



142,92

Рейтинг

NIX Solutions

Компания

NIX_Solutions 24 июля в 11:06

Памятки по искусственному интеллекту, машинному обучению, глубокому обучению и большим данным

<https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Машинное обучение, Искусственный интеллект, Big Data, Блог компании NIX Solutions

[Перевод](#)**ИНФОРМАЦИЯ**

Дата основания 1994 год

Локация Харьков
УкраинаСайт nixsolutions.comЧисленность 1001–5000
человек

Дата регистрации 25 февраля 2015

FACEBOOK



В течение нескольких месяцев мы собирали памятки по искусственному интеллекту, которыми периодически делились с друзьями и коллегами. В последнее время сложилась целая коллекция, и мы добавили к памяткам описания и/или цитаты, чтобы было интереснее читать. А в конце вас ждёт подборка по сложности «О большое» (Big-O). Наслаждайтесь.

UPD. Многие картинки будут читабельнее, если открыть их в отдельных вкладках или сохранить на диск.

Нейронные сети

A mostly complete chart of

Neural Networks



TWITTER

nixsolutions 4 h

Есть в Никсах настоящие укротители, которых не пугает даже восстание машин 🤖. Они просто перезагрузят их, и те снов... <https://t.co/LaXMPW9ZRD>

nixsolutions yesterday

Чувствуете, что лето вас слишком расслабило? Добавьте в свою жизнь немного приключений и покорите новые горизонты в...
<https://t.co/d1vJZ3GtXO>

nixsolutions 3 d

Помните все эти сотни способов сделать незаметную шпаргалку? Хорошо, что во взрослой жизни все

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

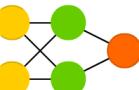
Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

Perceptron (P)



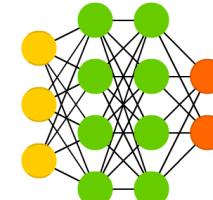
Feed Forward (FF)



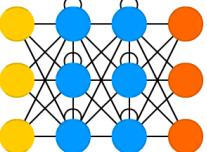
Radial Basis Network (RBF)



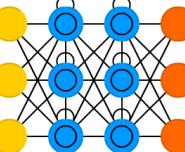
Deep Feed Forward (DFF)



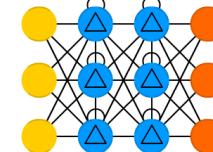
Recurrent Neural Network (RNN)



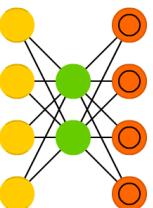
Long / Short Term Memory (LSTM)



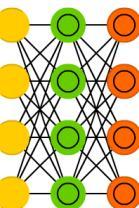
Gated Recurrent Unit (GRU)



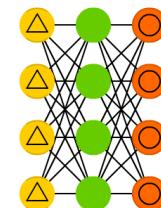
Auto Encoder (AE)



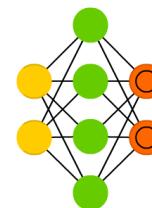
Variational AE (VAE)



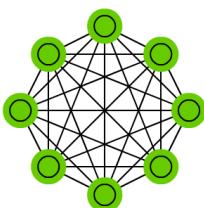
Denoising AE (DAE)



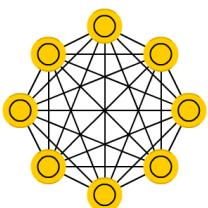
Sparse AE (SAE)



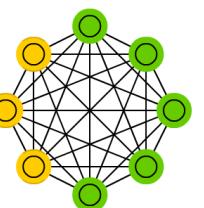
Markov Chain (MC)



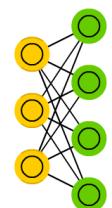
Hopfield Network (HN)



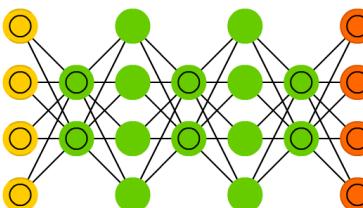
Boltzmann Machine (BM)



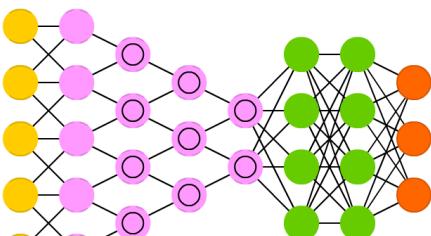
Restricted BM (RBM)



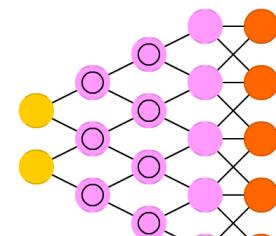
Deep Belief Network (DBN)



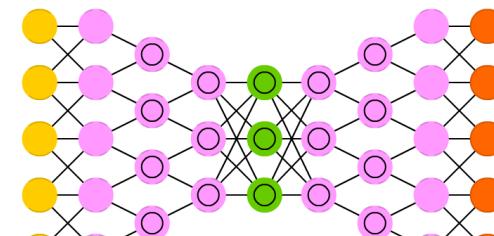
Deep Convolutional Network (DCN)



Deconvolutional Network (DN)



Deep Convolutional Inverse Graphics Network (DCIGN)



немного проще, и за...

<https://t.co/UC3LvubzPx>

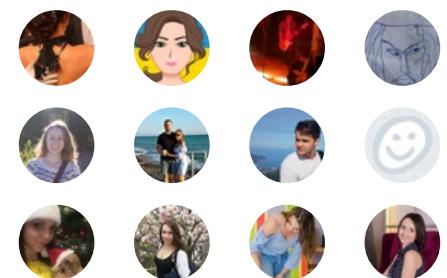
Читать @nixsolutions

ВКОНТАКТЕ



NIX Solutions

3,414 followers



[Follow on VK](#)

БЛОГ НА ХАБРЕ

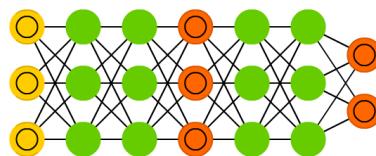
Памятки по искусственному интеллекту, машинному обучению, глубокому обучению и большим данным

11,1k

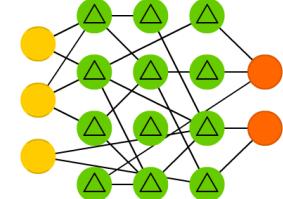
9



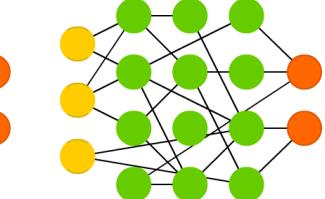
Generative Adversarial Network (GAN)



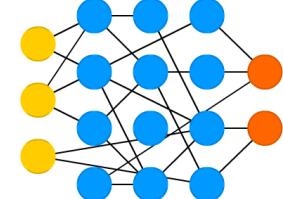
Liquid State Machine (LSM)



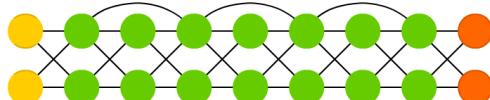
Extreme Learning Machine (ELM)



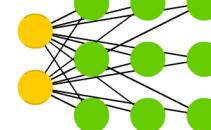
Echo State Network (ESN)



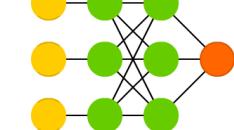
Deep Residual Network (DRN)



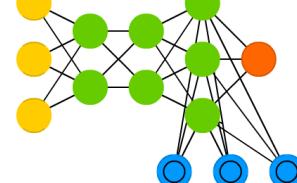
Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)



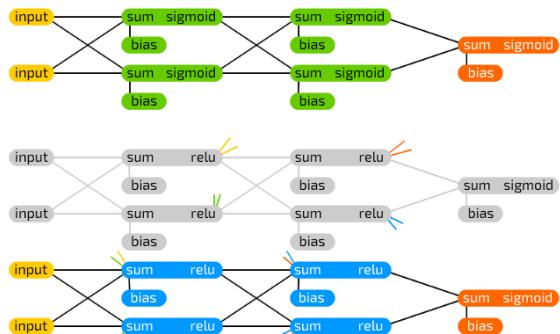
Памятка по нейронным сетям

Графы нейронных сетей

An informative chart to build

Neural Network Graphs

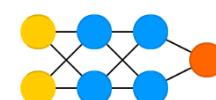
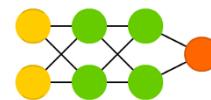
©2016 Fjodor van Veen - asimovinstitute.org



Deep Feed Forward Example

Deep Recurrent Example
(previous iteration)

Deep Recurrent Example



Как ИИ учится генерировать изображения кошек

8,3k 0

27 отличных open source-инструментов для веб-разработки

38,8k 10

Как в React избавиться от сложности в управлении состоянием — отчёт по итогам поездки на React Amsterdam

5,8k 3

Как машины анализируют большие данные: введение в алгоритмы кластеризации

7,3k 7

Я сделал PWA и выложил в трёх магазинах приложений. И вот что я выяснил

17,4k 58

Анонс митапа RubyRoars #1 в Харькове

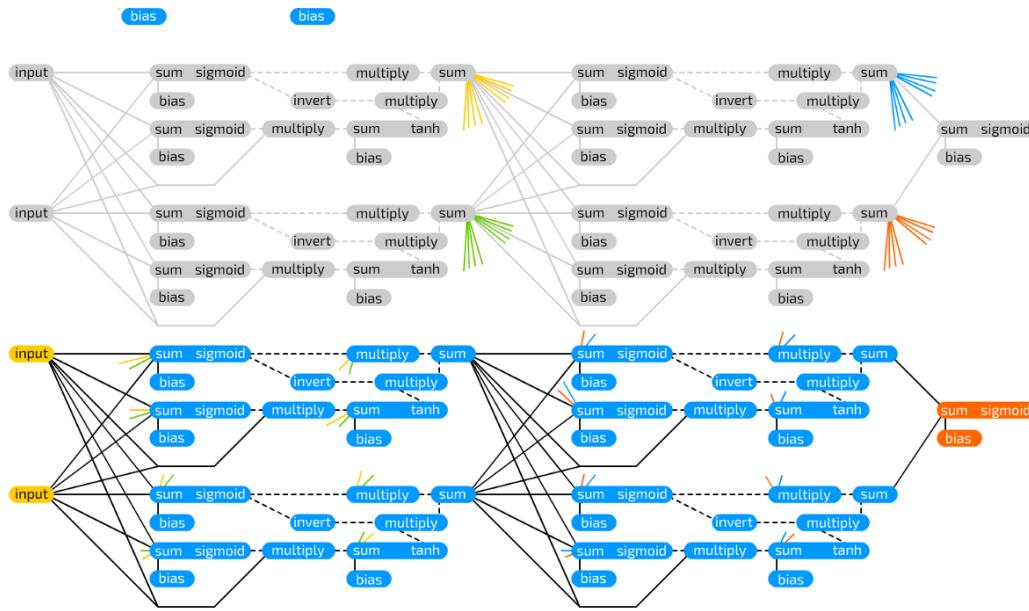
667 0

Анонс митапа Sync.NET #6 в Харькове

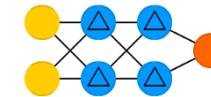
1,3k 2

Анонс митапа ThinkPHP #16 в Харькове

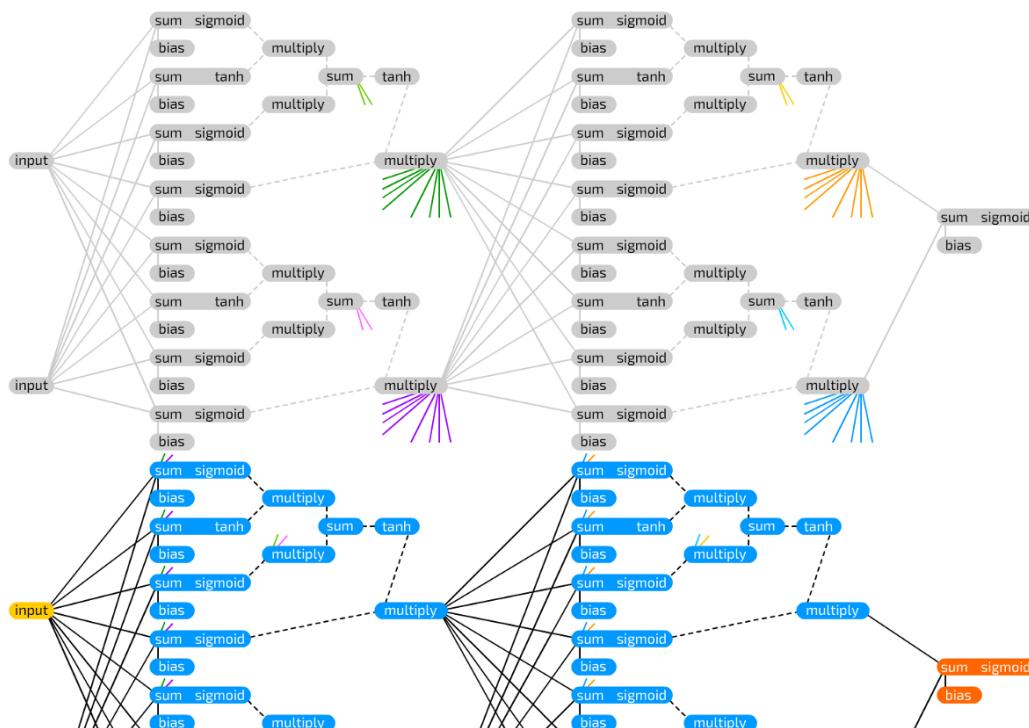
1,7k 0



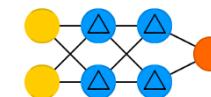
Deep GRU Example
(previous iteration)



Deep GRU Example



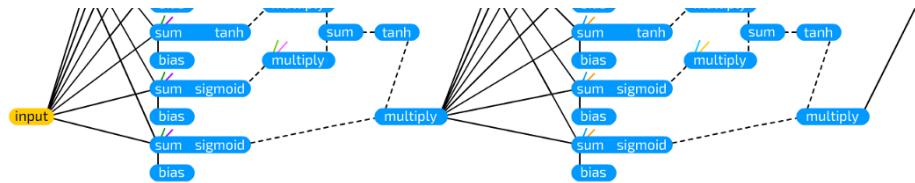
Deep LSTM Example
(previous iteration)



Deep LSTM Example

Как создать нейросеть всего из 30
строк JavaScript-кода

16,6k 10



Памятка по графикам нейронных сетей

Linear Vector Spaces:

Definition: A linear vector space, X , is a set of elements (vectors) defined over a scalar field, F , that satisfies the following conditions:

- 1) if $x \in X$ and $y \in X$ then $x+y \in X$.
- 2) $x+y=y+x$.
- 3) $(x+y)+z=x+(y+z)$.
- 4) There is a unique vector $0 \in X$, such that $x+0=x$ for all $x \in X$.
- 5) For each vector $x \in X$ there is a unique vector in X , to be called $(-x)$, such that $x+(-x)=0$.
- 6) multiplication, for all scalars $a \in F$, and all vectors $x \in X$,
- 7) For any $x \in X$, $1x=x$ (for scalar 1).
- 8) For any two scalars $a \in F$ and $b \in F$ and any $x \in X$, $a(bx)=(ab)x$.
- 9) $(a+b)x=a x+b x$.
- 10) $a(x+y)=a x+a y$.

