Introduction to Machine Learning

Fedor Ratnikov

NRU Higher School of Economics,
Yandex School of Data Analysis

Credits

- This presentation have been prepared with a great use of well established academic courses and ideas from
  - K. Vorontsov
    - http://www.machinelearning.ru/wiki/index.php?title=Участник:Vokov
  - A. Artemov
    - https://www.hse.ru/en/org/persons/200432883
  - A. Rogozhnikov
    - http://arogozhnikov.github.io
- ... with a great assistance from A. Panin

Outlook

- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Illustrations
- Concluding remarks

Outlook

- Models and data
  - fundamental and Generic Models
  - types of Generic Models
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Fundamental Models

- Paradigm: **model first, then data**
- Some specific dependency is assumed *a priori* (e.g. a Nature low)
  - "fit parabola to data"
- Number of degrees of freedom corresponds to the problem
  - smaller dimensions
    - faster and more stable converging
- Extendable beyond the domain of train parameters
- Interpretable representation
- Can not accommodate difference between parametric model and actual data

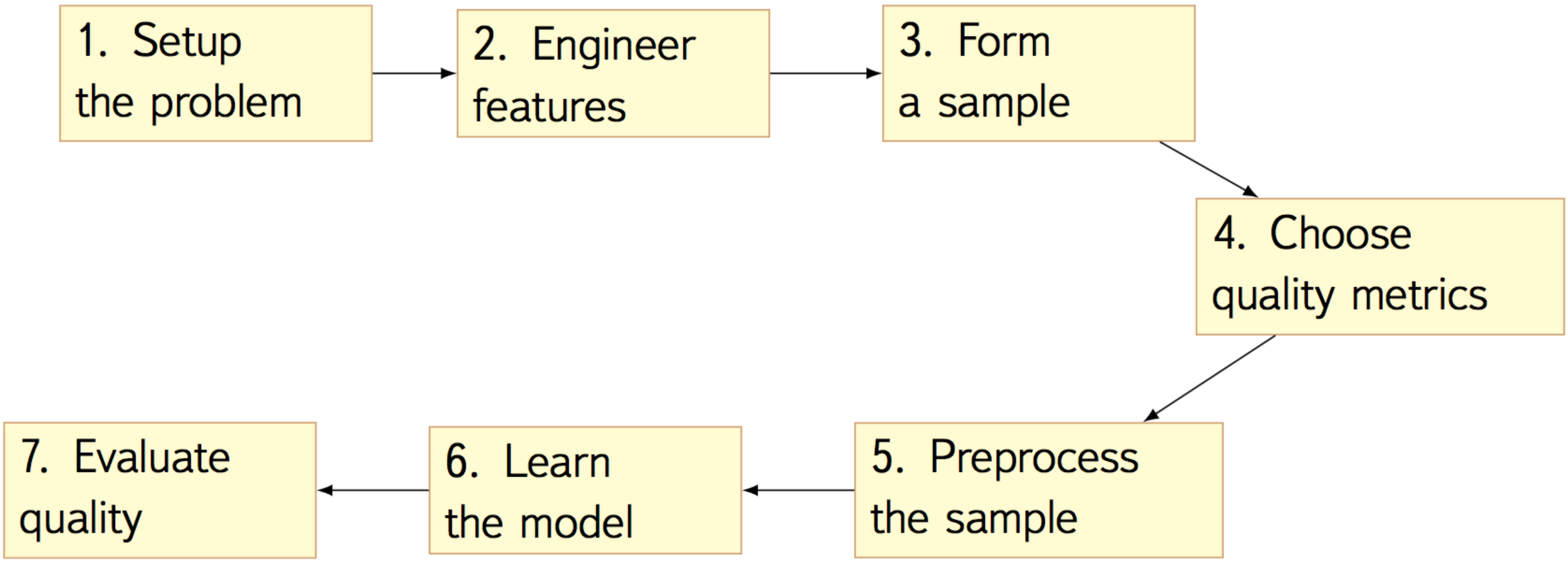
Generic Models

- Paradigm: **data first, then model**
  - assumes nothing about dependency *a priori*
  - "find reasonable representation for data"
    - model ≡ data
  - universal - can accommodate most actual data
  - significantly over-defined problem
    - special regularisations are needed
    - less stable converging
    - significant computing resources needed
      - many speed up tricks are developed
  - extrapolation can not be trusted at all
  - hard to interpret
- Machine Learning is all about Generic Models

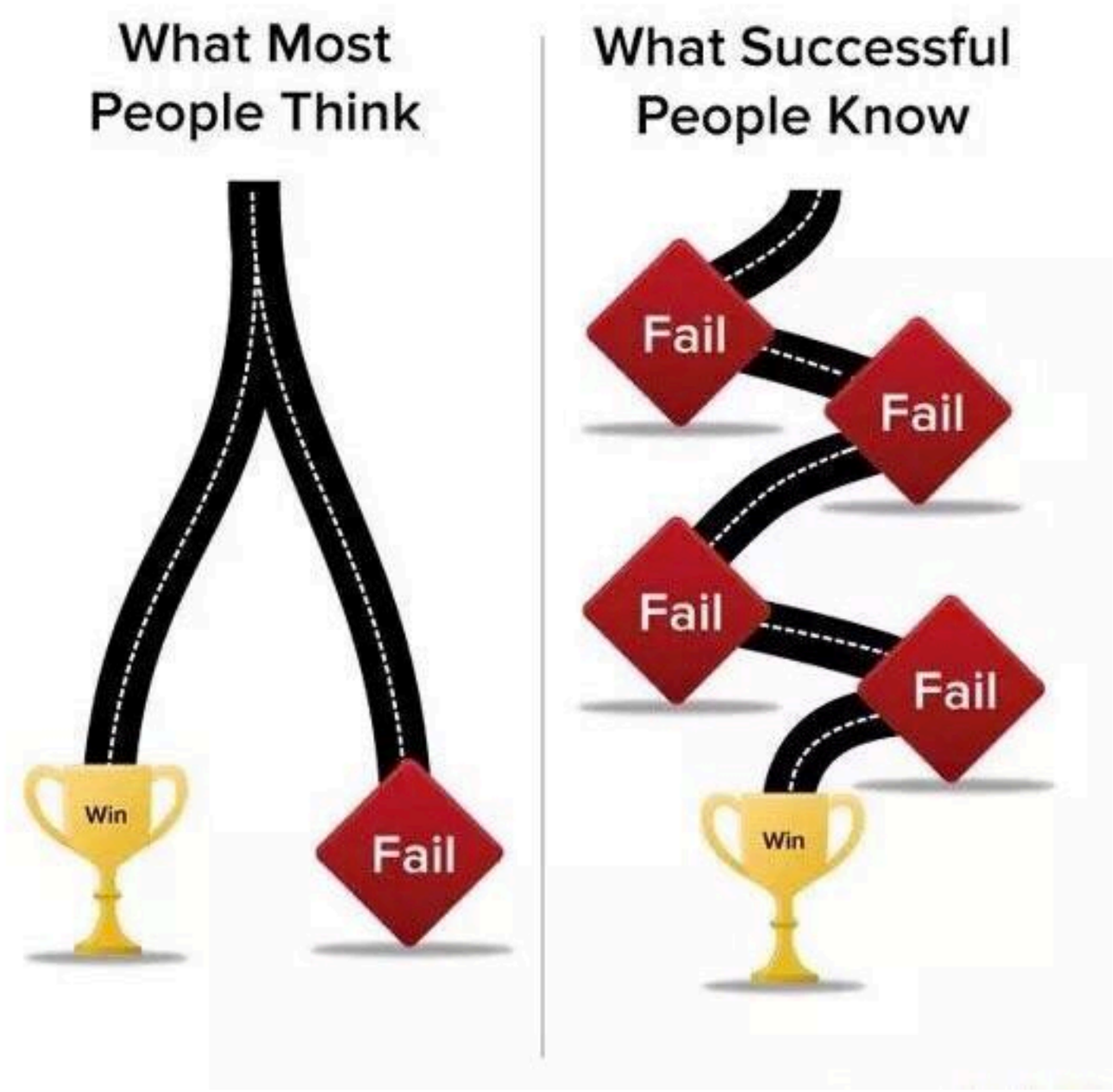
Types of Learning

- There is a set of well labelled data
  - we want to learn how to produce labels for more data like this
    - automation of routine
      - **supervised learning**
- There is a set of labelled data, and unlabelled data which are assumed to be similar
  - we want to derive labels
    - propagation of knowledge
      - **supervised learning**
- There is a set of unlabelled data with some properties
  - we want to infer properties of this data
    - inferring new knowledge
      - **unsupervised learning**

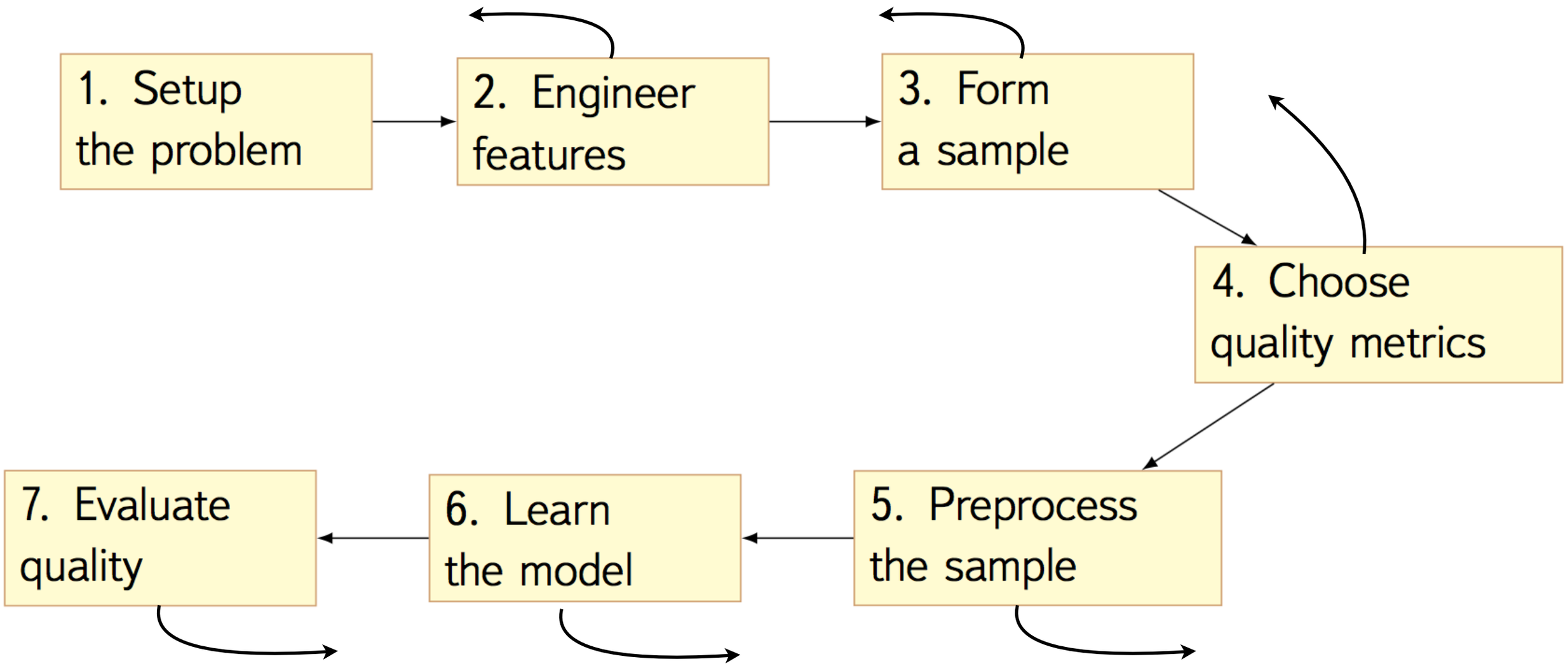
Machine Learning Pipeline



Machine Learning Pipeline



Machine Learning Pipeline



Outlook

- Models and data
- Formal set of the ML problem
  - main concepts and definitions
  - KNN classification example
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Basic Problem of the Supervised Learning

X - set of objects

Y - set of answers

$y : X \to Y$ - unknown dependency

Given:

$\{x_1, x_2, \ldots, x_k\} \subset X$ - *training sample*

$\{y_1, y_2, \ldots, y_k\} \subset Y$ - answers for objects in training sample

Target:

find $a : X \to Y$ - *decision function* approximating y on the full dataset X

Machine Learning is all about:

- what are possible objects and answers?
- what is the meaning of "*a* approximates *y*"?
- how to build function *a*?

Object Features

Objects of X are completely defined by set of features:

$f_j : X \to D_j$. $j=1, 2, \ldots, n$ - object *features*

$D_j = \{0,1\}$ - *binary feature*

e.g. {“charged”, “neutral”} for particles

$D_j = \{c_1, c_2, \ldots, c_l\}$ - *categorial feature*

e.g. {“electron”, “muon”, “pion”, “kaon”, “proton”}

$D_j = \mathbb{R}$ - *numeric feature*

e.g. subatomic particle momentum

MC particle: $f : x \to \{PDGid, p_x, p_y, p_z, v_x, v_y, v_z\}$ $n=7$

Types of the Supervised ML Problems

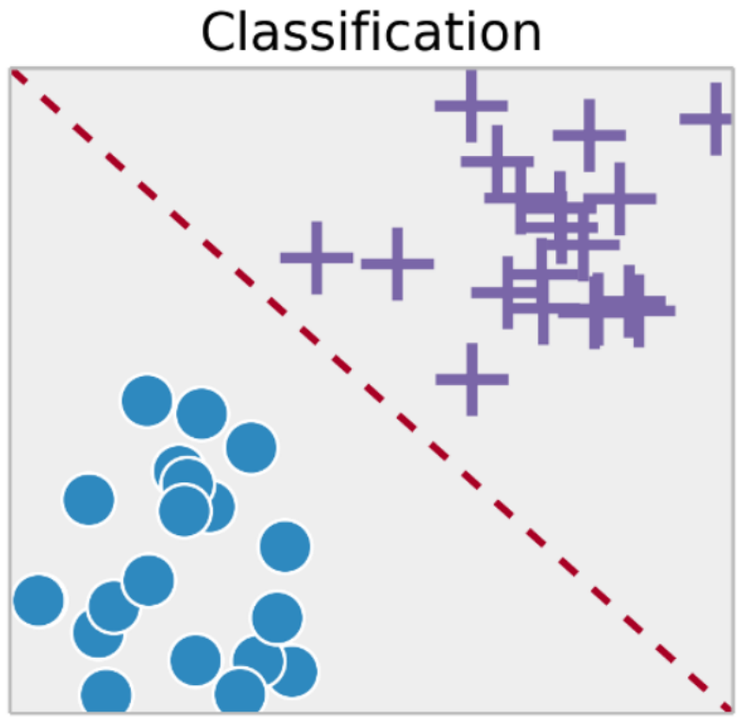
Classification:

$Y = \{-1, +1\}$ - two-class classification

e.g. “signal”, “background”

$Y = \{1, 2, \ldots, M\}$ - multi-class classification

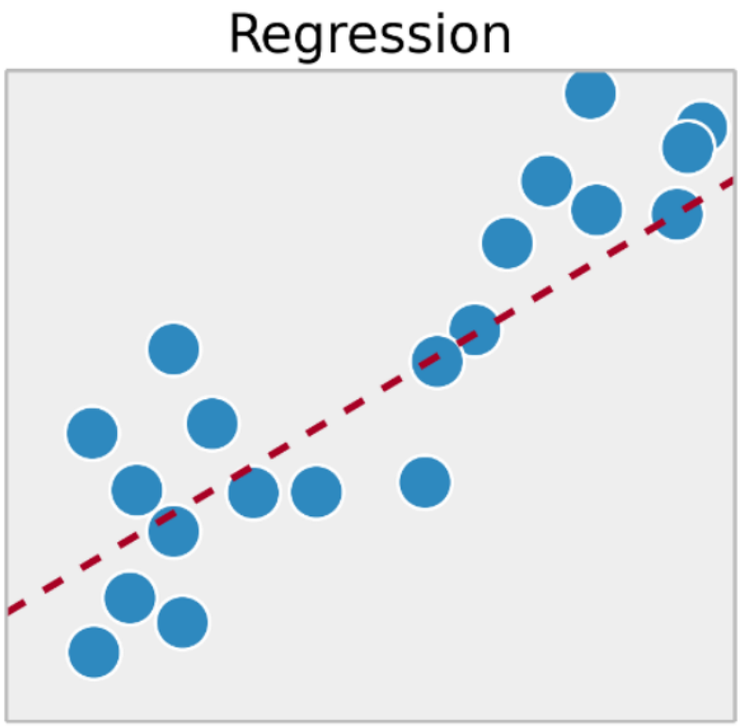
e.g. particle ID



Regression:

$Y = \mathbb{R}$ or $Y = \mathbb{R}^m$

e.g. reconstructed subatomic particle kinematics

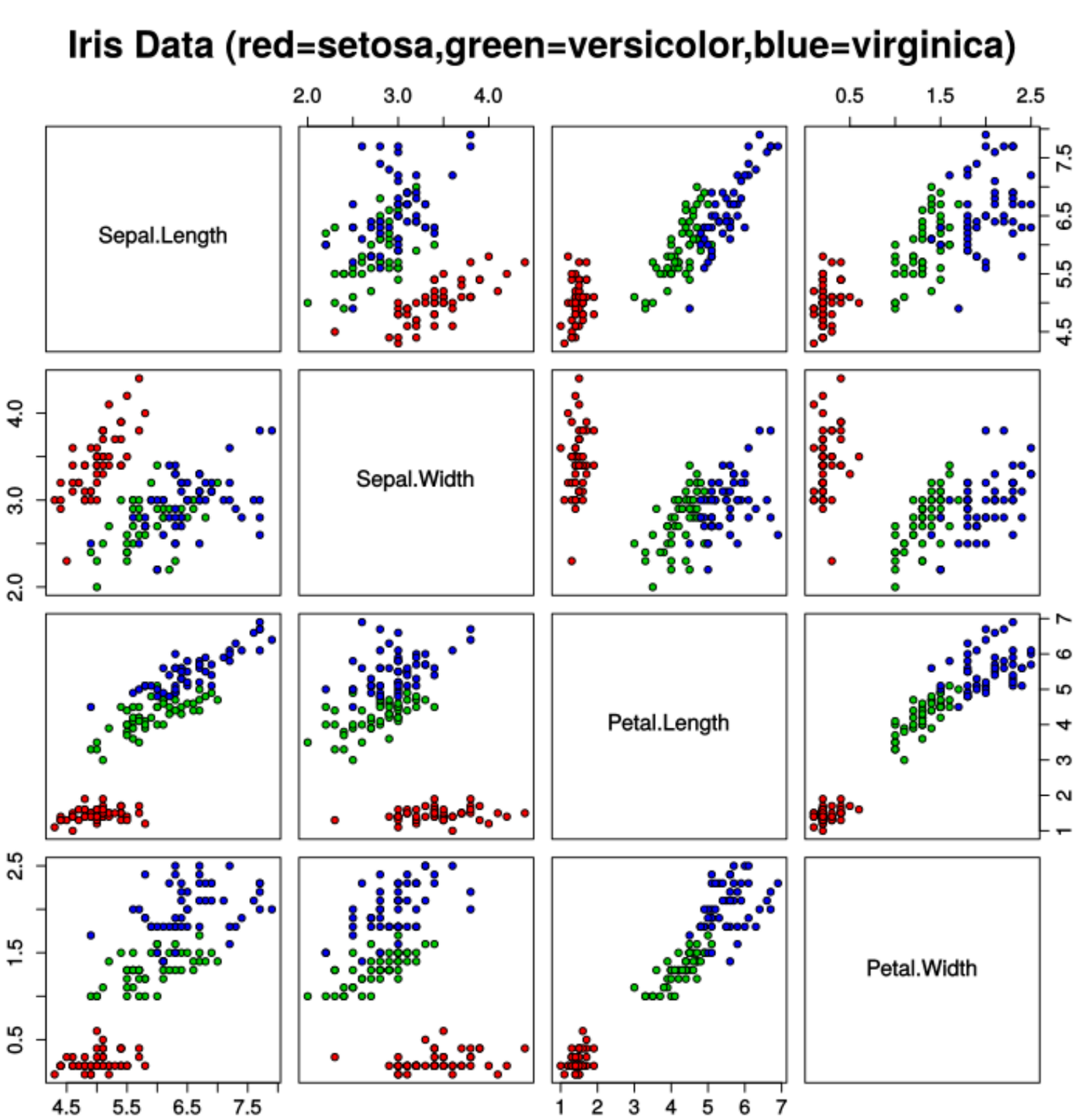


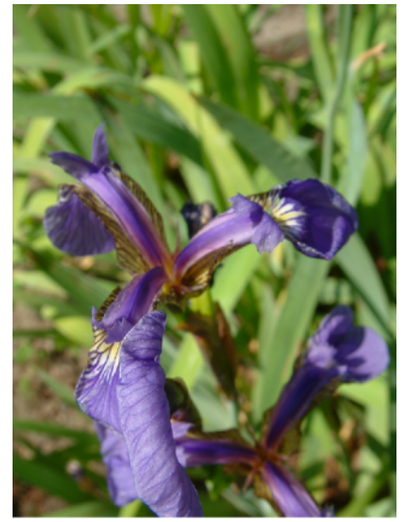
grey area:

classification may be in terms of probabilities

regression may be in terms of e.g. integers

Iris Flowers Dataset (Fisher, 1936)





