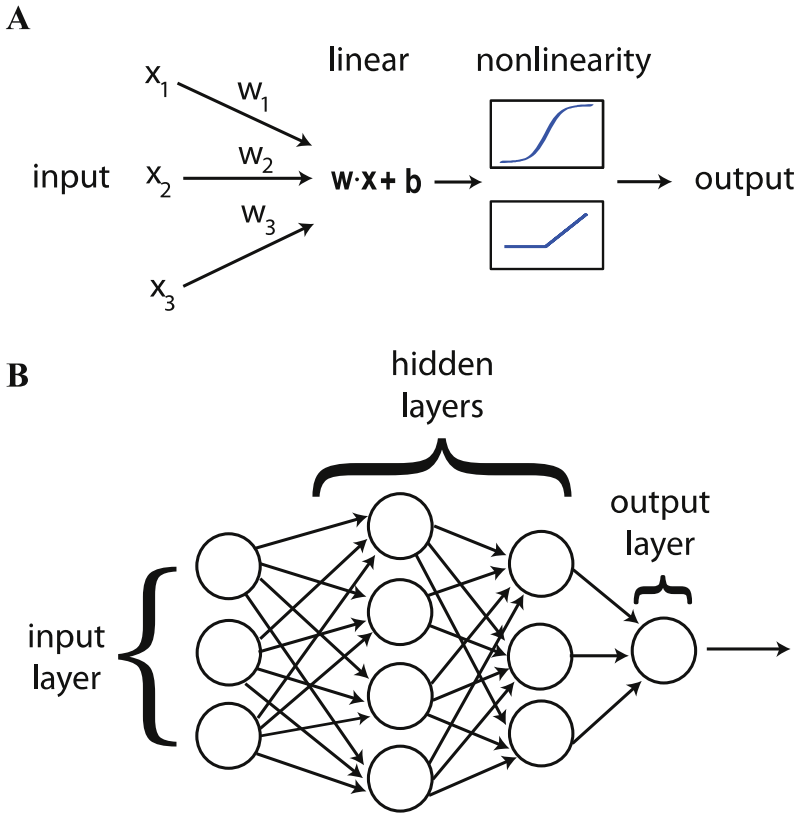
Перцептрон,
F. Rosenblat (1957)



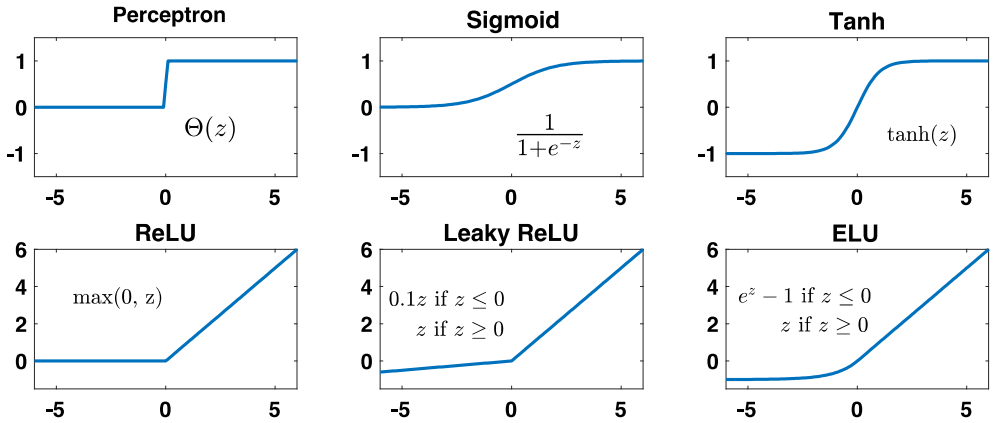
Биологически неврон (Wikipedia)



Невронни мрежи (NN)



Популярни нелинейни функции на активиране



Невронна мрежа: йерархична структура на взаимосвързани слоеве от неврони.

Теорема за универсалната апроксимация: NN с един скрит слой може да апроксимира с произволна точност всяка непрекъсната векторна функция с векторен аргумент

P. Mehta, M. Bukov, C.-H. Wang et al. / Physics Reports 810 (2019) 1–124

Задачи на NN: регресия и класификация

РЕГРЕСИЯ

Многомерна регресия: тренировъчният масив се състои от двойки вектори ($\mathbf{x}_i$,$\mathbf{y}_i$), където **y** са значенията в точка **x**. Цел – да се предскажат значенията в произволна точка

- Един изходен неврон за измерение, обикновено без активационна функция за да покрие пълния диапазон
- Емпиричен риск: обикновено средна квадратична грешка
- Скрити слоеве 1-5 с 10-100 неврона, RELU или ELU активиране

КЛАСИФИКАЦИЯ

Тренировка с вектори от атрибути и етикет.

- Кодиране на изхода: един бит за етикет
- Изходните неврони предсказват вероятности

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross entropy	Cross entropy	Cross entropy

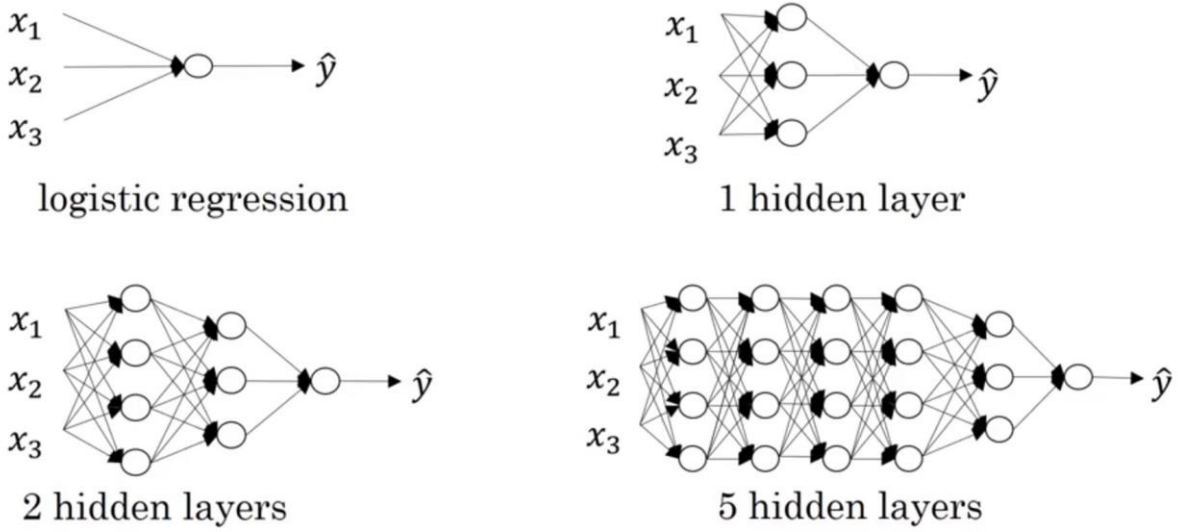
Обучение на NN: обратно разпространение на грешката

Изобретено през 1986 от David Rumelhart, Geoffrey Hinton, and Ronald Williams.

- Ефективен алгоритъм за пресмятане на градиентите по **W** и **b** отчитащ спецификата на NN
- Две преминавания през NN: напред за изчисляване изходите на всеки неврон и назад, за пресмятане на градиентите
- Реализира мини-групово GD и няколко преминавания през тренировъчните данни, наречени епохи
- При преминаване напред запазва всички промежутъчни резултати, необходими за обратното разпространение
- С помощта на емпирическия риск пресмята грешката на NN върху всяка мини-група
- По правилото за диференциране на сложна функция пресмята приноса на всеки NN параметър за грешката, от изходния слой към входния
- Използвайки пресметнатите градиенти обновява всички параметри на NN в GD стъпка

Обратното разпространение е специфично за NN. То позволява създаването на дълбоки невронни мрежи, където „дълбоки“ означава много скрити (вътрешни) слоеве.

Дълбоки невронни мрежи (DNN)



Диференцирането на сложна функция в обратното разпространяване изисква умножение на производните, но *sigmoid* и *tanh* имат относително малки значения, така че наблюдаваме “изчезващи градиенти” => ***sigmoid* и *tanh* не се използват в DNN**

ReLU, Leaky ReLU и ELU са предпочитани.

Нужна е регуляризация (включително отпадане и групова нормировка)

Хиперпараметри: брой скрити слоеве, брой неврони във всеки слой, активационна функция, метод за минимизиране, стъпка на обучение, размер на мини-група, итерации

Случайна инициализация (Glorot, He, LeCun): трябва да се „нарушат всички симетрии“.

Дълбоко обучение: защо има толкова много параметри?

- Устойчивост. Устойчивостта е способност на мрежата да се справя с малки промени.
- Bubeck and Sellke показват, че свръхпараметризацията е необходима, за да бъде една мрежа стабилна.
- Колко параметри са нужни за апроксимация на данни с гладка функция (гладкостта в математиката е еквивалентна на устойчивост)?
- Те показват, че гладка апроксимация на данни с висока размерност изисква не само n параметъра, но n × d параметъра, където d е размерността на входния вектор (например, 784 за 784-пикселна картинка). Иначе казано, ако мрежата трябва да запомни устойчиво тренировъчните данни, свръхпараметризацията не само помага, но е задължителна.
- Популярна статия в quantamagazine.org
- Математически е невъзможно да има едновременно възможност за интерпретация (в смисъл на чувствителност към изменения на атрибут за оценка на важността му) и устойчивост (обичайното изискване към статистически модел малките промени във входа да не променят много изхода): https://arxiv.org/abs/2205.15834 .

DNN

ПРЕДИМСТВА

Много мощен инструмент

Добри резултати в класификацията и регресията

Достъпни във всички популярни пакети и библиотеки

Много натрупан опит, развита интуиция, достъпни указания и трикове

НЕДОСТАТЪЦИ

Обучението може да е бавно и трудно

Необходими са големи тренировъчни масиви

Много хиперпараметри

DNN често е „черна кутия“

DNN не отчита симетриите на системата, която апроксимира

Конволюционни невронни мрежи (CNN)

CNN: Концепция

Революция в обработката на изображения: детектиране на обекти, класификация, сервис за търсене на изображения, самоуправляващи се коли, автоматична класификация на видео.

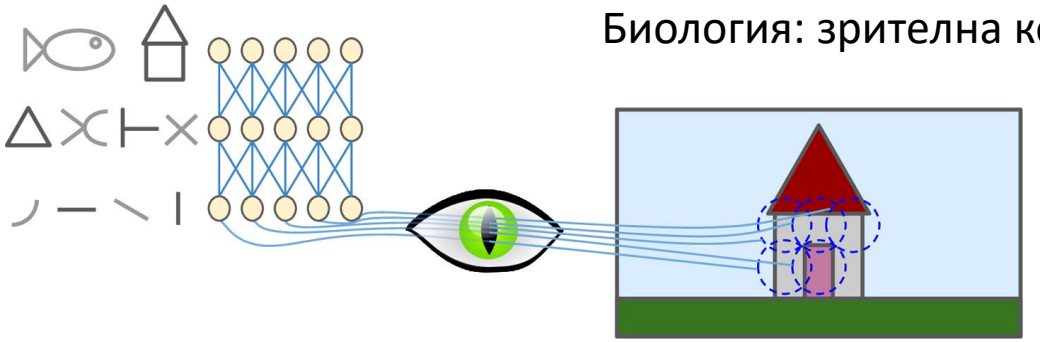
Възникват като резултат от изучаването на мозъчната кора, отговаряща за зрението през 1980-те години