Linear Independence: Consider n vectors $\{x_1, x_2, \dots, x_n\}$. If there exists n scalars a_1, a_2, \dots, a_n , at least one of which is nonzero, such that $a_1x_1 + a_2x_2 + \dots + a_nx_n = 0$, then the $\{x_i\}$ are linearly dependent.

Spanning a Space:

Let X be a linear vector space and let $\{u_1, u_2, \dots, u_n\}$ be a subset of vectors in X . This subset spans X if and only if for every vector $x \in X$ there exist scalars x_1, x_2, \dots, x_n such that $x = x_1u_1 + x_2u_2 + \dots + x_nu_n$.

Inner Product: $\langle x, y \rangle$ for any scalar function of x and y .

$$1. \langle x, y \rangle = \langle y, x \rangle \quad 2. \langle ax_1 + bx_2, y \rangle = a \langle x_1, y \rangle + b \langle x_2, y \rangle$$

$$3. \langle x, x \rangle \geq 0, \text{ where equality holds iff } x \text{ is the zero vector.}$$

Norm: A scalar function $\|x\|$ is called a norm if it satisfies:

1. $\|x\| \geq 0$
2. $\|x\| = 0$ if and only if $x = 0$.
3. $\|ax\| = |a|\|x\|$
4. $\|x + y\| \leq \|x\| + \|y\|$

Angle: The angle θ bet. 2 vectors x and y is defined by $\cos \theta = \frac{\langle x, y \rangle}{\|x\| \|y\|}$

Orthogonality: 2 vectors $x, y \in X$ are said to be orthogonal if $\langle x, y \rangle = 0$.

Gram Schmidt Orthogonalization:

Assume that we have n independent vectors y_1, y_2, \dots, y_n . From these vectors we will obtain n orthogonal vectors v_1, v_2, \dots, v_n .

$$v_1 = y_1, \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{\langle v_i, y_k \rangle}{\langle v_i, v_i \rangle} v_i,$$

where $\frac{\langle v_i, y_k \rangle}{\langle v_i, v_i \rangle} v_i$ is the projection of y_k on v_i

Vector Expansions:

$$x = \sum_{i=1}^n x_i v_i = x_1 v_1 + x_2 v_2 + \dots + x_n v_n,$$

Perceptron Architecture:

$$\mathbf{a} = \text{hardlim}(\mathbf{W}\mathbf{p} + \mathbf{b}), \quad \mathbf{W} = [\mathbf{w}_1^T \ \mathbf{w}_2^T \ \dots \ \mathbf{w}_S^T]^T, \\ a_i = \text{hardlim}(n_i) = \text{hardlim}(\mathbf{w}_i^T \mathbf{p} + b_i)$$

$$\text{Decision Boundary: } \mathbf{w}^T \mathbf{p} + b_i = 0$$

The decision boundary is always orthogonal to the weight vector.

Single-layer perceptrons can only classify linearly separable vectors.

Perceptron Learning Rule

$$\mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} + \mathbf{e} \mathbf{p}^T, \quad \mathbf{b}^{\text{new}} = \mathbf{b}^{\text{old}} + \mathbf{e}, \\ \text{where } \mathbf{e} = \mathbf{t} - \mathbf{a},$$

Hebb's Postulate: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

Linear Associator: $\mathbf{a} = \text{purelin}(\mathbf{W}\mathbf{p})$

$$\text{The Hebb Rule: Supervised Form: } w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + t_{qi} P_{qi} \\ \mathbf{W} = \mathbf{t}_1 \mathbf{P}_1^T + \mathbf{t}_2 \mathbf{P}_2^T + \dots + \mathbf{t}_Q \mathbf{P}_Q^T$$

$$\mathbf{W} = [\mathbf{t}_1 \ \mathbf{t}_2 \ \dots \ \mathbf{t}_Q] \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \\ \vdots \\ \mathbf{P}_Q^T \end{bmatrix} = \mathbf{T} \mathbf{P}^T$$

Pseudoinverse Rule: $\mathbf{W} = \mathbf{T} \mathbf{P}^+$

When the number, R , of rows of \mathbf{P} is greater than the number of columns, Q , of \mathbf{P} and the columns of \mathbf{P} are independent, then the pseudoinverse can be computed by $\mathbf{P}^+ = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T$

Variations of Hebbian Learning:

$$\text{Filtered Learning (Ch.14): } \mathbf{W}^{\text{new}} = (1 - \gamma) \mathbf{W}^{\text{old}} + \alpha \mathbf{t}_q \mathbf{p}_q^T$$

$$\text{Delta Rule (Ch.10): } \mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} + \alpha (\mathbf{t}_q - \mathbf{a}_q) \mathbf{p}_q^T$$

$$\text{Unsupervised Hebb (Ch.13): } \mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} + \alpha \mathbf{a}_q \mathbf{p}_q^T$$

$$\text{Taylor: } F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \\ \frac{1}{2} (\mathbf{x} - \mathbf{x}^*) \nabla^2 F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots$$

for orthogonal vectors, $x_j = \frac{(v_j, x)}{(v_j, v_j)}$

Reciprocal Basis Vectors:

$$(r_i, v_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}, \quad x_j = (r_j, x)$$

To compute the reciprocal basis vectors: set $\mathbf{B} = [v_1 \ v_2 \ \dots \ v_n]$,

$\mathbf{R} = [r_1 \ r_2 \ \dots \ r_n]$, $\mathbf{R}^T = \mathbf{B}^{-1}$ In matrix form: $\mathbf{x}^v = \mathbf{B}^{-1} \mathbf{x}^s$

Transformations:

A *transformation* consists of three parts:

domain: $X = \{x_i\}$, range: $Y = \{y_i\}$, and a rule relating each $x_i \in X$ to an element $y_i \in Y$.

Linear Transformations: transformation A is *linear* if:

1. for all $x_1, x_2 \in X$, $A(x_1 + x_2) = A(x_1) + A(x_2)$
2. for all $x \in X, a \in R$, $A(ax) = aA(x)$

Matrix Representations:

Let $\{v_1, v_2, \dots, v_n\}$ be a basis for vector space X , and let $\{u_1, u_2, \dots, u_n\}$ be a basis for vector space Y . Let A be a linear transformation with domain X and range Y : $A(x) = y$

The coefficients of the matrix representation are obtained from

$$A(v_j) = \sum_{i=1}^m a_{ij} u_i$$

Change of Basis: $\mathbf{B}_t = [t_1 \ t_2 \ \dots \ t_n]$, $\mathbf{B}_w = [w_1 \ w_2 \ \dots \ w_n]$
 $A' = [\mathbf{B}_w^{-1} \mathbf{A} \mathbf{B}_t]$

Eigenvalues & Eigenvectors: $Az = \lambda z$, $|(A - \lambda I)| = 0$

Diagonalization: $\mathbf{B} = [z_1 \ z_2 \ \dots \ z_n]$,

where $\{z_1, z_2, \dots, z_n\}$ are the eigenvectors of a square matrix A ,
 $[\mathbf{B}^{-1} \mathbf{A} \mathbf{B}] = \text{diag}([\lambda_1 \ \lambda_2 \ \dots \ \lambda_n])$

$$\underline{\text{Grad}} \nabla F(\mathbf{x}) = \left[\frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \frac{\partial}{\partial x_2} F(\mathbf{x}) \quad \dots \quad \frac{\partial}{\partial x_n} F(\mathbf{x}) \right]^T$$

Hessian: $\nabla^2 F(\mathbf{x}) =$

$$\begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_2 \partial x_n} F(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix}$$

Directional Derivatives:

$$1^{\text{st}} \text{ Dir.Der.: } \frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}, \quad 2^{\text{nd}} \text{ Dir.Der.: } \frac{\mathbf{p}^T \nabla^2 F(\mathbf{x}) \mathbf{p}}{\|\mathbf{p}\|^2}$$

Minima:

Strong Minimum: if a scalar $\delta > 0$ exists, such that $F(x) < F(x + \Delta x)$ for all Δx such that $\delta > \|\Delta x\| > 0$.

Global Minimum: if $F(x) < F(x + \Delta x)$ for all $\Delta x \neq 0$

Weak Minimum: if it is not a strong minimum, and a scalar $\delta > 0$ exists, such that $F(x) \leq F(x + \Delta x)$ for all Δx such that $\delta > \|\Delta x\| > 0$.

Necessary Conditions for Optimality:

1st-Order Condition: $\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = 0$ (Stationary Points)

2nd-Order Condition: $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \geq 0$ (Positive Semi-definite Hessian Matrix).

Quadratic fn.: $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}, \quad \nabla^2 F(\mathbf{x}) = \mathbf{A}, \quad \lambda_{min} \leq \frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2} \leq \lambda_{max}$$

Памятка по нейронным сетям

Обзор по машинному обучению

MACHINE LEARNING IN EMOJI

SUPERVISED UNSUPERVISED REINFORCEMENT

SUPERVISED human builds model based on input / output
UNSUPERVISED human input, machine output
 human utilizes if satisfactory

CLUSTER ANALYSIS

K-MEANS

cluster.KMeans()

 REINFORCEMENT human input, machine output
human reward/punish, cycle continues

BASIC REGRESSION

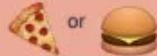
LINEAR

Lots of numerical data



LOGISTIC

Target variable is categorical



linear_model.LinearRegression()
linear_model.LogisticRegression()

CLASSIFICATION

NEURAL NET

neural_network.MLPClassifier()

Complex relationships. Prone to overfitting

Basically magic.



K-NN

neighbors.KNeighborsClassifier()

Group membership based on proximity



DECISION TREE

tree.DecisionTreeClassifier()

If/then/else. Non-contiguous data

Can also be regression



RANDOM FOREST

ensemble.RandomForestClassifier()

Find best split randomly

Can also be regression



SVM

svm.SVC() svm.LinearSVC()

Maximum margin classifier. Fundamental

Data Science algorithm



NAIVE BAYES

GaussianNB() MultinomialNB() BernoulliNB()

Updating knowledge step by step with new info



 Similar datum into groups
based on centroids


ANOMALY DETECTION

Finding outliers
through grouping

covariance.
EllipticalEnvelope()



FEATURE REDUCTION

T-DISTRIB STOCHASTIC NEIGH EMBEDDING

manifold.TSNE()

Visualize high dimensional data. Convert
similarity to joint probabilities



PRINCIPLE COMPONENT ANALYSIS

decomposition.PCA()

Distill feature space into components that
describe greatest variance



CANONICAL CORRELATION ANALYSIS

decomposition.CCA()

Making sense of cross-correlation
matrices



LINEAR DISCRIMINANT ANALYSIS

linalg.LDA()

Linear combination of features that
separates classes



OTHER IMPORTANT CONCEPTS



BIAS VARIANCE TRADEOFF



UNDERFITTING / OVERFITTING



INERTIA

$$(TP + TN) / (P + N)$$



ACCURACY FUNCTION

$$TP / (TP + FP)$$



Precision Function

$$TN / (FP + TN)$$



SPECIFICITY FUNCTION

$$TP / (TP + FN)$$

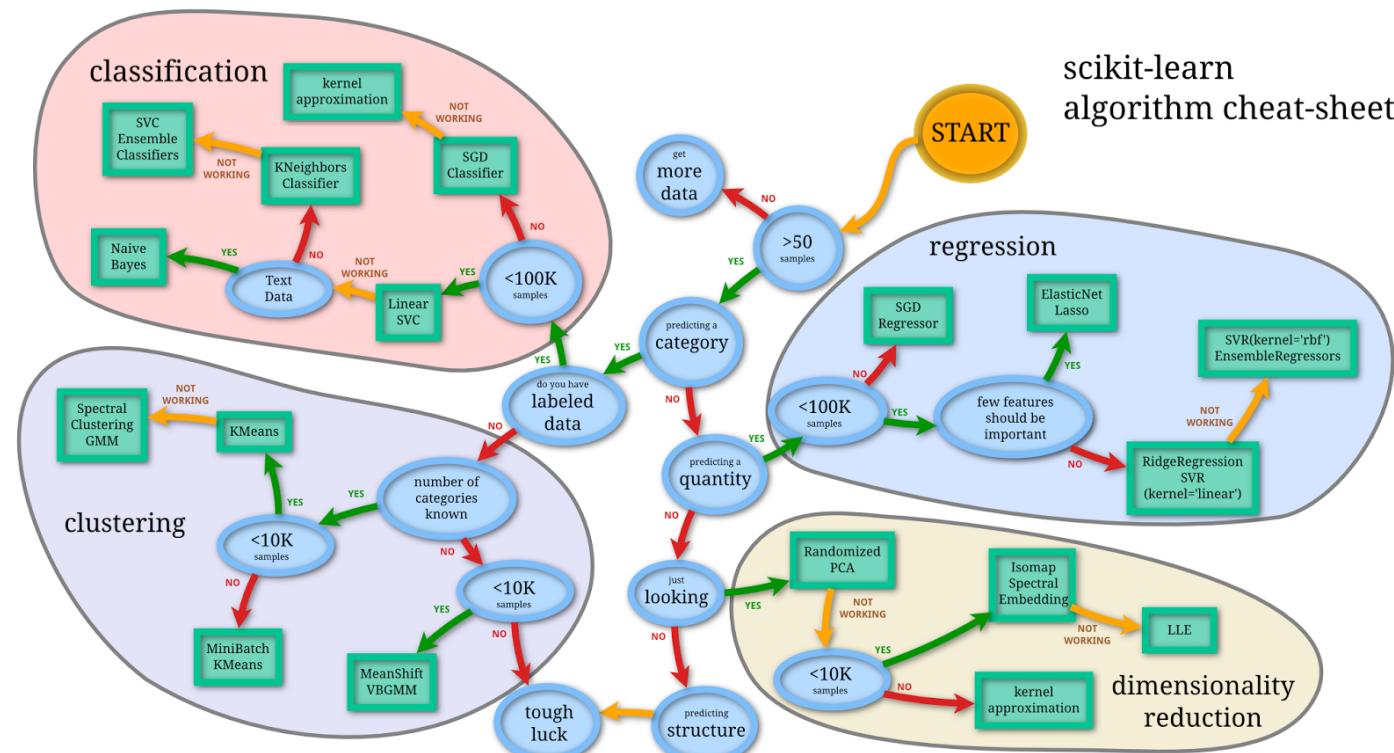
@emilyinamillion made this

Памятка по машинному обучению

Алгоритм Scikit-learn

Эта памятка по машинному обучению поможет найти подходящий алгоритм для оценки, что является наиболее сложной частью работы. Блок-схема поможет проверить документацию и задаст общее направление по каждому алгоритму. Это позволит лучше понять стоящие перед вами проблемы и способы их решения.

Scikit-learn (ранее известная как **scikits.learn**) — это бесплатная библиотека машинного обучения для Python. В нее входят различные виды **классификации**, **регрессии** и **алгоритмы кластеризации**, включающие **метод опорных векторов**, алгоритм **Random forest** («случайный лес»), **градиентный бустинг**, **метод k-средних** и **DBSCAN**. Scikit-learn предназначена для взаимодействия с вычислительными и научными библиотеками Python **NumPy** и **SciPy**.



Python For Data Science Cheat Sheet

Scikit-Learn

Learn Python for data science interactively at www.DataCamp.com



Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.