4 features, 3 classes, 150 objects

Predictive Model

$$A = \{ g(x, \theta) \mid \theta \in \Theta \}$$

where:

g: X $\times$ Θ → *Y* - well defined function

Θ - set of possible values for parameter θ

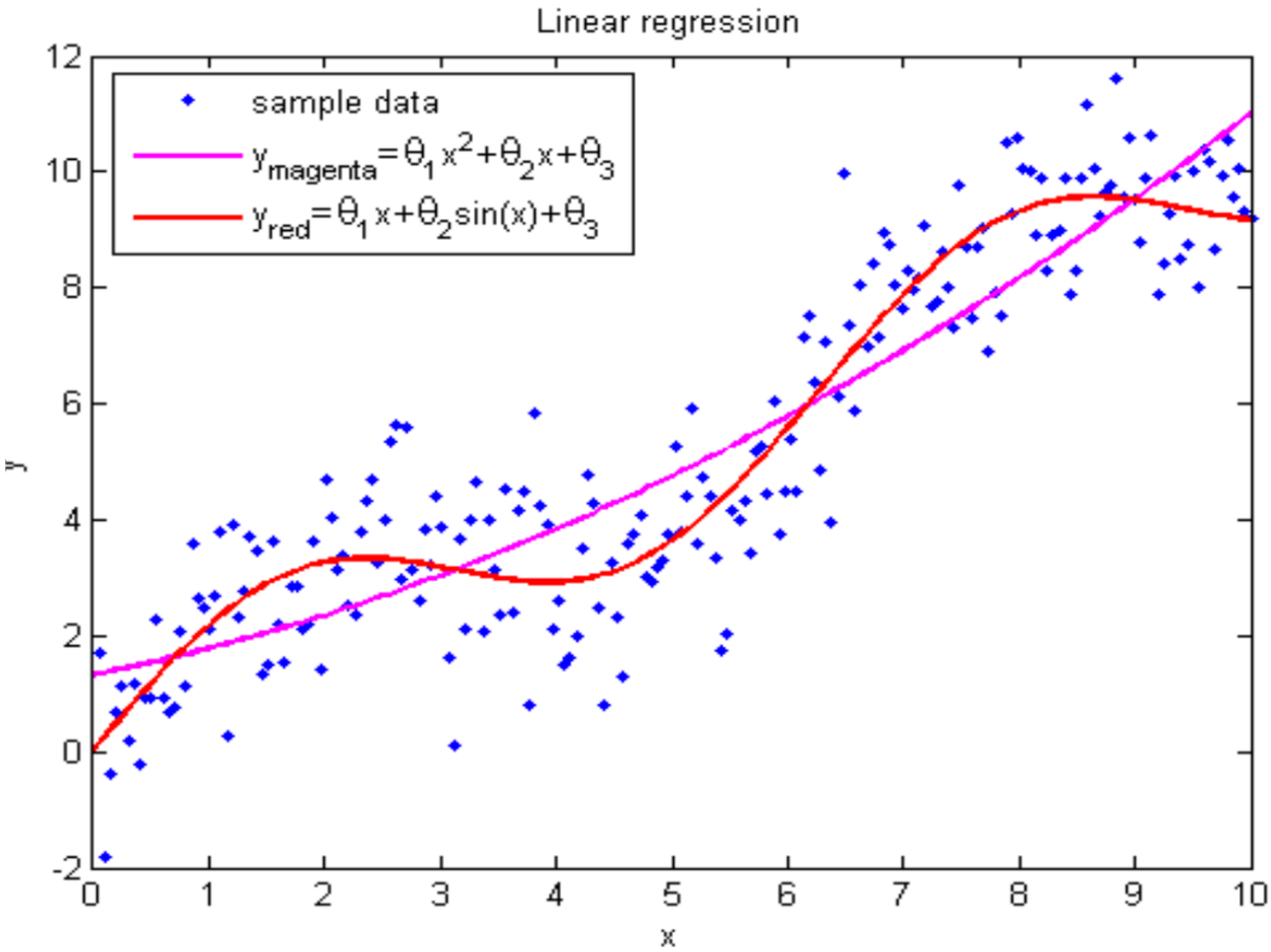
e.g. linear model with parameter vector $\theta = (\theta_1, \ldots, \theta_n), \ \Theta = \mathbb{R}^n$

$$g(x, \theta) = \sum_{j=1}^{n} \theta_j f_j(x)$$ - regression

$$g(x, \theta) = \text{sign} \sum_{j=1}^{n} \theta_j f_j(x)$$ - classification

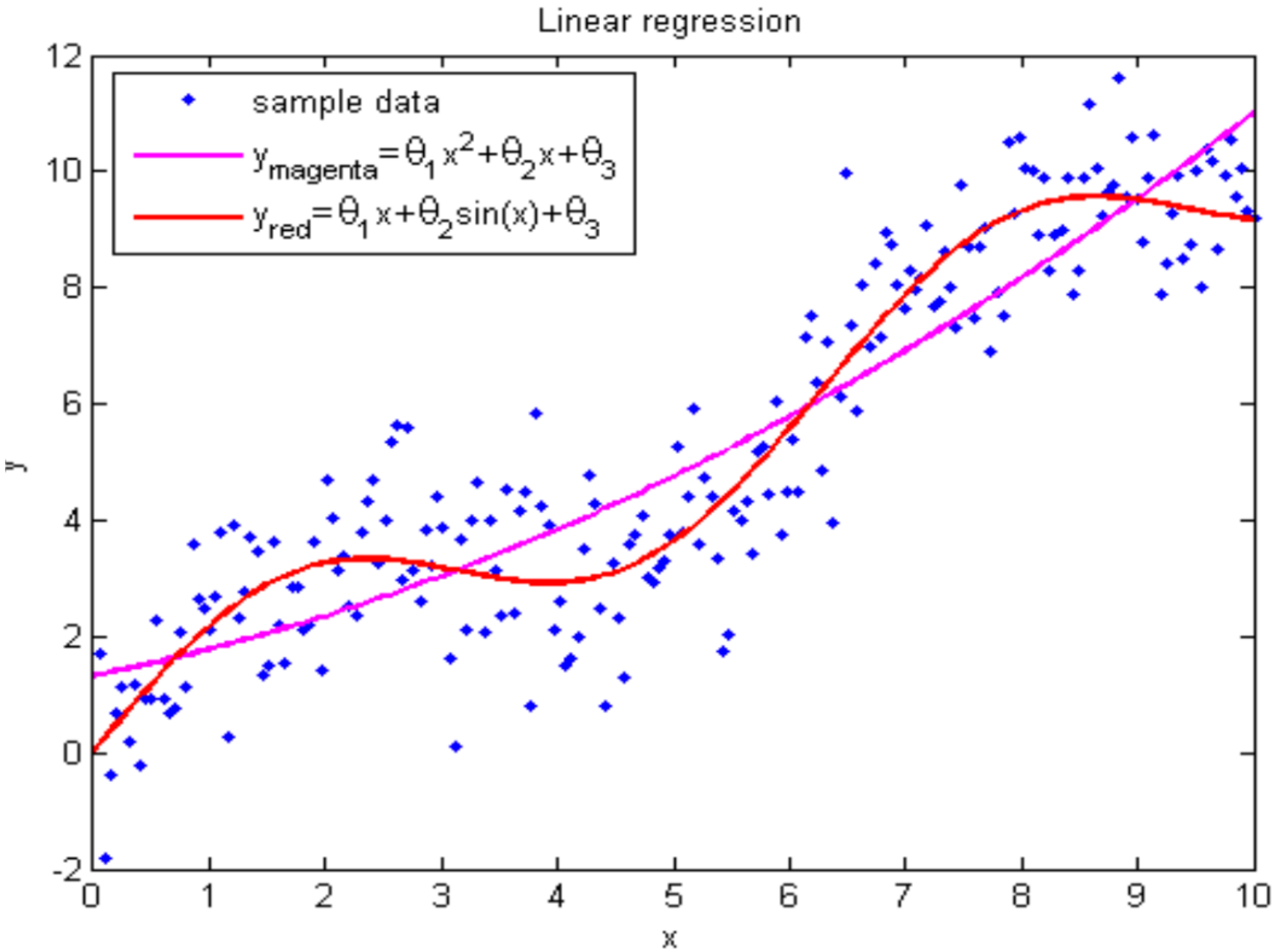
Features May Be Different

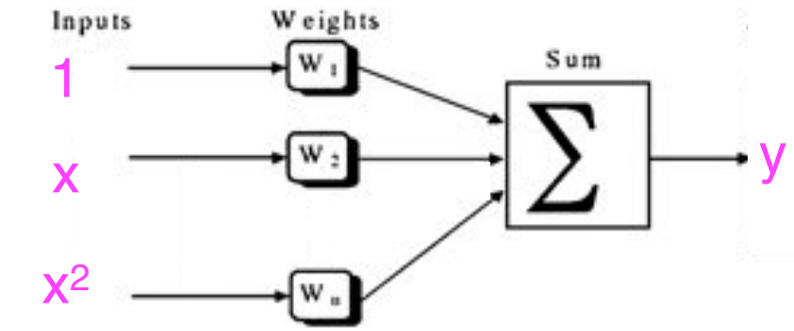
Regression, X = Y = $\mathbb{R}$, *3* features*: {x, x^2, 1}* or *{x, sin(x), 1}*

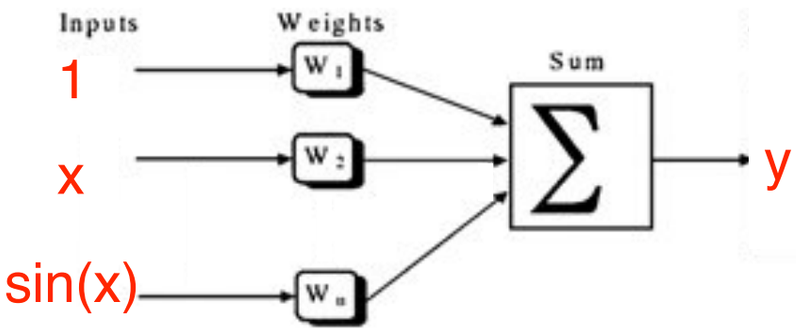


Features May Be Different

Regression, X = Y = $\mathbb{R}$, *3* features*: {x, x^2, 1}* or *{x, sin(x), 1}*







Learning Algorithm

learning algorithm μ is a function to convert a particular data sample into predictive model:

$\mu : \{(x_i, y_i)\}_{i=1..l} \rightarrow a \in A$

$$A = \{g(x, \theta) \mid \theta \in \Theta\}$$

Two stages for the supervised learning

training: μ builds a using train set $\{(x_i, y_i)\}_{i=1..l}$

testing: a predicts y for $x \in X$

Train and Test

$$\begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots \\ f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix} \xrightarrow{y} \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix} \xrightarrow{\mu} a$$

$$\begin{pmatrix} f_1(x'_1) & \dots & f_n(x'_1) \\ \dots & \dots & \dots \\ f_1(x'_k) & \dots & f_n(x'_k) \end{pmatrix} \xrightarrow{a} \begin{pmatrix} a(x'_1) \\ \dots \\ a(x'_k) \end{pmatrix}$$

Loss Function

What is the meaning of "**a approximates y**"?

loss function: $\mathscr{L}(x,a)$ — prediction error of algorithm $a \in A$ on the object $x \in X$

classification:

$\mathscr{L}(x,a) = [a(x) \neq y(x)]$ — error indicator

regression:

$\mathscr{L}(x,a) = |a(x) - y(x)|$ — absolute error

$\mathscr{L}(x,a) = (a(x) - y(x))^2$ — quadratic error

…

empirical risk: mean loss on the data sample X^l:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i)$$

Nearest Neighbour Classification

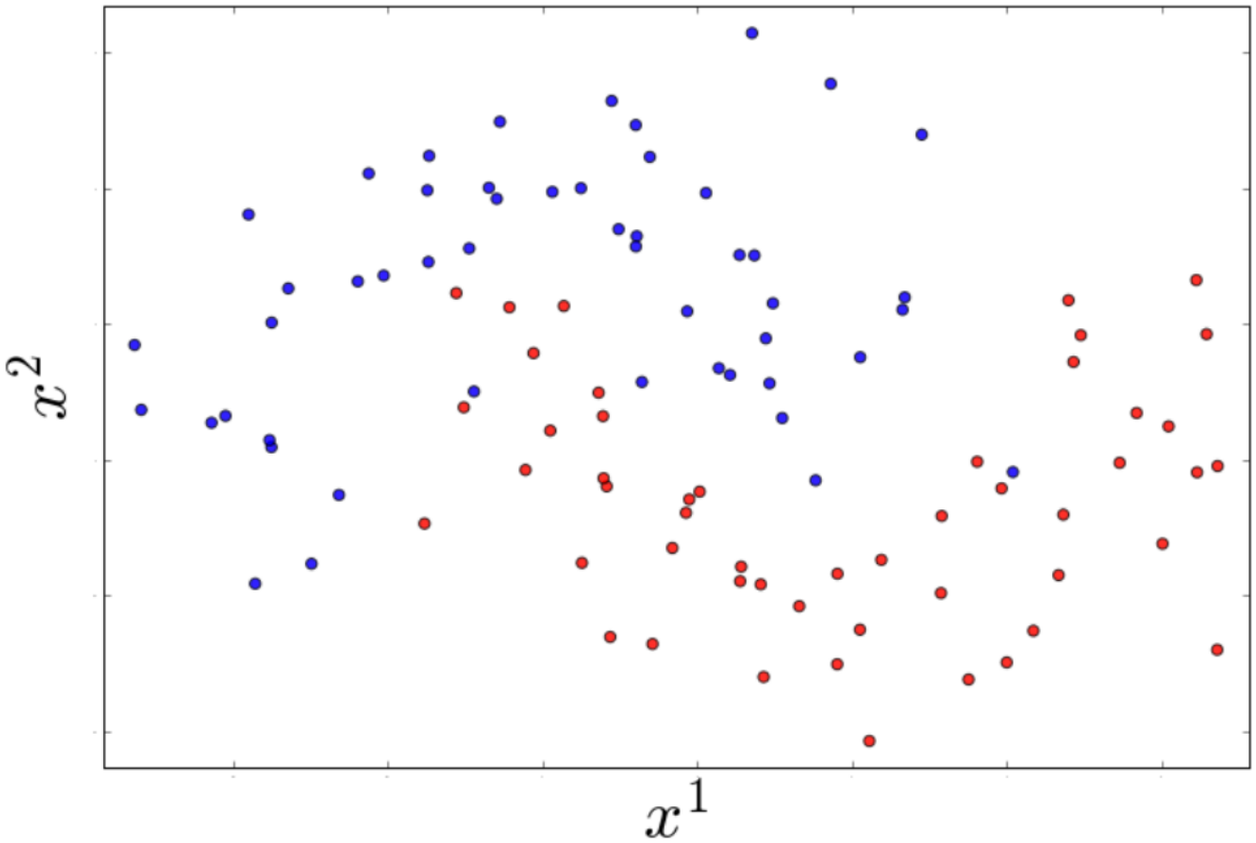
we have a train sample $X = \{x_i, y_i\}$, $i=1..l$

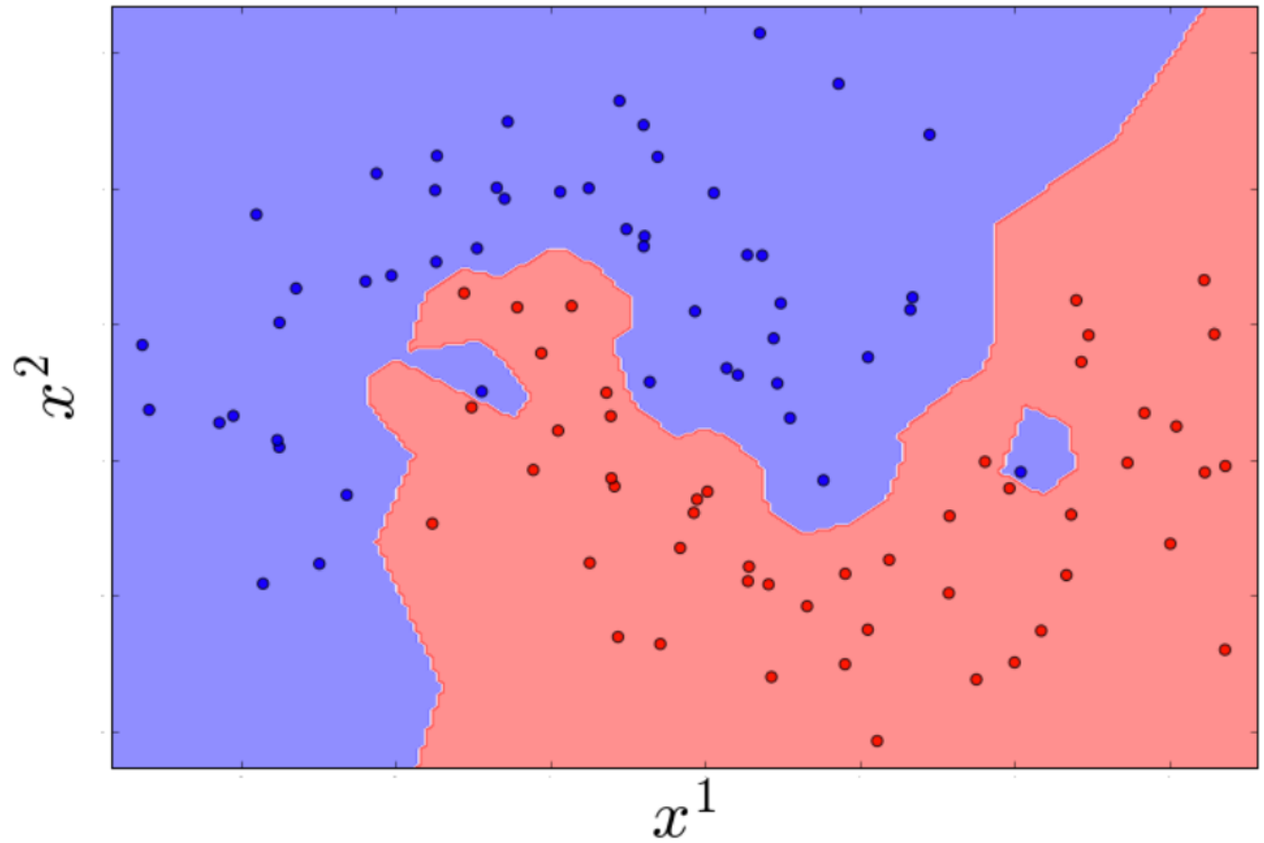
there is a distance $\rho\ (x', x'')$ defined in space of features

for any x algorithm predicts $y = y_i$, $i = arg\ min\ \rho\ (x, x_i)$

algorithm: search in train dataset, no training *per se*

2-class Classification With 2 Features

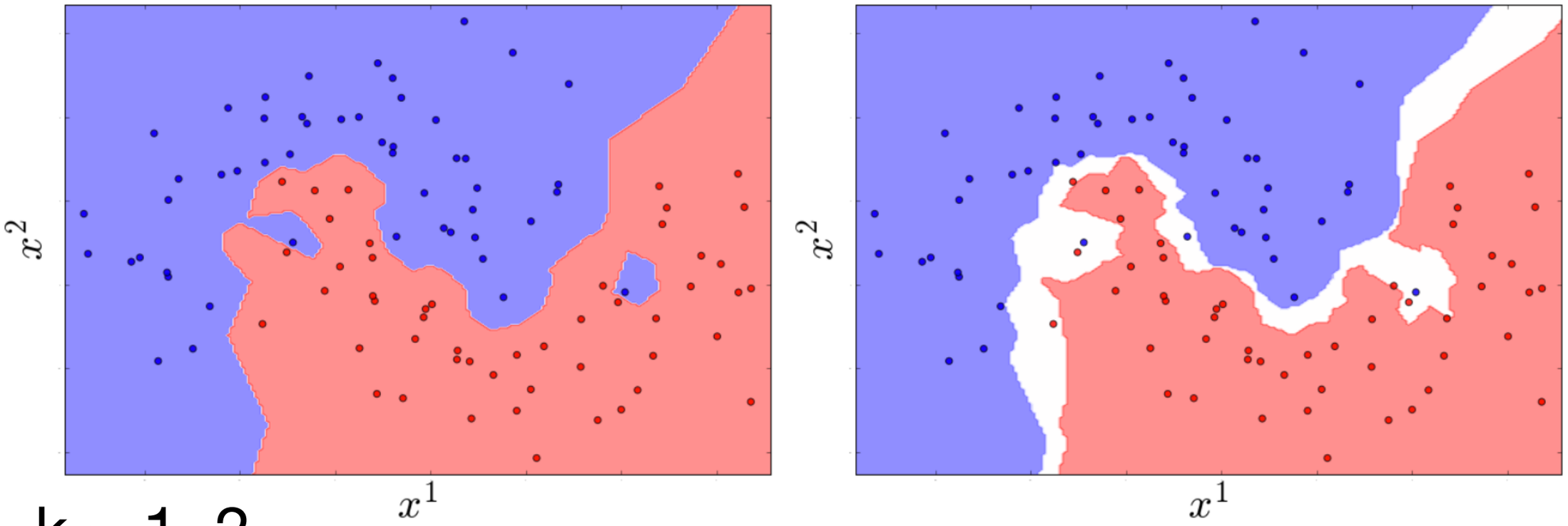




k Nearest Neighbours (kNN)

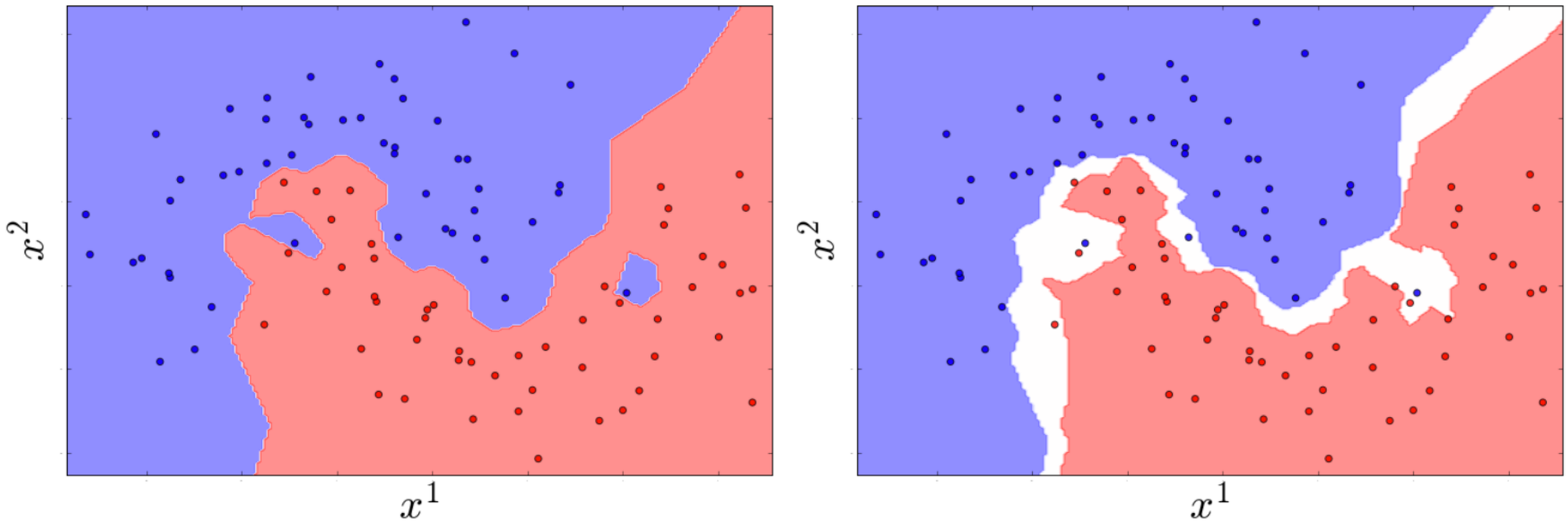
using k neighbours improves stability

p(x) = # (kNN events in X of class {1}) / k



k = 1, 2

k Nearest Neighbours (kNN)



k = 1, 2, 5, 30

Overfitting

What is the quality of classification on training dataset with k=1?

answer: it is ideal (closest neighbour is the event itself)

quality is lower when k>1

however this doesn't mean k=1 is the best

it means we **cannot use training events to estimate quality**

when classifier's decision rule is too complex it captures details from training data that are not relevant to distribution

we call this an overfitting (more on this later)

Outlook

- Models and data
- Formal set of the ML problem
- Practical points
  - optimization
  - overfitting
  - regularization
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Supervised Learning IS Optimization

empirical risk:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i)$$

learning algorithm:

$$\mu(X^\ell) = \arg\min_{a \in A} Q(a, X^\ell)$$

e.g. least squares method:

$$\mu(X^\ell) = \arg\min_{\theta} \sum_{i=1}^{\ell} \left(g(x_i, \theta) - y_i\right)^2$$

How Generic is Result?

$$\mu(X^\ell) = \arg\min_\theta \sum_{i=1}^{\ell} \left(g(x_i, \theta) - y_i\right)^2$$

- is $g(x,\theta)$ generic for all $x \in X$, or is it tuned for train set X^l?
- will $Q(a, X^k)$ keep being small on different, test subset of X?