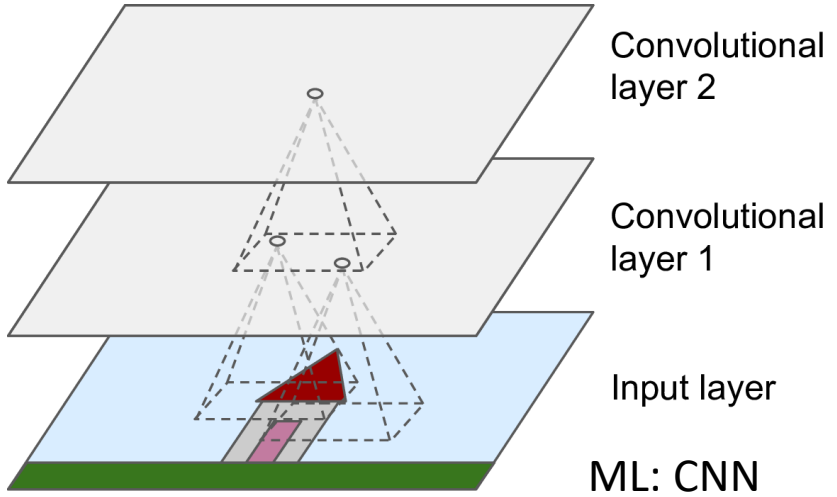
Използват транслационната инвариантност

Научават локални структури и ги обобщават в последователни слоеве

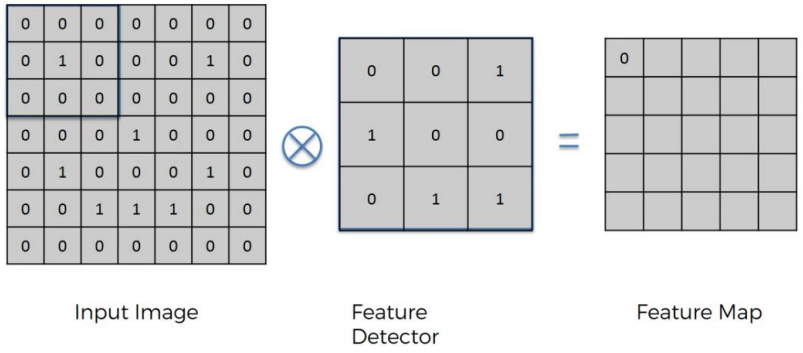
Конволюция във физиката: функция на Грин за линейно диф. уравнение, сигнален филтър с ядро K

$$g(x) = \int K(x-y)f(y)dy$$





Елементи на CNN: конволюционни слоеве

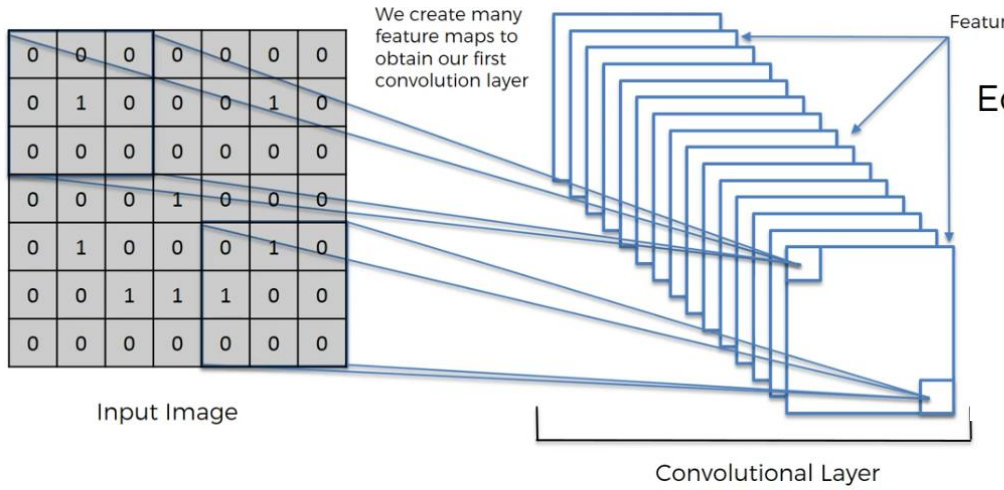


Sharpen:

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

Edge Enhance:

	0	0	0	
	-1	1	0	
	0	0	0	



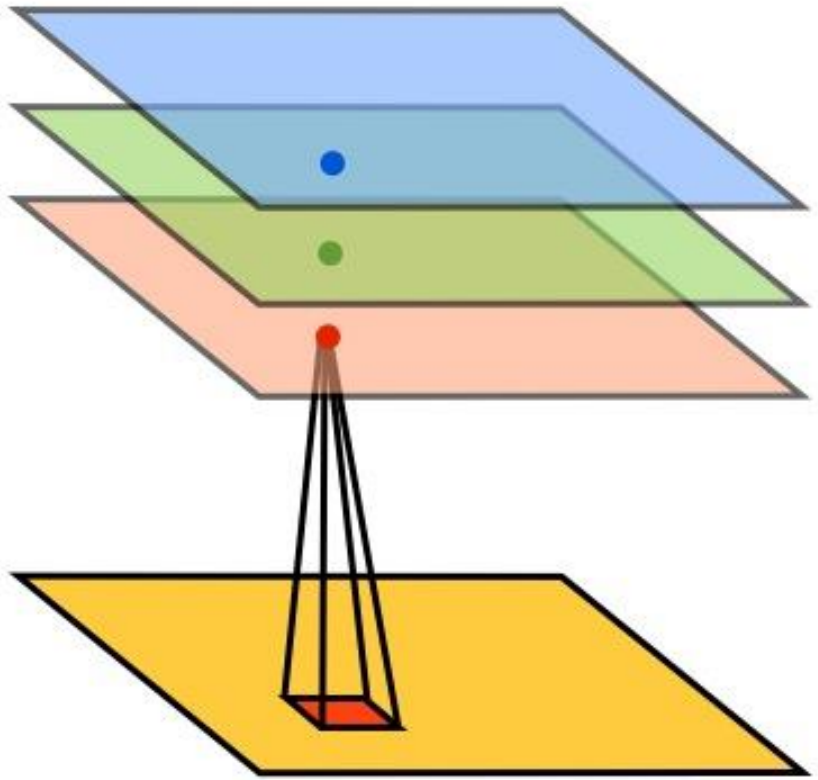
Edge Detect:

	0	1	0	
	1	-4	1	
	0	1	0	

Emboss:

	-2	-1	0	
	-1	1	1	
	0	1	2	

Конволюционни слоеве



Много по-малък брой тегла, независим от размера на изображението

- Напълно свързана мрежа: N^2 (N=размер на слой/изображение)
- Конволюционна мрежа: M (M=размер на ядро)

Много ефективна обработка на изображения
По-мощно обобщаване => по-малко изображения са нужни за обучение

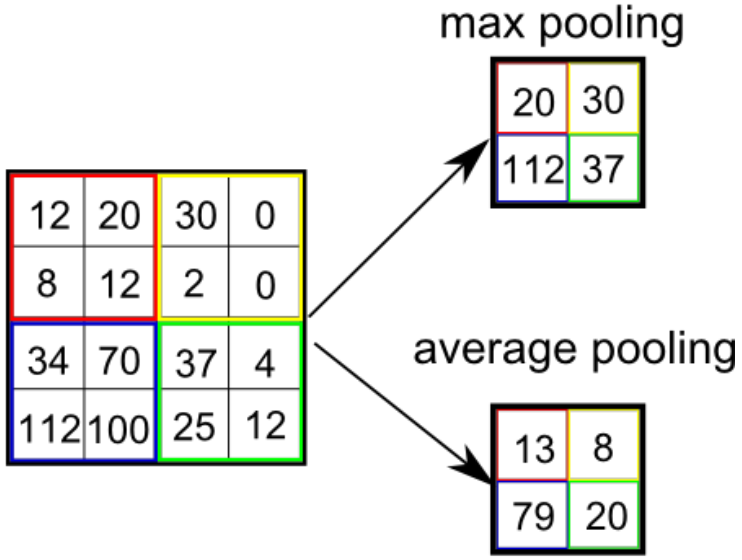
Могат да научат пространствената йерархия на изображенията: от основни геометрични структури към комплексни комбинации и концепции.

Извличане на множество характеристики с помощта на различни филтри

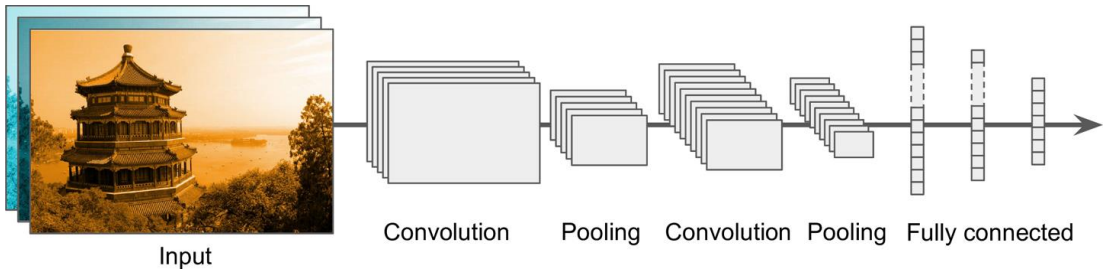
Елементни на CNN: обединяващи слоеве, „сплескване“/flattening

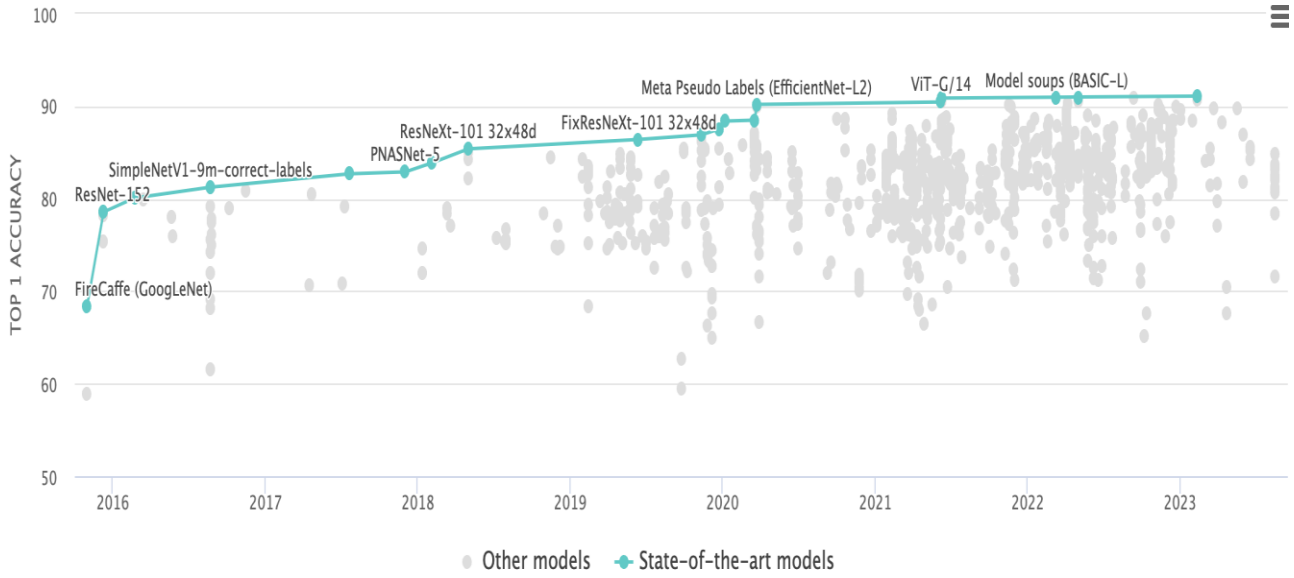
Използва се за намаляване размерността на картите на характеристиките (намалена „резолюция“, подизвадка)

Не научават параметри



CNN архитектури





Достъпни са много изключително успешни архитектури

- LeNet-5, Yann LeCun (1998)
- AlexNet (2012)
- GoogLeNet (2014)
- Visual Geometry Group (VGG) VGGNet (2014)
- Residual Network, ResNet (2015)
- Xception (2016)
- Squeeze-and-Excitation Networks, SENet (2017)

Повече за Kaggle competition: ImageNet Winning CNN Architectures (ILSVRC)

Image Classification on ImageNet

CNN накратко

ПРЕДИМСТВА

Най-съвременна област в компютърното зрение

Много активни изследвания

Бързо обучение

Изискват относително малки тренировъчни масиви в сравнение с напълно свързани DNN

Могат да се изпозват на само за обработка на изображения

НЕДОСТАТЪЦИ

Най-успешните архитектури са много комплексни и изобщо не са интуитивни

Работят много по-добре на GPU

Не са „универсално решение“: могат да се провалят в конкретни задачи извън обработката на изображения.