A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, 2:], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10, 5))
>>> y = np.array(['M', 'M', 'F', 'F', 'M', 'M', 'M', 'F', 'F', 'F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
...                                                    y,
...                                                    random_state=0)
```

Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning <pre>>>> lr.fit(X, y) >>> knn.fit(X_train, y_train) >>> svc.fit(X_train, y_train)</pre> Unsupervised Learning <pre>>>> k_means.fit(X_train) >>> pca_model = pca.fit_transform(X_train)</pre>	Fit the model to the data Fit the model to the data Fit to data, then transform it
---	--

Prediction

Supervised Estimators <pre>>>> y_pred = svc.predict(np.random.random(2,5)) >>> y_pred = lr.predict(X_test) >>> y_pred = knn.predict_proba(X_test)</pre> Unsupervised Estimators <pre>>>> y_pred = k_means.predict(X_test)</pre>	Predict labels Predict labels Estimate probability of a label Predict labels in clustering algos
---	--

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Classification Report

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R² Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> lr.fit(X, y)
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1, 3),
...            "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
...                      param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": random.sample(range(1, 3),
...                                         2),
...            "weights": ["uniform", "distance"]}
>>> research = RandomizedSearchCV(estimator=knn,
...                                 param_distributions=params,
...                                 cv=4,
...                                 n_iter=8,
...                                 random_state=5)
>>> research.fit(X_train, y_train)
>>> print(research.best_score_)
```

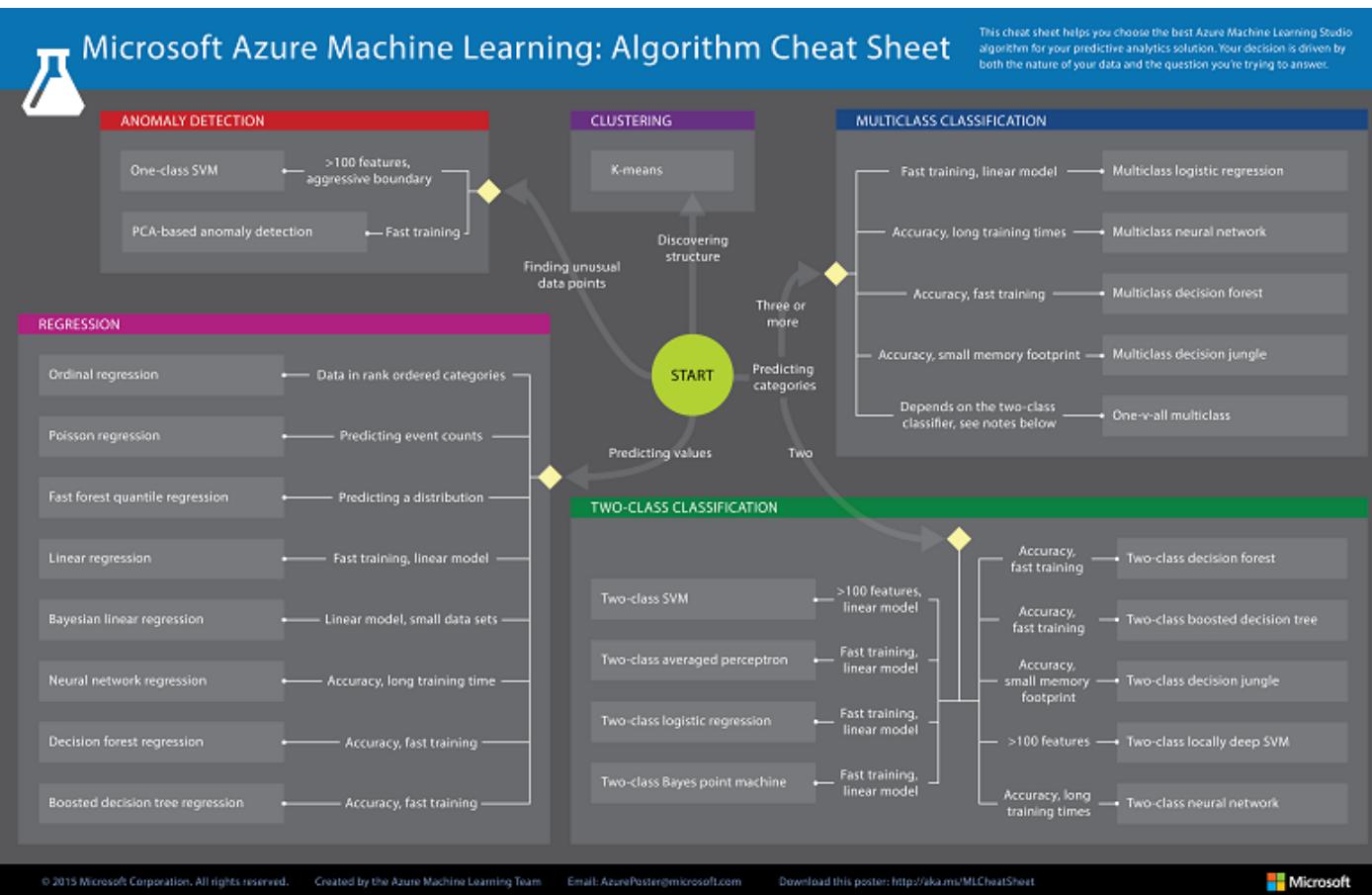
DataCamp
Learn Python for Data Science interactively



Памятка по Scikit-learn

Памятка по алгоритмам машинного обучения

Эта памятка от Microsoft Azure поможет с выбором подходящих алгоритмов машинного обучения для вашего предсказательного аналитического решения. Вначале памятка спросит о природе данных, а затем посоветует наилучший алгоритм.



Python для Data Science

Python For Data Science Cheat Sheet

Python Basics

Learn More Python for Data Science interactively at www.datacamp.com



Variables and Data Types

Variable Assignment

```
>>> x=5  
>>> x  
5
```

Calculations With Variables

>>> x+2 7	Sum of two variables
>>> x-2 3	Subtraction of two variables
>>> x*2 10	Multiplication of two variables
>>> x**2 25	Exponentiation of a variable
>>> x%2 1	Remainder of a variable
>>> x/float(2) 2.5	Division of a variable

Types and Type Conversion

str()	'5', '3.45', 'True'	Variables to strings
int()	5, 3, 1	Variables to integers
float()	5.0, 1.0	Variables to floats
bool()	True, True, True	Variables to booleans

Asking For Help

```
>>> help(str)
```

Strings

```
>>> my_string = 'thisStringIsAwesome'  
>>> my_string  
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2  
'thisStringIsAwesomethisStringIsAwesome'  
>>> my_string + 'Init'  
'thisStringIsAwesomeInit'  
>>> 'm' in my_string  
True
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'  
>>> b = 'nice'  
>>> my_list = ['my', 'list', a, b]  
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset
>>> my_list[1]
>>> my_list[-3]

Select item at index 1
Select 3rd last item

Slice
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]

Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists
>>> my_list2[1][0]
>>> my_list2[1][:2]

my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list  
'my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice'  
>>> my_list * 2  
'my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice'  
>>> my_list2 > 4  
True
```

List Methods

```
>>> my_list.index('a')  
>>> my_list.count('a')  
>>> my_list.append('!')  
>>> my_list.remove('!')  
>>> del(my_list[0:1])  
>>> my_list.reverse()  
>>> my_list.extend('!')  
>>> my_list.pop(-1)  
>>> my_list.insert(0, '!')  
>>> my_list.sort()
```

Get the index of an item
Count an item
Append an item at a time
Remove an item
Remove an item
Reverse the list
Append an item
Remove an item
Insert an item
Sort the list

Libraries

Import libraries
>>> import numpy
>>> import numpy as np
Selective import
>>> from math import pi

pandas Data analysis
NumPy Scientific computing
matplotlib 2D plotting

Install Python



Leading open data science platform
powered by Python
Free IDE that is included
with Anaconda
Create and share
documents with live code,
visualizations, text, ...

Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]  
>>> my_array = np.array(my_list)  
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset
>>> my_array[1]
2
Slice
>>> my_array[0:2]
array([1, 2])
Subset 2D Numpy arrays
>>> my_2darray[:,0]
array([1, 4])

Select item at index 1
Select items at index 0 and 1
my_2darray[rows, columns]

Numpy Array Operations

```
>>> my_array > 3  
array([False, False, False, True], dtype=bool)  
>>> my_array * 2  
array([2, 4, 6, 8])  
>>> my_array + np.array([5, 6, 7, 8])  
array([6, 8, 10, 12])
```

Numpy Array Functions

```
>>> my_array.shape  
>>> np.append(other_array)  
>>> np.insert(my_array, 1, 5)  
>>> np.delete(my_array, [1])  
>>> np.mean(my_array)  
>>> np.median(my_array)  
>>> my_array.correlcoef()  
>>> np.std(my_array)
```

Get the dimensions of the array
Append items to an array
Insert items in an array
Delete items in an array
Mean of the array
Median of the array
Correlation coefficient
Standard deviation

DataCamp

Learn Python for Data Science interactively



Памятка по Python для Data Science

Python For Data Science *Cheat Sheet*

Bokeh

Learn Bokeh **Interactively** at www.DataCamp.com
taught by Bryan Van de Ven, core contributor



Plotting With Bokeh

The Python interactive visualization library Bokeh enables high-performance visual presentation of large datasets in modern web browsers.

Bokeh's mid-level general purpose `bokeh.plotting` interface is centered around two main components: data and glyphs.



The basic steps to creating plots with the `bokeh.plotting` interface are:

1. Prepare some data:
Python lists, NumPy arrays, Pandas DataFrames and other sequences of values
 2. Create a new plot
 3. Add renderers for your data, with visual customizations
 4. Specify where to generate the output
 5. Show or save the results

```
>>> from bokeh.plotting import figure  
>>> from bokeh.io import output_file, show  
>>> x = [1, 2, 3, 4, 5] # Step 1  
>>> y = [6, 7, 2, 4, 5]  
>>> p = figure(title="simple line example", # Step 2  
             x_axis_label='x',  
             y_axis_label='y')  
>>> p.line(x, y, legend="Temp.", line_width=2) # Step 3  
>>> output_file("lines.html") # Step 4  
>>> show(p) # Step 5
```

1 Data

Also see Lists, NumPy & Pandas

Under the hood, your data is converted to Column Data

Sources. You can also do this manually:

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.array([[33.9, 4.65, 'US'],
... [32.4, 4.66, 'Asia'],
... [21.4, 4.109, 'Europe'],
... ]], columns=['mpg', 'hp', 'origin'],
... index=['Toyota', 'Fiat', 'Volvo'])

>>> from bokeh.models import ColumnDataSource
>>> cds_df = ColumnDataSource(df)
```

2 Plotting

```
>>> from bokeh.plotting import figure  
>>> p1 = figure(plot_width=300, tools='pan,box_zoom')  
>>> p2 = figure(plot_width=300, plot_height=300,  
             x_range=(0, 8), y_range=(0, 8))  
>>> p3 = figure()
```

Памятка по большим данным

TensorFlow

В мае 2017 года Google анонсировал TPU второго поколения, а также их доступность в [Google Compute Engine](#). TPU второго поколения обладают производительностью до 180 терафлопов, а при кластеризации по 64 TPU — до 11.5 петафлопов.

Create PDF in your applications with the Pdfcrowd [HTML to PDF API](#)

PDFCROWD

About

TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Skflow

Scikit Flow provides a set of high level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code. Scikit Flow is a simplified interface for TensorFlow, to get people started on predictive analytics and data mining. Scikit Flow has been merged into TensorFlow since version 0.8 and now called TensorFlow Learn.

Keras

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano

Installation

How to install new package in Python:

```
pip install <package-name>
Example: pip install requests
How to install tensorflow?
device = cpu/gpu
python_version = cp27/cp34
sudo pip install
https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.8.0-$python_version-none-linux_x86_64.whl
```

How to install Skflow

```
pip install sklearn
```

How to install Keras

```
pip install keras
update ~/.keras/keras.json - replace
"theano" by "tensorflow"
```

Helpers

Python helper

Important functions

`id(object)`

Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.

```
import __builtin__
dir(__builtin__)
```

Other built-in functions

TensorFlow

Main classes

```
tf.Graph()
tf.Operation()
tf.Tensor()
tf.Session()
```

Some useful functions

```
tf.get_default_session()
tf.get_default_graph()
tf.reset_default_graph()
ops.reset_default_graph()
tf.device("/cpu:0")
tf.name_scope(value)
tf.convert_to_tensor(value)
```

TensorFlow Optimizers

```
GradientDescentOptimizer
AdadeltaOptimizer
AdagradOptimizer
MomentumOptimizer
AdamOptimizer
FtrlOptimizer
RMSPropOptimizer
```

Reduction

```
reduce_sum
reduce_prod
reduce_min
reduce_max
reduce_mean
reduce_all
reduce_any
accumulate_n
```

Activation functions

```
tf.nn?
relu
relu6
elu
softplus
softsign
dropout
bias_add
```

TensorFlowEstimator

Each classifier and regressor have

following fields

`n_classes=0` (Regressor), `n_classes` are expected to be input (Classifiers)

```
batch_size=32,
steps=200, // except
TensorFlowRNNClassifier - there is 50
optimizer='Adagrad',
learning_rate=0.1,
```

Important functions

type(object)
Get object type

help(object)
Get help for object (list of available methods, attributes, signatures and so on)

dir(object)
Get list of object attributes (fields, functions)

str(object)
Transform an object to string

object?
Shows documentations about the object

globals()
Return the dictionary containing the current scope's global variables.

locals()
Update and return a dictionary containing the current scope's local variables.

sigmoid
tanh
sigmoid_cross_entropy_with_logits
softmax
log_softmax
softmax_cross_entropy_with_logits
sparse_softmax_cross_entropy_with_logits
weighted_cross_entropy_with_logits
etc.

Skflow

Main classes

TensorFlowClassifier
TensorFlowRegressor
TensorFlowDNNClassifier
TensorFlowDNNRegressor
TensorFlowLinearClassifier
TensorFlowLinearRegressor
TensorFlowRNNClassifier
TensorFlowRNNRegressor

Памятка по TensorFlow

Keras

В 2017 году команда TensorFlow в Google решила внедрить поддержку Keras в основную библиотеку TensorFlow. Шолле (Chollet) объяснил, что Keras является, скорее, интерфейсом, чем сквозной системой машинного обучения. Он предоставляет высокоуровневый, более интуитивный набор абстракций, который упрощает конфигурирование нейронных сетей, вне зависимости от используемой в бэкенде библиотеки научных вычислений

Python For Data Science Cheat Sheet

Keras

Learn Python for data science interactively at www.DataCamp.com



Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2, size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32,
    activation='relu',
    input_dim=100))
>>> model.add(Dense(1,activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['accuracy'])
>>> model.fit(data,labels,epochs=10,batch_size=32)
>>> predictions = model.predict(data)
```

Data

Also see NumPy, Pandas & Scikit-Learn

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

Keras Data Sets

```
>>> from keras.datasets import boston_housing,
    mnist,
    cifar10,
    imdb
>>> (x_train,y_train), (x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2), (x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3), (x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4), (x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"), delimiter=",")
>>> X = data[:,0:8]
>>> y = data[:,8]
```

Preprocessing

Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

Multilayer Perceptron (MLP)

```
>>> from keras.layers import Dense
>>> model.add(Dense(12,
    input_dim=8,
    kernel_initializer='uniform',
    activation='relu'))
>>> model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
```