Overtraining problem

Illustration

consider function:

$$y(x) = \frac{1}{1 + 25x^2} \qquad x \in [-2, 2]$$

features — powers of x: $x \mapsto (1, x^1, x^2, \ldots, x^n)$

linear regression: $a(x, \theta) = \theta_0 + \theta_1 x + \cdots + \theta_n x^n$

use least squares learning algorithm:

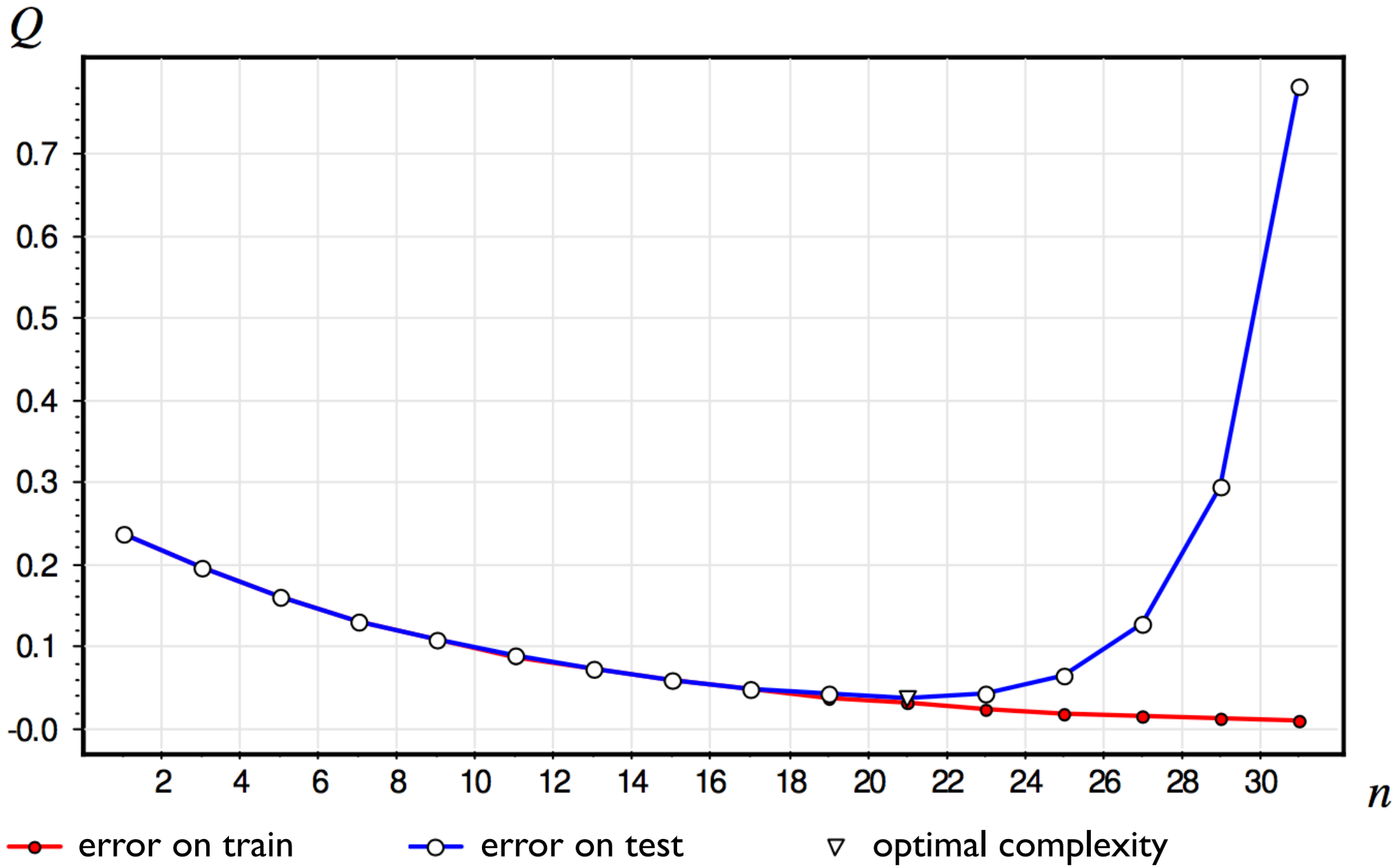
$$Q(\theta, X^\ell) = \sum_{i=1}^{\ell} (\theta_0 + \theta_1 x_i + \cdots + \theta_n x_i^n - y_i)^2 \to \min_{\theta_0, \ldots, \theta_n}.$$

train sample: $X^\ell = \left\{ x_i = 4\frac{i-1}{\ell-1} - 2 \;\middle|\; i = 1, \ldots, \ell \right\}$

test sample: $X^k = \left\{ x_i = 4\frac{i-0.5}{\ell-1} - 2 \;\middle|\; i = 1, \ldots, \ell - 1 \right\}$

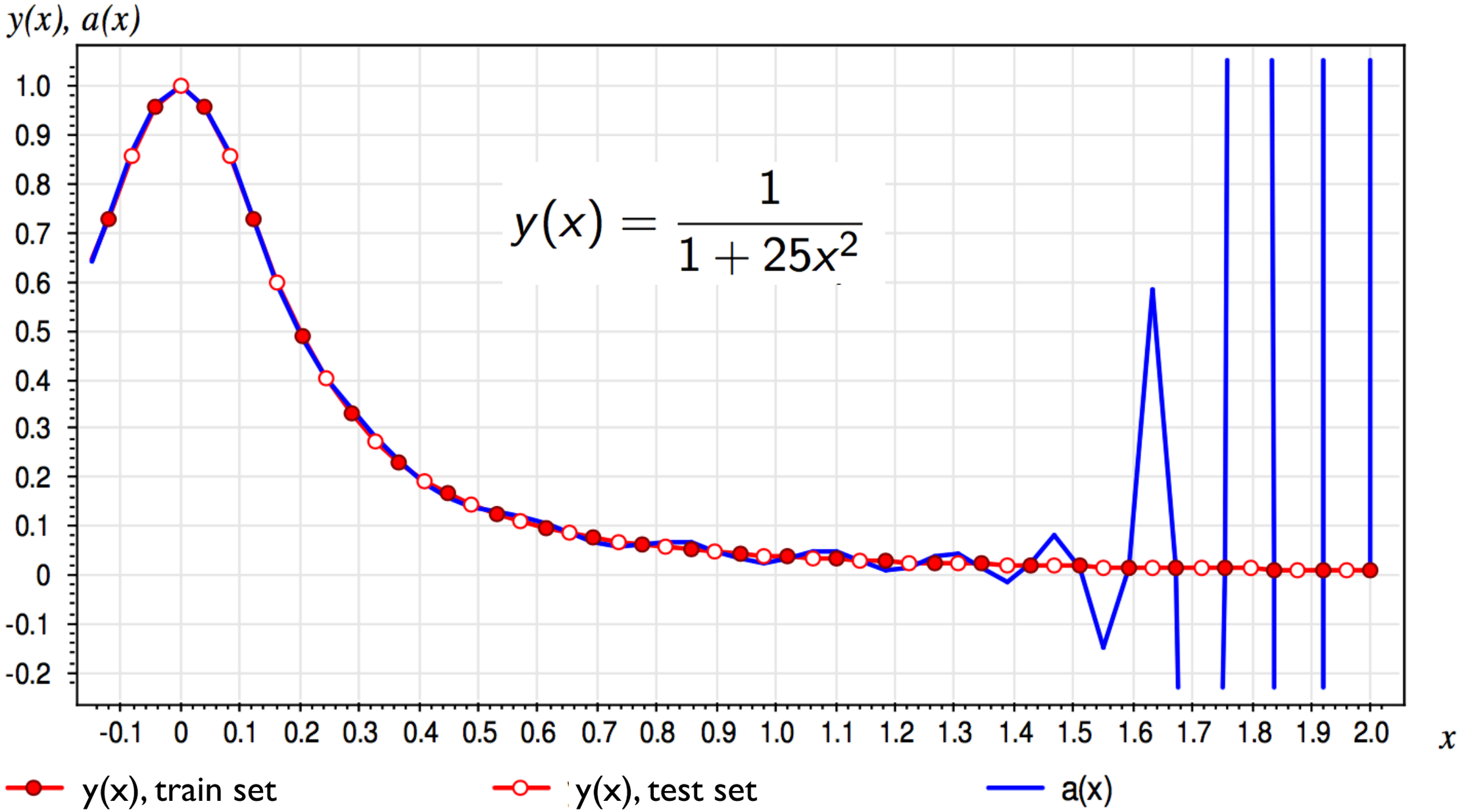
How Q behaves as n increases?

Empirical Risks (n=1..31)



overtraining (overfitting): $Q\left(\mu(X^\ell), X^k\right) \gg Q\left(\mu(X^\ell), X^\ell\right)$

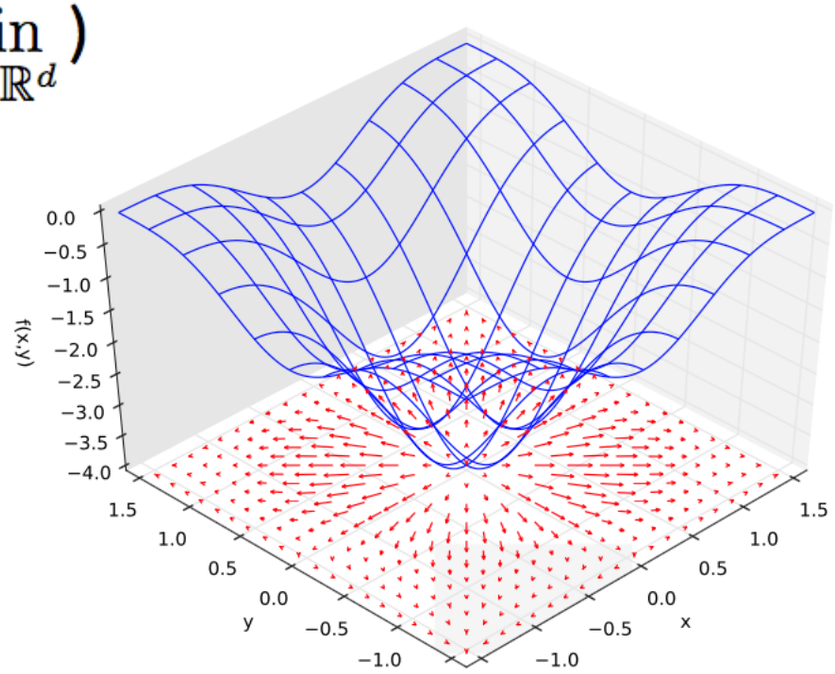
$a(x) = P_{38}(x)$


y(x), a(x)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
x
$y(x) = \frac{1}{1 + 25x^2}$
y(x), train set
y(x), test set
a(x)

Numerical Optimization

Consider optimization problem:

$$f(\mathbf{x}) \to \min_{\boldsymbol{x}\in\mathbb{R}^d} \qquad \text{(such as } \sum_{i=1}^{\ell}\Big(y_i - \sum_{k=1}^{d} w_k x_{ki}\Big)^2 \to \min_{\boldsymbol{w}\in\mathbb{R}^d}\text{)}$$

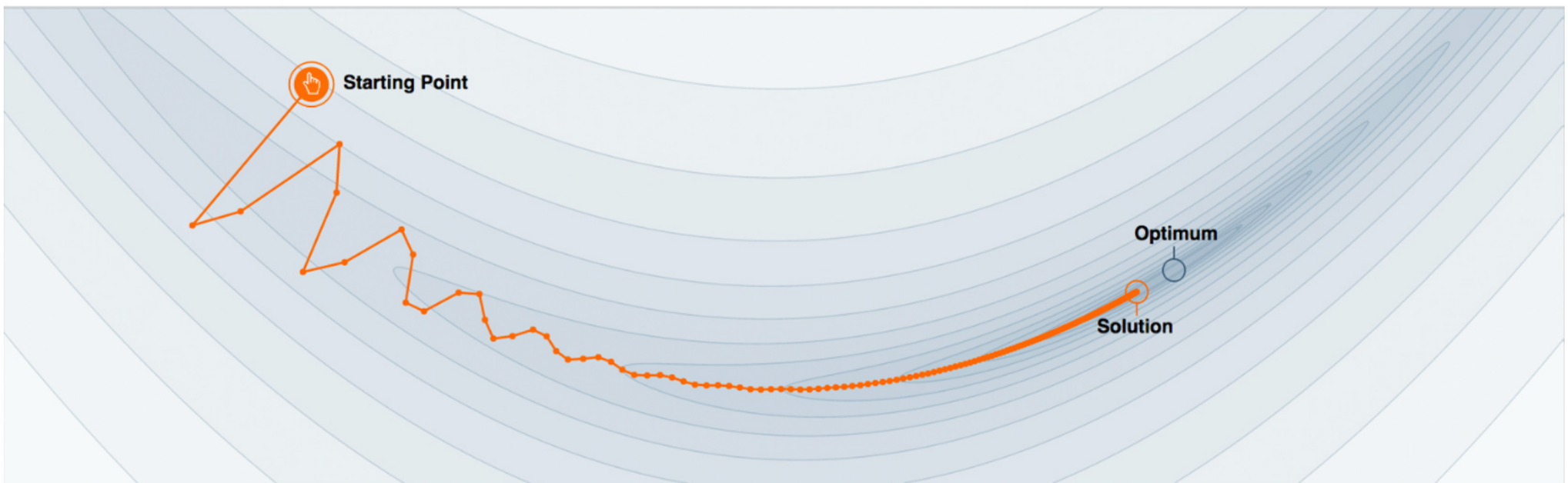


Usually solved by **gradient descent** method

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \equiv \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d}\right)$$

$$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} - \alpha_k \nabla_{\mathbf{x}} f(\mathbf{x}^{(k-1)})$$

converges as $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) = \mathcal{O}(1/k)$ $0<\alpha_k<1$ — **learning rate**



Stochastic Gradient Discent

ML approaches usually use empirical risk function which is a sum of losses over train dataset:

$$f(\mathbf{w}) = \sum_{i=1}^{\ell} f_i(\mathbf{w})$$

computationally heavy for $\ell \gg 1$

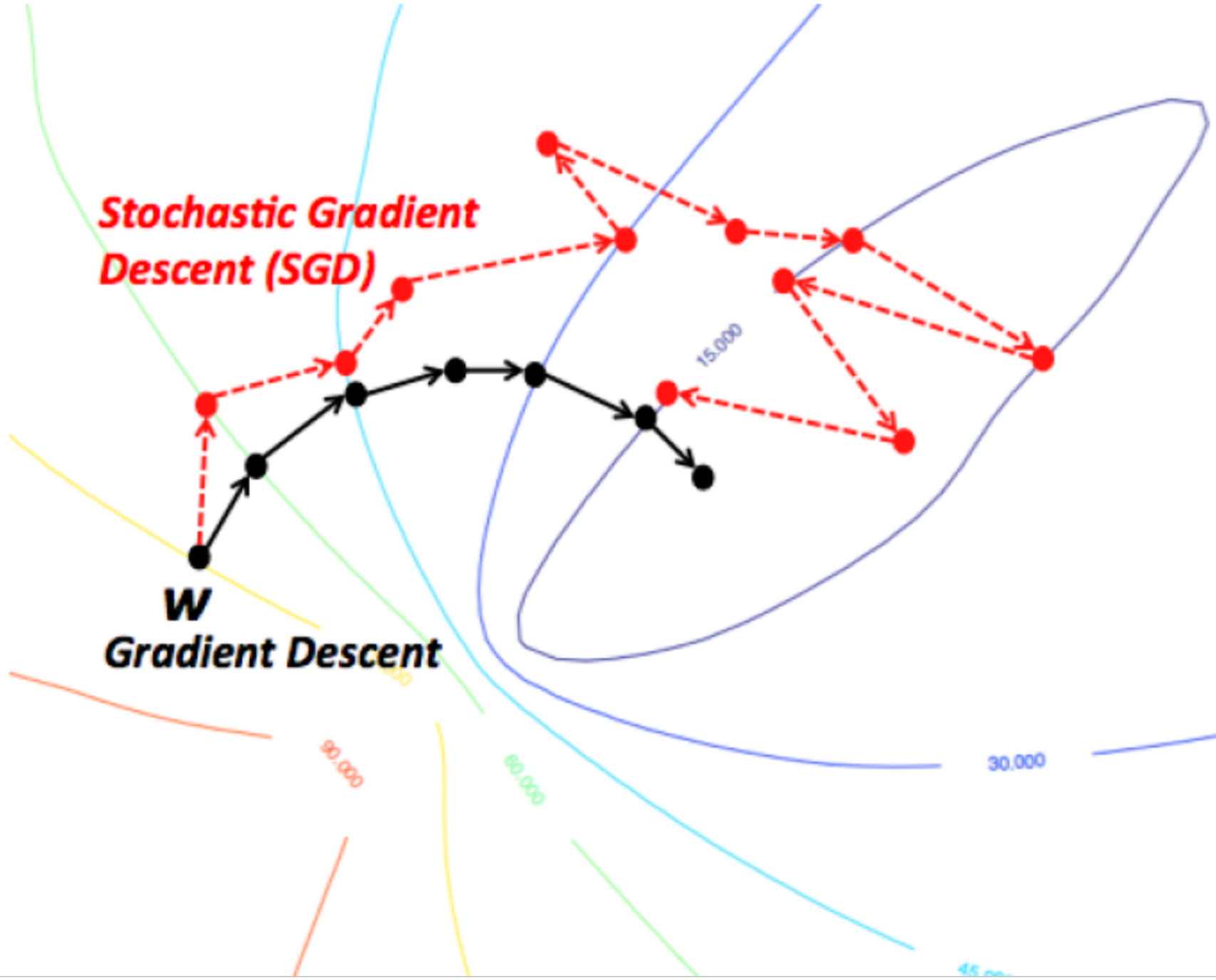
trick: on each iteration randomly pick one $1 \leq i_k \leq \ell$ and step in that direction:

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$$

converges as $f(\mathbf{w}^{(k)}) - f(\mathbf{w}^*) = \mathcal{O}(1/\sqrt{k})$

batches, momentum and other tricks may improve converging rates

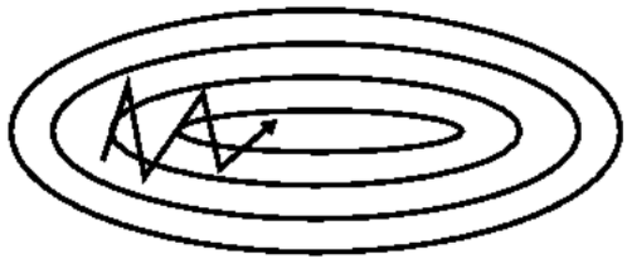
Gradient Descent vs Stochastic Gradient Discent



SDG With Momentum

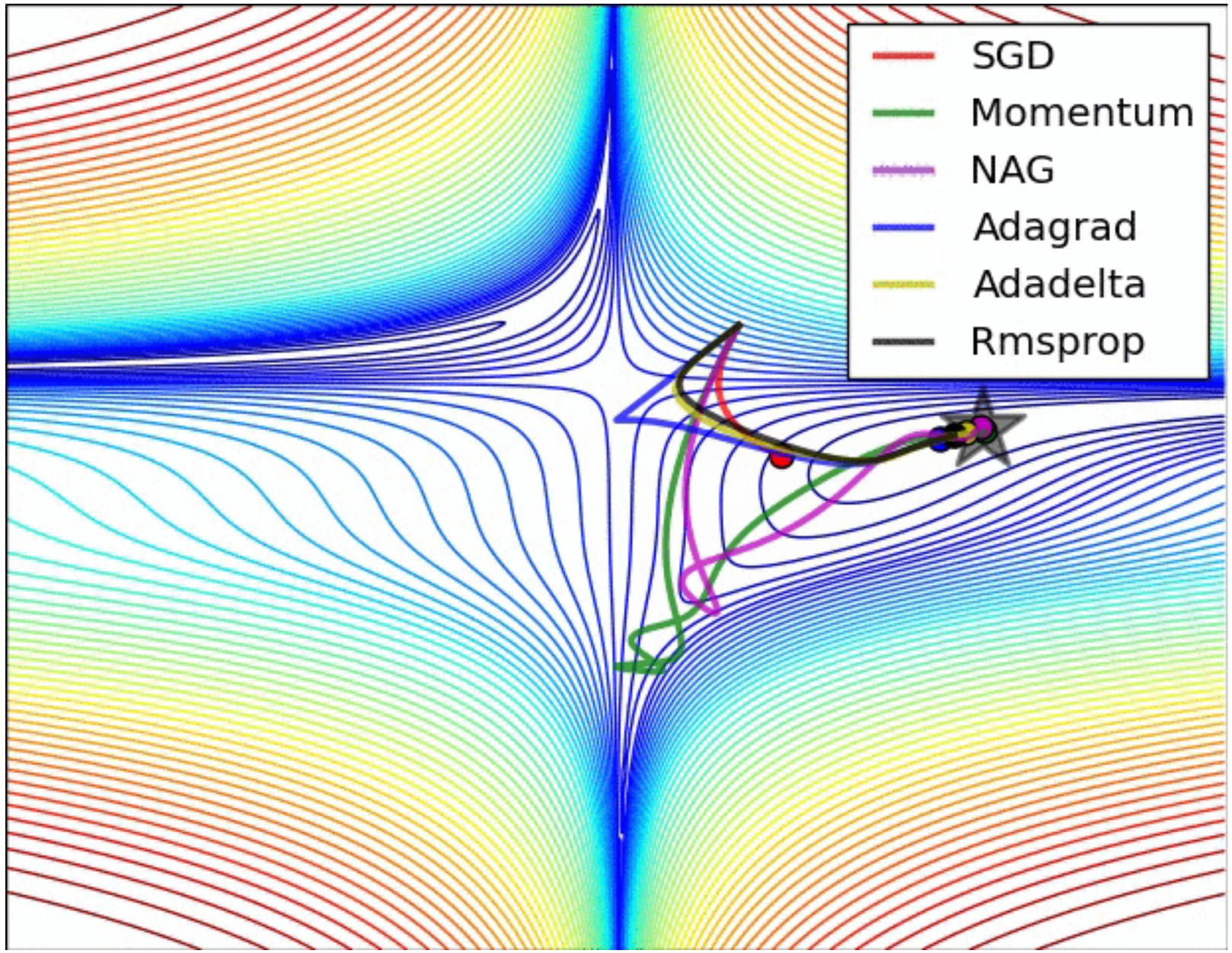
$$\Delta \boldsymbol{w}^{(k)} = -\alpha \nabla_{\boldsymbol{w}} f_i(\boldsymbol{w}^{(k-1)})$$
$$\boldsymbol{w}^{(k)} = \boldsymbol{w}^{(k-1)} + \Delta \boldsymbol{w}^{(k)}$$

$$\Delta \boldsymbol{w}^{(k)} = \gamma\, \Delta \boldsymbol{w}^{(k-1)} - \alpha \nabla_{\boldsymbol{w}} f_i(\boldsymbol{w}^{(k-1)})$$
$$\boldsymbol{w}^{(k)} = \boldsymbol{w}^{(k-1)} + \Delta \boldsymbol{w}^{(k)}$$



and many other approaches with adaptive learning rate

Various Optimizers Illustration

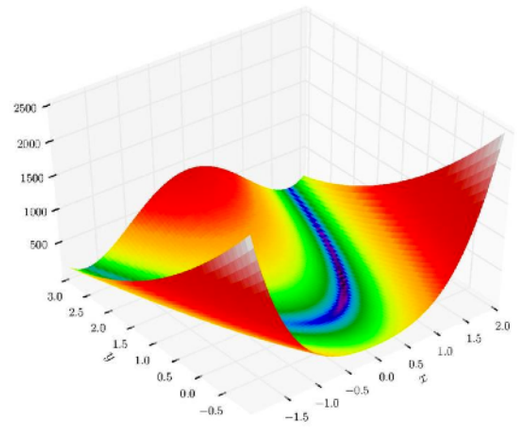


https://imgur.com/a/Hqolp

adaptive learning rate and momentum helps

Regularization

generic models usually use more parameters than actual dimensions of the function



under-defined optimization problem → collinear features → multidimensional optimum valleys → computational problems

fit → fit + penalty for large values — regularization:

$Q(\boldsymbol{w}) \rightarrow Q_\alpha(\boldsymbol{w}) = Q(\boldsymbol{w}) + \alpha R(\boldsymbol{w})$

$R(\boldsymbol{w})$ — regulizer,

α — regularization constant

Regularization Types

L2 regularization:

$$R(\boldsymbol{w}) = \boldsymbol{\Sigma} w_i^2$$

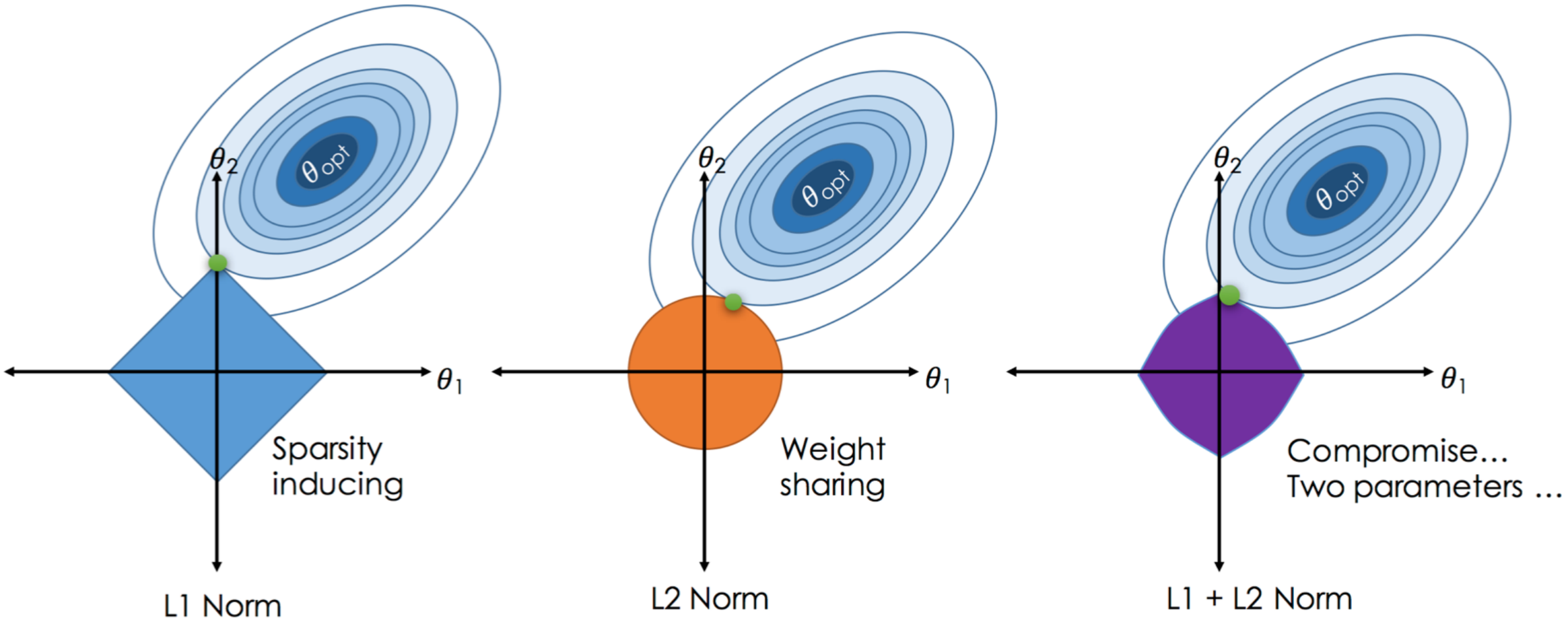
L1 (LASSO) regularization:

$$R(\boldsymbol{w}) = \boldsymbol{\Sigma} |w_i|$$

L1/L2 regularization (Elastic Net)

$$\alpha \boldsymbol{\Sigma} w_i^2 + \beta \boldsymbol{\Sigma} |w_i|$$

Geometric Interpretation



Picture credit: http://www.ds100.org/sp17/assets/notebooks/linear_regression/Regularization.html