Автоенкодери

Лекции:

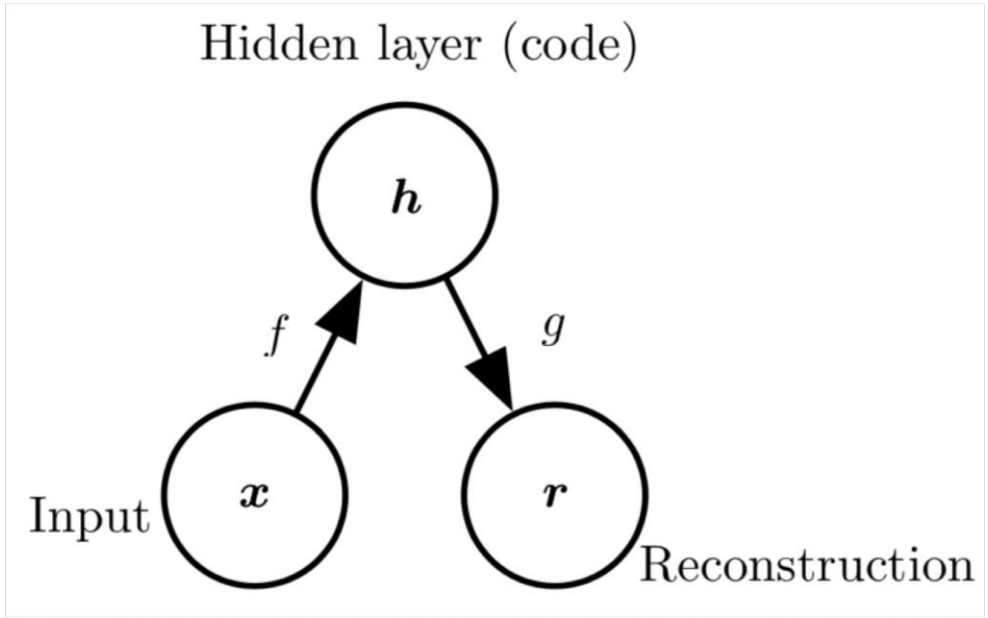
Ian Goodfellow

Andreas Wichert

Pavlos Protopapas and Mark Glickman

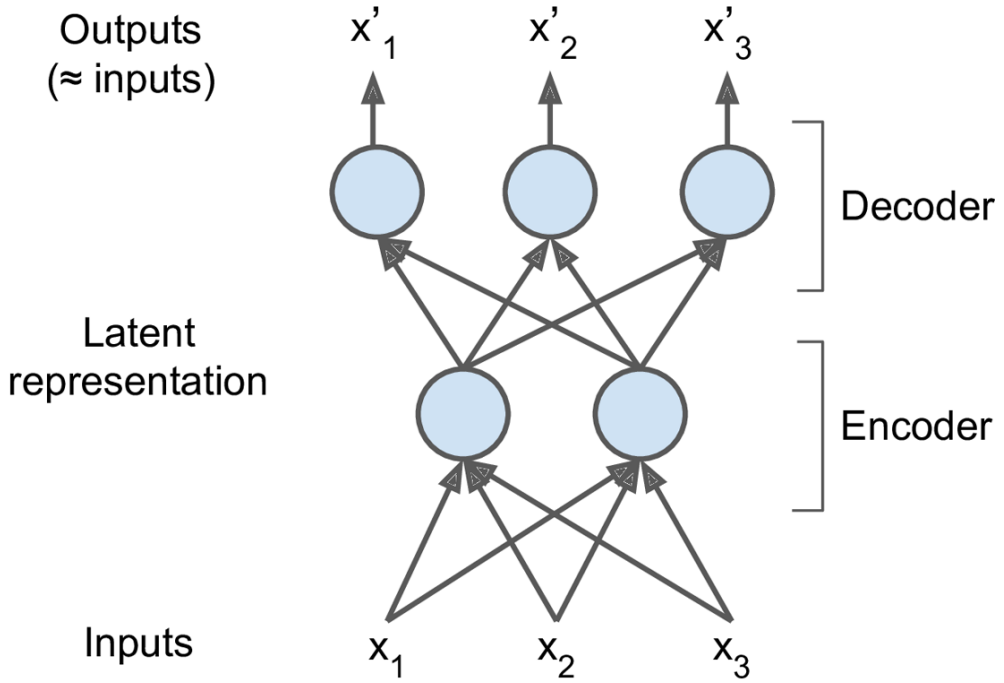
Автоенкодери: общи сведения

- Автоенкодерите са пример за невронна мрежа, която се обучава без учител
- Задачата им е да възпроизведат входовете върху изходите
- Скритият слой **h** представя структурата, свойствата и особеностите на данните
- Можем да разглеждаме две части на мрежата
  - Енкодер **h**=f(**x**), създаващ вътрешното (кодирано) представяне на данните
  - Декодер **r**=g(**h**), реконструиращ оригиналния вид на данните



Автоенкодери: използване

- Сравнително лесен и евтин достъп до данни без етикети
- Използване
  - Намаляване на размерността на данните
  - Намиране на по-подходящо представяне за дадена задача
  - Смесване на входовете
  - Намаляване на шума, възстановяване на пропуснати данни
  - Предварително обучение на дълбоки мрежи
- Част от входната информацията може да се губи



От тривиален резултат **x’= x** към извличане на структура и свойства

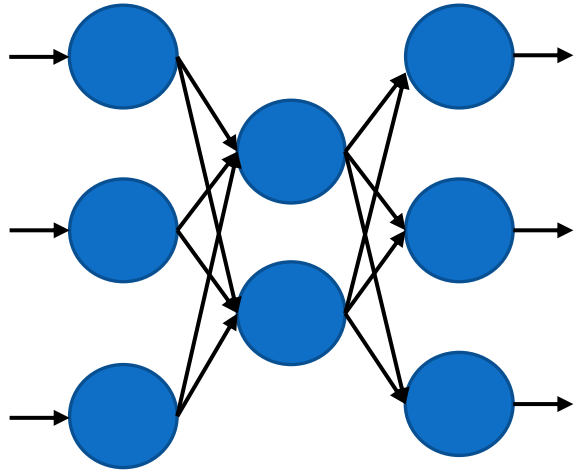
- Непълни (undercomplete) автоенкодери
  - ***h*** има по-малка размерност от ***x***
  - ***f*** или ***g*** имат малък капацитет **(**линейна ***g*****)**
  - Част от информацията се изхвърля в ***h***
- Препълнени (overcomplete) автоенкодери
  - ***h*** има по-голяма размерност от ***x***
  - Необходима е регуляризация, която ефективно намалява степените на свобода в скрития слой
- Методи за регуляризация
  - Разредени (sparse) автоенкодери
    - Регуляризираща добавка на Кулбак-Лейблер
  - Обозначаващи (denoising) автоенкодери
    - Автоенкодери с отпадане (dropout)
    - AE с добавен шум
  - Контрактивни AE
    - Производните на активационната функция в кодиращия слой от значенията на входовете са малки

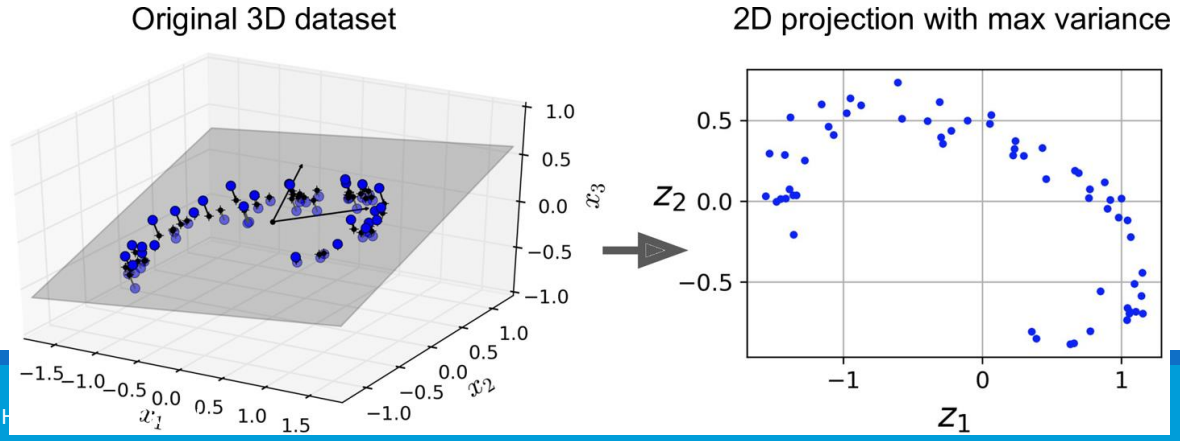
Непълен автоенкодер за определяне на принципни компоненти

Две компоненти – енкодер (3→2) и декодер (2→3)

Броят на входовете и изходите е еднакъв (3)

Линейни неврони (няма нелинейна активационна функция)





Последователни автоенкодери

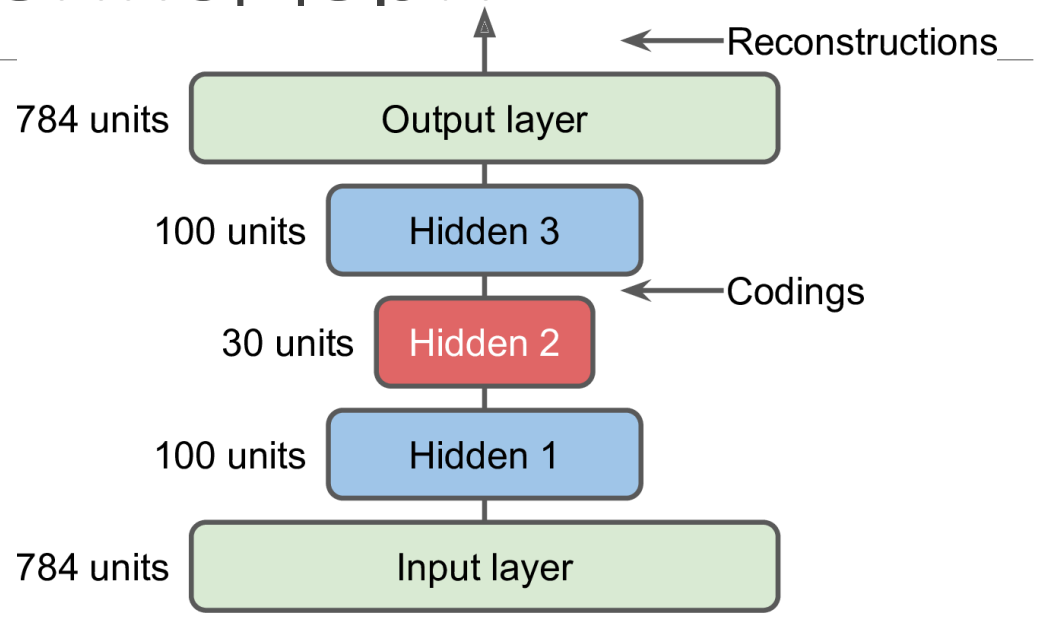
На входа задаваме картинки 28x28=784 пиксела от базата данни MNIST

Симетрия относително централния скрит (кодиращ) слой

Последователно намаляване на размерността („bottleneck”)

Обучени на всички слоеве заедно

Достигната „компресия“ 784→30



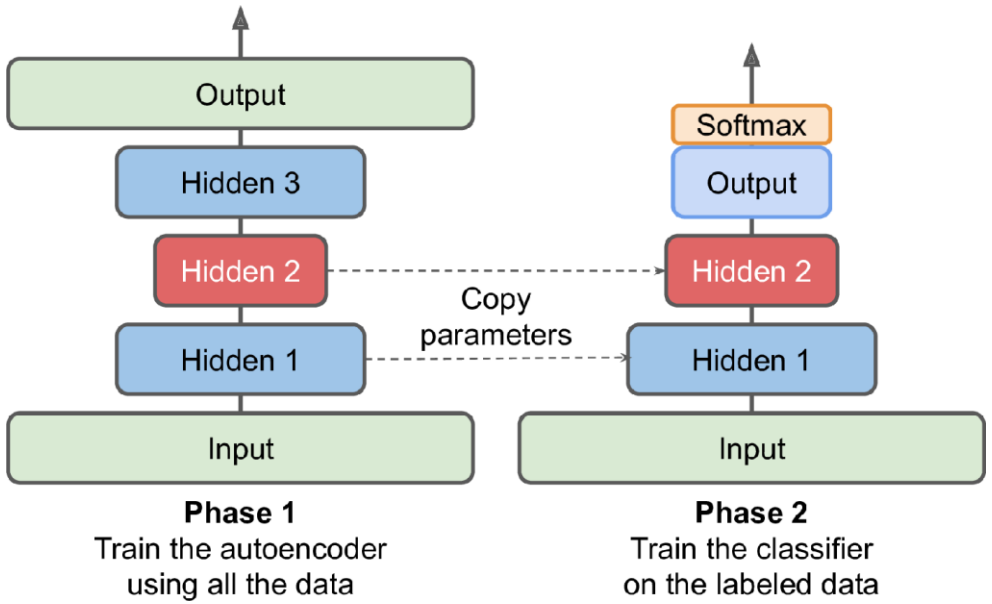
Използване на последователни автоенкодери за предварително обучение на дълбоки мрежи