Multi-Class Classification

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

Regression

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation,Conv2D,MaxPooling2D,Flatten
>>> model2.add(Conv2D(32, (3,3),padding='same',input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32, (3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Conv2D(64, (3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64, (3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

```
>>> from keras.layers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

Also see NumPy & Scikit-Learn

Inspect Model

```
>>> model.output_shape
>>> model.summary()
>>> model.get_config()
>>> model.get_weights()
```

Model output shape
Model summary representation
Model configuration
List all weight tensors in the model

Compile Model

MLP: Binary Classification

```
>>> model.compile(optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='rmsprop',
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
    loss='mse',
    metrics=['mae'])
```

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy'])
```

Model Training

```
>>> model3.fit(x_train4,
    y_train4,
    batch_size=32,
    epochs=15,
    verbose=1,
    validation_data=(x_test4,y_test4))
```

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
    y_test,
    batch_size=32)
```

Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4, batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

```
>>> from keras.optimizers import RMSProp
>>> opt = RMSProp(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
    optimizer=opt,
    metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
    y_train4,
    batch_size=32,
    epochs=15,
    validation_data=(x_test4,y_test4),
    callbacks=[early_stopping_monitor])
```

DataCamp
Learn Python for Data Science interactively



NumPy

NumPy предназначен для CPython, эталонной реализации Python, которая является не оптимизирующим интерпретатором байт-кода. Математические алгоритмы, написанные для этой версии Python, часто работают гораздо медленнее скомпилированных аналогов. Библиотека NumPy частично решает проблему скорости за счет многомерных массивов, а также функций и операторов, оптимизированных для работы с массивами. Необходимо будет переписать часть кода с

использованием NumPy, в основном внутренние циклы.

Python For Data Science Cheat Sheet

NumPy Basics

Learn Python for Data Science interactively at www.DataCamp.com



NumPy

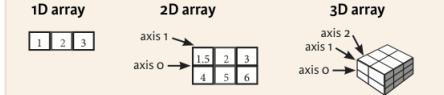
The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```



NumPy Arrays



Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1,5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([(1,5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]),
       dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4))
>>> np.ones((2,3,4),dtype=np.int16)
>>> d = np.arange(10,25,5)
>>> np.linspace(0,2,9)
>>> e = np.full((2,2),7)
>>> f = np.eye(2)
>>> np.random.random((2,2))
>>> np.empty((3,2))
```

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savetxt('array.npy', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.loadtxt("myfile.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

<code>>>> np.int64</code>	Signed 64-bit integer types
<code>>>> np.float32</code>	Standard double-precision floating point
<code>>>> np.complex</code>	Complex numbers represented by 128 floats
<code>>>> np.bool</code>	Boolean type storing TRUE and FALSE values
<code>>>> np.object</code>	Python object type
<code>>>> np.string_</code>	Fixed-length string type
<code>>>> np.unicode_</code>	Fixed-length unicode type

Inspecting Your Array

```
>>> a.shape
>>> len(a)
>>> a.ndim
>>> a.size
>>> a.dtype
>>> a.dtype.name
>>> a.astype(int)
```

Array dimensions
Length of array
Number of array dimensions
Number of array elements
Data type of array elements
Name of data type
Convert an array to a different type

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> a = b - c
array([-0.5,  0. , -0. ], [-3. , -3. , -3. ])
>>> np.subtract(a,b)
>>> b + a
array([ 2.5,  4. ,  6. ], [ 5. ,  7. ,  9. ])
>>> np.add(b,a)
>>> a / b
array([ 0.66666667,  1.        ,  1.        ], [ 0.25,  0.4,  0.5 ])
>>> np.divide(a,b)
>>> a * b
array([ 1.5,  4. ,  9. ], [ 4. , 10. , 18. ])
>>> np.multiply(a,b)
>>> np.exp(b)
>>> np.sqrt(b)
>>> np.sin(a)
>>> np.cos(b)
>>> np.log(a)
>>> e.dot(f)
array([ 7. ,  7. ], [ 7. ,  7. ])
```

Subtraction

```
Subtraction
Addition
Division
Multiplication
```

```
Multiplication
Exponentiation
Square root
Print sines of an array
Element-wise cosine
Element-wise natural logarithm
Dot product
```

Comparison

```
>>> a == b
array([False, True, True],
      [False, False, False]), dtype=bool)
>>> a < 2
array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)
```

Element-wise comparison
Element-wise comparison
Array-wise comparison

Aggregate Functions

```
>>> a.sum()
>>> a.min()
>>> b.max(axis=0)
>>> b.cumsum(axis=1)
>>> a.mean()
>>> b.median()
>>> a.corrcoef()
>>> np.std(b)
```

Array-wise sum
Array-wise minimum value
Maximum value of an array row
Cumulative sum of the elements
Mean
Median
Correlation coefficient
Standard deviation

Copying Arrays

```
>>> h = a.view()
>>> np.copy(a)
>>> h = a.copy()
```

Create a view of the array with the same data
Create a copy of the array
Create a deep copy of the array

Sorting Arrays

```
>>> a.argsort()
>>> c.argsort(axis=0)
```

Sort an array
Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Subsetting

```
>>> a[2]
3.0
>>> b[1,2]
6.0
```

1	2	3
4	5	6
7	8	9

Select the element at the 2nd index
Select the element at row 0 column 2 (equivalent to `b[1][2]`)

Slicing

```
>>> a[0:2]
array([[1, 2, 3],
       [4, 5, 6]])
>>> b[0:2,1]
array([ 2.,  5.])
```

1	2	3
4	5	6
7	8	9

Select items at index 0 and 1 in column 1
Select all items at row 0 (equivalent to `b[0:1, :]`)

c[1,...]

```
>>> c[1,...]
array([[ 3.,  2.,  1.],
       [ 4.,  5.,  6.]])
```

1	2	3
4	5	6
7	8	9

Same as `[1,:,:]`

a[::1]

```
>>> a[::1]
```

1	2	3
4	5	6
7	8	9

Reversed array a

a[1:2]

```
>>> a[1:2]
```

2	3
4	5

Select elements from a less than 2

Fancy Indexing

```
>>> b[1, 0, 1, 0]
```

1	2	3	4
5	6	7	8
9	10	11	12

Select elements (1,0),(0,1),(1,2) and (0,0)

b[1:1]

```
>>> b[1:1]
```

1	2	3
4	5	6

Select a subset of the matrix's rows and columns

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)
```

Permute array dimensions

i.T

```
>>> i.T
```

Permute array dimensions

Changing Array Shape

```
>>> g.reshape(3,-2)
```

Flatten the array

Adding/Removing Elements

```
>>> h.resize((2,6))
```

Reshape, but don't change data

```
>>> np.append(h,g)
```

Return a new array with shape (2,6)

```
>>> np.insert(a, 1, 5)
```

Append items to an array

```
>>> np.delete(a, [1])
```

Insert items in an array

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
```

Delete items from an array

```
>>> np.r_[e,f]
```

Concatenate arrays

```
>>> np.vstack((e,f))
```

Stack arrays vertically (row-wise)

```
>>> np.hstack((e,f))
```

Stack arrays horizontally (column-wise)

Creating Stacked Arrays

```
>>> np.column_stack((a,d))
```

Create stacked column-wise arrays

```
>>> np.c_[a,d]
```

Create stacked column-wise arrays

Splitting Arrays

```
>>> np.hsplit(a,3)
```

Split the array horizontally at the 3rd index

```
>>> np.vsplit(c,2)
```

Split the array vertically at the 2nd index

DataCamp

Learn Python for Data Science interactively



Памятка по NumPy

Pandas

Название «Pandas» происходит от эконометрического термина "panel data", который применяется для многомерных структурированных наборов данных.

Python For Data Science Cheat Sheet

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com



Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.



Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type

A	3
B	-5
C	7
D	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528

A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
   ...: 'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
   ...: 'Population': [11190846, 1303171035, 207847528]}
```

```
>>> df = pd.DataFrame(data,
   ...: columns=['Country', 'Capital', 'Population'])
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
-5
>>> df[1:]
   Country    Capital  Population
1  India      New Delhi  1303171035
2  Brazil     Brasilia  207847528
```

Get one element

Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0], [0]]
'Belgium'
>>> df.iat[[0], [0]]
'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at[[0], ['Country']]
'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
   Country    Brazil
   Capital    Brasilia
   Population 207847528
```

Select single row or subset of rows

```
>>> df.ix[:, 'Capital']
```

Select a single column or subset of columns

```
>>> df.ix[1, 'Capital']
```

Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```

Series s where value is not >1
is where value is <-1 or >2
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```

Set index a of Series s to 6

Dropping

```
>>> s.drop(['a', 'c'])
Drop values from rows (axis=0)
>>> df.drop('Country', axis=1)
Drop values from columns(axis=1)
```

Sort & Rank

```
>>> df.sort_index(by='Country')
>>> s.order()
>>> df.rank()
```

Sort by row or column index
Sort a series by its values
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
(rows,columns)
>>> df.index
Describe index
>>> df.columns
Describe DataFrame columns
>>> df.info()
Info on DataFrame
>>> df.count()
Number of non-NA values
```

Summary

```
>>> df.sum()
Sum of values
>>> df.cumsum()
Cumulative sum of values
>>> df.min() / df.max()
Minimum/maximum values
>>> df.idmin() / df.idmax()
Minimum/maximum index value
>>> df.describe()
Summary statistics
>>> df.mean()
Mean of values
>>> df.median()
Median of values
```

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
Apply function
>>> df.applymap(f)
Apply function element-wise
```

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b    NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
a    8.0
b    -7.0
c     3.0
d     5.0
>>> s.div(s3, fill_value=4)
a    2.5
b    -1.25
c     1.25
d     1.75
>>> s.mul(s3, fill_value=3)
a    21.0
b    -6.0
c     9.0
d     12.0
```

DataCamp
Learn Python for Data Science Interactively



Памятка по Pandas

Data Wrangling

Data Wrangling («выпас» данных, первичная обработка данных) — этот термин начинает проникать в поп-культуру. В фильме 2017 «Конг: Остров черепа» один из героев представлен как «Стив Вудвард, наш data wrangler».

Data Wrangling

with pandas

Cheat Sheet

<http://pandas.pydata.org>

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = [1, 2, 3])
Specify values for each column.
```

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
Specify values for each row.
```

	a	b	c
n			
d	1	4	7
e	2	5	11
f	3	6	9
g			12

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
        names=['n','v']))
Create DataFrame with a MultiIndex
```

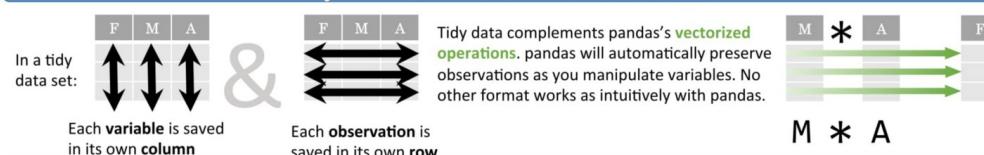
Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

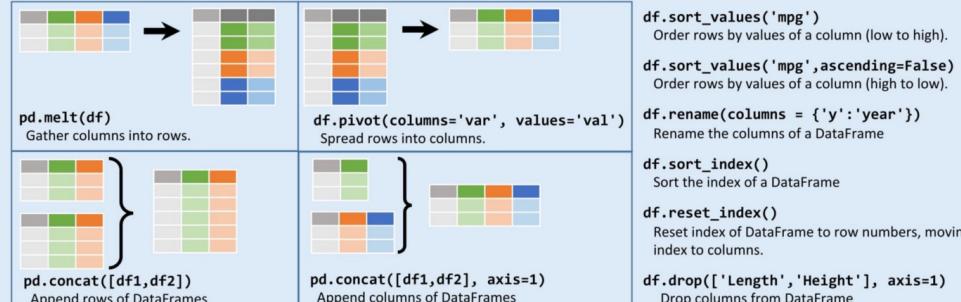
```
df = (pd.melt(df)
      .rename(columns={
          'variable': 'var',
          'value' : 'val'})
      .query('val >= 200')
    )
```

Памятка по Data Wrangling

Tidy Data – A foundation for wrangling in pandas



Reshaping Data – Change the layout of a data set



Subset Observations (Rows)



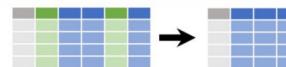
```
df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.
```

Subset Variables (Columns)



```
df[['width', 'length', 'species']]
Select multiple columns with specific names.

df['width'] or df.width
Select single column with specific name.

df.filter(regex='regex')
Select columns whose name matches regular expression regex.
```

regex (Regular Expressions) Examples

'.'	Matches strings containing a period.'
'Length\$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]\$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species\$).*	Matches strings except the string 'Species'

```
df.loc[:, 'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:, [1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a', 'c']]
Select rows meeting logical condition, and only the specific columns.
```

Summarize Data

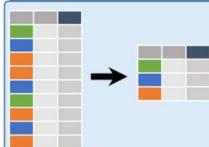
```
df['w'].value_counts()  
Count number of rows with each unique value of variable  
len(df)  
# of rows in DataFrame.  
df['w'].nunique()  
# of distinct values in a column.  
df.describe()  
Basic descriptive statistics for each column (or GroupBy)
```



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

sum()	min()
Sum values of each object.	Minimum value in each object.
count()	max()
Count non-NA/null values of each object.	Maximum value in each object.
median()	mean()
Median value of each object.	Mean value of each object.
quantile([0.25, 0.75])	var()
Quantiles of each object.	Variance of each object.
apply(function)	std()
Apply function to each object.	Standard deviation of each object.

Group Data



```
df.groupby(by="col")  
Return a GroupBy object, grouped by values in column named "col".  
  
df.groupby(level="ind")  
Return a GroupBy object, grouped by values in index level named "ind".
```

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

```
size()  
Size of each group.  
  
agg(function)  
Aggregate group using function.
```

Windows

```
df.expanding()  
Return an Expanding object allowing summary functions to be applied cumulatively.  
  
df.rolling(n)  
Return a Rolling object allowing summary functions to be applied to windows of length n.
```

Handling Missing Data

```
df.dropna()  
Drop rows with any column having NA/null data.  
df.fillna(value)  
Replace all NA/null data with value.
```

Make New Columns



```
df.assign(Area=lambda df: df.Length*df.Height)  
Compute and append one or more new columns.  
df['Volume'] = df.Length*df.Height*df.Depth  
Add single column.  
pd.qcut(df.col, n, labels=False)  
Bin column into n buckets.
```



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

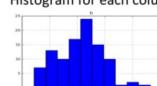
max(axis=1)	min(axis=1)
Element-wise max.	Element-wise min.
clip(lower=-10, upper=10)	abs()
Trim values at input thresholds	Absolute value.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

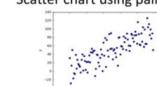
shift(1)	shift(-1)
Copy with values shifted by 1.	Copy with values lagged by 1.
rank(method='dense')	cumsum()
Ranks with no gaps.	Cumulative sum.
rank(method='min')	cummax()
Ranks. Ties get min rank.	Cumulative max.
rank(pct=True)	cummin()
Ranks rescaled to interval [0, 1].	Cumulative min.
rank(method='first')	cumprod()
Ranks. Ties go to first value.	Cumulative product.

Plotting