Another Interpretation

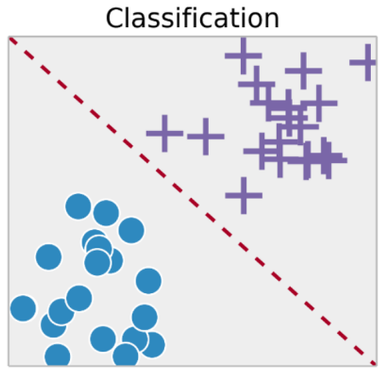
Figure: Large parameter space

Figure: Regularized models

Outlook

- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Logistic Regression

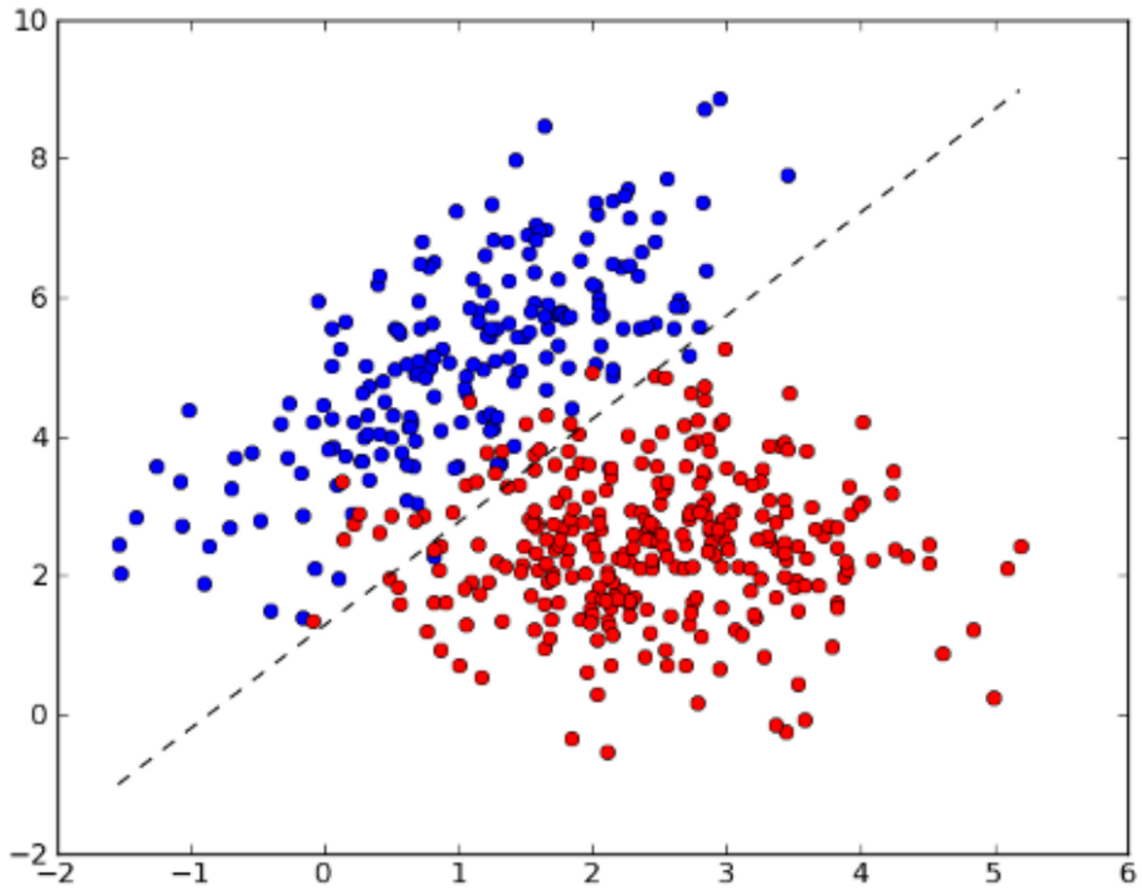


We want to separate objects of two classes

Domains for classes essentially intersect

Probabilistic separation is appropriate in this case:

logistic regression



Logistic Regression Model

Training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$

We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$ (model probability!)

A probability that an instance $(\mathbf{x}_i, y_i)$ is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

Entire X^ℓ likelihood:

$$L(X^\ell) = \prod_{i=1}^{\ell} h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

is often written via log-likelihood (of which the negative is log-loss)

$$\log L(X^\ell) = \sum_{i=1}^{\ell} [y_i = +1] \log h(\mathbf{x}_i) + [y_i = -1] \log(1 - h(\mathbf{x}_i))$$

NB: classic classification decision boundary is set by $h(x) = 0.5$

Logistic Regression Model

The choice of h: sigmoid function

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\intercal \mathbf{x})$$

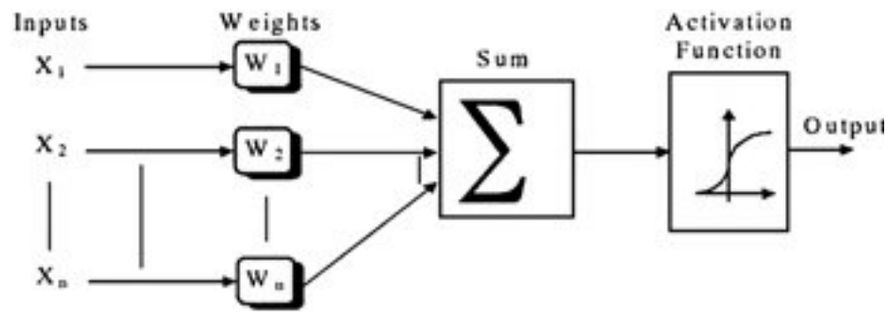
where $\sigma(x) \in [0, 1]$

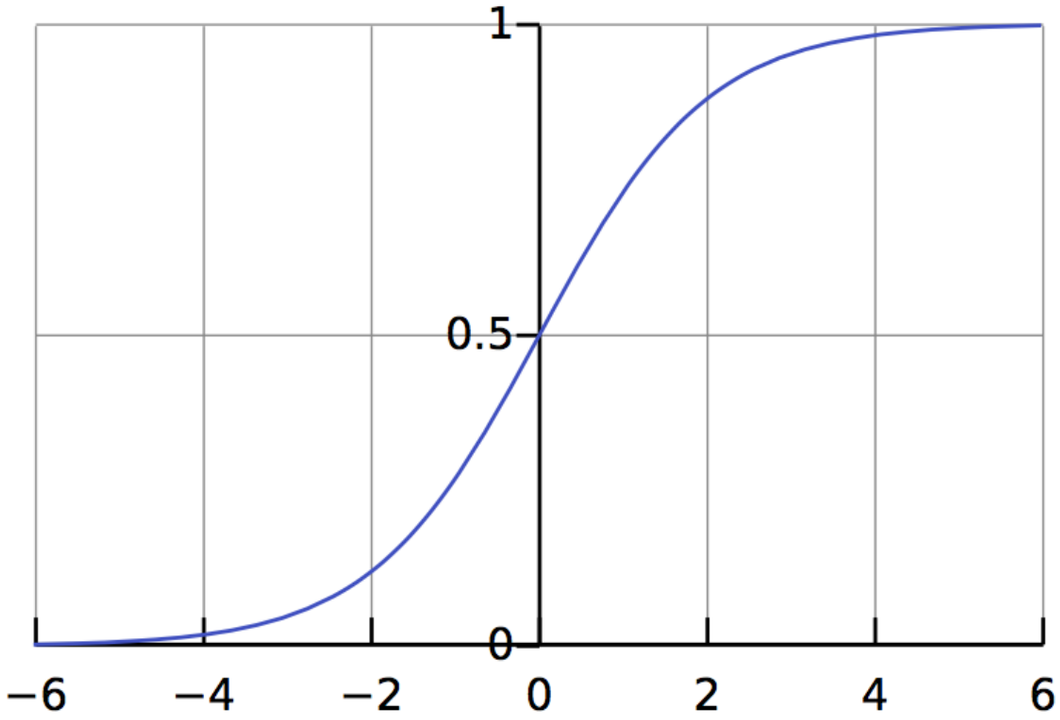
Typical choice: the logistic function

$$\sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\intercal \mathbf{x})}$$

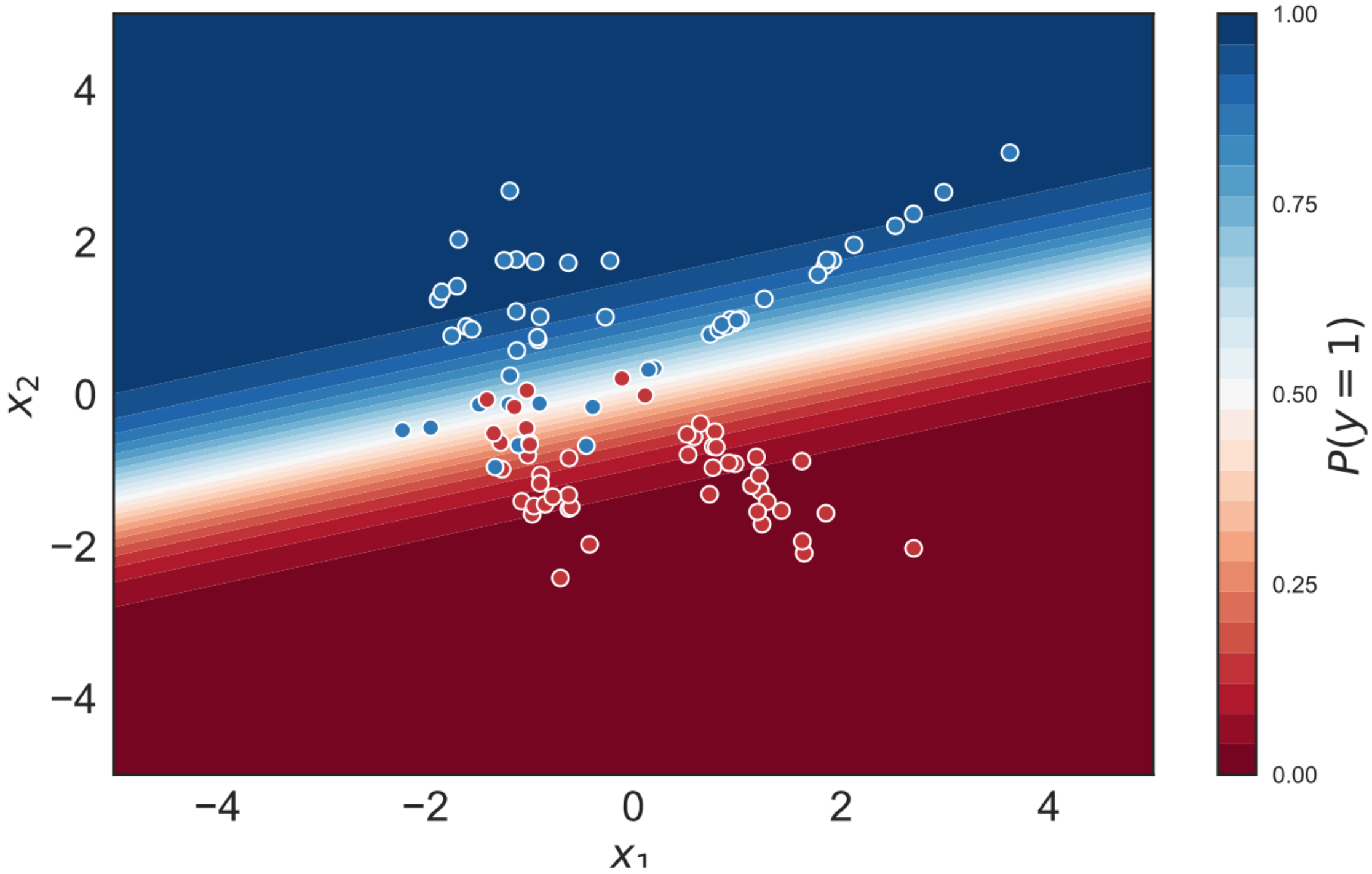
Plugging the logistic function into the loss yields an approximation of accuracy

$$\sum_{i=1}^{\ell} (1 + \exp(\mathbf{w}^\intercal \mathbf{x})) \to \min_{\mathbf{w} \in \mathbb{R}^d}$$



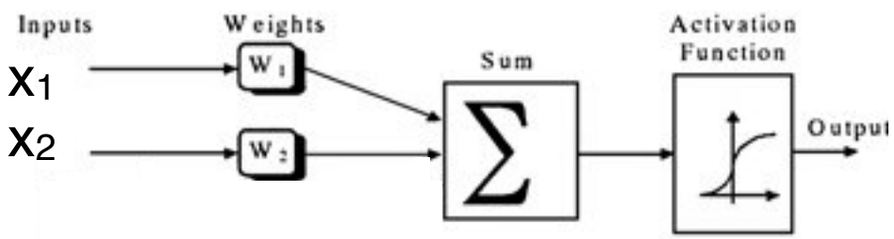


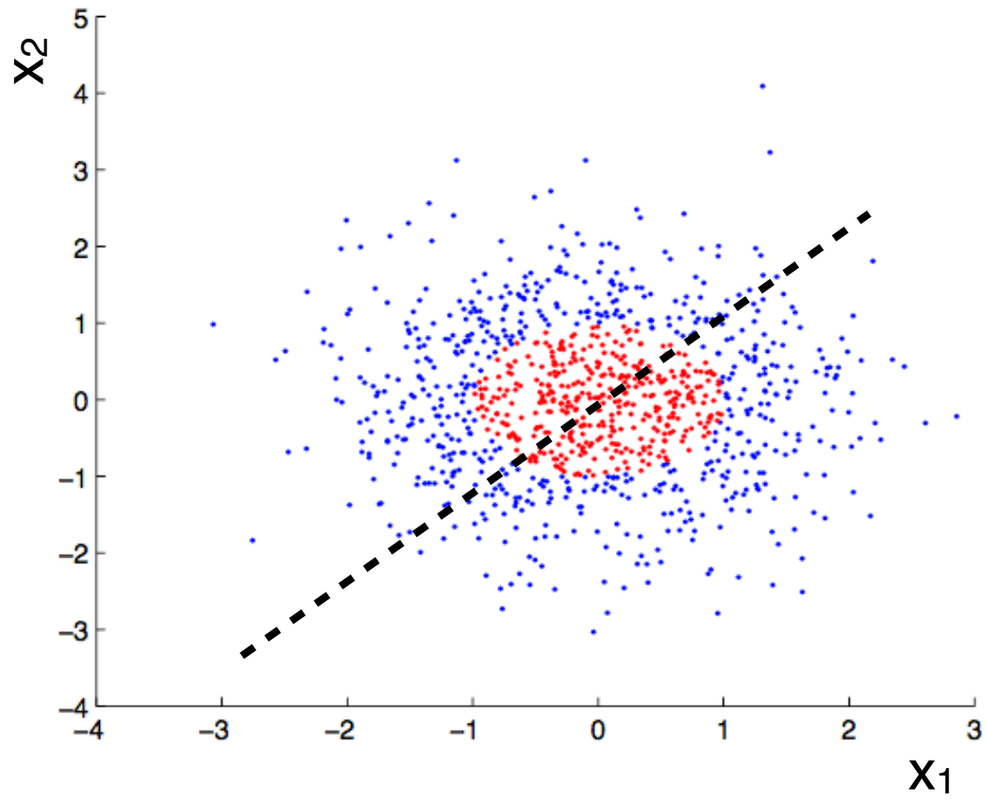
Logistic Regression Model



classic classification decision boundary is set by h*(x)* = 0.5

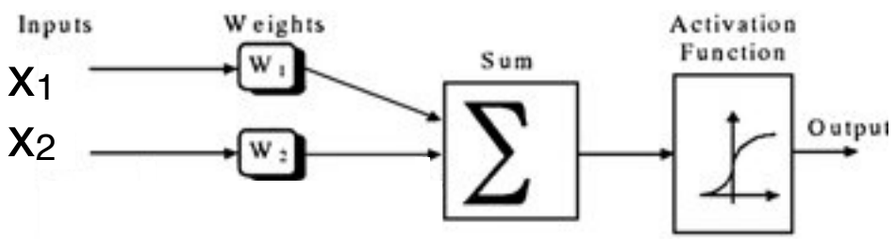
Is Linear Regression Too Restrictive?

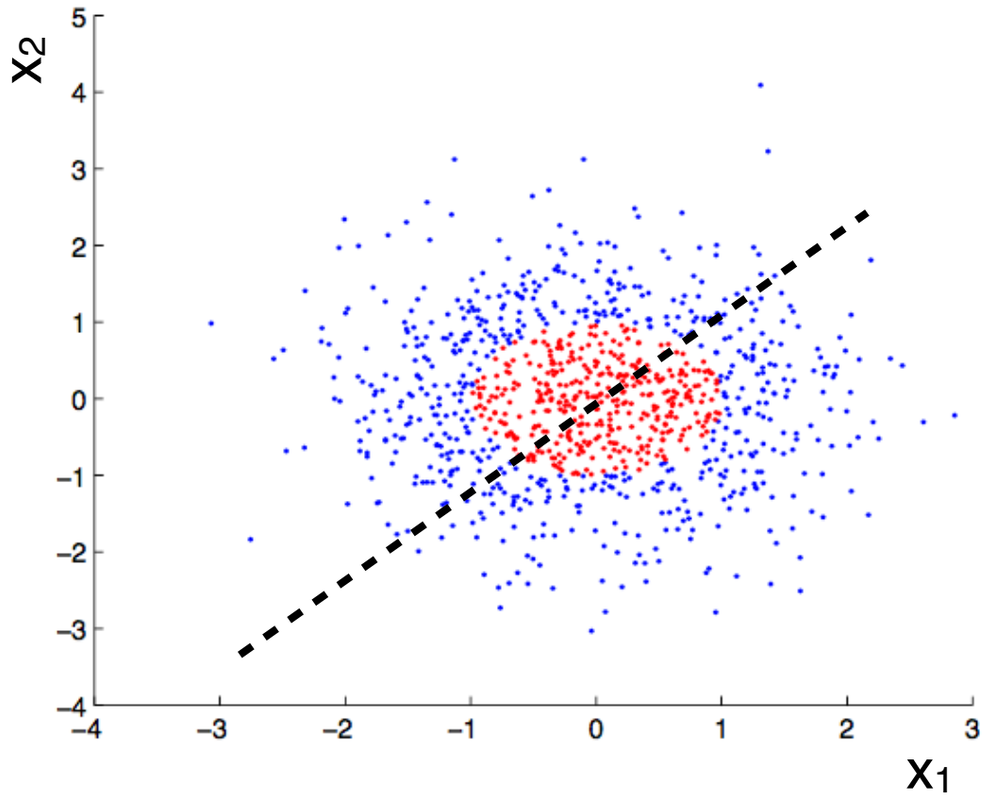




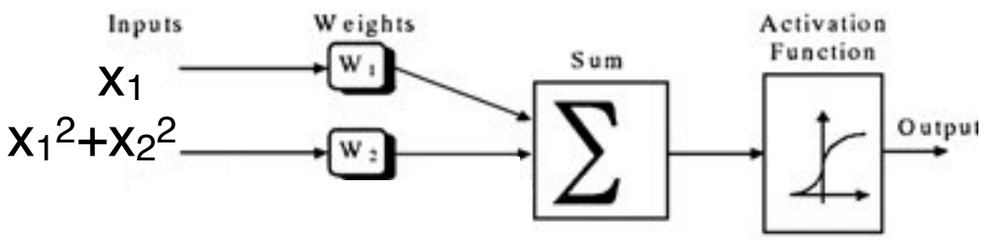
useless?

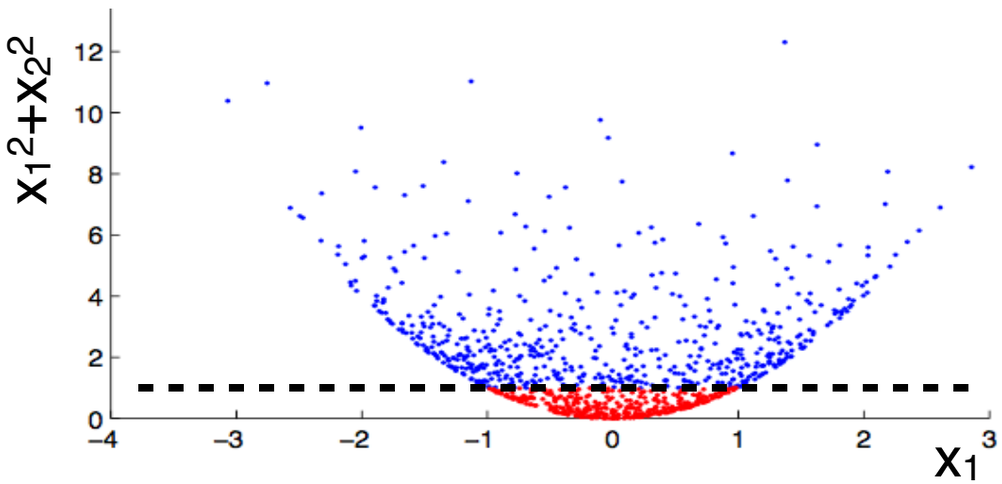
Is Linear Regression Too Restrictive?





useless?





easy!

It's all about selecting good features

Outlook

- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Figures Of Merit

For the regression e.g. χ^2 on the test sample is a reasonable FOM

What is a good FOM for classification problem?

Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?