Трениране автоенкодер на базата на данни без етикети

Копираме параметрите от обучението в първите слоеве на мрежата: така задаваме начално приближения

Продължаваме да обучаваме мрежата с учител (данни с етикети)

В последните години този метод губи популярност и е изместен от случайна инициализация



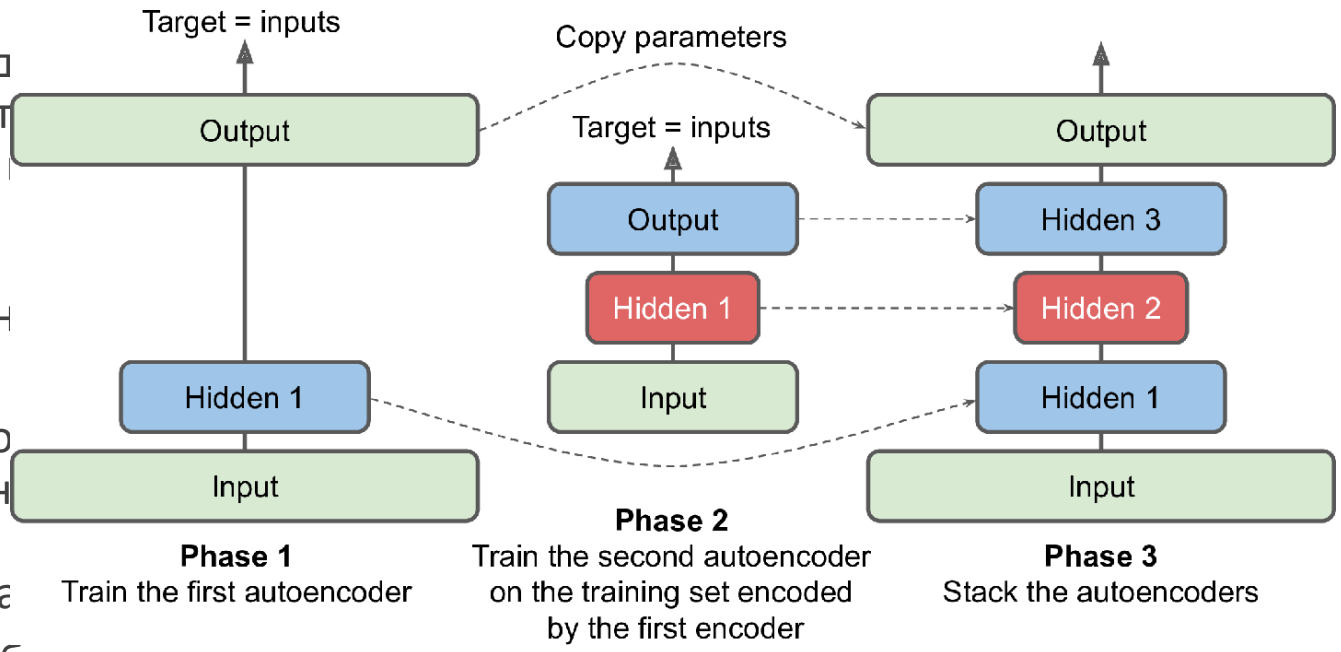
Полезни трикове

Отчитане на симетрията

- Ако автоенкодерът е симетричен, има възможност теглата от енкодера да бъд приравнени на теглата в декодера. По т начин броят на параметрите намалява двойно

Последователно трениране на вложен автоенкодери

- Отначало трениране най-външните сло след това използваме кодираните данн тренировка на следващите слоеве и продължаваме към средата на мрежата
- Това е особено полезно при много дълбока архитектура

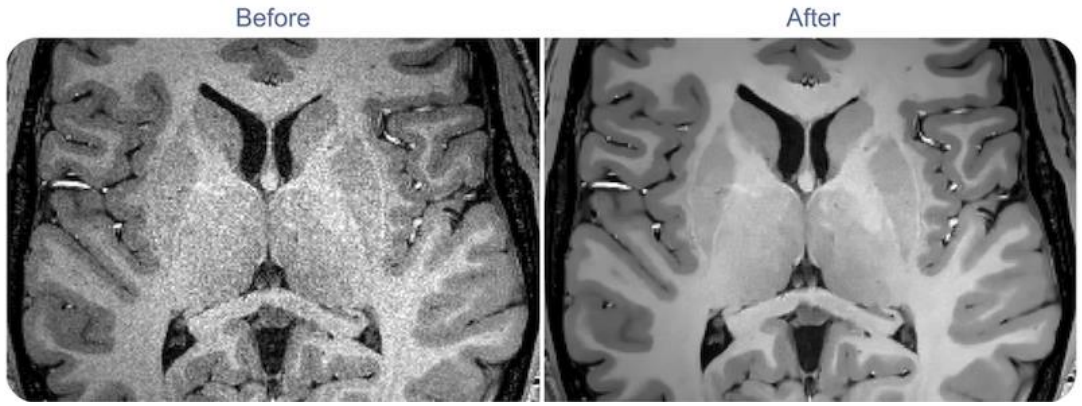


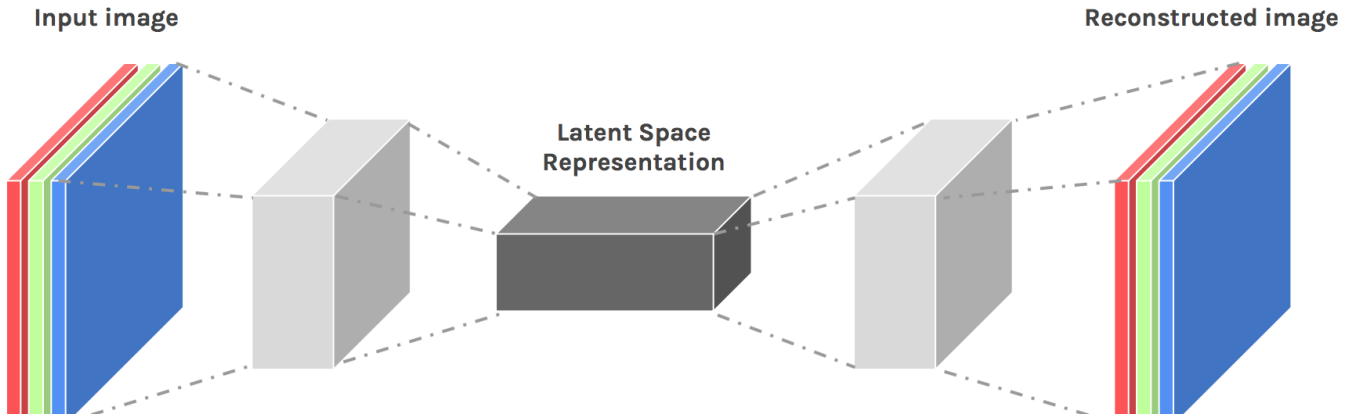
Конволюционни автоенкодери

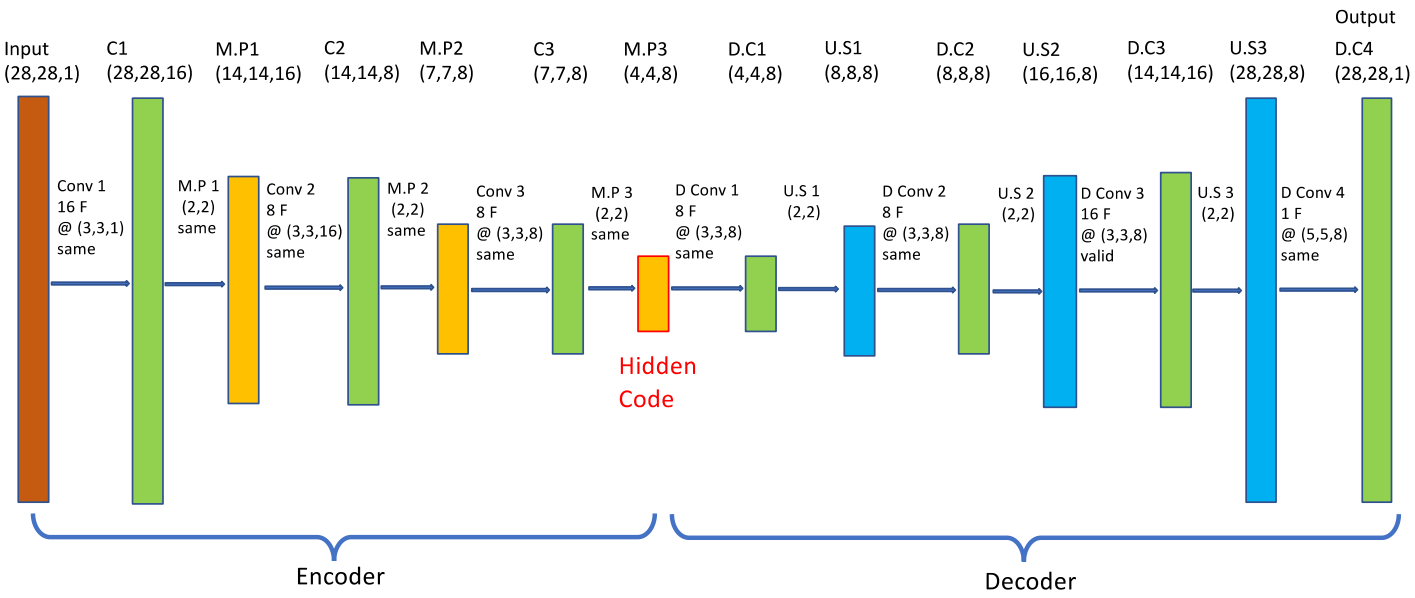
Много подходящи за обработка на изображения.

Автоенкодерът е обикновена конволюционна невронна мрежа с конволюционни и обединяващи (pooling) слоеве.

Намалява размерността за сметка на дълбочината.