```
df.plot.hist()  
Histogram for each column
```



```
df.plot.scatter(x='w',y='h')  
Scatter chart using pairs of points
```



Combine Data Sets



Standard Joins

x1 x2 x3	+	x1 x3	=
A 1 T		A T	
B 2 F		B F	
C 3 NaN		D T	

```
pd.merge(adf, bdf,  
        how='left', on='x1')  
Join matching rows from bdf to adf.
```

x1 x2 x3	+	x1 x3	=
A 1.0 T		A T	
B 2.0 F		B F	
D NaN T			

```
pd.merge(adf, bdf,  
        how='right', on='x1')  
Join matching rows from adf to bdf.
```

x1 x2 x3	+	x1 x3	=
A 1 T		A T	
B 2 F		B F	

```
pd.merge(adf, bdf,  
        how='inner', on='x1')  
Join data. Retain only rows in both sets.
```

x1 x2 x3	+	x1 x3	=
A 1 T		A T	
B 2 F		B F	
C 3 NaN		C NaN	
D NaN T		D T	

```
pd.merge(adf, bdf,  
        how='outer', on='x1')  
Join data. Retain all values, all rows.
```

Filtering Joins

x1 x2	+	x1 x2	=
A 1		A 1	
B 2			

```
adf[adf.x1.isin(bdf.x1)]  
All rows in adf that have a match in bdf.
```

x1 x2	+	x1 x2	=
C 3			

```
adf[~adf.x1.isin(bdf.x1)]  
All rows in adf that do not have a match in bdf.
```



x1 x2	+	x1 x2	=
A 1		B 2	
B 2		C 3	
C 3		D 4	

```
pd.merge(ydf, zdf)  
Rows that appear in both ydf and zdf (Intersection).
```

x1 x2	+	x1 x2	=
A 1		B 2	
B 2		C 3	
C 3		D 4	

```
pd.merge(ydf, zdf, how='outer')  
Rows that appear in either or both ydf and zdf (Union).
```

x1 x2	+	x1 x2	=
A 1			

```
pd.merge(ydf, zdf, how='outer',  
        indicator=True)  
.query('_merge == "left_only"')  
.drop(['_merge'], axis=1)  
Rows that appear in ydf but not zdf (Setdiff).
```

Памятка по Pandas Data Wrangling

Data Wrangling с помощью dplyr и tidyr

Data Wrangling with dplyr and tidyverse

Cheat Sheet



Syntax - Helpful conventions for wrangling

`dplyr::tbl_df(iris)`

Converts data to `tbl` class. `tbl`'s are easier to examine than data frames. R displays only the data that fits onscreen:

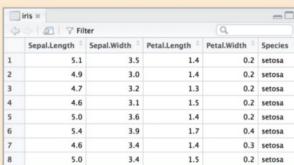
Source: local data frame [150 x 5]				
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
..
Variables not shown:	Petal.Width (dbl)			
	Species (fctr)			

`dplyr::glimpse(iris)`

Information dense summary of `tbl` data.

`utils::View(iris)`

View data set in spreadsheet-like display (note capital V).



Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa

`dplyr::%>%`

Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` is the same as `f(x, y)`
`y %>% f(x, .., z)` is the same as `f(x, y, z)`

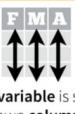
"Piping" with `%>%` makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

RStudio® is a trademark of RStudio, Inc. • [CC BY RStudio](#) • [info@rstudio.com](#) • 844-448-1212 • [rstudio.com](#)

Tidy Data - A foundation for wrangling in R

In a tidy data set:



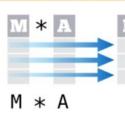
Each variable is saved in its own column

&



Each observation is saved in its own row

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



`M * A`

`F`

Reshaping Data - Change the layout of a data set



`tidy::gather(cases, "year", "n", 2:4)`

Gather columns into rows.



`tidy::spread(pollution, size, amount)`

Spread rows into columns.



`tidy::separate(storms, date, c("y", "m", "d"))`

Separate one column into several.



`tidy::unite(data, col, ..., sep)`

Unite several columns into one.

`dplyr::data_frame(a = 1:3, b = 4:6)`

Combine vectors into data frame (optimized).

`dplyr::arrange(mtcars, mpg)`

Order rows by values of a column (low to high).

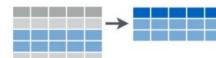
`dplyr::arrange(mtcars, desc(mpg))`

Order rows by values of a column (high to low).

`dplyr::rename(tb, y = year)`

Rename the columns of a data frame.

Subset Observations (Rows)



`dplyr::filter(iris, Sepal.Length > 7)`

Extract rows that meet logical criteria.

`dplyr::distinct(iris)`

Remove duplicate rows.

`dplyr::sample_frac(iris, 0.5, replace = TRUE)`

Randomly select fraction of rows.

`dplyr::sample_n(iris, 10, replace = TRUE)`

Randomly select n rows.

`dplyr::slice(iris, 10:15)`

Select rows by position.

`dplyr::top_n(storms, 2, date)`

Select and order top n entries (by group if grouped data).

Subset Variables (Columns)



`dplyr::select(iris, Sepal.Width, Petal.Length, Species)`

Select columns by name or helper function.

Helper functions for select - ?select

`select(iris, contains("x"))`

Select columns whose name contains a character string.

`select(iris, ends_with("Length"))`

Select columns whose name ends with a character string.

`select(iris, everything())`

Select every column.

`select(iris, matches("x"))`

Select columns whose name matches a regular expression.

`select(iris, num_range("x", 1:5))`

Select columns named x1, x2, x3, x4, x5.

`select(iris, one_of("Species", "Genus"))`

Select columns whose names are in a group of names.

`select(iris, starts_with("Sepal"))`

Select columns whose name starts with a character string.

`select(iris, Sepal.Length:Petal.Width)`

Select all columns between Sepal.Length and Petal.Width (inclusive).

`select(iris, -Species)`

Select all columns except Species.

Памятка по Data Wrangling с dplyr и tidyverse

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with [browseVignettes\(package = c\("dplyr", "tidyverse"\)\)](#) • dplyr 0.4.0 • tidyverse 2.0.0 • Updated: 1/15

Summarise Data



`dplyr::summarise(iris, avg = mean(Sepal.Length))`

Summarise data into single row of values.

`dplyr::summarise_each(iris, funs(mean))`

Apply summary function to each column.

`dplyr::count(iris, Species, wt = Sepal.Length)`

Count number of rows with each unique value of variable (with or without weights).



Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

`dplyr::first`

First value of a vector.

`dplyr::last`

Last value of a vector.

`dplyr::nth`

Nth value of a vector.

`dplyr::n`

of values in a vector.

`dplyr::n_distinct`

of distinct values in a vector.

`IQR`

IQR of a vector.

Make New Variables



`dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)`

Compute and append one or more new columns.

`dplyr::mutate_each(iris, funs(min_rank))`

Apply window function to each column.

`dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)`

Compute one or more new columns. Drop original columns.



Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

`dplyr::lead`

Copy with values shifted by 1.

`dplyr::lag`

Copy with values lagged by 1.

`dplyr::dense_rank`

Ranks with no gaps.

`dplyr::min_rank`

Ranks. Ties get min rank.

`dplyr::percent_rank`

Ranks rescaled to [0, 1].

`dplyr::row_number`

Ranks. Ties got to first value.

`dplyr::ntile`

Bin vector into n buckets.

`dplyr::between`

Are values between a and b?

`dplyr::cume_dist`

Cumulative distribution.

`dplyr::cumall`

Cumulative all

`dplyr::cumany`

Cumulative any

`dplyr::cummean`

Cumulative mean

`cumsum`

Cumulative sum

`cummax`

Cumulative max

`cummin`

Cumulative min

`cumprod`

Cumulative prod

`pmax`

Element-wise max

`pmin`

Element-wise min

Group Data

`dplyr::group_by(iris, Species)`

Group data into rows with the same value of Species.

`dplyr::ungroup(iris)`

Remove grouping information from data frame.

`iris %>% group_by(Species) %>% summarise(...)`

Compute separate summary row for each group.



RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with `browseVignettes(package = c("dplyr", "tidyverse"))` • dplyr 0.4.0 • tidyverse 2.0.0 • Updated: 1/15

Combine Data Sets



Mutating Joins

`dplyr::left_join(a, b, by = "x1")`

Join matching rows from b to a.

`dplyr::right_join(a, b, by = "x1")`

Join matching rows from a to b.

`dplyr::inner_join(a, b, by = "x1")`

Join data. Retain only rows in both sets.

`dplyr::full_join(a, b, by = "x1")`

Join data. Retain all values, all rows.

Filtering Joins

`dplyr::semi_join(a, b, by = "x1")`

All rows in a that have a match in b.

`dplyr::anti_join(a, b, by = "x1")`

All rows in a that do not have a match in b.



Set Operations

`dplyr::intersect(y, z)`

Rows that appear in both y and z.

`dplyr::union(y, z)`

Rows that appear in either or both y and z.

`dplyr::setdiff(y, z)`

Rows that appear in y but not z.

Binding

`dplyr::bind_rows(y, z)`

Append z to y as new rows.

`dplyr::bind_cols(y, z)`

Append z to y as new columns.

Caution: matches rows by position.

Памятка по Data Wrangling с dplyr и tidyverse

SciPy

В основе SciPy лежит объект-массив NumPy. Эта библиотека является частью стека NumPy, который включает такие инструменты, как Matplotlib, Pandas и SymPy, а также расширяющийся набор библиотек

для научных вычислений. У стека NumPy и приложений **MATLAB**, **GNU Octave** и **Scilab** одна и та же аудитория пользователей. Стек NumPy также иногда называют стеком SciPy.

Python For Data Science Cheat Sheet

SciPy - Linear Algebra

Learn More Python for Data Science [Interactively](#) at [www.datacamp.com](#)



SciPy

The SciPy library is one of the core packages for scientific computing that provides mathematical algorithms and convenience functions built on the NumPy extension of Python.



Interacting With NumPy

Also see NumPy

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
>>> b = np.array([(1+5), 2, 3], [(4, 5), 6])
>>> c = np.array([(1, 5, 2, 3), (4, 5, 6), (3, 2, 1), (4, 5, 6)])
```

Index Tricks

```
>>> np.mgrid[0:5, 0:5]
Create a dense meshgrid
>>> np.ogrid[0:2, 0:2]
Create an open meshgrid
>>> np.r_[3, [0]*5, -1:10]
Stack arrays vertically (row-wise)
>>> np.c_[b, c]
Create stacked column-wise arrays
```

Shape Manipulation

>>> np.transpose(b)	Permute array dimensions
>>> b.flatten()	Flatten the array
>>> np.hstack((b, c))	Stack arrays horizontally (column-wise)
>>> np.vstack((a, b))	Stack arrays vertically (row-wise)
>>> np.hsplit(c, 2)	Split the array horizontally at the 2nd index
>>> np.vsplit(d, 2)	Split the array vertically at the 2nd index

Polynomials

```
>>> from numpy import poly1d
>>> p = poly1d([3, 4, 5])
Create a polynomial object
```

Vectorizing Functions

```
>>> def myfunc(a):
...     if a < 0:
...         return a**2
...     else:
...         return a/2
>>> np.vectorize(myfunc)
Vectorize functions
```

Type Handling

```
>>> np.real(b)
Return the real part of the array elements
>>> np.imag(b)
Return the imaginary part of the array elements
>>> np.real_if_close(c, tol=1000)
Return a real array if complex parts close to 0
>>> np.cast['f'](np.pi)
Cast object to a data type
```

Other Useful Functions

```
>>> np.angle(b, deg=True)
>>> g = np.linspace(0, np.pi, num=5)
>>> g[3:] += np.pi
>>> np.unwrap(g)
>>> np.logspace(0, 10, 3)
>>> np.select((c<4), [c**2])
>>> misc.factorial(a)
>>> misc.comb(10, 3, exact=True)
>>> misc.central_diff_weights(3)
>>> misc.derivative(myfunc, 1.0)
```

Return the angle of the complex argument
Create an array of evenly spaced values (number of samples)
Unwrap
Create an array of evenly spaced values (log scale)
Return values from a list of arrays depending on conditions
Factorial
Combine N things taken at k time
Weights for N-point central derivative
Find the n-th derivative of a function at a point

Linear Algebra
Also see NumPy

You'll use the `linalg` and `sparse` modules. Note that `scipy.linalg` contains and expands on `numpy.linalg`.

Creating Matrices	
<pre>>>> A = np.matrix(np.random.random((2,2))) >>> B = np.asmatrix(b) >>> C = np.mat(np.random.random((10,5))) >>> D = np.mat([[3,4], [5,6]])</pre>	
Basic Matrix Routines	
Inverse <code>>>> A.I</code> <code>>>> linalg.inv(A)</code>	Inverse Transpose <code>>>> A.T</code> <code>>>> A.H</code>
Trace <code>>>> np.trace(A)</code>	Trace Norm <code>>>> linalg.norm(A)</code> <code>>>> linalg.norm(A,1)</code> <code>>>> linalg.norm(A,np.inf)</code>
Rank <code>>>> np.linalg.matrix_rank(C)</code>	Matrix rank Determinant <code>>>> linalg.det(A)</code>
Solving linear problems <code>>>> linalg.solve(A,b)</code> <code>>>> E = np.mat(a).T</code> <code>>>> linalg.lstsq(F,E)</code>	Solver for dense matrices Solver for dense matrices Least-squares solution to linear matrix equation <code>>>> linalg.pinv(C)</code>
Generalized inverse <code>>>> linalg.pinv2(C)</code>	Compute the pseudo-inverse of a matrix (least-squares solver) Compute the pseudo-inverse of a matrix (SVD) <code>>>> linalg.pinv2(C)</code>
Creating Sparse Matrices	
Inverse <code>>>> sparse.linalg.inv(I)</code>	Inverse <code>>>> sparse.linalg.norm(I)</code>
Norm <code>>>> sparse.linalg.norm(I)</code>	Norm <code>>>> sparse.linalg.spsolve(H,I)</code>
Sparse Matrix Routines	
Inverse <code>>>> sparse.linalg.inv(I)</code>	Inverse <code>>>> sparse.linalg.norm(I)</code>
Norm <code>>>> sparse.linalg.norm(I)</code>	Norm <code>>>> sparse.linalg.spsolve(H,I)</code>
Sparse Matrix Functions	
<code>>>> sparse.linalg.expm(I)</code>	Sparse matrix exponential <code>>>> sparse.linalg.expm(I)</code>
Asking For Help	
<code>>>> help(scipy.linalg.diagsvd)</code> <code>>>> np.info(np.matrix)</code>	

Matrix Functions

Addition	Subtraction
<code>>>> np.add(A, D)</code>	<code>>>> np.subtract(A, D)</code>
Division	Multiplication
<code>>>> np.divide(A, D)</code>	<code>>>> A @ D</code>
Multiplication <code>>>> np.multiply(D,A)</code>	Addition Subtraction Division Multiplication operator (Python 3) Dot product Vector dot product Inner product Outer product Tensor dot product Kronecker product
Exponential Functions	Logarithm Function
<code>>>> linalg.expm(A)</code>	<code>>>> linalg.logm(A)</code>
<code>>>> linalg.expm2(A)</code>	Matrix exponential Matrix exponential (Taylor Series) Matrix exponential (eigenvalue decomposition)
Trigonometric Functions	Hyperbolic Trigonometric Functions
<code>>>> linalg.sinm(D)</code>	<code>>>> linalg.sinh(D)</code>
<code>>>> linalg.cosm(D)</code>	<code>>>> linalg.coshm(D)</code>
<code>>>> linalg.tanm(A)</code>	<code>>>> linalg.tanhm(A)</code>
Matrix Sign Function	Matrix Sign Function
<code>>>> np.sigm(A)</code>	Matrix sign function
Matrix Square Root	Matrix square root
<code>>>> linalg.sqrtm(A)</code>	Matrix square root
Arbitrary Functions	
<code>>>> linalg.funm(A, lambda x: x*x)</code>	
Decompositions	
Eigenvalues and Eigenvectors	
<code>>>> I, v = linalg.eig(A)</code>	
Solve ordinary or generalized eigenvalue problem for square matrix Unpack eigenvalues First eigenvector Second eigenvector Unpack eigenvalues	
Singular Value Decomposition	
<code>>>> U, s, Vh = linalg.svd(B)</code>	
Singular Value Decomposition (SVD) Construct sigma matrix in SVD	
LU Decomposition	
<code>>>> P, L, U = linalg.lu(C)</code>	
Sparse Matrix Decompositions	
<code>>>> la, v = sparse.linalg.eigs(F, 1)</code>	
Eigenvalues and eigenvectors SVD	

DataCamp
Learn Python for Data Science Interactively

Памятка по SciPy

Matplotlib

Matplotlib — это библиотека для построения графиков для Python и его вычислительного

Create PDF in your applications with the Pdfcrowd [HTML to PDF API](#)

PDFCROWD

математического расширения NumPy. Она предоставляет объектно-ориентированный API для встраивания графиков в приложения с использованием универсальных GUI-инструментов, таких как Tkinter, wxPython, Qt, или GTK+. Существует также процедурный интерфейс «pylab» на основе конечного автомата (например, OpenGL), разработанный так, чтобы походить MATLAB, хотя его использование не рекомендуется. SciPy использует matplotlib.