Let the thresholded decision rule be $a(x) = [h(x) > t]$ (t: hyperparameter)

Obvious choice: accuracy

$$\text{accuracy}(a, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}[a(\mathbf{x}_i) = y_i]$$

However e.g. for sample with 50 signal and 950 background events, classifier “all is background” has accuracy = 0.95

Confusion Matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

Rates are often more informative:

background fake rate → $\text{False Positive Rate aka FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$,

signal efficiency → $\text{True Positive Rate aka TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$,

While accuracy can be expressed, too

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

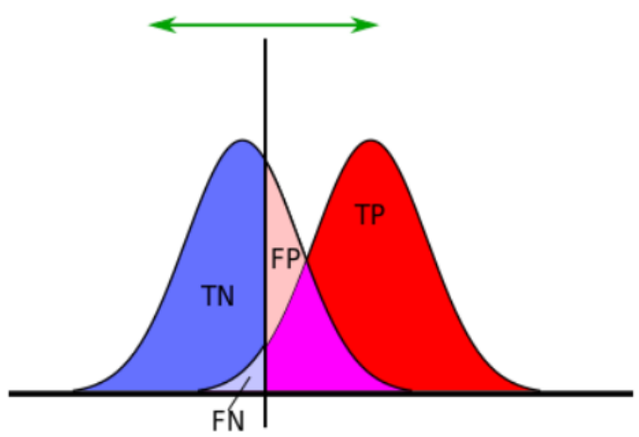
Receiver Operating Curve

Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$

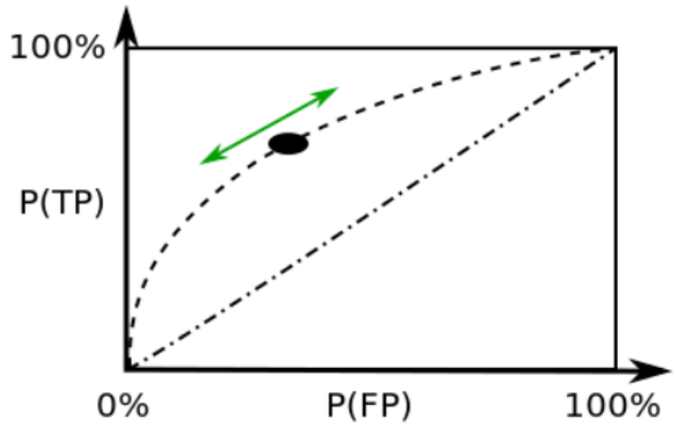
Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t

The plot TPR(t) vs. FPR(t) is called the receiver operating characteristic (ROC) curve

Area under curve (ROC-AUC) reflects classification quality



TP	FP
FN	TN

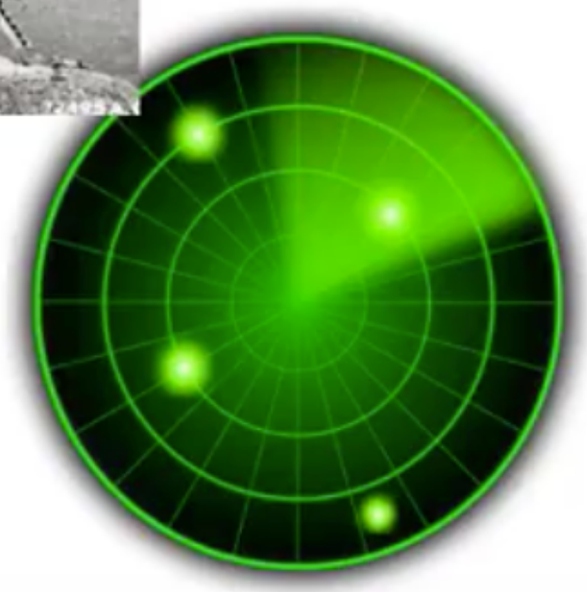


Source: Wikipedia

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

ROC History

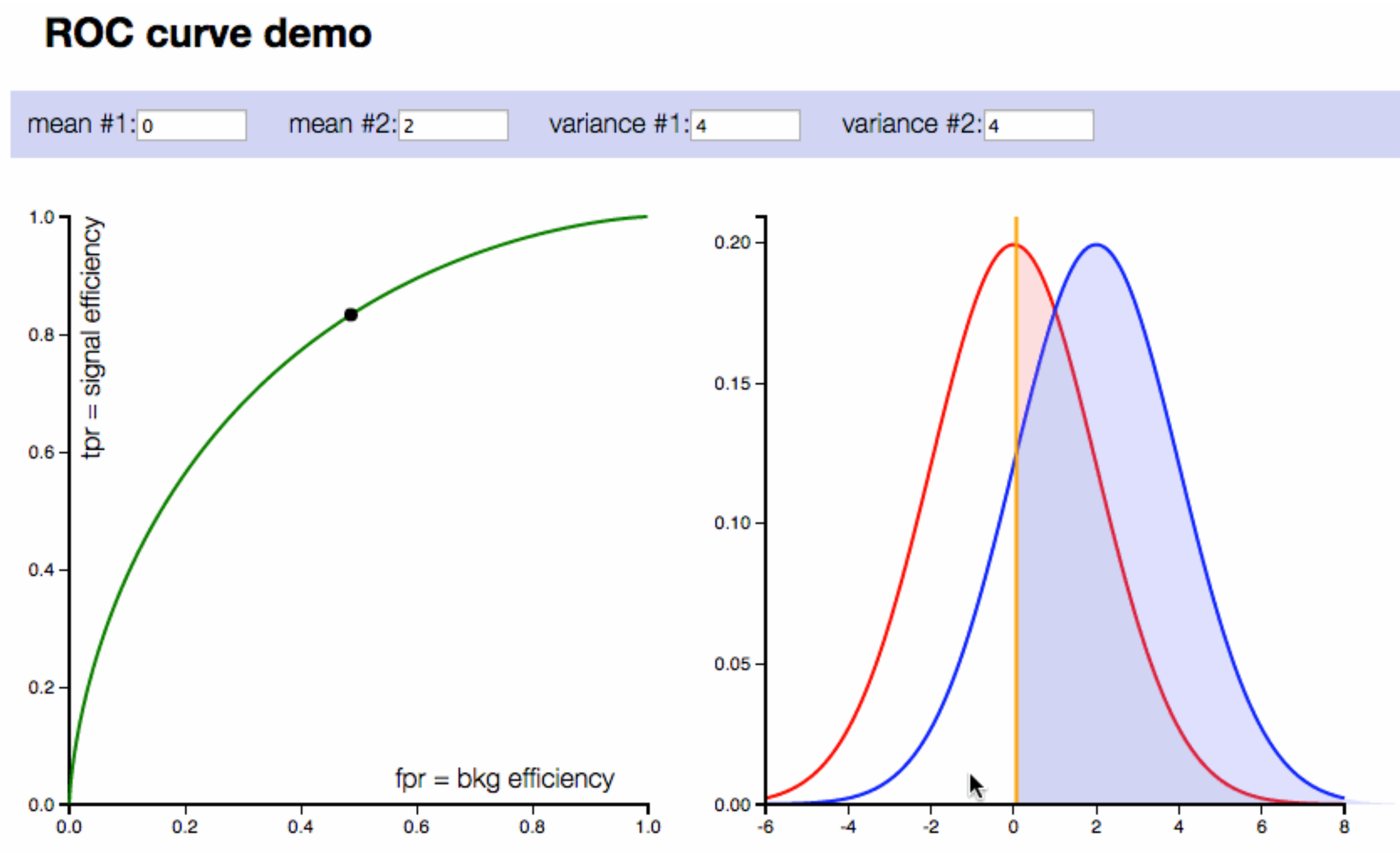
WWII: studying algorithms to correctly detect Japanese aircrafts using radars



ROC and ROC-AUC were found a good figures to quantify receiver ability to correctly detect aircraft from the radar signal

ROC Curve Demonstration

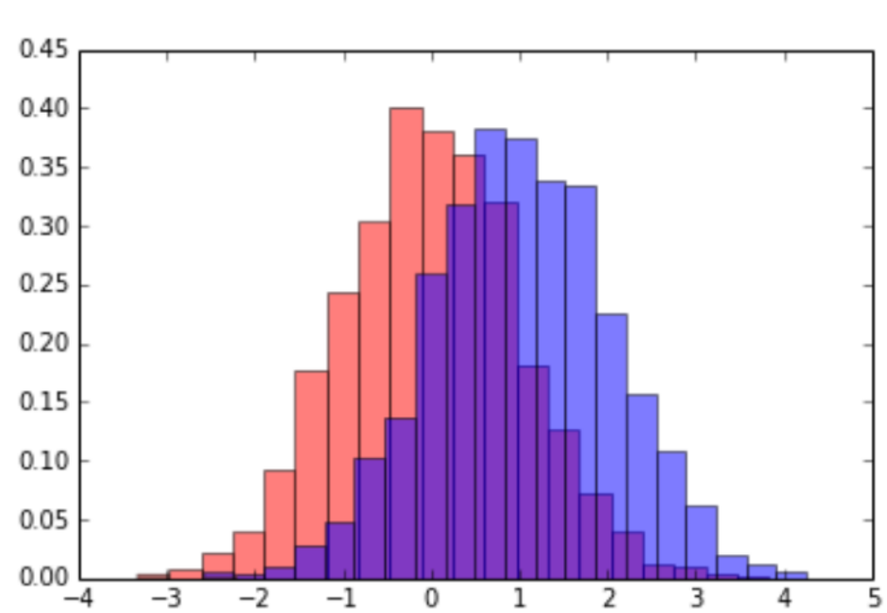
http://arogozhnikov.github.io/2015/10/05/roc-curve.html

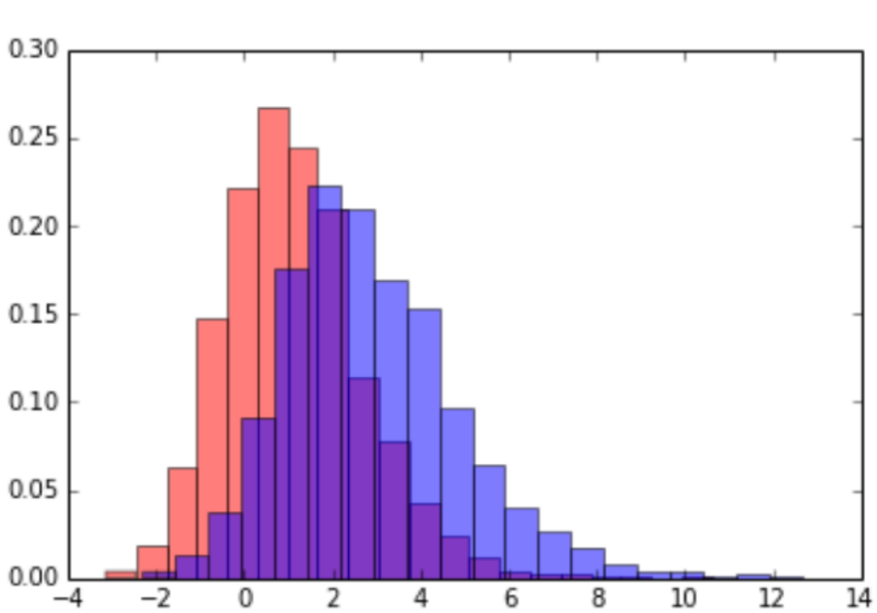


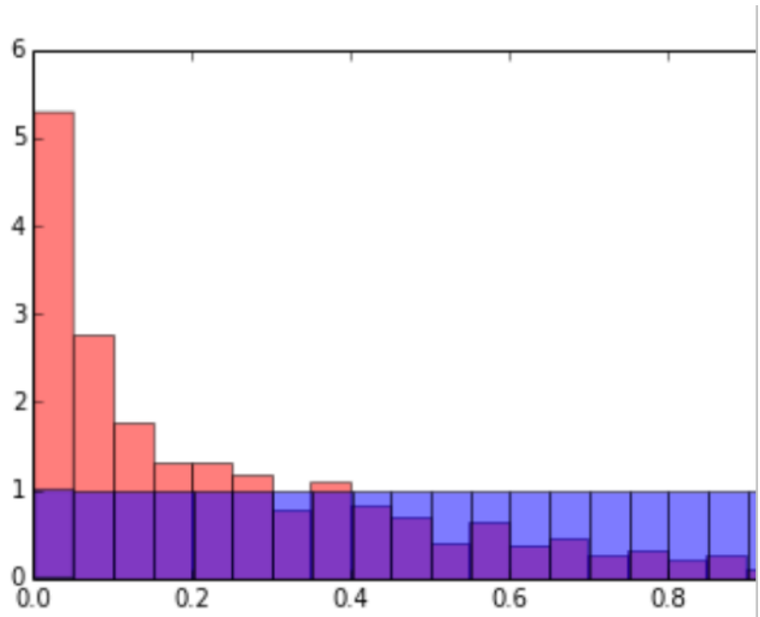
ROC Curve

output of binary classifier is a real variable

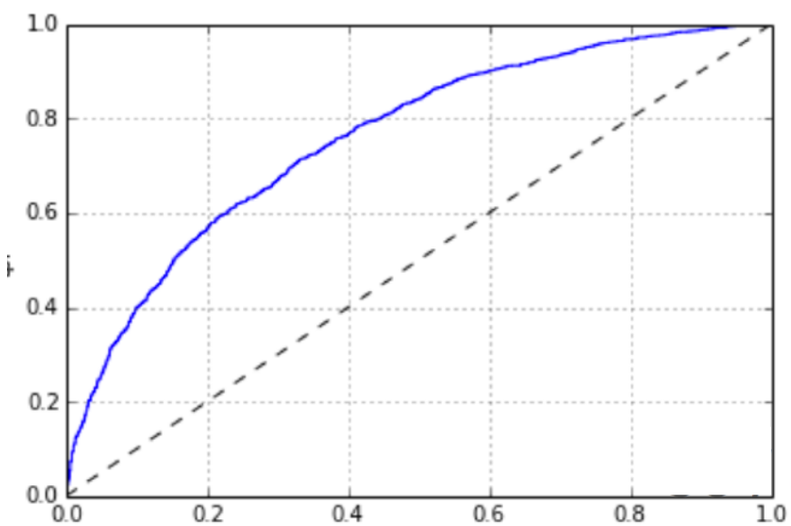
say, signal and background







which classifier provides better discrimination?



ROC Curve

Defined only for binary classification

Contains important information:

all possible combinations of signal and background efficiencies you may achieve by setting threshold

Particular values of thresholds (and initial pdfs) don't matter, ROC curve doesn't contain this information

ROC curve = information about order of events:

bbsbsb ... ssbss

Comparison of algorithms should be based on the information from ROC curve

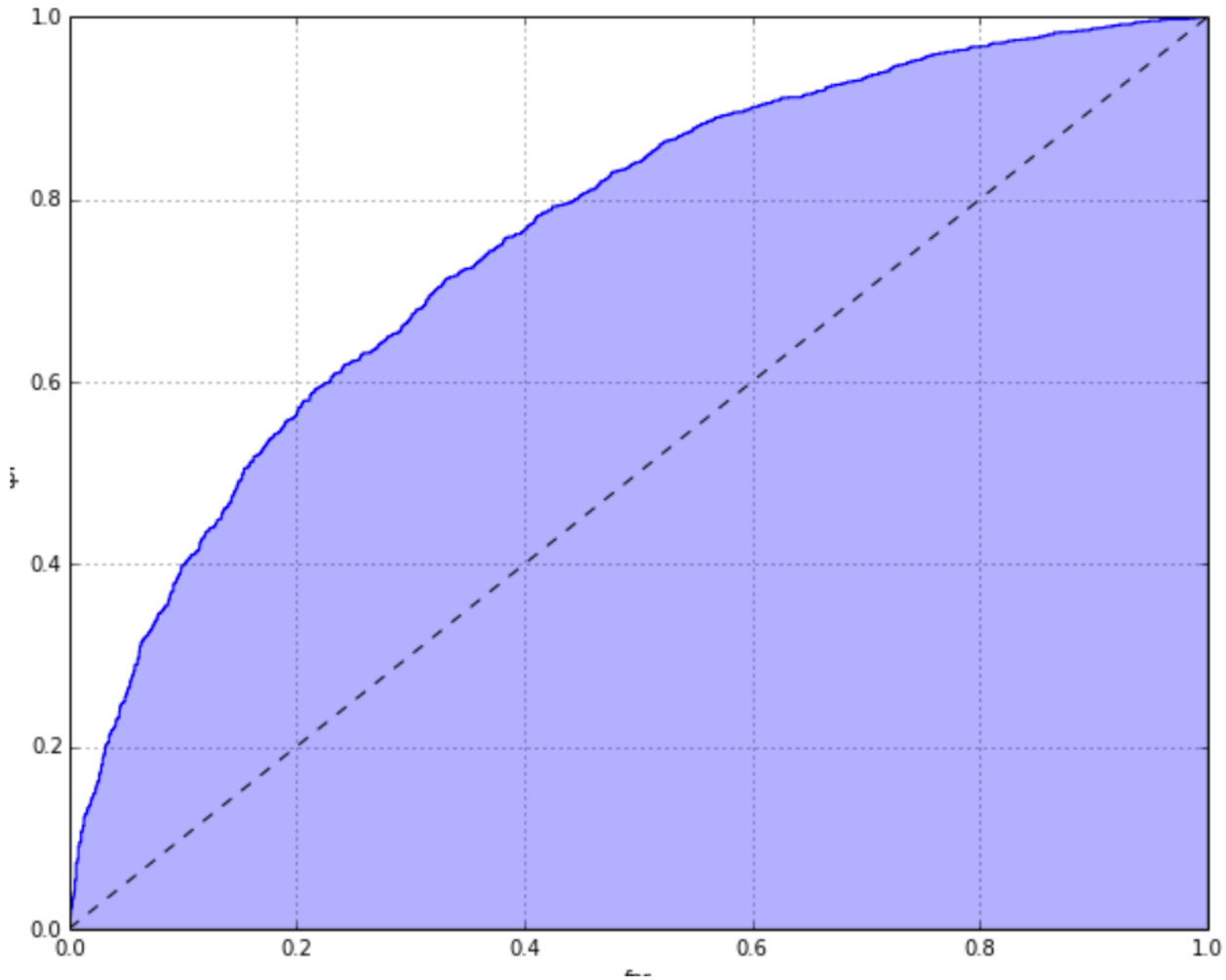
ROC-AUC

ROC-AUC: Area Under ROC Curve

ROC-AUC = $P(r_b < r_s)$

r_b and r_s are predictions for random background and signal events

The bigger ROC-AUC, the stronger separation is



Selecting Good Classifier

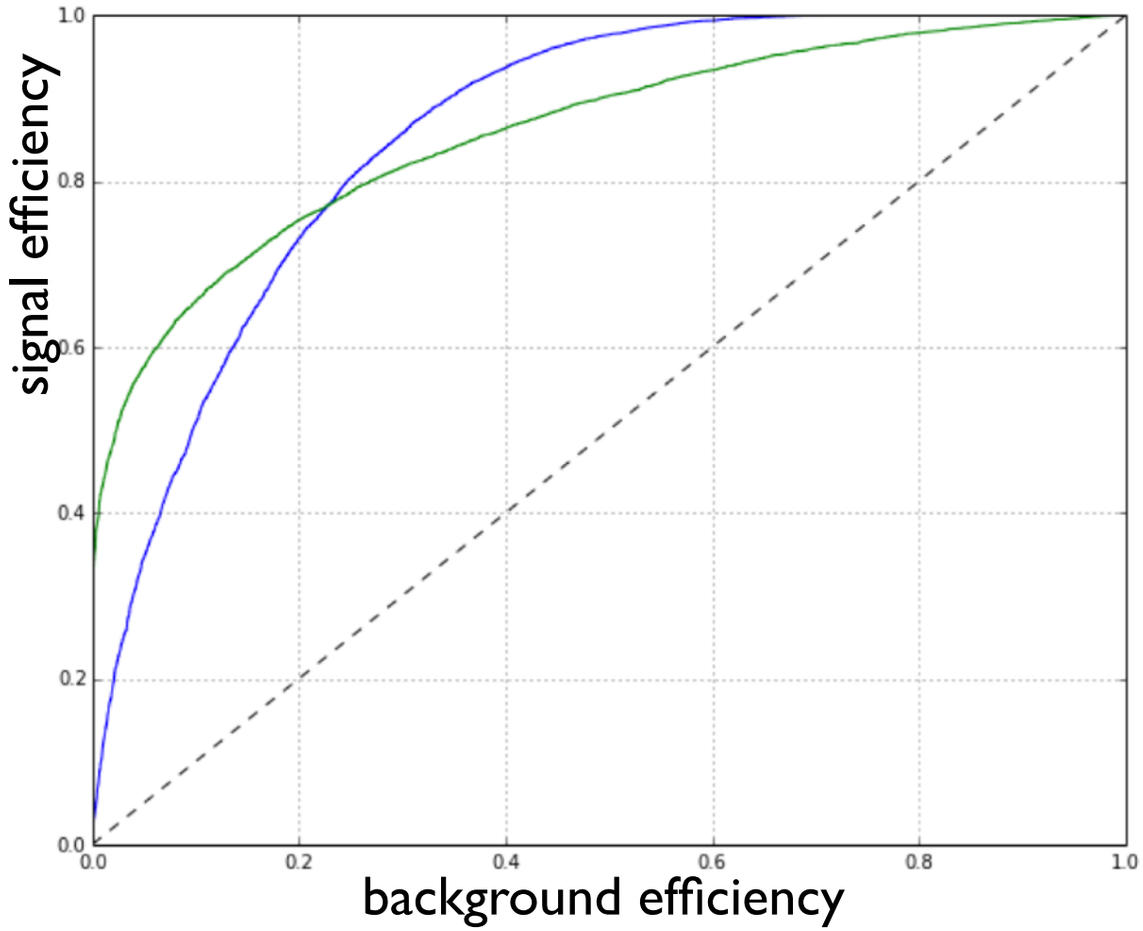
ROC AUC is a measure of separation ability of the classifier

Is larger always better?

These these are two classifiers for trigger selection at LHC

which one should be used?

NB we are limited by the trigger rate and need few background events to pass



Outlook

- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
  - assess model quality: cross-validation
  - optimizing model parameters: hyperparameters tuning
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Model Validation and Selection

our model has free parameters (hyperparameters)

e.g. polynomial degree n for P_n regression, subset of features in multivariate regression etc.

model selection: how to choose optimal hyperparameters for the given problem

after training we obtain some model performance

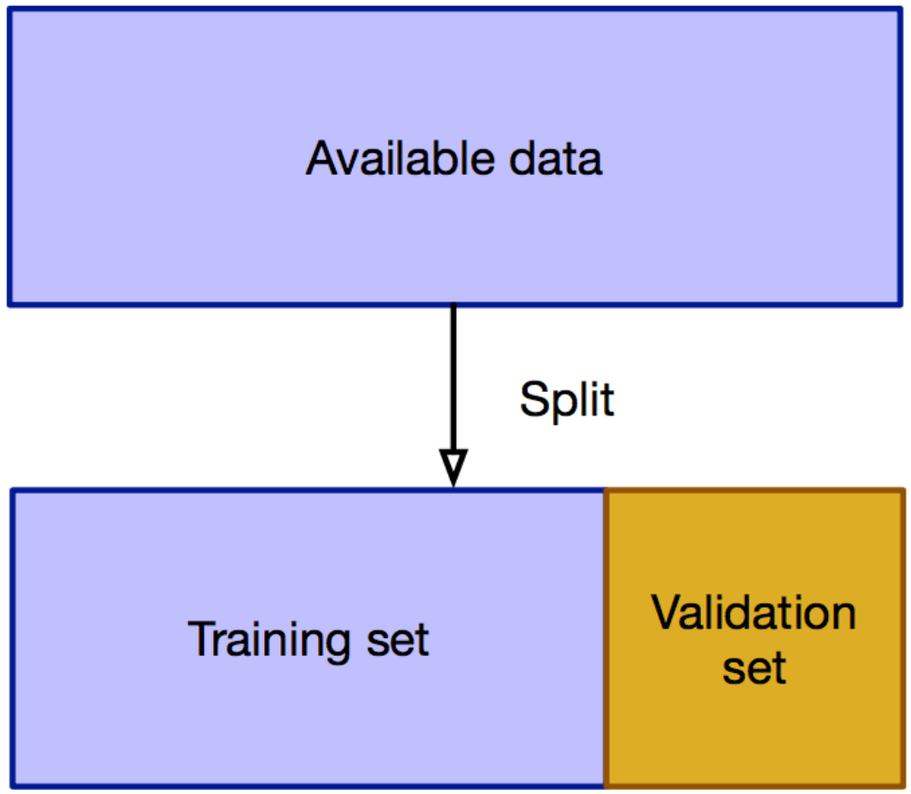
model validation: how to evaluate true performance

(NB overfitting)

solution: hold some data apart when training the model, use them for *unbiased* validation and optimization of the model

Model Validation

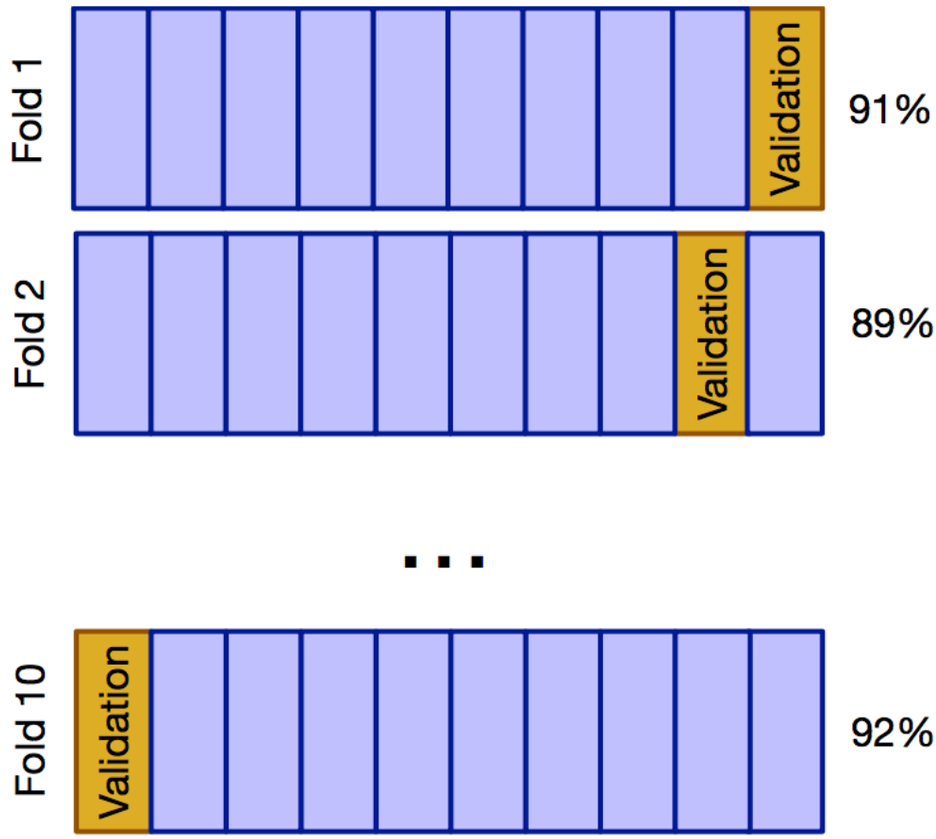
- split data $X = X_{train} + X_{val}$
- **train model** $g(x, \theta)$ on X_{train}
  - find optimal parameters θ_0 and optimal model $g_0(x) = g(x, \theta_0)$
- ***a*ssess quality**, e.g. $Q(g_0(x), X_{val})$