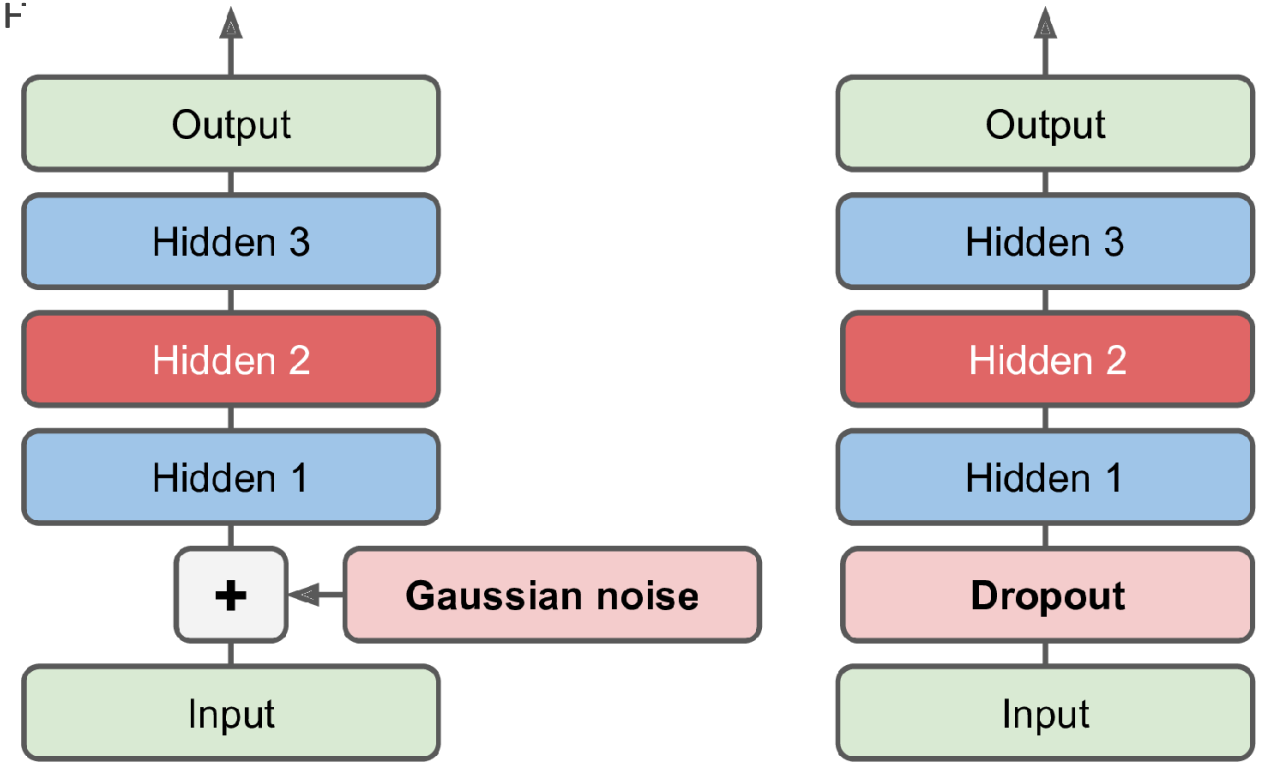


Обозначаващ (denoising) автоенкодер

Представлява обикновен последователен автоенкодер, но информацията се модифицира:

- Чрез добавяне на шум (например гаусов)
- Чрез отпадане на информация (dropout)

Обучението се извършва върху оригиналната информация



Предимства и недостатъци на AE

ПРЕДИМСТВА

Обучение без учител, данни без етикети.

Сравнително бързо обучение.

Лесна интерпретация и визуализация на резултатите.

Работи добре с непрекъснати и дискретни данни.

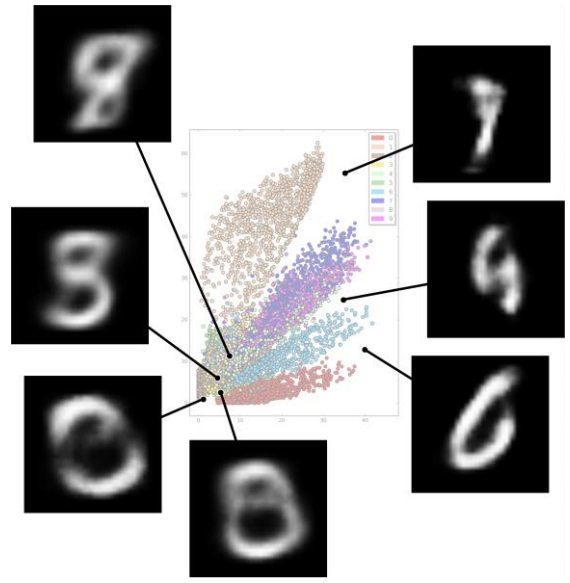
Помагат при обучение на по-сложни мрежи

НЕДОСТАТЪЦИ

Опасност от прекалено нагаждане.

Понякога в латентното пространство (пространството на кода) се наблюдават:

- Празнини (дупки)
- Припокриване на класовете
- Дискретизация



Вариационни Автоенкодери (VAE)

Lectures on VAE and GAN:

Yann LeCun & Alfredo Canziani, Training VAE and Training GANs

Stanford: CS231n: Convolutional Neural Networks for Visual Recognition

Generative models

Roger Grosse, Intro to Neural Networks and Machine Learning

- GANs

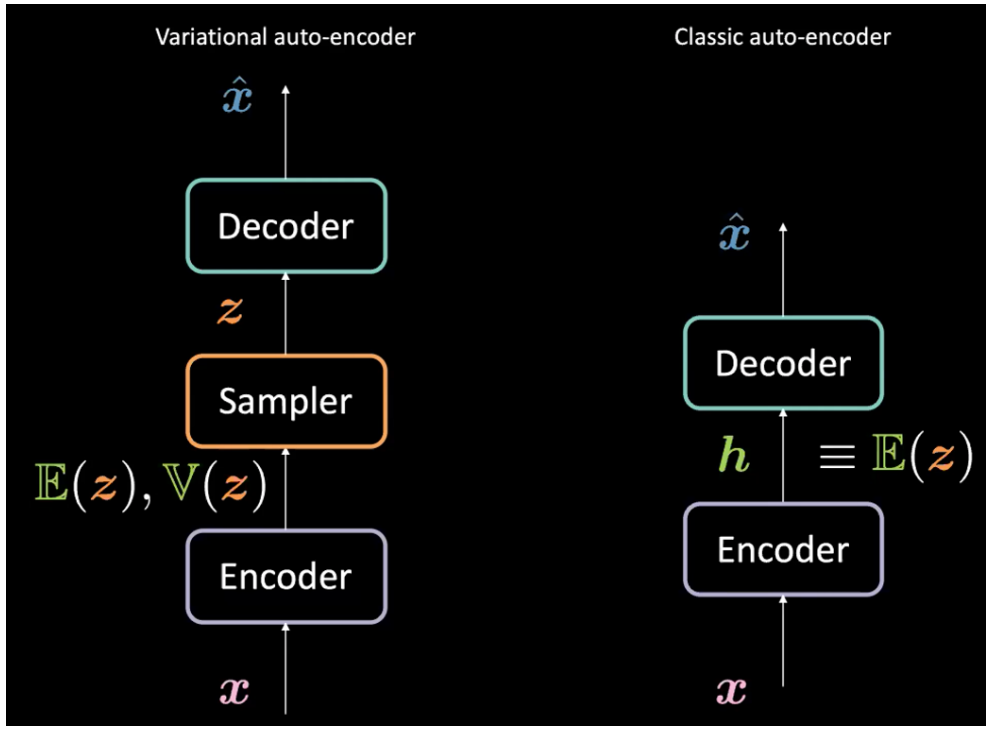
Вариационни автоенкодери и обикновени автоенкодери

ВА са предложени през 2013

- Вероятностни енкодери, изходът частично зависи от случайни променливи;
- Генеративни АЕ – дават възможност за създаване на изход, приличащ на истински входове

ВА съдържат в кодиращия слой не само значенията (обикновен АЕ), но също така и асоциираната с тях вариация

- Обикновено кодиращият слой на ВА използва гаусово разпределение, но може и друго.
- Обикновено два параметъра: средно **E(z)=μ** и вариация **V(z)=σ²**



Вариационни автоенкодери

Сандвич от енкодери и декодери

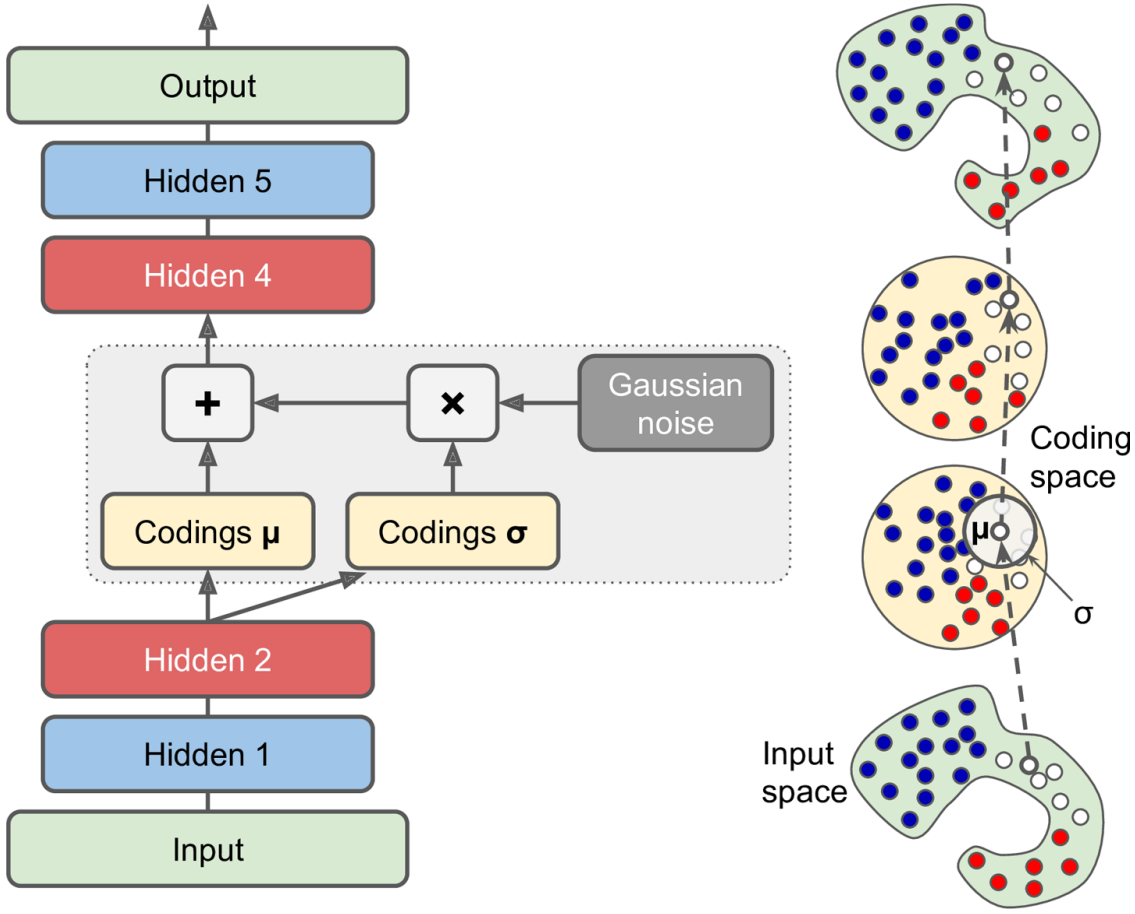
Кодиращият слой генерира средно **μ** и стандартно отклонение **σ**

На базата на средното и стандартното отклонение семплерът генерира случаен код за текущия пример

Декодерът работи като в обикновения AE

При тренировката функцията на загубите кара кодовете да мигрират и да образуват гаусови клъстери

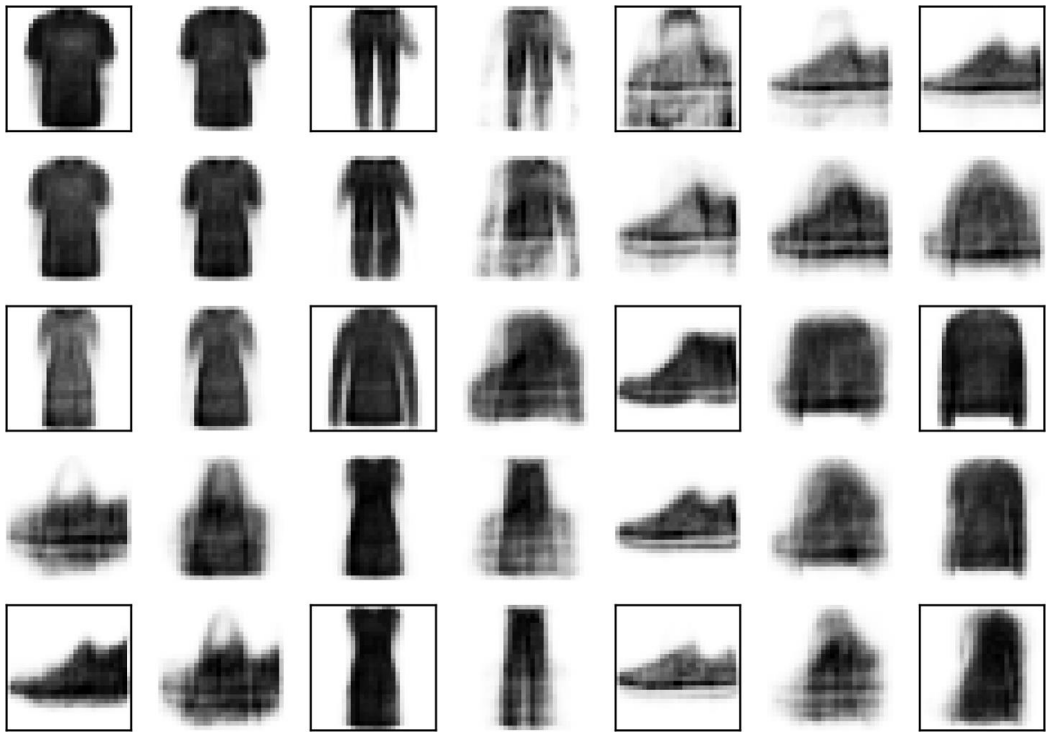
Достатъчно е да генерираме кодове от обученото разпределение, за да получим нови изходи.