Pyplot — это модуль matplotlib, который предоставляет интерфейс наподобие MATLAB. Matplotlib применяется так же, как и MATLAB, позволяет использовать Python, и к тому же бесплатен.

Python For Data Science Cheat Sheet

Matplotlib

Learn Python Interactively at www.DataCamp.com



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



1) Prepare The Data

Also see [Lists & NumPy](#)

1D Data

```
>>> import numpy as np  
>>> x = np.linspace(0, 10, 100)  
>>> y = np.cos(x)  
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))  
>>> data2 = 3 * np.random.random((10, 10))  
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]  
>>> U = -1 - X**2 + Y  
>>> V = 1 + X - Y**2  
>>> from matplotlib.cbook import get_sample_data  
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2) Create Plot

```
>>> import matplotlib.pyplot as plt  
  
Figure  
>>> fig = plt.figure()  
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()  
>>> ax1 = fig.add_subplot(221) # row-col-num  
>>> ax2 = fig.add_subplot(212)  
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)  
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3) Plotting Routines

1D Data

```
>>> fig, ax = plt.subplots()  
>>> lines = ax.plot(x,y)  
>>> ax.scatter(x,y)  
>>> axes[0,0].bar([1,2,3],[3,4,5])  
>>> axes[0,0].barh([1,2,5],[0,1,2])  
>>> axes[1,0].axhline(0.45)  
>>> axes[0,1].axvline(0.65)  
>>> ax.fill(x,y,color='blue')  
>>> ax.fill_between(x,y,color='yellow')
```

Draw points with lines or markers connecting them
Draw unconnected points, scaled or colored
Plot vertical rectangles (constant width)
Draw a horizontal line across axes
Draw filled polygons
Fill between y-values and o

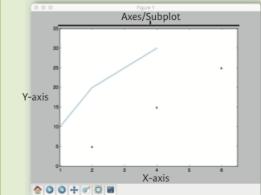
2D Data or Images

```
>>> fig, ax = plt.subplots()  
>>> im = ax.imshow(img, cmap='gist_earth',  
 interpolation='nearest',  
 vmin=-2,  
 vmax=2)
```

Colormapped or RGB arrays

Plot Anatomy & Workflow

Plot Anatomy



Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
 - 2 Create plot
 - 3 Plot
 - 4 Customize plot
 - 5 Save plot
 - 6 Show plot
- ```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4] Step 1
>>> y = [10,20,25,30] Step 2
>>> fig = plt.figure() Step 3
>>> ax = fig.add_subplot(111) Step 4
>>> ax.plot(x, y, color='lightblue', linewidth=3) Step 3, 4
>>> ax.scatter([2,4,6],
 [5,15,25],
 color='darkgreen',
 marker='*')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show() Step 6
```

#### 4) Customize Plot

##### Colors, Color Bars & Color Maps

```
>>> plt.plot(x, y, alpha=0.4)
>>> ax.plot(x, y, 'r+')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img, cmap='seismic')
```

##### Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".")
>>> ax.plot(x,y,marker="o")
```

##### LineStyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'-',x**2,y**2,'-')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

##### Text & Annotations

```
>>> ax.text(1,-2,1,
'Example Graph',
style='italic')
>>> ax.annotate("Some text",
(8,0),
xycoords='data',
xytext=(10.5, 0),
textcoords='data',
arrowprops=dict(arrowsize=2,
connectionstyle="arc3"),)
```

##### Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)
>>> axes[1,1].quiver(y,z)
>>> axes[0,1].streamplot(X,Y,U,V)
```

##### Data Distributions

```
>>> ax1.hist(y)
>>> ax3.bxpplot(y)
>>> ax3.violinplot(z)
```

##### MathText

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

##### Limits, Legends & Layouts

```
>>> ax.margins(x=0,y=0.1)
>>> ax.axis('equal')
>>> ax.set_xlim([0,10.5],ylim=[-1.5,1.5])
>>> ax.set_xlim(0,10.5)
```

##### Legends

```
>>> ax.set(title='An Example Axes',
 xlabel='x-Axis',
 ylabel='y-Axis')
>>> ax.legend(loc='best')
```

##### Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),
 ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',
 direction='inout',
 length=10)
```

##### Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,
 hspace=0.3,
 left=0.125,
 right=0.9,
 top=0.9,
 bottom=0.1)
>>> fig.tight_layout()
>>> ax1.spines['top'].set_visible(False)
>>> ax1.spines['bottom'].set_position(('outward',10))
```

Make the top axis line for a plot invisible  
Move the bottom axis line outward

#### 5) Save Plot

```
>>> plt.savefig('foo.png')
>>> plt.savefig('foo.png', transparent=True)
```

#### 6) Show Plot

```
>>> plt.show()
```

#### Close & Clear

```
>>> plt.cla()
>>> plt.clf()
>>> plt.close()
```

DataCamp  
Learn Python for Data Science [Interactively](http://www.DataCamp.com)



Памятка по Matplotlib

## Визуализация данных

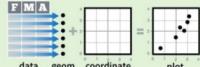
# Data Visualization with ggplot2

Cheat Sheet

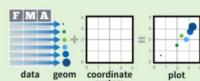


## Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data set**, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

**aesthetic mappings**    **data**    **geom**

```
aqplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
```

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

**ggplot(data = mpg, aes(x = cty, y = hwy))**

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
data +
 geom_point(aes(color = cyl)) +
 geom_smooth(method = "lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

**add layers, elements with +**

**additional elements**

Add a new layer to a plot with a **geom\_\*** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

**last\_plot()**

Returns the last plot

**ggsave("plot.png", width = 5, height = 5)**

Saves last plot as 5'x5' file named "plot.png" in working directory. Matches file type to file extension.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

## Памятка по визуализации данных

| Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.                                                                                                                                                                                                                                                                       |  |  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| <b>One Variable</b>                                                                                                                                                                                                                                                                                                                                                                                           |  |  |
| <b>Continuous</b><br><pre>a + geom_area(stat = "bin")</pre><br><pre>b + geom_area(aes(y..density..), stat = "bin")</pre> <pre>a + geom_density(kernel = "gaussian")</pre><br><pre>a + geom_dotplot()</pre><br><pre>a + geom_freqpoly()</pre><br><pre>b + geom_freqpoly(aes(y = ..density..))</pre> <pre>a + geom_histogram(binwidth = 5)</pre><br><pre>x, y, alpha, color, fill, linetype, size, weight</pre> |  |  |
| <b>Discrete</b><br><pre>b &lt;- ggplot(mpg, aes(flt))</pre><br><pre>b + geom_bar()</pre><br><pre>x, alpha, color, fill, linetype, size, weight</pre>                                                                                                                                                                                                                                                          |  |  |
| <b>Graphical Primitives</b>                                                                                                                                                                                                                                                                                                                                                                                   |  |  |
| <b>Continuous X, Continuous Y</b><br><pre>f &lt;- ggplot(mpg, aes(cty, hwy))</pre><br><pre>f + geom_blank()</pre><br><pre>f + geom_jitter()</pre><br><pre>x, y, alpha, color, fill, shape, size</pre>                                                                                                                                                                                                         |  |  |
| <b>Continuous Function</b><br><pre>j &lt;- ggplot(economics, aes(date, unemploy))</pre><br><pre>j + geom_area()</pre><br><pre>x, y, alpha, color, fill, linetype, size</pre>                                                                                                                                                                                                                                  |  |  |
| <b>Two Variables</b>                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| <b>Continuous X, Continuous Y</b><br><pre>f + geom_point()</pre><br><pre>x, y, alpha, color, fill, shape, size</pre>                                                                                                                                                                                                                                                                                          |  |  |
| <b>Continuous Bivariate Distribution</b><br><pre>i &lt;- ggplot(movies, aes(year, rating))</pre><br><pre>i + geom_bin2d(binwidth = c(5, 0.5))</pre> <pre>xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight</pre>                                                                                                                                                                             |  |  |
| <b>Continuous Function</b><br><pre>j + geom_hex()</pre><br><pre>x, y, alpha, colour, fill size</pre>                                                                                                                                                                                                                                                                                                          |  |  |
| <b>Discrete X, Continuous Y</b><br><pre>df &lt;- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)</pre> <pre>k &lt;- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))</pre><br><pre>g + geom_crossbar(fatten = 2)</pre><br><pre>x, y, ymax, ymin, alpha, color, fill, linetype, size</pre>                                                                                                           |  |  |
| <b>Discrete X, Discrete Y</b><br><pre>h &lt;- ggplot(diamonds, aes(cut, color))</pre><br><pre>h + geom_jitter()</pre><br><pre>x, y, alpha, color, fill, shape, size</pre>                                                                                                                                                                                                                                     |  |  |
| <b>Three Variables</b>                                                                                                                                                                                                                                                                                                                                                                                        |  |  |
| <b>Discrete X, Continuous Y</b><br><pre>sealsSz &lt;- with(seals, sqrt(delta_long^2 + delta_lat^2))</pre> <pre>m &lt;- ggplot(seals, aes(long, lat))</pre><br><pre>e + geom_rect(aes(xmin = long, ymin = lat, xmax = long + delta_long, ymax = lat + delta_lat))</pre><br><pre>xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size</pre>                                                               |  |  |
| <b>Three Variables</b><br><pre>m + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)</pre><br><pre>x, y, alpha, fill</pre>                                                                                                                                                                                                                                                                  |  |  |
| <b>Three Variables</b><br><pre>m + geom_contour(aes(z = z))</pre><br><pre>x, y, z, alpha, colour, linetype, size, weight</pre>                                                                                                                                                                                                                                                                                |  |  |
| <b>Three Variables</b><br><pre>m + geom_tile(aes(fill = z))</pre><br><pre>x, y, alpha, color, fill, linetype, size</pre>                                                                                                                                                                                                                                                                                      |  |  |

Learn more at [docs.ggplot2.org](http://docs.ggplot2.org) • ggplot2 0.9.3.1 • Updated: 3/15

## Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. `a + geom_bar(stat = "bin")`

Each stat creates additional variables to map aesthetics to. These variables use a common `.name..` syntax.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. `stat_bin(geom="bar")` does the same as `geom_bar(stat="bin")`

| stat function                                                                                                      | layer specific mappings | variable created by transformation |
|--------------------------------------------------------------------------------------------------------------------|-------------------------|------------------------------------|
| <code>i + stat_density2d(aes(fill = ..level..), geom = "polygon", n = 100)</code>                                  |                         |                                    |
| <code>a + stat_bin(binwidth = 1, origin = 10)</code>                                                               | 1D distributions        |                                    |
| <code>x, y   ..count.., ..density..</code>                                                                         |                         |                                    |
| <code>a + stat_bindot(binwidth = 1, binaxis = "x")</code>                                                          |                         |                                    |
| <code>x, y   ..count..,</code>                                                                                     |                         |                                    |
| <code>a + stat_density(adjust = 1, kernel = "gaussian")</code>                                                     |                         |                                    |
| <code>x, y   ..density.., ..scaled..</code>                                                                        |                         |                                    |
| <code>f + stat_bin2d(bins = 30, drop = TRUE)</code>                                                                | 2D distributions        |                                    |
| <code>x, y, fill   ..count.., ..density..</code>                                                                   |                         |                                    |
| <code>f + stat_binehex(bins = 30)</code>                                                                           |                         |                                    |
| <code>x, y, fill   ..count.., ..density..</code>                                                                   |                         |                                    |
| <code>f + stat_density2d(contour = TRUE, n = 100)</code>                                                           |                         |                                    |
| <code>x, y, color, size   ..level..</code>                                                                         |                         |                                    |
| <code>m + stat_contour(aes(z = z))</code>                                                                          | 3 Variables             |                                    |
| <code>x, y, z, order   ..level..</code>                                                                            |                         |                                    |
| <code>m + stat_speckle(aes(radius = z, angle = z))</code>                                                          |                         |                                    |
| <code>angle, radius, x, xend, y, yend   ..xend.., ..yend.., ..yend..</code>                                        |                         |                                    |
| <code>m + stat_summary_hex(aes(z = z), bins = 30, fun = mean)</code>                                               |                         |                                    |
| <code>x, y, z, fill   ..value..</code>                                                                             |                         |                                    |
| <code>m + stat_summary2d(aes(z = z), bins = 30, fun = mean)</code>                                                 |                         |                                    |
| <code>x, y, z, fill   ..value..</code>                                                                             |                         |                                    |
| <code>g + stat_boxplot(coef = 1.5)</code>                                                                          | Comparisons             |                                    |
| <code>x, y   ..lower.., ..middle.., ..upper.., ..outliers..</code>                                                 |                         |                                    |
| <code>g + stat_ydensity(adjust = 1, kernel = "gaussian", scale = "area")</code>                                    |                         |                                    |
| <code>x, y   ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..</code>                          |                         |                                    |
| <code>f + stat_ecdf(n = 40)</code>                                                                                 | Functions               |                                    |
| <code>x, y   ..x.., ..y..</code>                                                                                   |                         |                                    |
| <code>f + stat_quantile(qu quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")</code>             |                         |                                    |
| <code>x, y   ..quantile.., ..y..</code>                                                                            |                         |                                    |
| <code>f + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)</code> |                         |                                    |
| <code>x, y   ..se.., ..y.., ..ymin.., ..ymax..</code>                                                              |                         |                                    |
| <code>ggplot() + stat_function(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5))</code>                    | General Purpose         |                                    |
| <code>x   ..y..</code>                                                                                             |                         |                                    |
| <code>f + stat_identity()</code>                                                                                   |                         |                                    |
| <code>ggplot() + stat_qq(aes(sample=1:100), distribution = qt, dparams = list(fit=0))</code>                       |                         |                                    |
| <code>sample, x, y   ..x.., ..y..</code>                                                                           |                         |                                    |
| <code>f + stat_sum()</code>                                                                                        |                         |                                    |
| <code>x, y, size   ..size..</code>                                                                                 |                         |                                    |
| <code>f + stat_summary(fun.data = "mean_cl_boot")</code>                                                           |                         |                                    |
| <code>f + stat_unique()</code>                                                                                     |                         |                                    |