- we got unbiased quality estimation, but no idea about its precision
  - result depends on the particular data split

K-fold Cross Validation

- split data $X = X_1 + X_2 + \ldots + X_K$ of equal sizes
- **train K models** $g_i(x)$, $i=1..K$
  - each model is trained on all subsets except X_i
- ***a*ssess K qualities**, on remaining subset, e.g. $Q(g_i(x), X_i)$
- we got K estimations for quality, may derive average and variance
- most effective use of available dataset

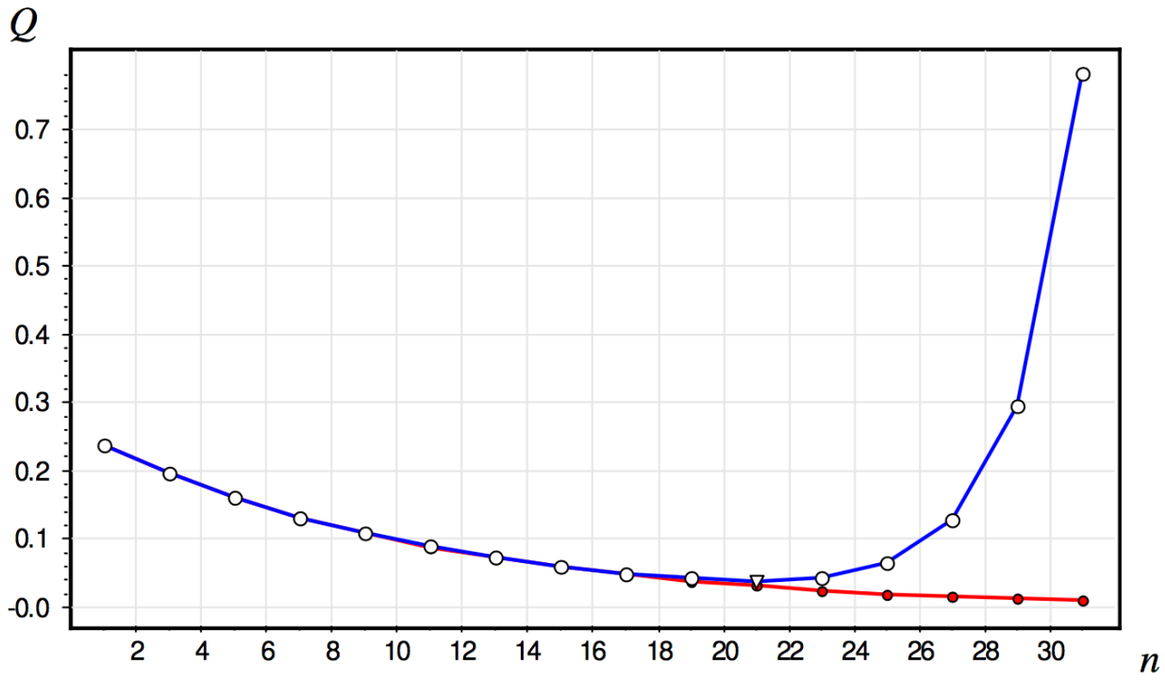


K-Fold Cross Validation Tradeoffs

- many folds:
  - small bias comparing to training on the full dataset
  - large variance due to small test size
  - large computational time due to many experiments
- few folds:
  - computationally effective
  - small variance
  - larger bias with respect to the training on the full dataset
- mean and variance of the Q in CV gives a good idea about quality and stability of the model

Model Optimization

- with cross-validation we can evaluate quality of the model
- but we can use different models
  - how to select the best one?



  - e.g. which degree *d* to select for P*n* regression?

- solution: consider it as a yet another optimization problem
  - describe variety of models by few **hyperparameters**
  - optimize hyperparameters

Hyperparameters Optimization

- what is the figure of merit?
  - e.g. mean quality on cross-validation
- on which data?
  - validation subset must be **statistically independent** from both train and validation subsets
- which approach?
  - until number of hyperparameters is small, simple grid search may be good enough
  - otherwise black-box optimization approaches/tools may be employed
    - to speedup converging and increasing precision

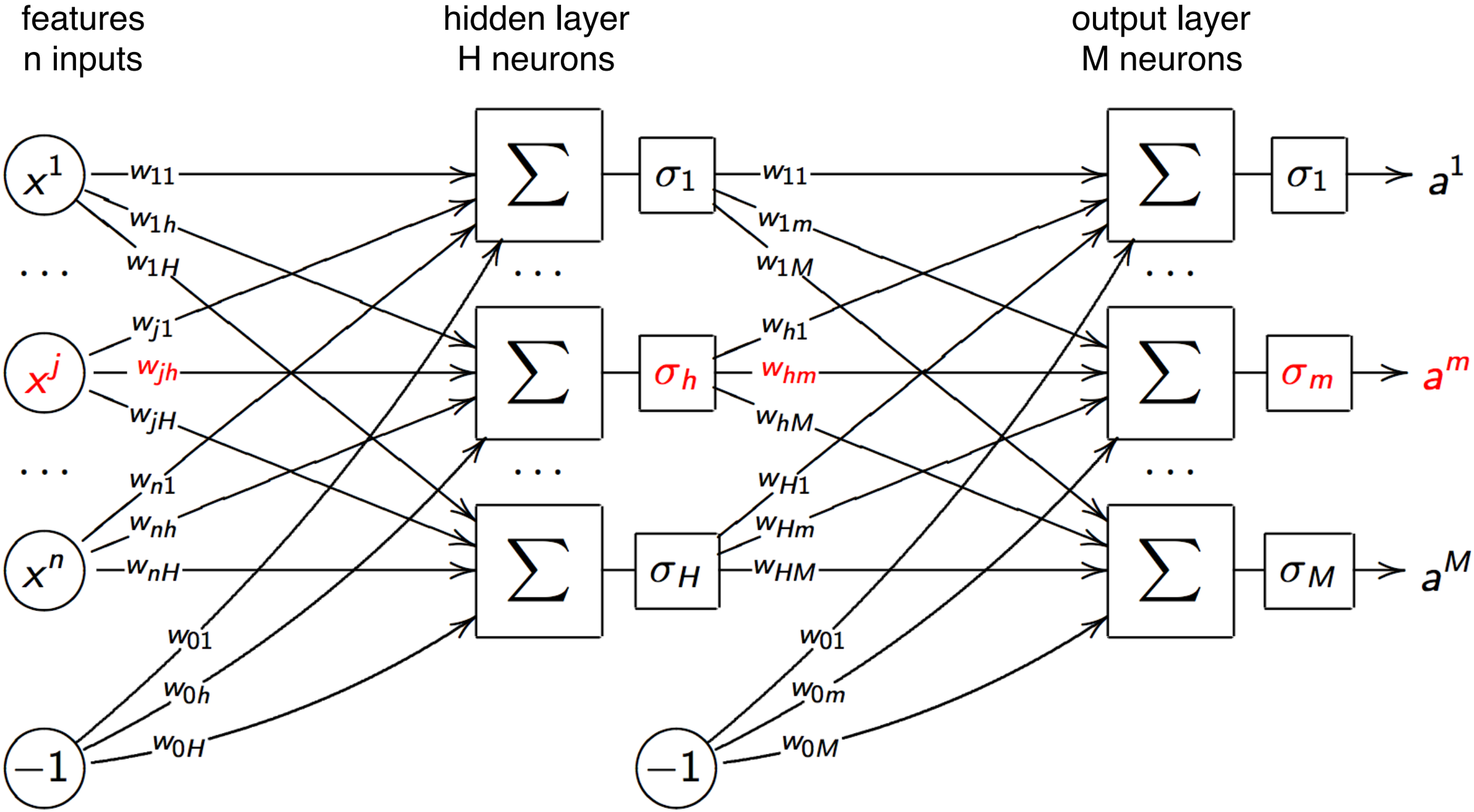
Hyperparameters Optimization

- optimisation tools
  - hyperopt
  - SMAC3
  - GPyOpt
  - scikit-optimize
  - bayesopt
  - modelgym
- difference
  - sequential/distributed/parallel execution
  - surrogate model family
  - support from the developers

Outlook

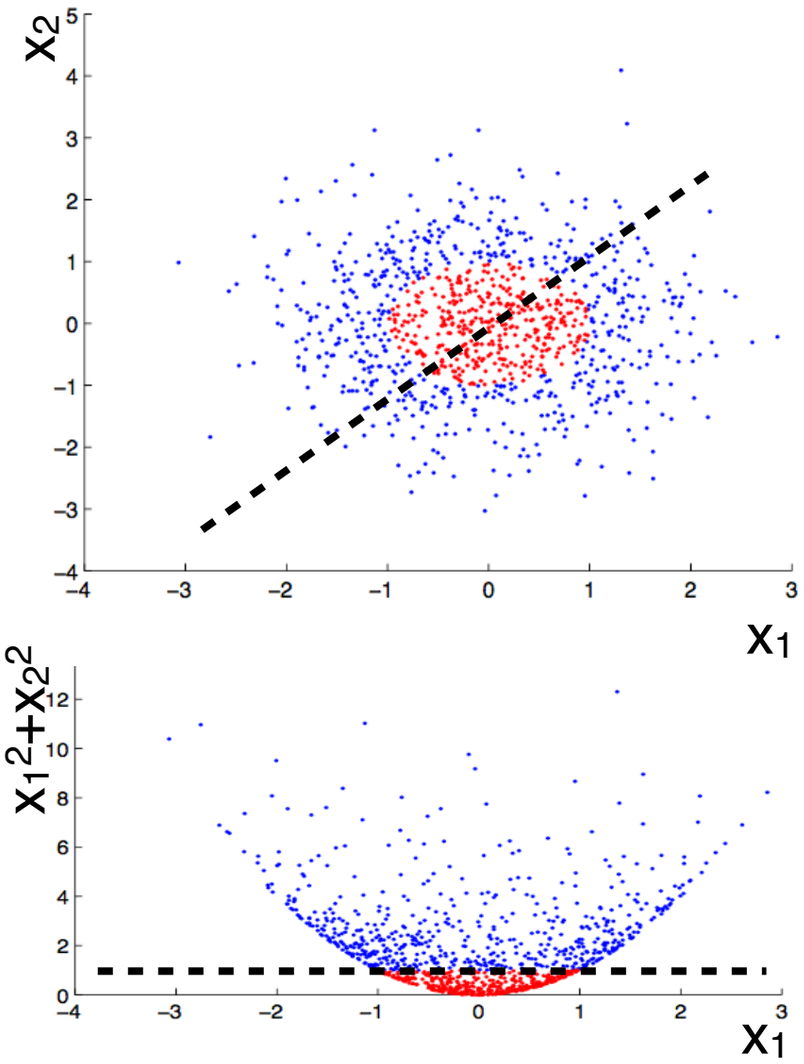
- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
  - intuition
  - mathematical basics
  - backpropagation
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

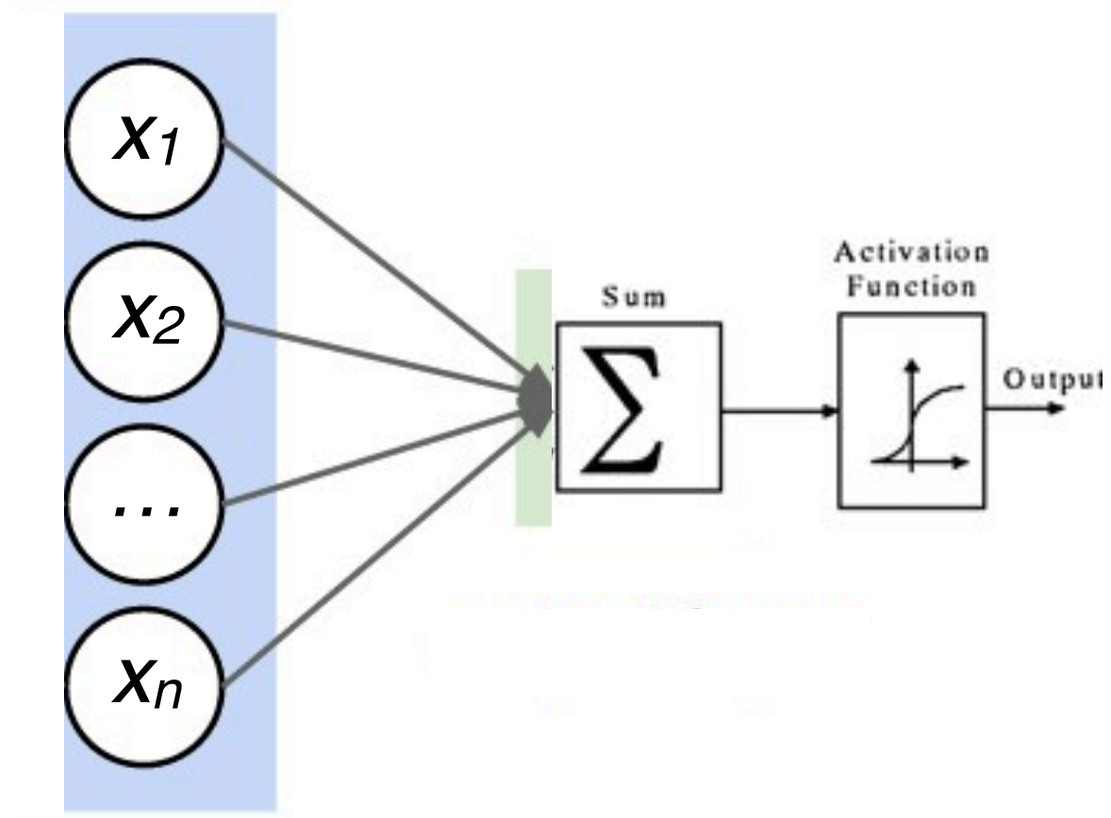
Neural Networks



(n+1)H+(H+1)M total model parameters

Neural Network Intuition: Classifier



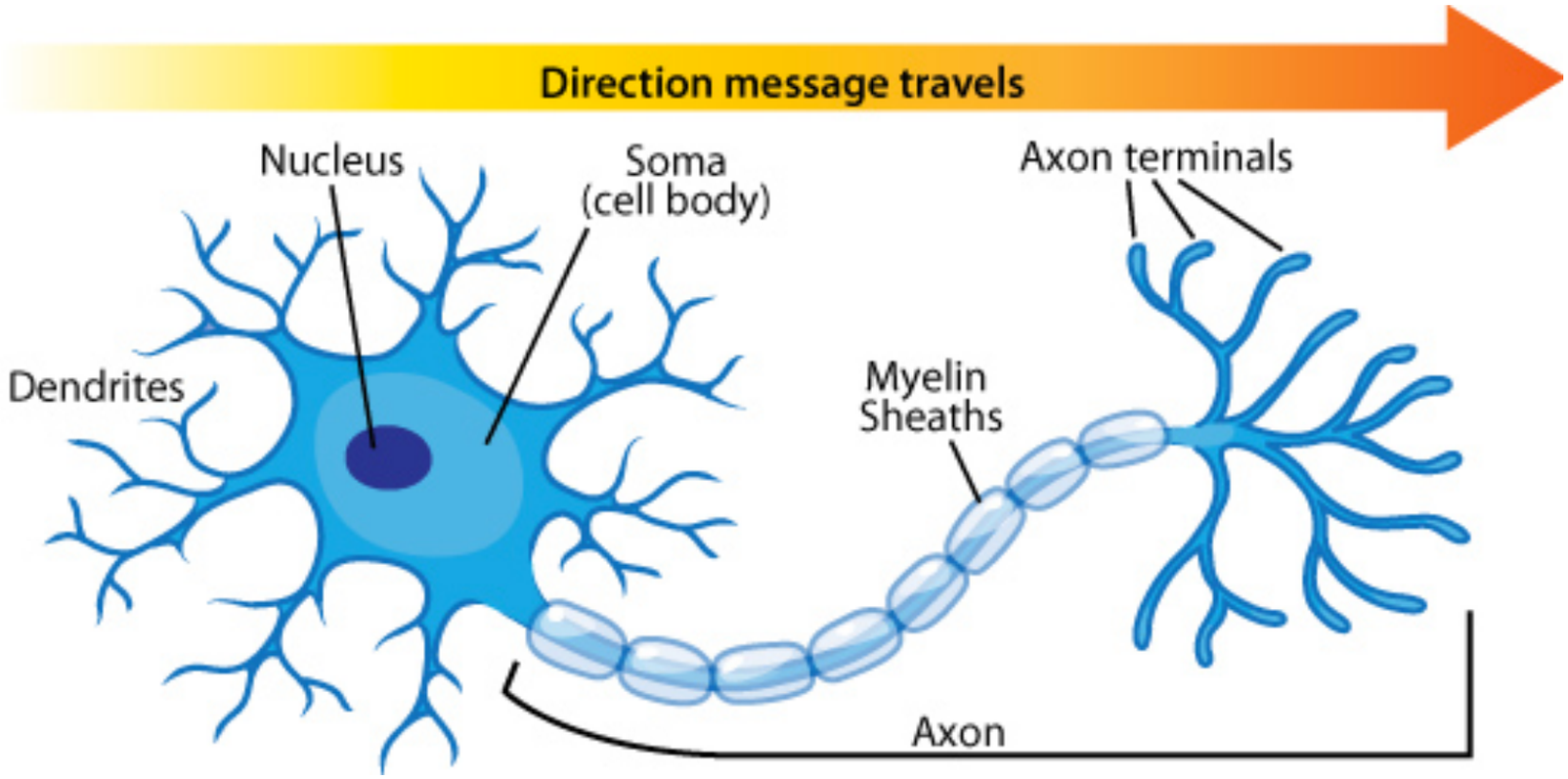


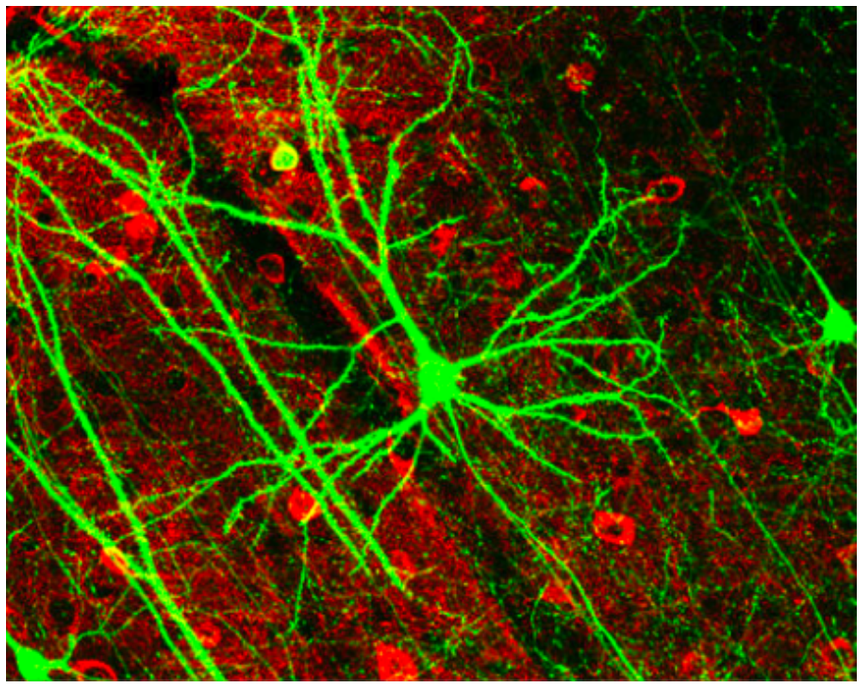
input features

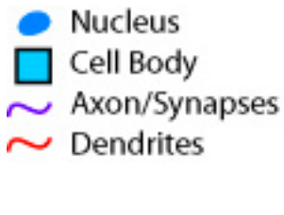
- Linear logistic regression
  - requires decision boundary be a plane in terms of input features

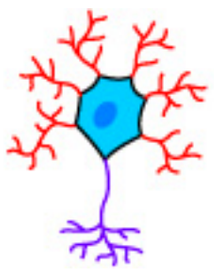
Historical Basis

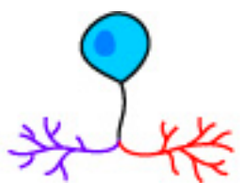
https://askabiologist.asu.edu/neuron-anatomy

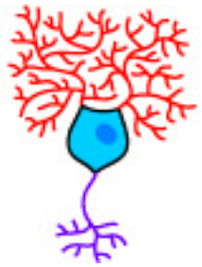












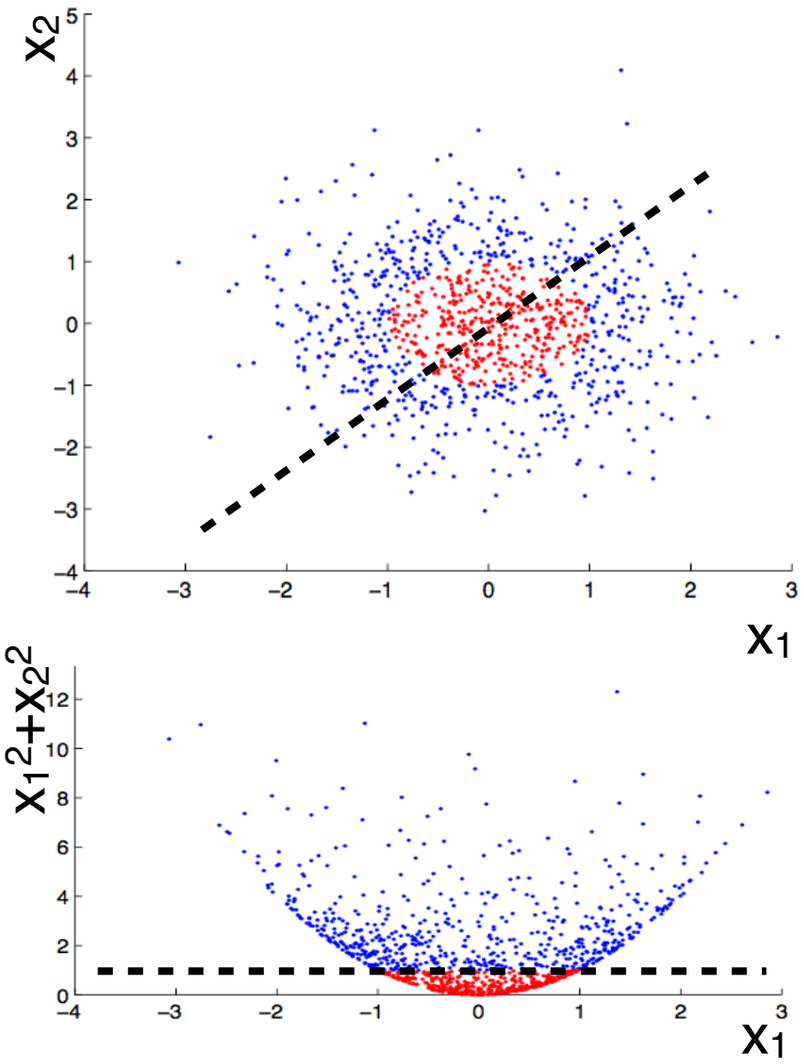
Theoretical Basis

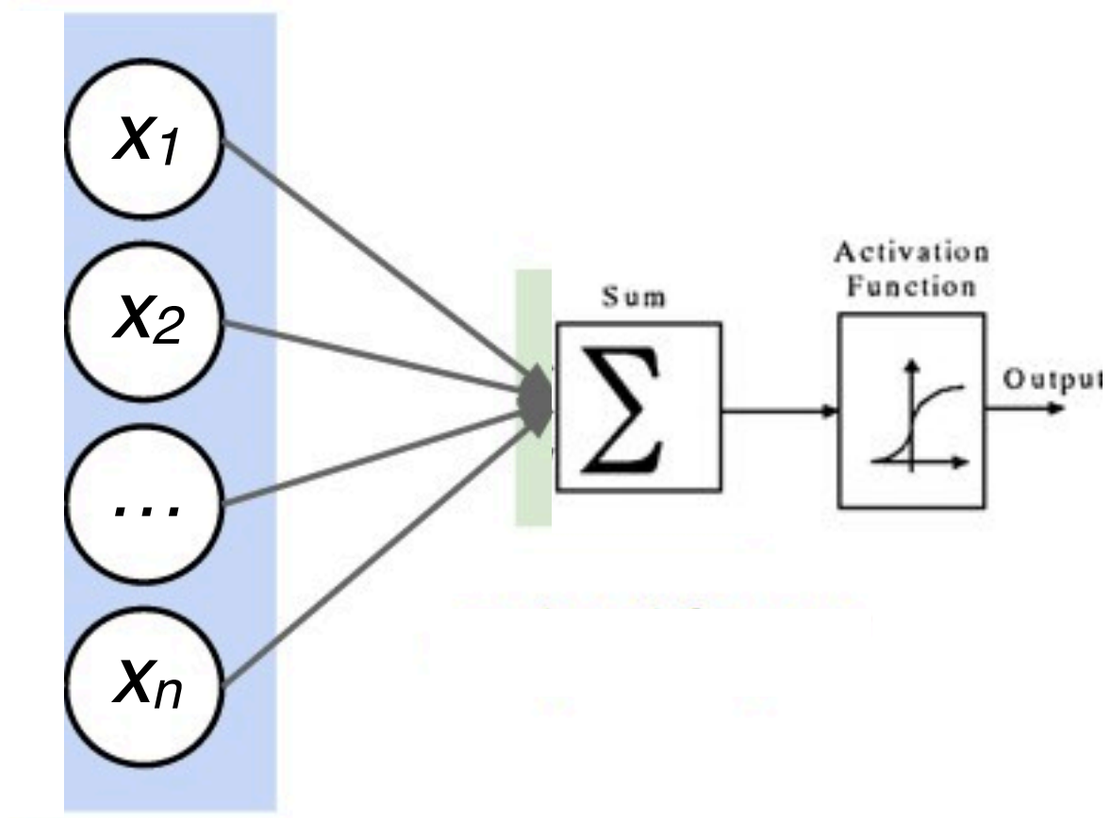
- Arnold - Kolmogorov (1957):
  - if f is a multivariate continuous function, then f can be written as a finite compositionof continuous functions of a single variable and the binary operation of addition
- Gorban (1998)

$$f(x^1, x^2, \ldots, x^n) = \sum_{k=1}^{2n+1} h_k \left(\sum_{i=1}^{n} \varphi_{ik}(x^i) \right)$$