Генерирани изображения

Семантична интерполация:
Оригинални (в рамка) и генерирани изображения.

Вместо на ниво входни пиксели, използваме генерирания код за две изображения, усредняваме и декодираме

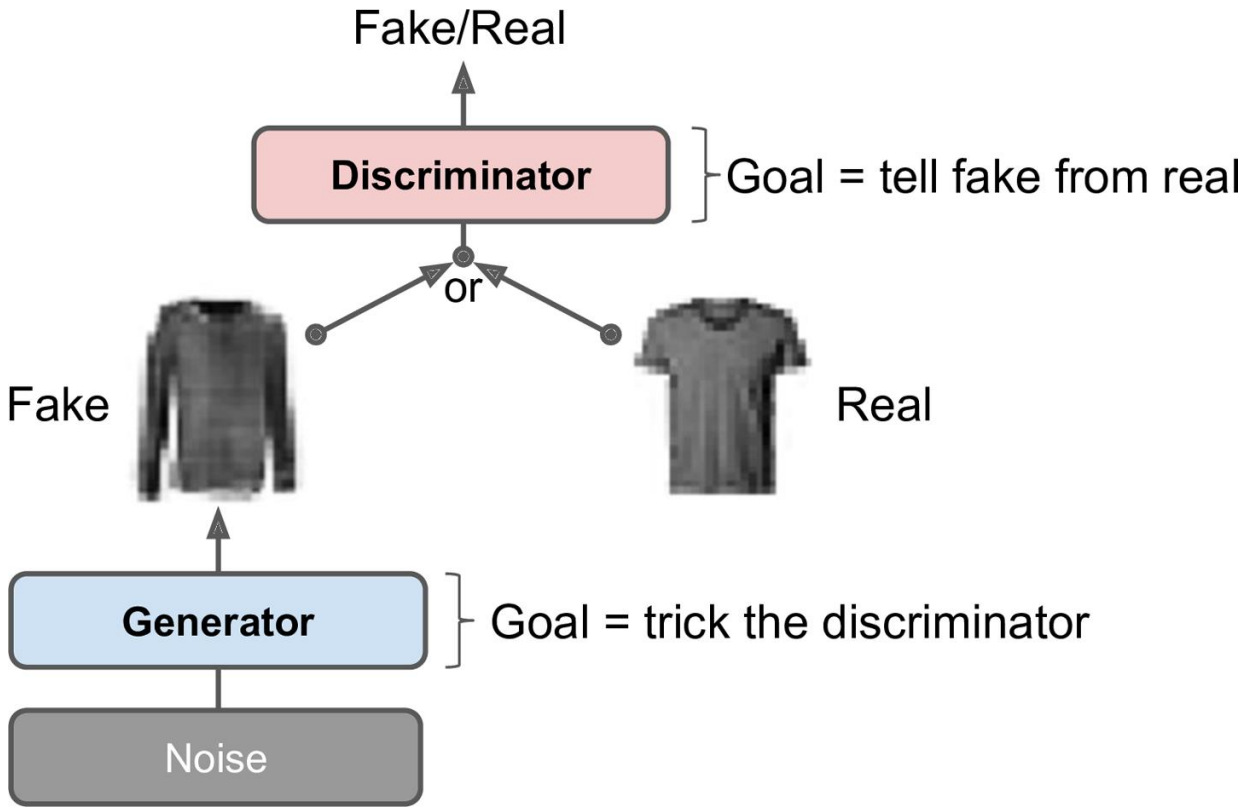


Генеративни състезателни мрежи (GAN)

Обща информация

GAN са предложени през 2014. Идеята е проста: две невронни мрежи се състезават помежду си, като всяка се усъвършенства в процеса на състезанието:

- Генераторът получава на входа случайно разпределение (кодирано изображение в термините на ВА) и генерира изображение. Задачата му е да създаде максимално реалистично изображение за всеки случаен вход.
- Дискриминаторът получава на входа си истински и фалшиви изображения. Задачата му е да ги раздели.



Особености на обучението на GAN: две мрежи, две повтарящи се фази

1. ОБУЧЕНИЕ НА ДИСКРИМИНАТОРА

Генераторът произвежда фалшиви изображения (етикет 0)

Допълваме със същия брой истински изображения (етикет 1)

Обучаваме дискриминатора да класифицира истински и фалшиви. Функцията на загубите е кръстосана ентропия.

Обратното разпространение обновява само теглата в дискриминатора.

2. ОБУЧЕНИЕ НА ГЕНЕРАТОРА

Генерираме нов масив изображения с етикет 1 (лъжем дискриминатора). Не използваме истински картинки.

Обучаваме генератора да произвежда изображения, които дискриминаторът разпознава като истински. Функциата на загубите е кръстосана ентропия.

Обратното разпространение обновява само теглата на генератора.

Трудности при обучението на GAN

След известен брой итерации резултатите спират да се подобряват.

- Системата достига равновесна точка на Наш от теория на игрите, когато смяна на стратегията не носи печалба на играч, ако другите не сменят своята стратегия.
- Теоретически резултат: мрежата спира в точка на Наш, където генераторът прави перфектни изображения, а дискриминаторът ги бракува с вероятност 50%.
- Практически спира далеч от тази точка…

Колапс на варианти (mode collapse):

- Наблюдава се, когато сред всички варианти (типове изображения) генераторът специализира в един конкретен тип, а дискриминаторът подсилва тази специализация. Възможна е и осцилация между варианти.
- Начини за борба:
  - Повторение на опит (experience replay), когато част от генерираните изображения се съхраняват и използват при следваща итерация.
  - Изхвърляне на итерации (mini-batch discrimination), измерва се близостта между последователни итерации и прекалено близките генерирани данни се изхвърлят.

Дълбоки конволюционни GAN

Рецепта, 2015:

В дискриминатора заменете обединяващите слоеве конволюционни.

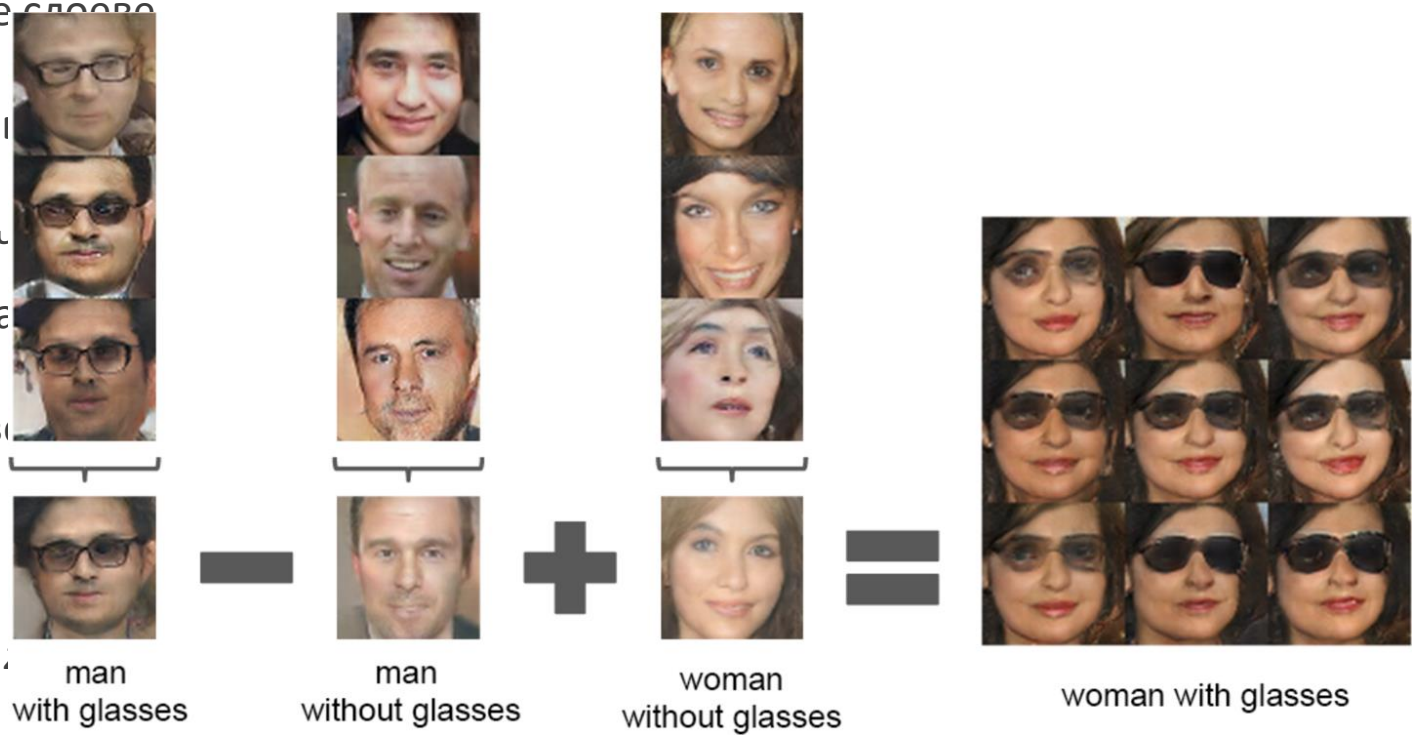
В генератора заменете обединяващите слоеве транспонирани конволюционни слоеве.

Използвайте групова нормализация във всич слоеве на дискриминатора и генератора, с изключение на изхода на генератора и входа дискриминатора.

Премахнете всички напълно свързани слоеве дълбоката архитектура.

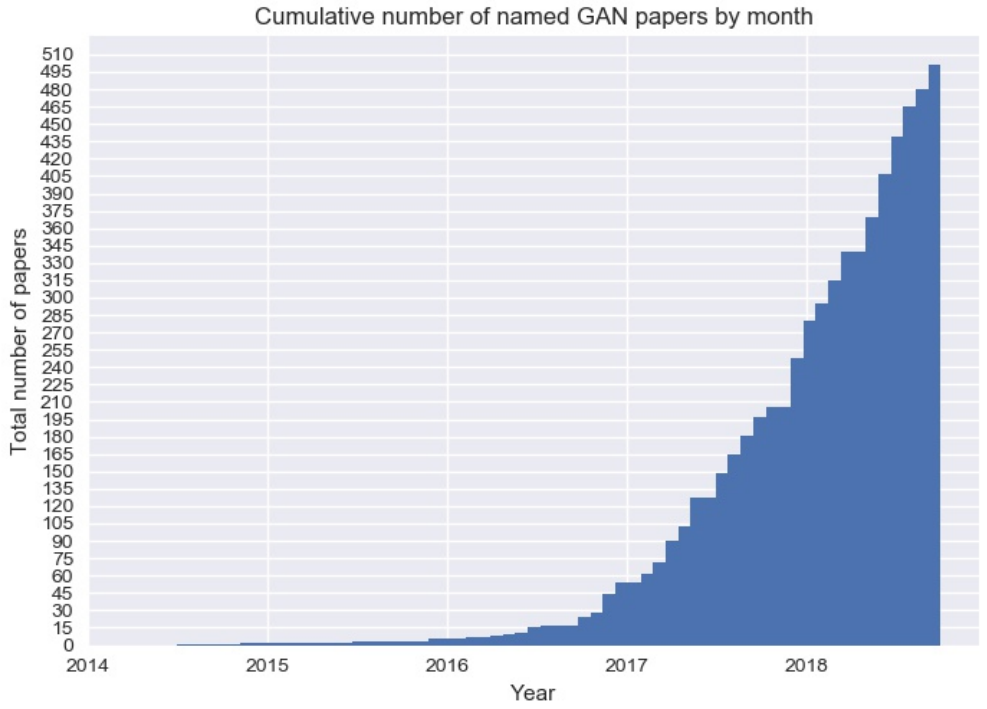
За изхода на генератора използвайте tanh-активация, за останалите: ReLU.