RStudio® is a trademark of RStudio, Inc. • [CC BY](#) RStudio • [info@rstudio.com](mailto:info@rstudio.com) • 844-448-1212 • [rstudio.com](http://rstudio.com)

## Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

```
n <- b + geom_bar(aes(fill = fl))
n + scale_fill_manual(
 values = c("skyblue", "royalblue", "blue", "navy"),
 limits = c("d", "e", "p", "r"),
 name = "fuel",
 labels = c("D", "E", "P", "R"))
```

range of values to include in mapping title to use in legend/axis labels to use in legend/axis breaks to use in legend/axis

**General Purpose scales**  
Use with any aesthetic: alpha, color, fill, linetype, shape, size

- `scale_*_continuous()` - map cont'ous values to visual values
- `scale_*_discrete()` - map discrete values to visual values
- `scale_*_identity()` - use data values as visual values
- `scale_*_manual(values = c())` - map discrete values to manually specified visual values

**X and Y location scales**  
Use with x or y aesthetics (shown here)

- `scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks"))` - treat x values as dates. See `lubridate` for label formats.
- `scale_x_datetime()` - treat x values as date times. Use same arguments as `scale_x_date()`.
- `scale_x_log10()` - Plot x on log10 scale
- `scale_x_reverse()` - Reverse direction of x axis
- `scale_x_sqrt()` - Plot x on square root scale

**Color and fill scales**

| Discrete                                                                   | Continuous                                                               |
|----------------------------------------------------------------------------|--------------------------------------------------------------------------|
| <code>n &lt;- b + geom_bar(aes(fill = fl))</code>                          | <code>o &lt;- a + geom_dotplot(aes(fill = ..x..))</code>                 |
| <code>n + scale_fill_brewer(palette = "Blues")</code>                      | <code>o + scale_fill_gradient(low = "red", high = "yellow")</code>       |
| For palette choices:<br>library(RColorBrewer)<br>display.brewer.all()      | <code>o + scale_fill_grey(mid = "white", midpoint = 25)</code>           |
| <code>n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")</code> | <code>o + scale_fill_gradients(colours = terrain.colors(6))</code>       |
|                                                                            | Also: RdYlBu, RdYlGn, topo.colors, cm.colors, RColorBrewer::brewer.pal() |

**Shape scales**

| Manual shape values                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 □ 6 ▽ 12 ▨ 18 ◆ 24 ▲<br>1 ○ 7 △ 13 ▧ 19 ● 25 ▷<br>2 △ 8 ▴ 14 ▷ 20 ▪ 21 ▲<br>3 ▾ 9 ▫ 15 ▨ 21 ▲ 22 ▢<br>4 ✕ 10 ▩ 16 ● 22 □ 20 ▢<br>5 ▽ 11 ▾ 17 ▲ 23 ▢ 21 ▢ |

**Size scales**

| Value mapped to area of circle (not radius)          |                                           |
|------------------------------------------------------|-------------------------------------------|
| <code>q &lt;- f + geom_point(aes(size = cyl))</code> | <code>q + scale_size_area(max = 6)</code> |

## Coordinate Systems

```
r <- b + geom_bar()
r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
```

The default cartesian coordinate system

## Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
```

| t + facet_grid(~ fl)      | facet into columns based on fl        |
|---------------------------|---------------------------------------|
| t + facet_grid(year ~ .)  | facet into rows based on year         |
| t + facet_grid(year ~ fl) | facet into both rows and columns      |
| t + facet_wrap(~ fl)      | wrap facets into a rectangular layout |

Set `scales` to let axis limits vary across facets

```
t + facet_grid(y ~ x, scales = "free")
```

x and y axis limits adjust to individual facets

- `"free_x"` - x axis limits adjust
- `"free_y"` - y axis limits adjust

Set `labeler` to adjust facet labels

| t + facet_grid(~ fl, labeler = label_both)                 | fl: c fl: d fl: e fl: p fl: r                          |
|------------------------------------------------------------|--------------------------------------------------------|
| t + facet_grid(~ fl, labeler = label_bquote(alpha ^ .(x))) | $\alpha^c$ $\alpha^d$ $\alpha^e$ $\alpha^p$ $\alpha^r$ |
| t + facet_grid(~ fl, labeler = label_parsed)               | c d e p r                                              |

## Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))
```

Use scale functions to update legend labels

| s + geom_bar(position = "dodge")    | Arrange elements side by side                                              |
|-------------------------------------|----------------------------------------------------------------------------|
| s + geom_bar(position = "fill")     | Stack elements on top of one another, normalize height                     |
| s + geom_bar(position = "stack")    | Stack elements on top of one another                                       |
| f + geom_point(position = "jitter") | Add random noise to X and Y position of each element to avoid overplotting |

Each position adjustment can be recast as a function with manual `width` and `height` arguments

```
s + geom_bar(position = position_dodge(width = 1))
```

## Labels

**Legends**

- `t + theme(legend.position = "bottom")` Place legend at "bottom", "top", "left", or "right"
- `t + guides(color = "none")` Set legend type for each aesthetic: colorbar, legend, or none (no legend)
- `t + scale_fill_discrete(name = "Title", labels = c("A", "B", "C"))` Set legend title and labels with a scale function.

## Themes

**Themes**

| r + theme_bw()      | White background with grid lines |
|---------------------|----------------------------------|
| r + theme_classic() | White background no gridlines    |
| r + theme_minimal() | Minimal theme                    |
| r + theme_grey()    | Grey background (default theme)  |

**Zooming**

**Without clipping (preferred)**

```
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))
```

**With clipping (removes unseen data points)**

```
t + xlim(0, 100) + ylim(10, 20)
```

```
t + scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 100))
```

Learn more at [docs.ggplot2.org](http://docs.ggplot2.org) • ggplot2 0.9.3.1 • Updated: 3/15

## Памятка по ggplot

## PySpark

## Python For Data Science Cheat Sheet

### PySpark Basics

Learn Python for data science interactively at [www.DataCamp.com](http://www.DataCamp.com)



### Spark

PySpark is the Spark Python API that exposes the Spark programming model to Python



### Initializing Spark

#### SparkContext

```
>>> from pyspark import SparkContext
>>> sc = SparkContext(master = 'local[2]')
```

#### Inspect SparkContext

```
>>> sc.version
>>> sc.pythonVer
>>> sc.master
>>> str(sc.sparkHome)
>>> str(sc.sparkUser())

>>> sc appName
>>> sc.applicationId
>>> sc.defaultParallelism
>>> sc.defaultMinPartitions
```

Retrieve SparkContext version  
Retrieve Python version  
Master URL to connect to  
Path where Spark is installed on worker nodes  
Retrieve name of the Spark User running SparkContext  
Return application name  
Retrieve application ID  
Return default level of parallelism  
Default minimum number of partitions for RDDs

#### Configuration

```
>>> from pyspark import SparkConf, SparkContext
>>> conf = (SparkConf()
 .setMaster("local")
 .setAppName("My App")
 .set("spark.executor.memory", "1g"))
>>> sc = SparkContext(conf = conf)
```

#### Using The Shell

In the PySpark shell, a special interpreter-aware SparkContext is already created in the variable called `sc`.

```
$./bin/spark-shell --master local[2]
$./bin/pyspark --master local[4] --py-files code.py
```

Select which master the context connects to with the `--master` argument, and add Python .zip, .egg or .py files to the runtime path by passing a comma-separated list to `--py-files`.

#### Loading Data

##### Parallelized Collections

```
>>> rdd = sc.parallelize([('a', 7), ('a', 2), ('b', 2)])
>>> rdd2 = sc.parallelize([('a', 2), ('d', 1), ('b', 1)])
>>> rdd3 = sc.parallelize(range(100))
>>> rdd4 = sc.parallelize([('a', ["x", "y", "z"]),
 ('b', ["p", "q", "r"])])
```

##### External Data

Read either one text file from HDFS, a local file system or any Hadoop-supported file system URL with `textFile()`, or read in a directory of text files with `wholeTextFiles()`.

```
>>> textFile = sc.textFile("/my/directory/*txt")
>>> textFile2 = sc.wholeTextFiles("/my/directory/*")
```

## Памятка по PySpark

## «О большое» (Big-O)

### Retrieving RDD Information

#### Basic Information

```
>>> rdd.getNumPartitions()
>>> rdd.count()
3
>>> rdd.collectByKey()
defaultdict(<type 'int'>, {'a':2, 'b':1})
>>> rdd.countByValue()
>>> rdd.collectAsMap()
{'a': 2, 'b': 1}
>>> rdd3.sum()
4950
>>> sc.parallelize([]).isEmpty()
True
```

List the number of partitions  
Count RDD instances  
Count RDD instances by key  
Count RDD instances by value  
Return (key,value) pairs as a dictionary  
Sum of RDD elements  
Check whether RDD is empty

#### Summary

```
>>> rdd3.max()
99
>>> rdd3.min()
0
>>> rdd3.mean()
49.5
>>> rdd3.stdev()
28.8660700472218
>>> rdd3.variance()
831.25
>>> rdd3.histogram(3)
([0,33,66,99], [33,33,34])
>>> rdd3.stats()
Summary statistics (count, mean, stdev, max & min)
```

Maximum value of RDD elements  
Minimum value of RDD elements  
Mean value of RDD elements  
Standard deviation of RDD elements  
Compute variance of RDD elements  
Compute histogram by bins  
Summary statistics (count, mean, stdev, max & min)

#### Applying Functions

```
>>> rdd.map(lambda x: x*(x[1],x[0]))
.collect()
[('a',7,7),('a',1),('a',2,2,'a'),('b',2,2,'b')]
>>> rdd5 = rdd.flatMap(lambda x: x*(x[1],x[0]))

>>> rdd5.collect()
[('a',7,7,'a','a',2,2,'a','b',2,2,'b')]
>>> rdd4 = rdd5.mapValues(lambda x: x)
.collect()
[('a','a'),('a','b'),('a','z'),('b','p'),('b','r')]
```

Apply a function to each RDD element  
Apply a function to each RDD element and flatten the result  
Apply a flatMap function to each (key,value) pair of `rdd4` without changing the keys

#### Selecting Data

```
>>> rdd.collect()
[('a', 7), ('a', 2), ('b', 2)]
>>> rdd.take(2)
[('a', 7), ('a', 2)]
>>> rdd.first()
('a', 2)
>>> rdd.top(2)
[('b', 2), ('a', 7)]

>>> rdd3.sample(False, 0.15, 81).collect()
[3, 4, 27, 31, 40, 41, 42, 43, 60, 76, 79, 80, 86, 97]

>>> rdd.filter(lambda x: "a" in x)
.collect()
[('a', 7), ('a', 2)]
>>> rdd5.distinct().collect()
['a', 2, 'b', 7]
>>> rdd.keys().collect()
['a', 'a', 'b']
```

Return a list with all RDD elements  
Take first 2 RDD elements  
Take first RDD element  
Take top 2 RDD elements  
Return sampled subset of `rdd3`  
Filter the RDD  
Return distinct RDD values  
Return (key,value) RDD's keys

#### Iterating

```
>>> def g(x): print(x)
>>> rdd.foreach(g)
('a', 7)
('b', 2)
('a', 2)
```

Apply a function to all RDD elements

### Reshaping Data

#### Reducing

```
>>> rdd.reduceByKey(lambda x,y : x+y)
.collect()
[('a',9),('b',2)]
>>> rdd.reduce(lambda a, b: a + b)
('a', 'a', 'b', 2)
```

Merge the RDD values for each key  
Merge the RDD values

#### Grouping by

```
>>> rdd3.groupByKey(lambda x: x % 2)
.mapValues(list)
.collect()
[('a', [7,2]), ('b', [2])]
```

Return RDD of grouped values  
Group RDD by key

#### Aggregating

```
>>> seqOp = (lambda x,y: (x[0]+y,x[1]+1))
>>> combOp = (lambda x,y:(x[0]*y[0],x[1]+y[1]))
>>> rdd1.aggregate((0,0),seqOp,combOp)
(4950,100)
>>> rdd2.aggregateByKey((0,0),seqOp,combOp)
.collect()
[('a',(9,2)), ('b',(2,1))]
>>> rdd3.fold(0,add)
4950
>>> rdd2.foldByKey(0, add)
.collect()
[('a',(9,2)), ('b',(2,1))]
>>> rdd3.keyBy(lambda x: x+x)
.collect()
```

Aggregate RDD elements of each partition and then the results  
Aggregate values of each RDD key  
Aggregate the elements of each partition, and then the results  
Merge the values for each key  
Create tuples of RDD elements by applying a function

#### Mathematical Operations

```
>>> rdd2.subtract(rdd2)
.collect()
[('b',2),('a',7)]
>>> rdd2.subtractByKey(rdd2)
.collect()
[('a', 1)]
>>> rdd2.cartesian(rdd2).collect()
[(), ()]
```

Return each RDD value not contained in `rdd2`  
Return each (key,value) pair of `rdd2` with no matching key in `rdd`  
Return the Cartesian product of `rdd2` and `rdd`

#### Repartitioning

```
>>> rdd2.repartition(4)
>>> rdd.coalesce(1)
```

New RDD with 4 partitions  
Decrease the number of partitions in the RDD to 1

#### Saving

```
>>> rdd.saveAsTextFile("rdd.txt")
>>> rdd.saveAsHadoopFile("hdfs://namenodehost/parent/child",
 'org.apache.hadoop.mapred.TextOutputFormat')
```