  - it is possible to obtain arbitrarily exact approximation of any continuous function of several variables using operations of summation and multiplication by number, superposition of functions, linear functions and one arbitrary continuous nonlinear function of one variable.
- Thus Neural Network is an universal approximator for any continuous function
  - necessary complexity of the NN is not specified though

Neural Network Intuition: Classifier

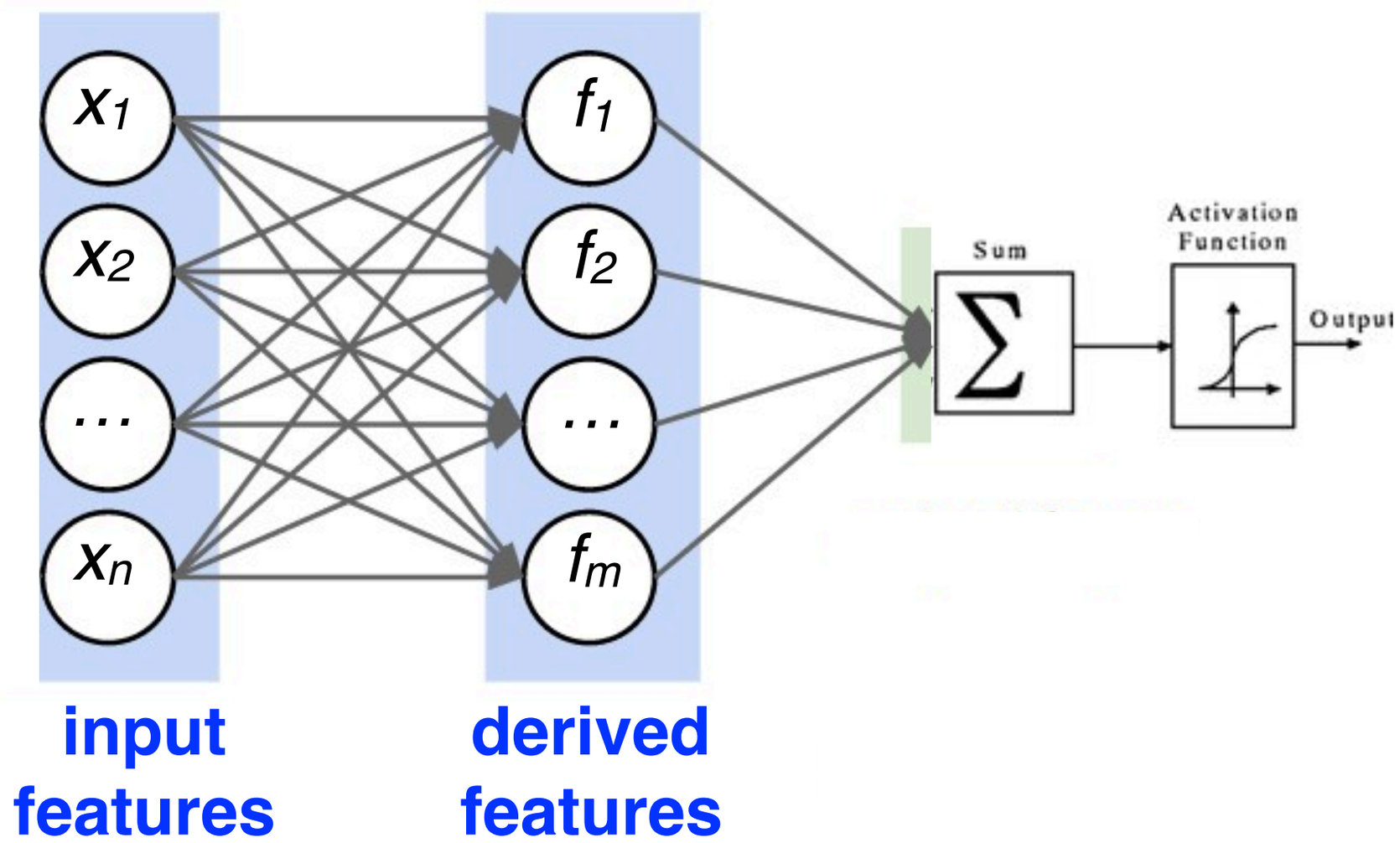




input features

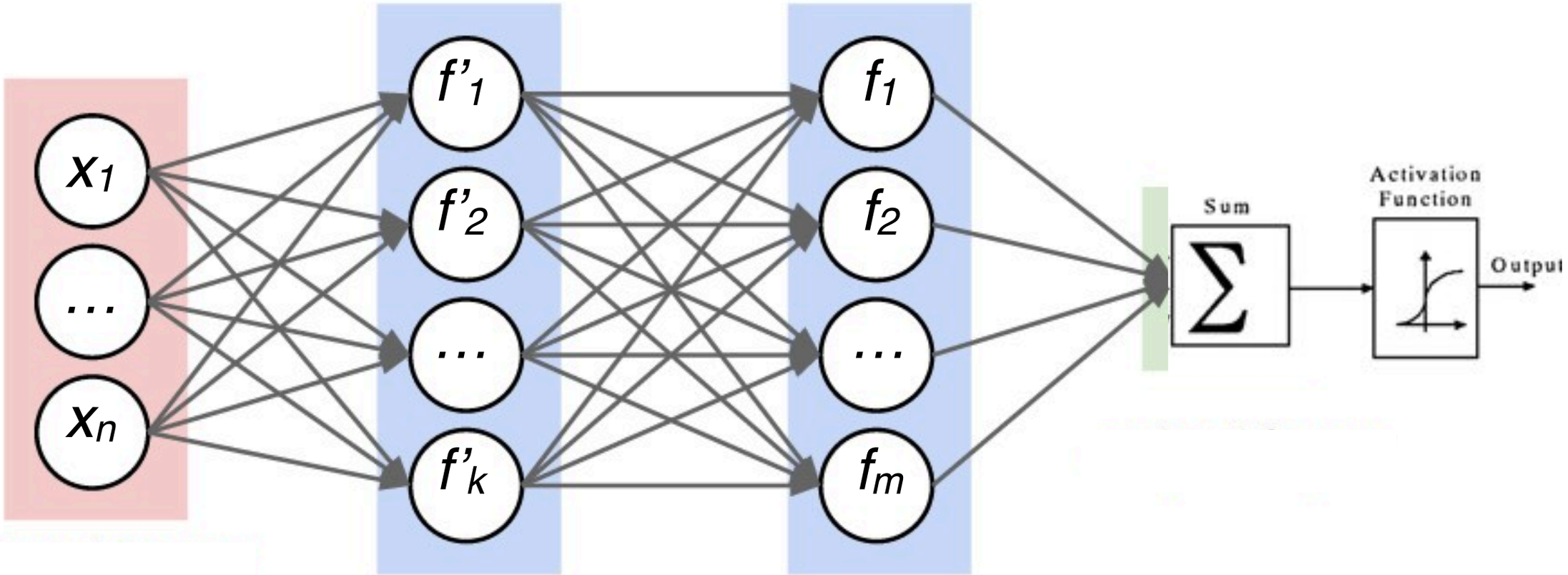
- Linear logistic regression
  - requires decision boundary be a plane in terms of input features

Neural Network Intuition: Building Features



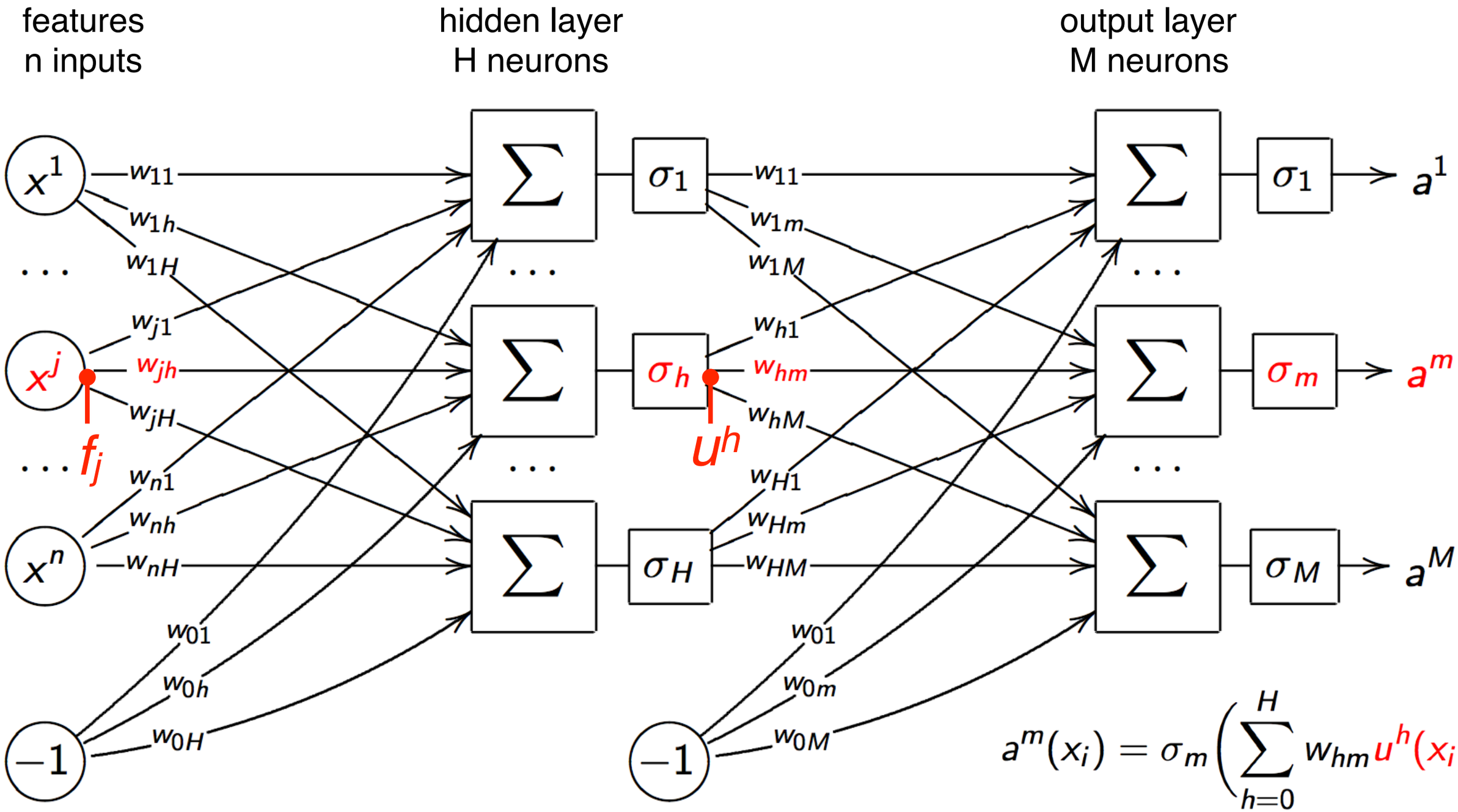
- Linear logistic regression
  - use NN layer to construct features more appropriate for linear logistic regression

Neural Network Intuition: Building Features



- Linear logistic regression
  - use extra NN layer to construct new features which are more appropriate to construct features which are appropriate for linear logistic regression

Backpropagation



$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm}\, u^h(x_i)\right)$$

(n+1)H+(H+1)M total model parameters

$$u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh}\, f_j(x_i)\right)$$

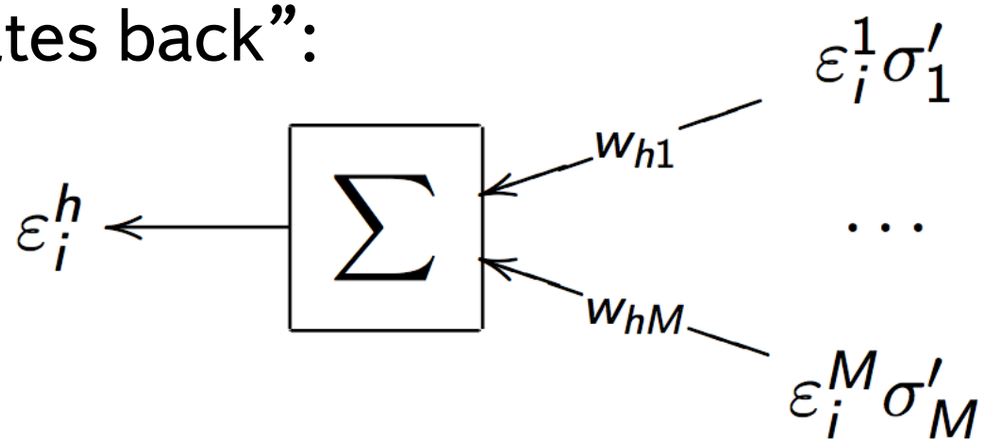
Loss Variation

- We want to correct (quadratic) loss by varying w appropriately
  - output layer error: $$\frac{\partial \mathscr{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$
- Let's propagate it back to u:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right)$$

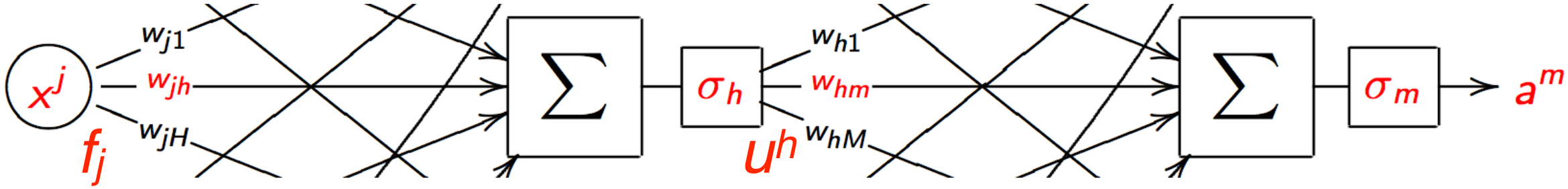
$$\frac{\partial \mathscr{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M} \frac{\partial \mathscr{L}_i(w)}{\partial a^m} \sigma'_m(\cdot) w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$$

  - this is previous (dense) layer error
    - note, errors "propagates back":



- This may be repeated further back for all layers

Stochastic Gradient Optimization



- Now we can get loss derivatives for all model parameters (weights):

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathscr{L}_i(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i), \quad m = 1..M, \quad h = 0..H;$$

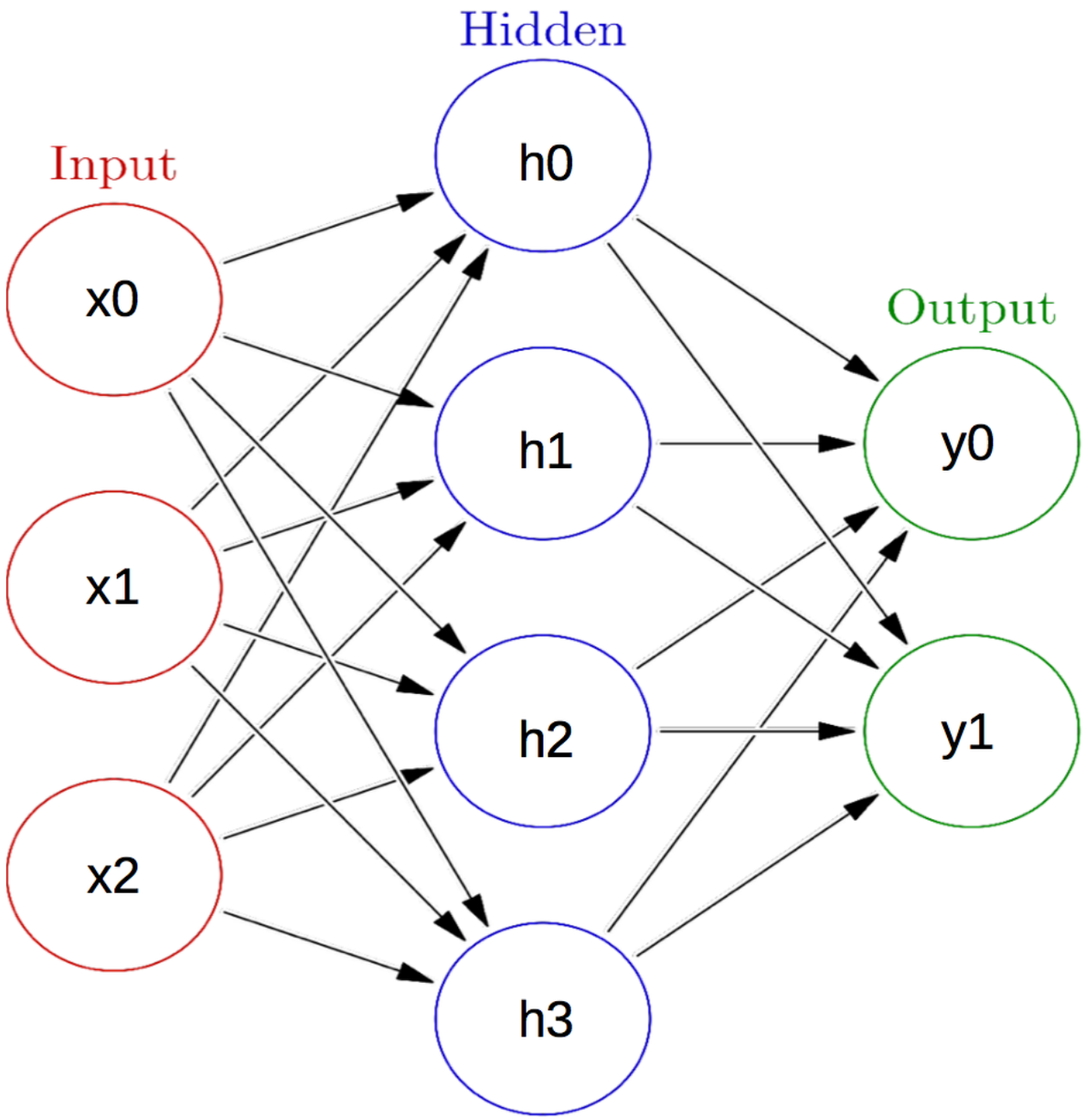
$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathscr{L}_i(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h f_j(x_i), \quad h = 1..H, \quad j = 0..n;$$

- These are used for Stochastic Gradient Descent optimization
- NB: massive tensor computations
  - how long would it take to compute 1000D gradient numerically?
  - calculations may be significantly speed up by using vector CPU operations or loading them into GPU

Outlook

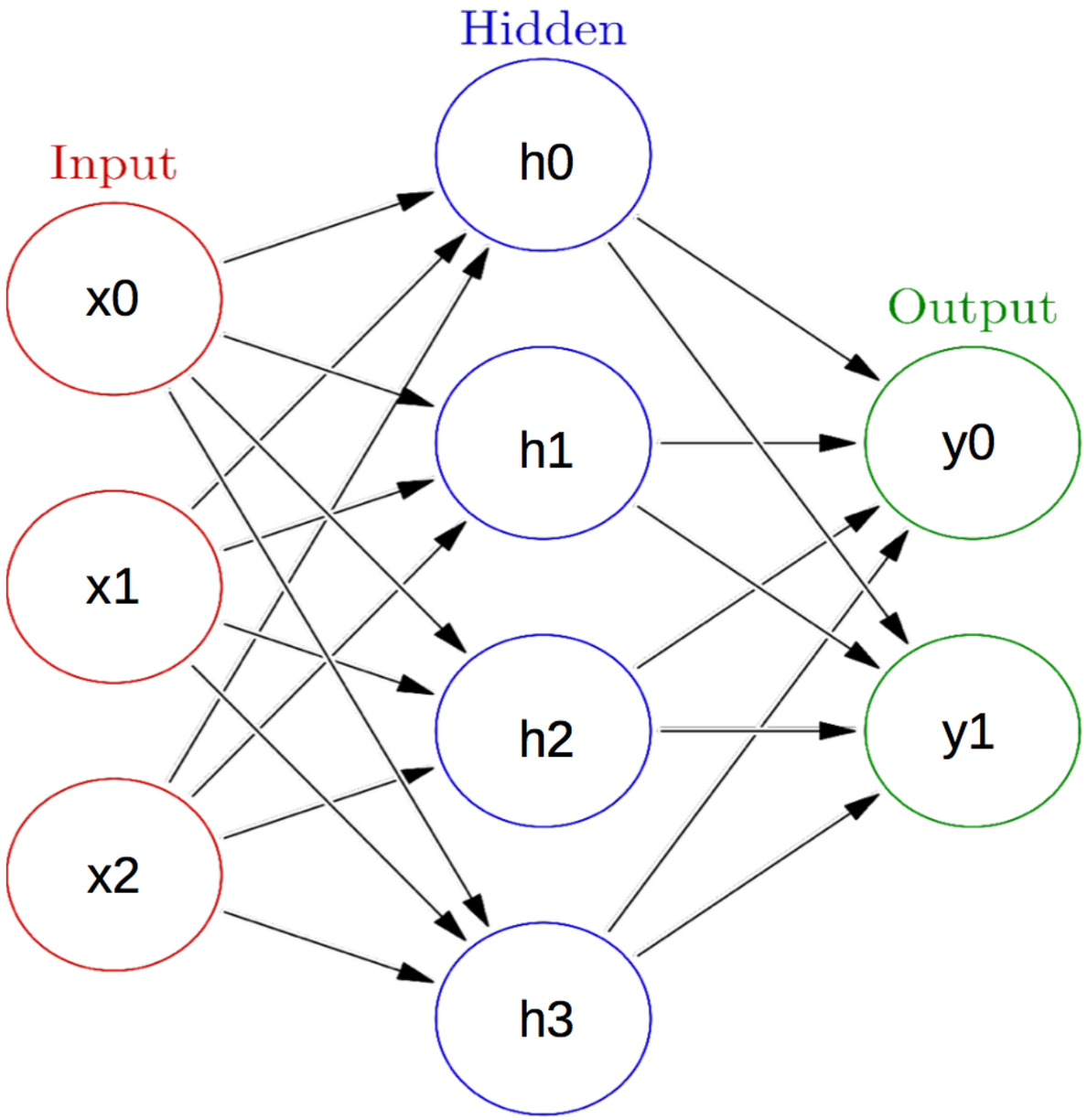
- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
  - initialization
  - dropout
  - activation function
  - batch normalization
- Decision Trees
- Illustrations
- Concluding remarks

Initialization



- Initialize with zeros
  - $0 \rightarrow w$
- What would be the first step?