За дискриминатора: $\text{LeakyReLU}_\alpha(z) = \max(\alpha z, z)$ активация



https://arxiv.org/pdf/1511.06434.pdf

Смайващо разнообразие на GAN: https://github.com/hindupuravinash/the-gan-zoo



Рекурентни Невронни Мрежи (RNN)

Лекции:

Yann LeCun & Alfredo Canziani

RNNs, GRUs, LSTMs...

Stanford: CS231n: Convolutional Neural Networks for Visual Recognition

RNN, LSTM

RNN: общи сведения

Конволюционните невронни мрежи се използват за обработки обработка на „мрежа“ от значения **X,** например изображения.

Рекуррентните невронни мрежи са предназначена за обработка на последователности от значения $\boldsymbol{x}(1), ..., \boldsymbol{x}(t)$ с произволна дължина, например времеви редове.

- Можем да говорим за време ***t*** или за елемент от последователност ***t***

Основна идея: RNN имат вътрешно състояние, което се обновява при обработката

Примери за използване на рекурентни невронни мрежи:

- Преобразуване на реч в текст и на текст в реч;
- Анотация на картинки;
- Автоматичен превод;
- Предсказание на борсови цени;
- Автоматично управление на автомобил.

Приложения във физиката

- Предсказания на траекториите на хаотична динамична система
- Метеорологични прогнози
- Идентификация на частици
- Физика на струи

Прост пример: рекурентни неврони и слоеве

Рекурентните неврони имат „обратна връзка“ и описват състоянието на динамична система. Част от информацията от изхода се подава на входа => системата има „памет“ за предното състояние. Използват се в разгънат по време вид.

$s_t = f_W(s_{t-1}, x_t)$: s_{t-1}, s_t – старо/ново състояние, x_t – входове, f_W – функция с параметри W

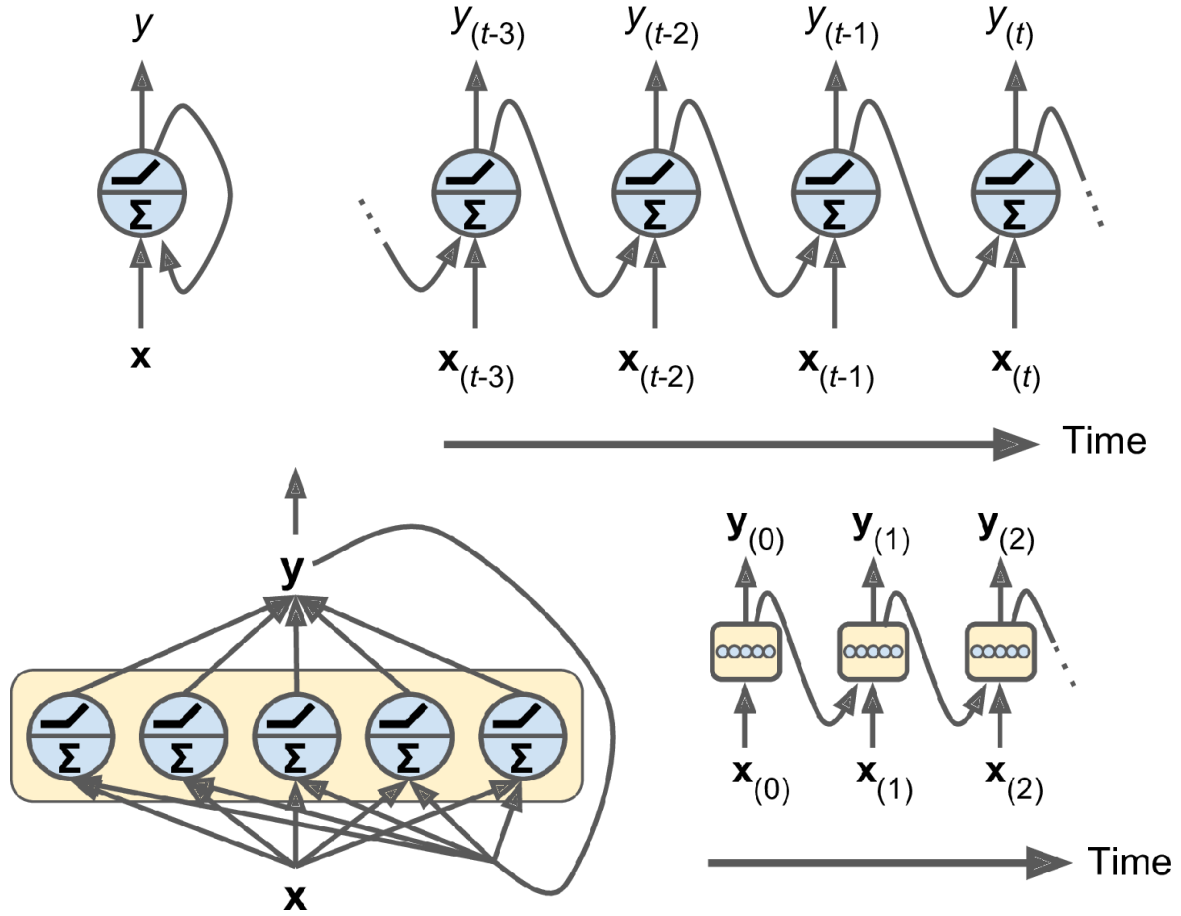
Рекурентните неврони обикновено се използват в слоеве.

Очевидни подобрения:

- Няколко слоя: дълбока мрежа
- Връзки между скритите слоеве: клетки на паметта

Сериозни подобрения:

- LSTM (Long Short-Term Memory)
- GRU (Gated Recurrent Units)



Архитектура на проста рекурентна невронна мрежа

Формално описание на RNN в момент ***t***

$a_t = b + Ws_{t-1} + Ux_t$, $s_t = f(a_t)$,

$o_t = c + Vs_t$, $y_t = h(o_t)$,

където

f – нелинейна активация (обикновено σ, tanh или ReLU)

h – функция за изчисление на изходите (например softmax)

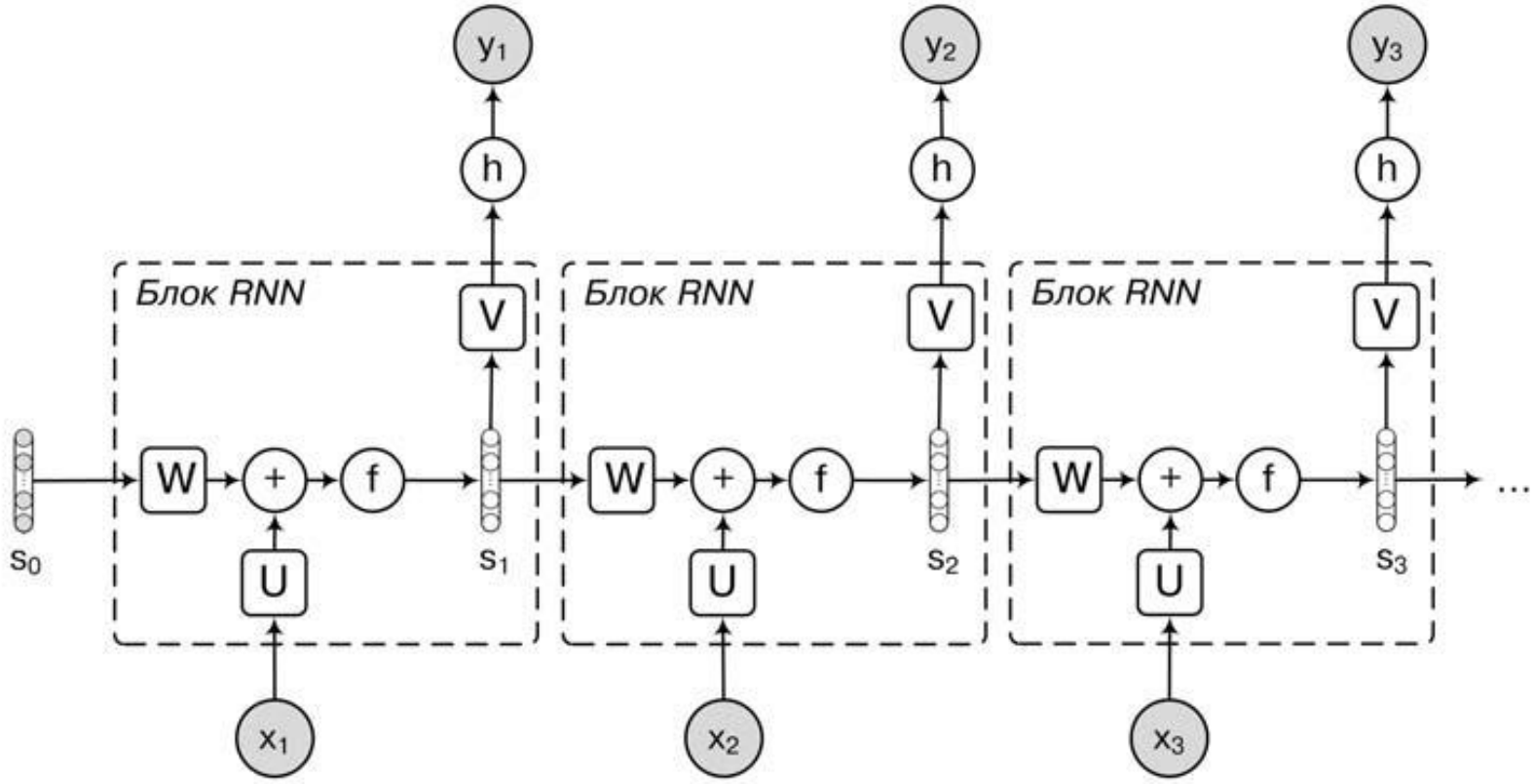
W – тегла на състоянието ***s***

U – тегла на входовете

V – тегла за изчисляване на изходите

c, b - отмествания

Важно: теглата са едни и същи за всеки момент от време!



Типове RNN в зависимост от преобразуването на информацията

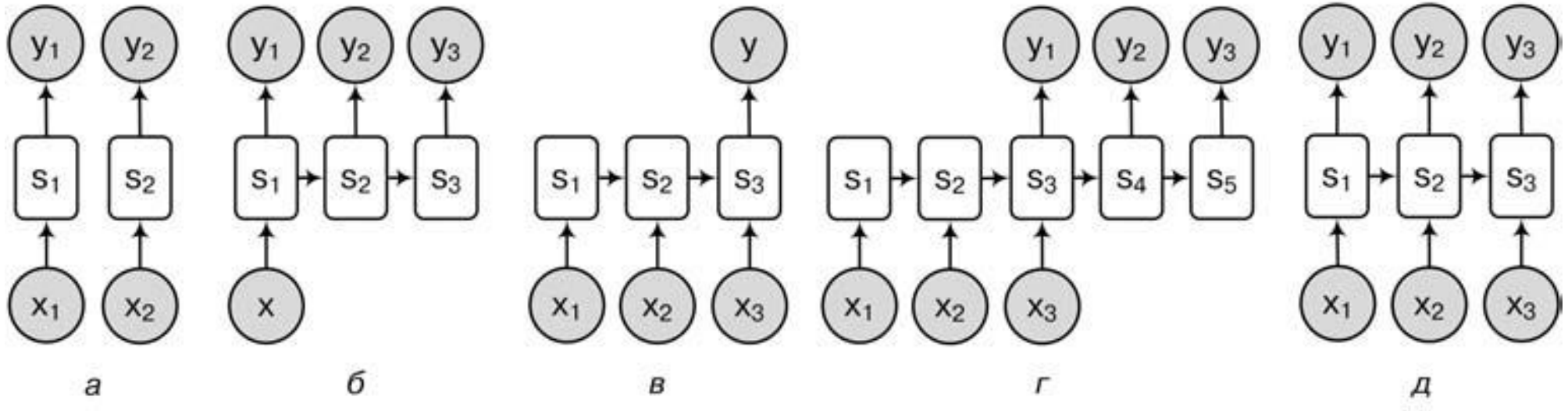
Един вход, един изход

Един вход, ред изходи

Ред входове, един изход

Ред входове, ред изходи

Синхронизирани редове от входове и изходи



Пример за „обикновена“ мрежа

Анотация на картинки: вход – картинка, изход – текст

Класификация на последователности: анализ на мнения (+,-)