#### Stopping SparkContext

```
>>> sc.stop()
```

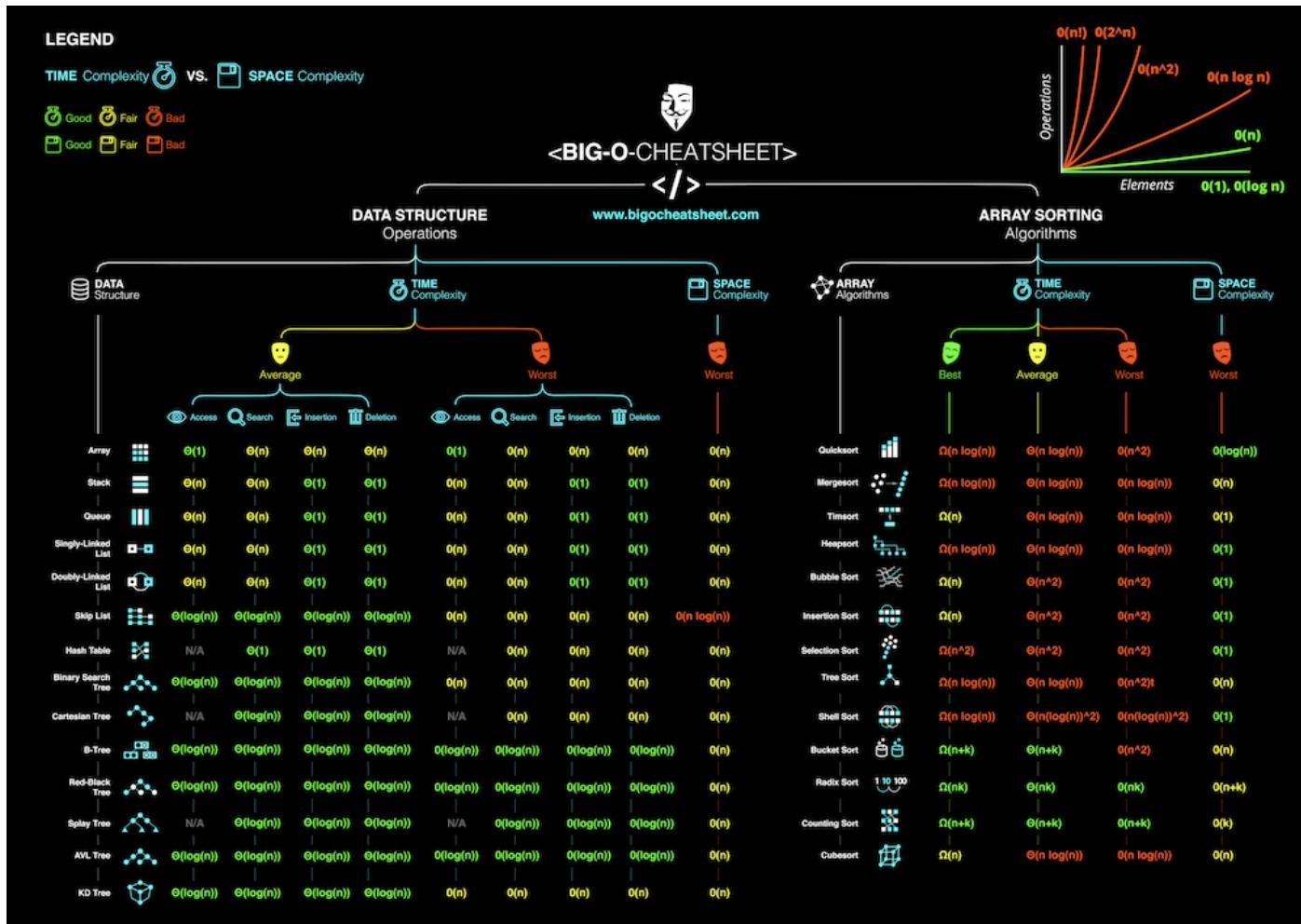
#### Execution

```
>>> ./bin/spark-submit examples/src/main/python/pi.py
```

DataCamp

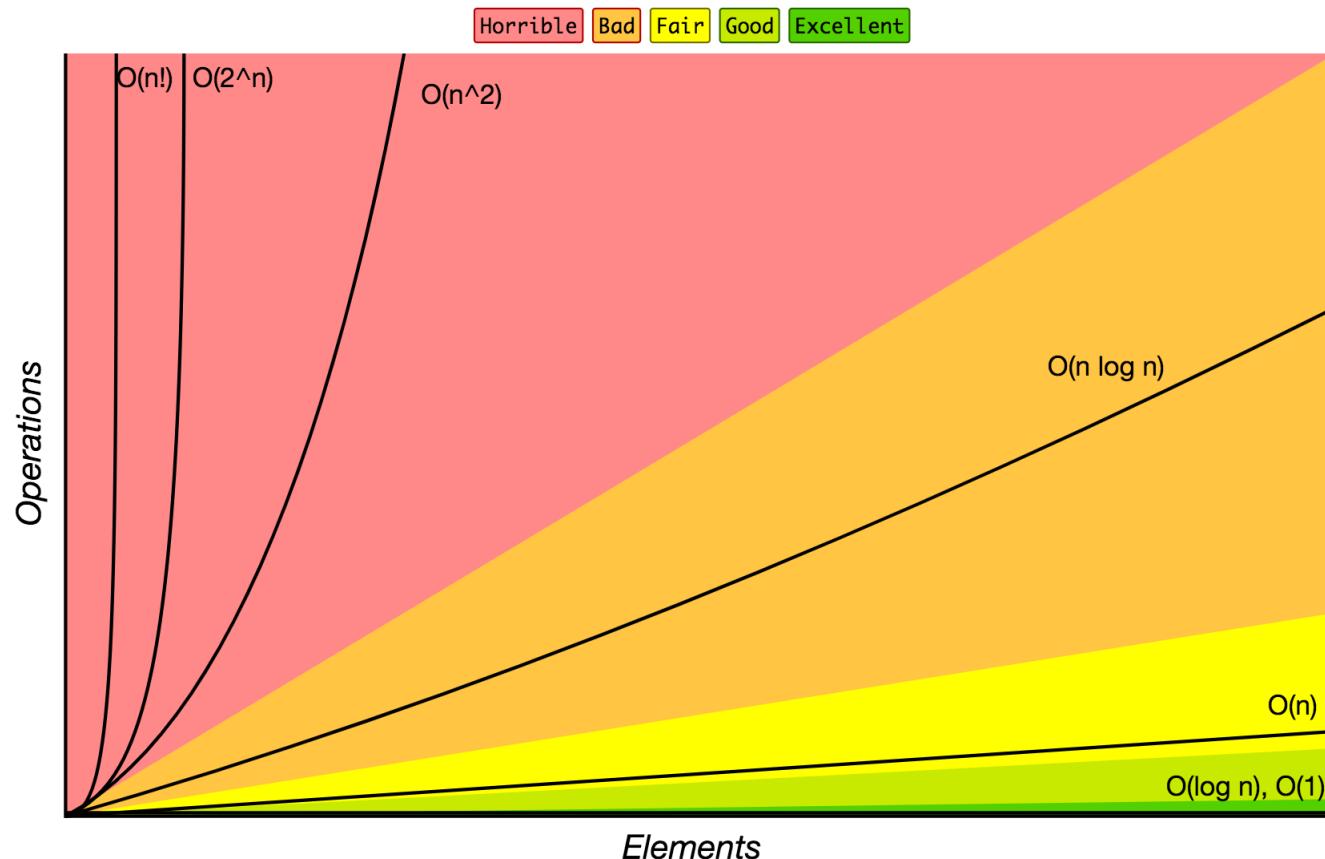
Learn Python for Data Science interactively





Памятка по сложности алгоритмов

## Big-O Complexity Chart



Памятка по сложности алгоритмов

## Common Data Structure Operations

| Data Structure     | Time Complexity   |                   |                   |                   |                   |                   |                   |                   | Space Complexity    |  |
|--------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|---------------------|--|
|                    | Average           |                   |                   |                   | Worst             |                   |                   |                   |                     |  |
|                    | Access            | Search            | Insertion         | Deletion          | Access            | Search            | Insertion         | Deletion          |                     |  |
| Array              | $\Theta(1)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$         |  |
| Stack              | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$         |  |
| Queue              | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$         |  |
| Singly-Linked List | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$         |  |
| Doubly-Linked List | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(1)$       | $\Theta(1)$       | $\Theta(n)$         |  |
| Skip List          | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n \log(n))$ |  |
| Hash Table         | N/A               | $\Theta(1)$       | $\Theta(1)$       | $\Theta(1)$       | N/A               | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$         |  |
| Binary Search Tree | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$         |  |
| Cartesian Tree     | N/A               | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | N/A               | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$         |  |
| B-Tree             | $\Theta(\log(n))$ | $\Theta(n)$         |  |
| Red-Black Tree     | $\Theta(\log(n))$ | $\Theta(n)$         |  |
| Splay Tree         | N/A               | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | N/A               | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(n)$         |  |
| AVL Tree           | $\Theta(\log(n))$ | $\Theta(n)$         |  |
| KD Tree            | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(\log(n))$ | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$       | $\Theta(n)$         |  |

Памятка по сложности операций со структурами данных в алгоритмах

# Array Sorting Algorithms

| Algorithm      | Time Complexity     |                        |                   | Space Complexity |
|----------------|---------------------|------------------------|-------------------|------------------|
|                | Best                | Average                | Worst             | Worst            |
| Quicksort      | $\Omega(n \log(n))$ | $\Theta(n \log(n))$    | $O(n^2)$          | $O(\log(n))$     |
| Mergesort      | $\Omega(n \log(n))$ | $\Theta(n \log(n))$    | $O(n \log(n))$    | $O(n)$           |
| Timsort        | $\Omega(n)$         | $\Theta(n \log(n))$    | $O(n \log(n))$    | $O(n)$           |
| Heapsort       | $\Omega(n \log(n))$ | $\Theta(n \log(n))$    | $O(n \log(n))$    | $O(1)$           |
| Bubble Sort    | $\Omega(n)$         | $\Theta(n^2)$          | $O(n^2)$          | $O(1)$           |
| Insertion Sort | $\Omega(n)$         | $\Theta(n^2)$          | $O(n^2)$          | $O(1)$           |
| Selection Sort | $\Omega(n^2)$       | $\Theta(n^2)$          | $O(n^2)$          | $O(1)$           |
| Tree Sort      | $\Omega(n \log(n))$ | $\Theta(n \log(n))$    | $O(n^2)$          | $O(n)$           |
| Shell Sort     | $\Omega(n \log(n))$ | $\Theta(n(\log(n))^2)$ | $O(n(\log(n))^2)$ | $O(1)$           |
| Bucket Sort    | $\Omega(n+k)$       | $\Theta(n+k)$          | $O(n^2)$          | $O(n)$           |
| Radix Sort     | $\Omega(nk)$        | $\Theta(nk)$           | $O(nk)$           | $O(n+k)$         |
| Counting Sort  | $\Omega(n+k)$       | $\Theta(n+k)$          | $O(n+k)$          | $O(k)$           |
| Cubesort       | $\Omega(n)$         | $\Theta(n \log(n))$    | $O(n \log(n))$    | $O(n)$           |

Памятка по сложности алгоритмов сортировки массива

## Источники

Памятка по сложности алгоритмов

[Памятка по Bokeh](#)  
[Памятка по Data Science](#)  
[Памятка по Data Wrangling](#)  
[Памятка по Ggplot](#)  
[Памятка по Keras](#)  
[Памятка по машинному обучению](#)  
[Памятка по машинному обучению](#)  
[Памятка по машинному обучению](#)  
[Памятка по Matplotlib](#)  
[Памятка по нейросетям](#)  
[Памятка по графикам нейросетей](#)  
[Нейросети](#)  
[Памятка по Numpy](#)  
[Памятка по Pandas](#)  
[Памятка по Pandas](#)  
[Памятка по PySpark](#)  
[Памятка по Scikit](#)  
[Памятка по Scikit-learn](#)  
[Памятка по Scipy](#)  
[Памятка по TensorFlow](#)

**Метки:** машинное обучение, модель, подсказка, памятка, большие данные, нейросеть





**NIX Solutions** 142,92  
Компания

---



**43,0** 124,8 43  
Карма Рейтинг Подписчики

@NIX\_Solutions

Пользователь

Поделиться публикацией



## ПОХОЖИЕ ПУБЛИКАЦИИ

1 декабря 2017 в 18:49

### Структура и модель выполнения .NET Core приложений

▲ +40    ⚡ 20,8k    📄 195    💬 10

23 мая 2017 в 11:01

### Как организовать большое React-приложение и сделать его масштабируемым

▲ +29    ⚡ 25,2k    📄 151    💬 16

15 марта 2017 в 11:29

### Модели памяти, лежащие в основе языков программирования

▲ +33    ⚡ 24,4k    📄 191    💬 10

## Комментарии 9

 DmitriyDev 24.07.18 в 17:42  

↑ 0 ↓

Большое человеческое спасибо!



phenik 24.07.18 в 17:44



↑ +3 ↓

Напоминает шпаргалки для сдачи экзаменов.

НЛО прилетело и опубликовало эту надпись здесь



delph 24.07.18 в 22:18



↑ 0 ↓

Отличная сборка, спасибо!



Dark\_Daiver 25.07.18 в 08:03



↑ +4 ↓

Если честно, то я бы убрал первую картинку, т.к. она откровенно вредная. Мало того, что она толком ничего не объясняет — одной топологии недостаточно чтобы объяснить чем отличается один тип нейронной сети от другой, так еще и топология местами нарисована с ошибками.

К примеру

- Что такое kernel в сверточной сети (deep convolutional network), и почему он идет перед первыми свертками?
- Чем отличаются feed forward и radial basis network?
- Почему у SVM несколько скрытых слоев? Если речь идет о нелинейных SVM, то возможно стоило влепить kernel сюда? Что за хитрые скрытые слои, которые принимают по одному входу? Ну и строго говоря причислять SVM к Neural Networks это несколько грубо.
- Почему в GAN дважды встречается Match input output Cell? Что такое Match input output Cell в принципе? Если это просто штука которая сравнивает поэлементно входы-выходы, то в GAN такого в чистом виде не происходит.
- Почему skip connections в Deep Residual Network связывают только часть нейронов? Строго говоря, не совсем корректно показывать skip connections так же как и обычные соединения. В DRN skip connections это всегда сложение, более сложные случаи уже не называют residual (Dense connections, например).



Calc 25.07.18 в 11:56



↑ 0 ↓

Наоборот, тем кто далек от этого это визуальное представление, что есть что то кроме простого перцептрона и 2x уровневой сетки. А дальше можно самому искать информацию по каждому типу. Много

раз видел поддержку сетей в opencv, но не так интересен просто список, сколько визуальное представление.

ps работа не связана с нейронными сетями, просто интерес

 Dark\_Daiver 25.07.18 в 20:21    

 0 

имхо, это не повод городить плохие «шпаргалки». На крайний случай можно было вставить топологии тех же Inception/ResNet — смотрятся солидно и при этом отображают реальную топологию.

 Iumaxy 25.07.18 в 18:17  

 0 

Только у меня некоторые картинки слишком мелкие, чтобы прочитать часть текста? «Открыть картинку в новой вкладке» не помогает, надо идти по ссылке в конце статьи.

 NIX\_Solutions 26.07.18 в 12:42    

 0 

Картинки перезалили.

Только [полноправные пользователи](#) могут оставлять комментарии. [Войдите](#), пожалуйста.

## САМОЕ ЧИТАЕМОЕ

[Сутки](#) [Неделя](#) [Месяц](#)

## Как я переехал в Израиль после блокировки Telegram

↑ +131    ⚡ 54,6k    📄 86    💬 364

## От разговорника вместо физики до CarPrice, или как я обосновался в Японии

↑ +54    ⚡ 22,6k    📄 60    💬 82

## Сапожники с сапогами: какие смартфоны выбирают для себя эксперты

↑ +17    ⚡ 35k    📄 24    💬 72

## Средства разработки для платформы «Байкал-T1» перешли на российский дистрибутив АЛТ

↑ +40    ⚡ 20,1k    📄 21    💬 128

## «Ручной» манипулятор

↑ +60    ⚡ 12,6k    📄 86    💬 15

### Аккаунт

[Войти](#)

[Регистрация](#)

### Разделы

[Публикации](#)

[Хабы](#)

[Компании](#)

[Пользователи](#)

[Песочница](#)

### Информация

[Правила](#)

[Помощь](#)

[Документация](#)

[Соглашение](#)

[Конфиденциальность](#)

### Услуги

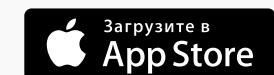
[Реклама](#)

[Тарифы](#)

[Контент](#)

[Семинары](#)

### Приложения



© 2006 – 2018 «TM»

[О сайте](#)

[Служба поддержки](#)

[Мобильная версия](#)