Автоматичен превод, диалогови системи

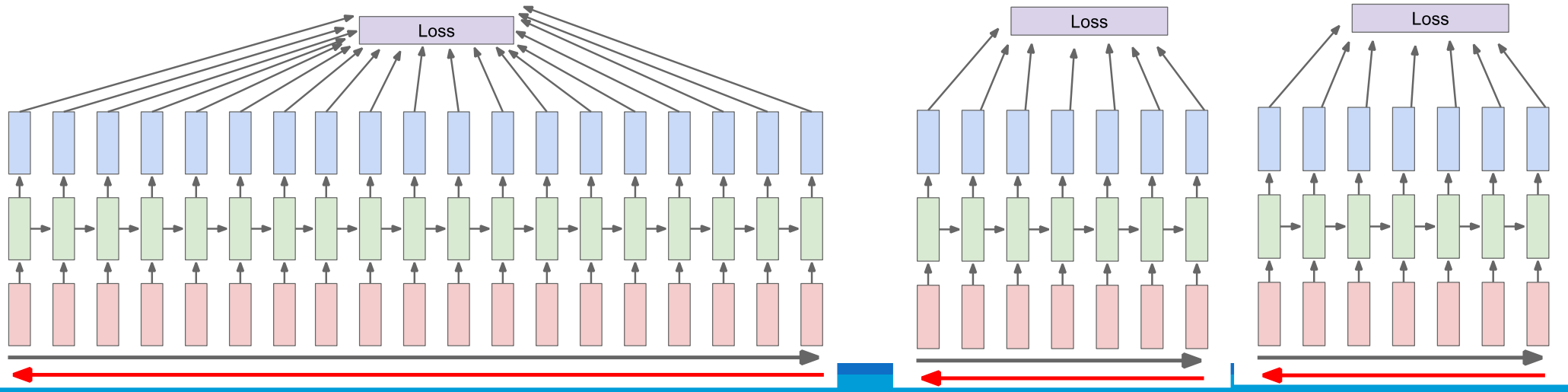
Анотация на видеокадри

Обучение на RNN

Изчисление на функцията на загубите: право разпространение

Изчисляване на градиентите: BPTT (back propagation through time), обратно разпространение във времето. На практика е обикновено обратно разпространение в разгънатата мрежа.

Вариант: право и обратно разпространение върху части от последователността



Stanford: CS231n

Трудности и полезни методи на обикновени RNN

НЕСТАБИЛНИ ГРАДИЕНТИ

Изчезващи градиенти. На всяка стъпка градиентите се умножават по теглата. Ако теглата са малки, градиентите намаляват експоненциално.

Експлозивни градиенти. Ако теглата са големи, а активацията е практически линейна, то градиентите растат експоненциално.

ПОЛЕЗНИ МЕТОДИ

„Подрязване“ (ограничаване по абс. стойност) на градиентите помага при експлозивни градиенти.

Специална инициализация на теглата, например като ортогонална случайна матрица (съхранява се нормата на векторите).

Нормировка в слоя преди активационната функция (центриране, единична дисперсия, скалиране).

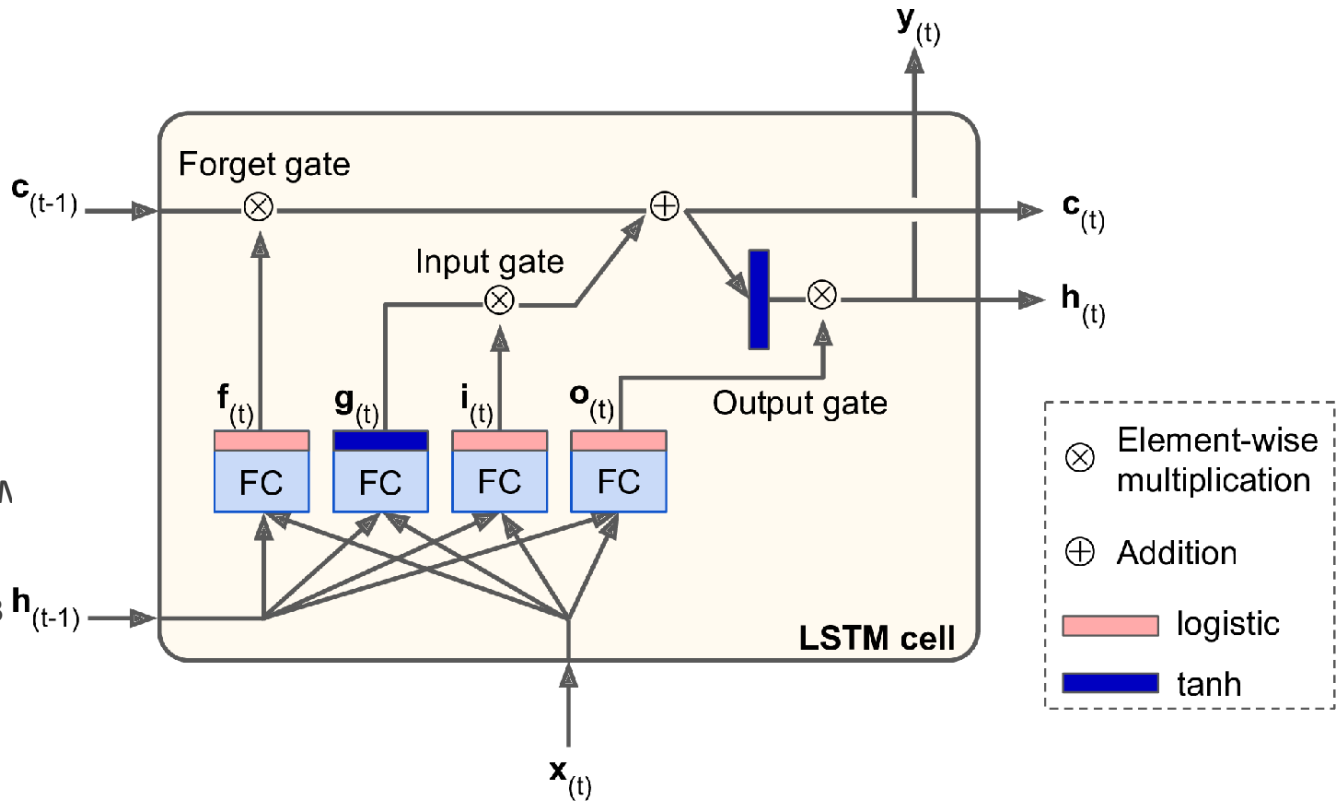
LSTM: Long Short-Term Memory

Състоянието се описва от два вектора:

- c_{t-1} – дългосрочно състояние (памет)
- h_{t-1} – краткосрочно състояние (памет)

Основна (***g***) и контролиращи (***f,i,o***) клетки мат. тегла, отместване)

- Забравяща (***f***): определя кои елементи на дългосрочната памет да бъдат „изтрити“
- Входна (***i***): определя какво да се добави към дългосрочната памет
- Изходна (***o***): контролира какво се записва в краткосрочната памет и отива към изхода



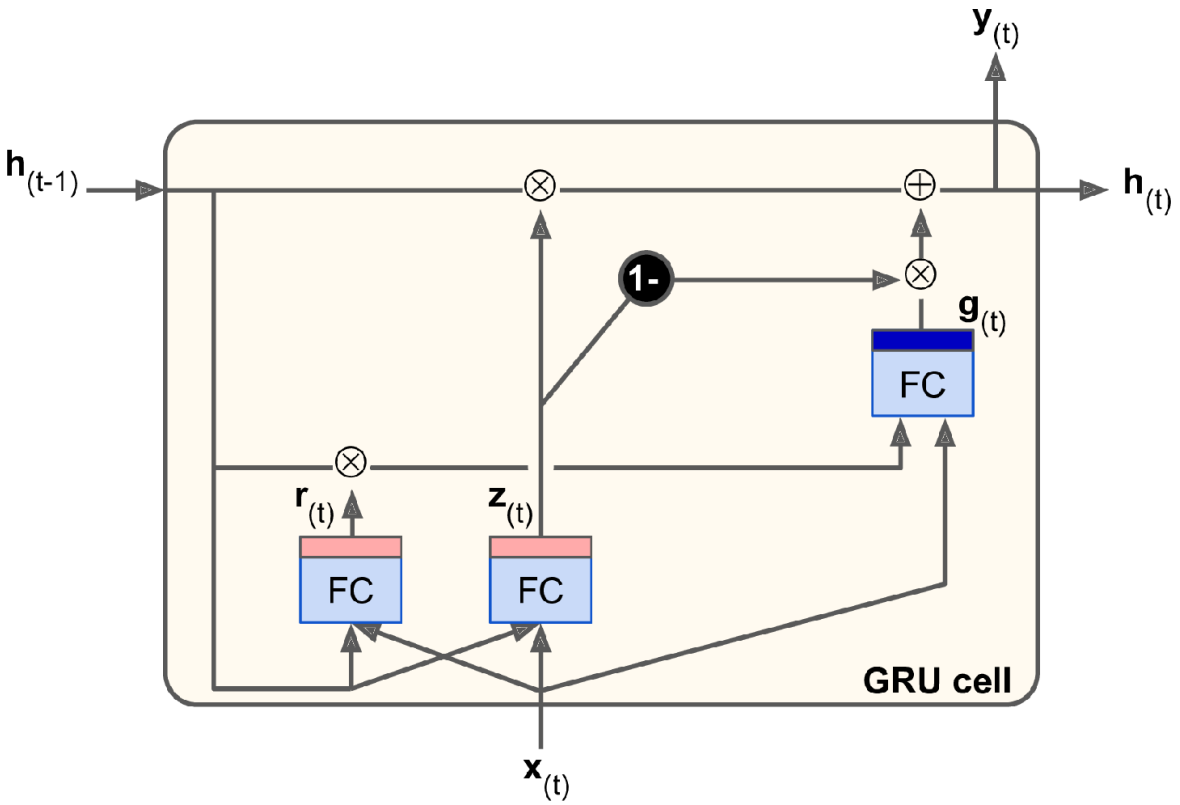
GRU

Опростена версия на LSTM

Един вектор на състояние (памет)

Основна клетка (***g***) и две контролиращи клетки (2 мат. тегла + отместване)

- Вход и забравяне (***z***): изтрива части от състоянието и ги замества с нова информация
- Състояние (***r***): определя кои части от състоянието се подават на входа на основната клетка



RNN

ПРЕДИМСТВА

Обработка на вход с произволна дължина

Възможност за използване на информацията от предходните стъпки

Фиксиран размер на мрежата за различни последователности

Фиксирани тегла за всички стъпки

НЕДОСТАТЪЦИ

Бавни за обучение и използване

Трудности при използване на прекалено много предходни стъпки

RNN: резюме

RNN предоставят гъвкави инструменти за обработка на последователности

Простите RNN не винаги работят добре

- Проблеми с нестабилните градиенти
- Изрязването на градиента, ортогоналната инициализация и нормализирането на слоя могат да помогнат

LSTM и GRU са най-популярните RNN:

- Добавянето на контролни потоци подобрява разпространението на градиента
- LSTM са добър начален избор, но може да са бавни
- GRU са по-бързи и имат по-малко параметри, но понякога дават по-лоши резултати

Активно търсене на нови RNN архитектури

- Необходимо е по-добро разбиране

Sources

Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, uses TensorFlow 2. This is my preferred textbook, very detailed and up-to-date. Jupyter notebooks. There is already third edition and the corresponding notebooks.

Machine learning for physicists, Florian Marquardt. Online lectures, lecture notes and slides, Jupyter notebooks, and a lot of information on the Web site. Last version from 2021. From the same author "**Online Course: Advanced Machine Learning for Physics, Science, and Artificial Scientific Discovery**"

Machine Learning and Deep Learning, Lara Lloret Iglesias, INFIERI School 2019, Wuhan. Introductory lecture and astrophysics lab.

AstroML: Machine Learning and Data Mining for Astronomy. Python package with textbook, user guide, Jupyter notebooks and a lot of information on the Web site.

CERN Academic Training, May 2022: A General Introduction to Machine Learning, with a twist towards accelerators