Initialization



- Break the symmetry!
  - N(0, 0.01) → *w*
  - U(0, 0.1) → *w*
  - …

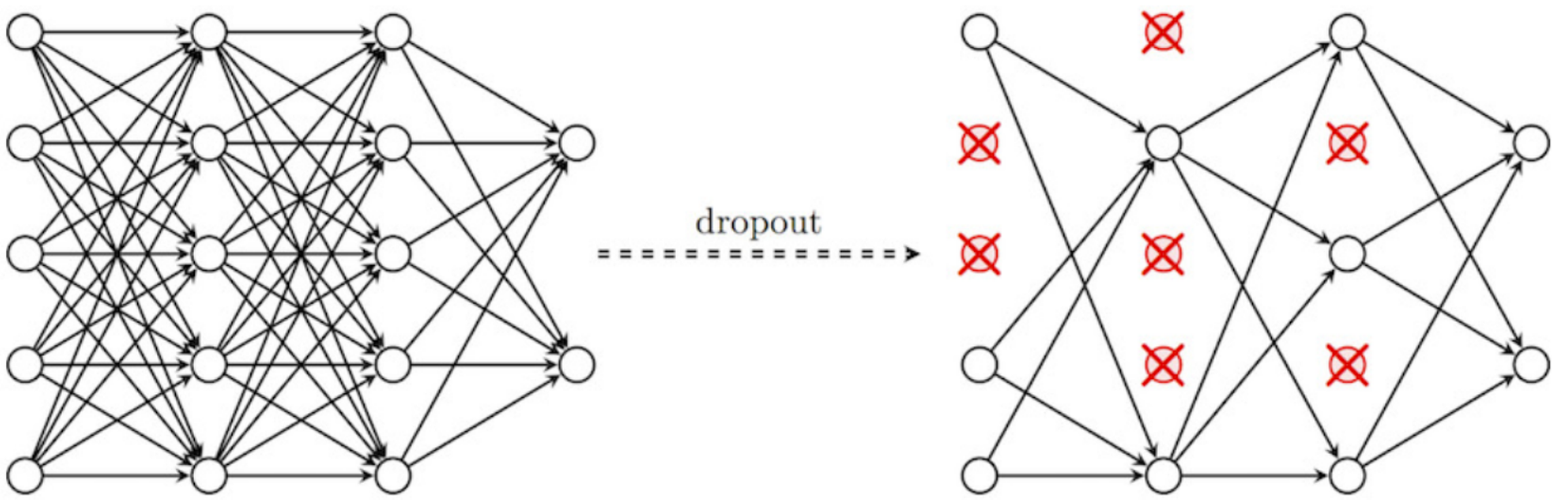
Dropout

- On every step of optimization disconnect neurone h of layer ℓ with probability p_ℓ:

$$x_{ih}^{\ell+1} = \xi_h^\ell \, \sigma_h\Big(\sum_j w_{jh} x_{ij}^\ell\Big), \qquad \mathrm{P}(\xi_h^\ell = 0) = p_\ell$$

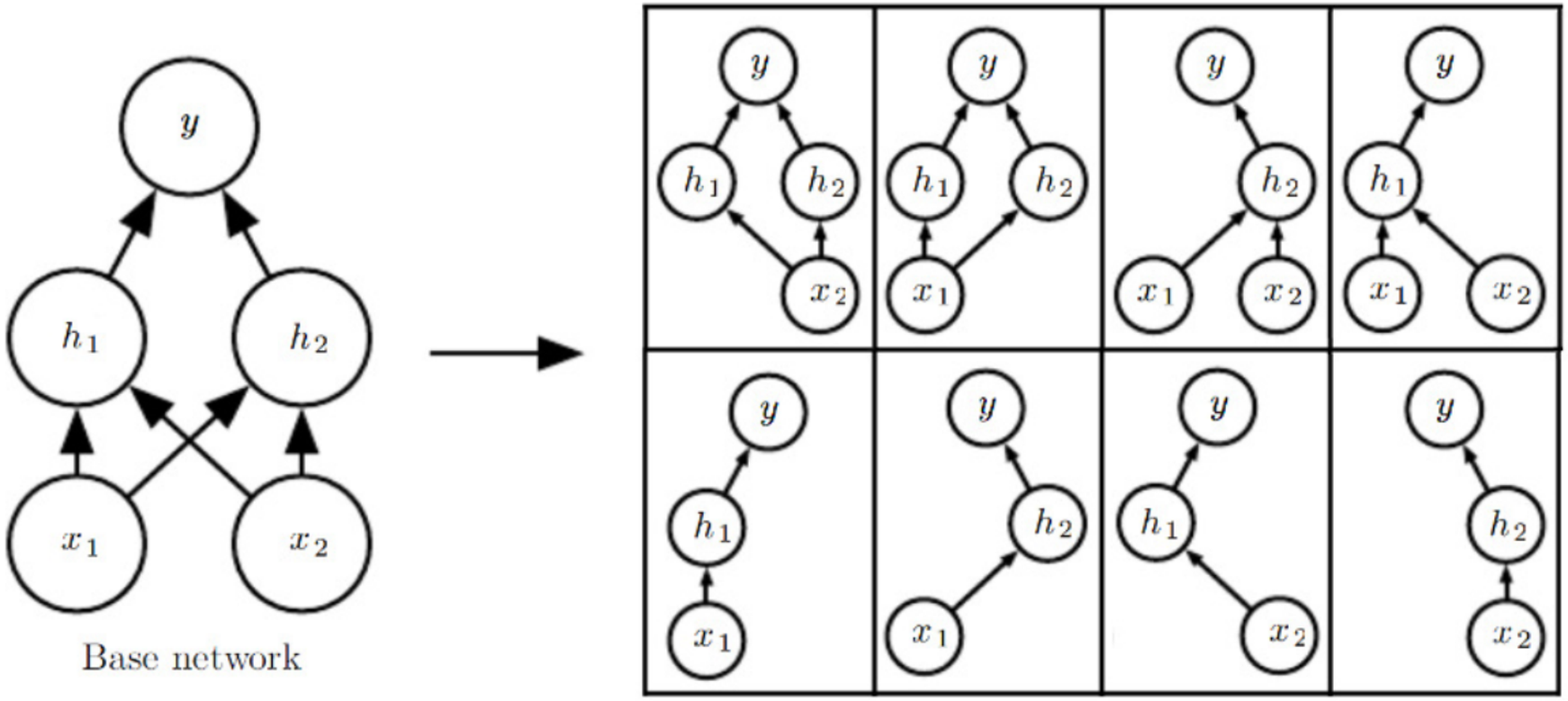
- When using, enabled all neurones with correction:

$$x_{ih}^{\ell+1} = (1 - p_\ell)\sigma_h\Big(\sum_i w_{jh} x_{ij}^\ell\Big)$$



Why Dropout?

- Regularization: train NN most sustainable to losing pN neurons
  - emulating brain sustainability
- Reduce overtraining of particular network domains by forcing them all to solve general problem rather than fine tune to compensate errors each other

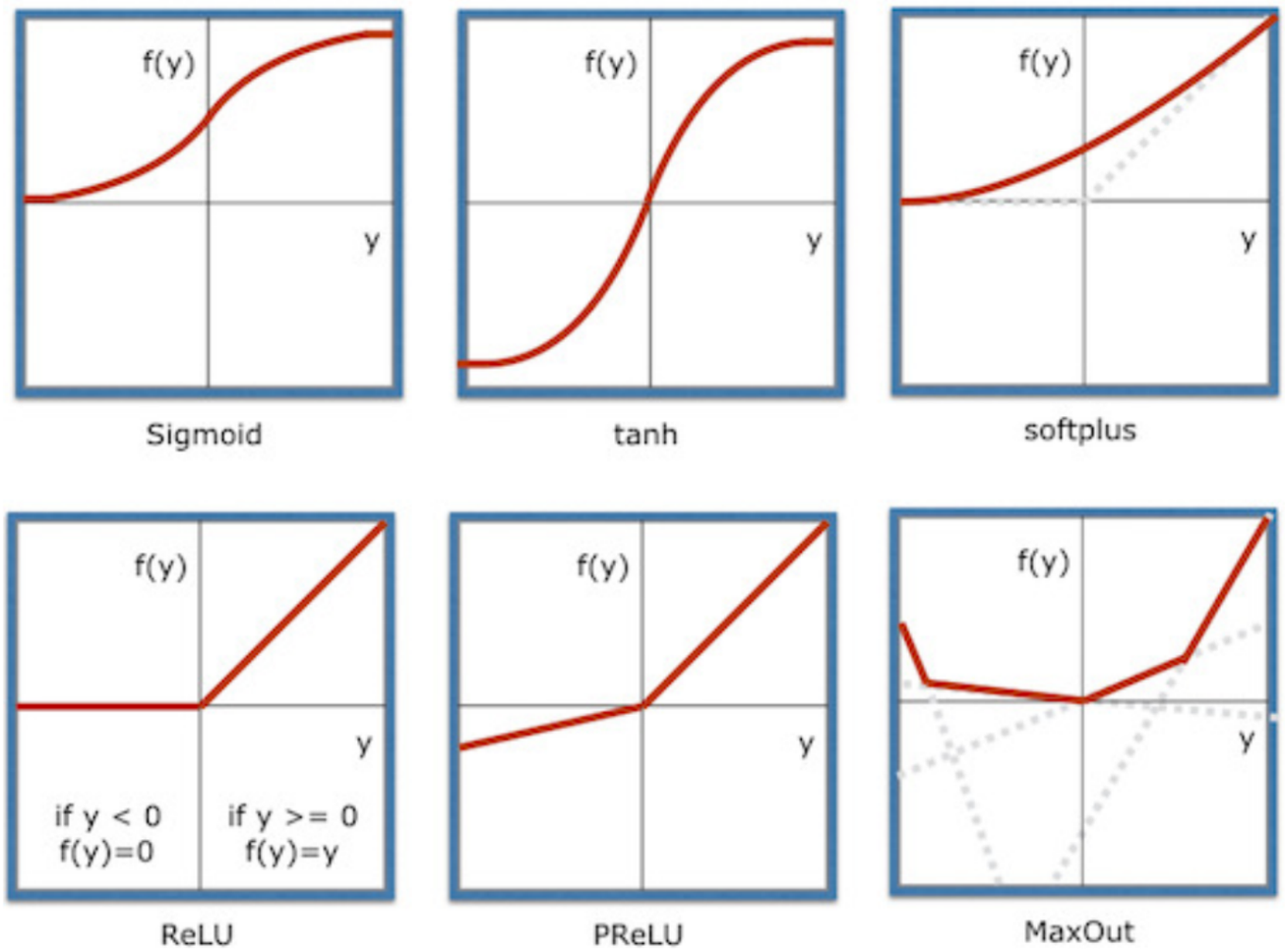


Base network

Activation Functions

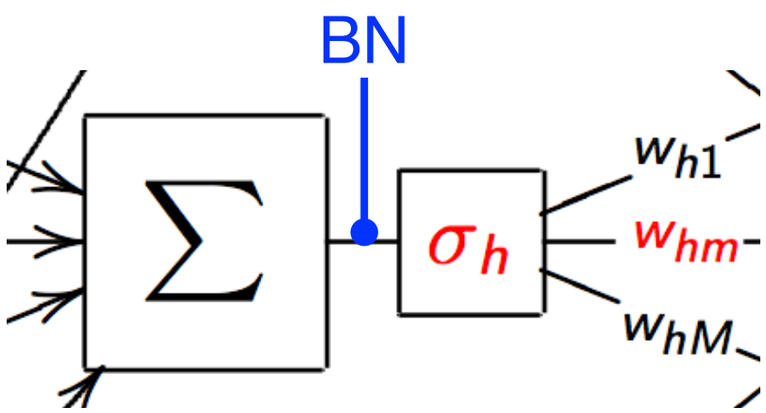
- Sigmoid-like functions are prompt to “vanishing gradients” problem: may cause “network paralysis”
- Solution: using rectified linear units like

$$\mathrm{ReLU}(y) = \max\{0, y\}; \qquad \mathrm{PReLU}(y) = \max\{0, y\} + \alpha \min\{0, y\}$$



Batch Normalization

- Process data in mini-batches
- For every mini-batch normalize inputs for every neuron to given mean β and variance γ
- β_{ij}, γi_j are model parameters now, learned as usual
- Heuristic: stabilising domain of input improve convergence



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

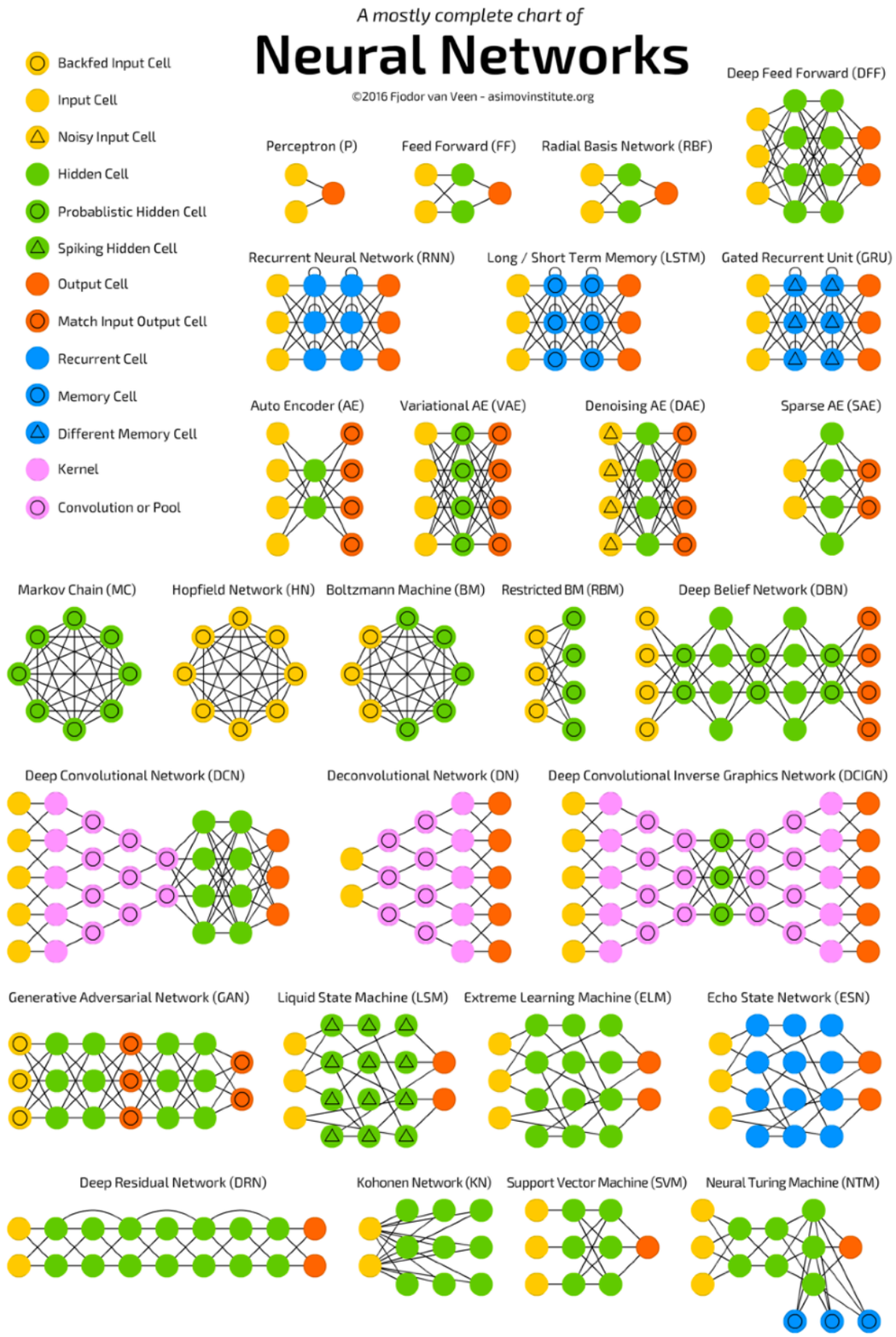
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Why NN?

- For classification problems NN doing as good as BDT
  - BDT even mor stable
- NN power is in flexibility
  - NN is a tool to build an arbitrary architecture, and as soon as appropriate loss function is set, it will learn the model eventually
- Different applications have different specifics
  - you are going to have a lot of fun by studying them in coming days!



https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

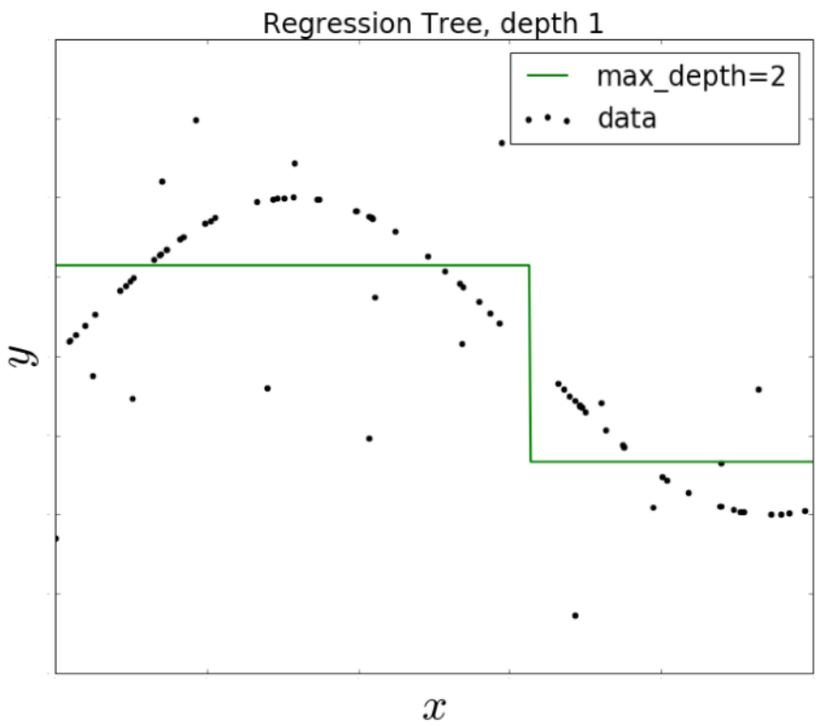
Outlook

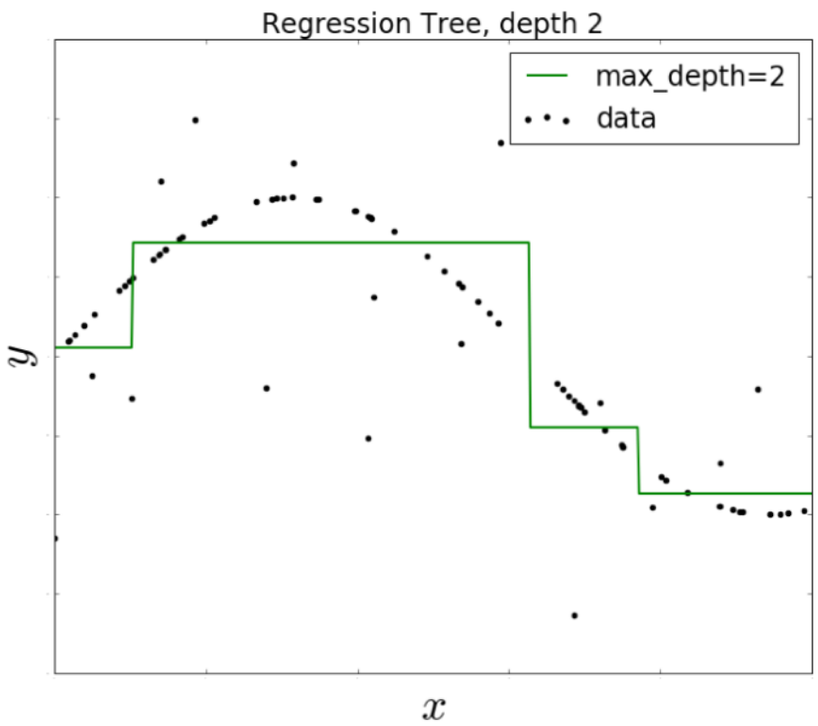
- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

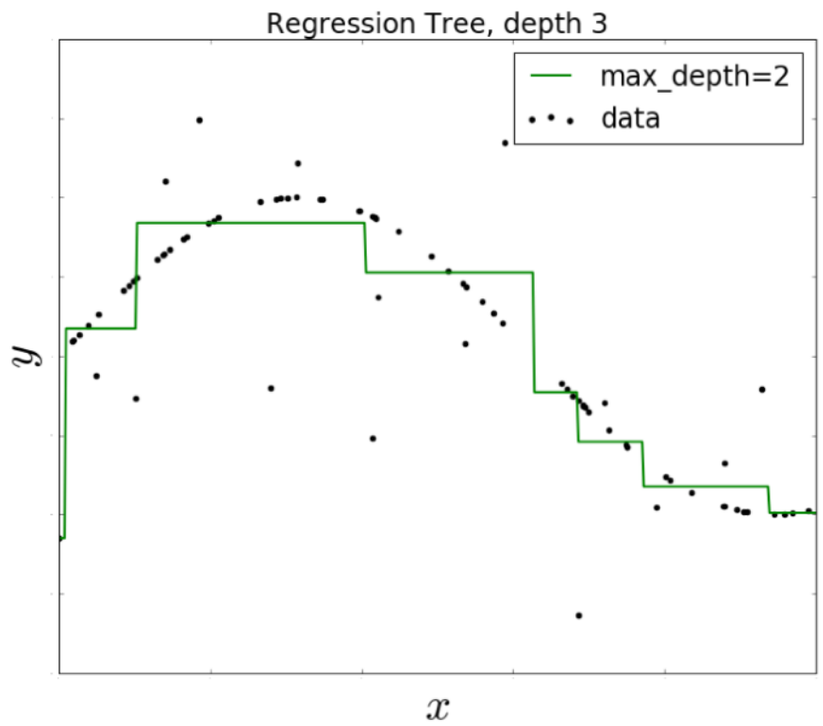
ML Basic Concept

- Machine Learning is a technical way to build generic model for the given dataset
  - inputs, outputs or goals may be very different though
    - classification
    - pattern recognition
      - text, speech, vision, etc.
    - new object generation
    - action control
      - games, driving, etc.
    - ...
- Key requirement: there is a way to effectively train the model

Decision Trees for Regression







- Build tree, optimize MSE in leafs
  - use greedy algorithm on every step

Ensembling Models

- Trees are not stable
  - small variations in data drastically affect the split to branches
- Use it for good
  - build different trees - ensemble of solutions
    - average them
    - get more stable result
- Approaches
  - bagging
  - boosting

Bagging (Bootstrap Aggregation)

- Select training subset from data randomly with replacement
- Build tree for every subset
- Average all predictions - Profit!

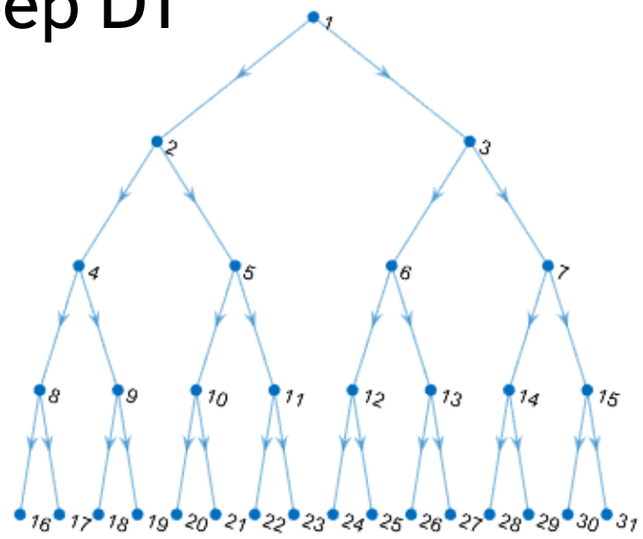
- Random Forest
  - bagging
  - and also chose a subset of input features randomly for every subset
- Allows to maintain high dimensibility in shallow trees

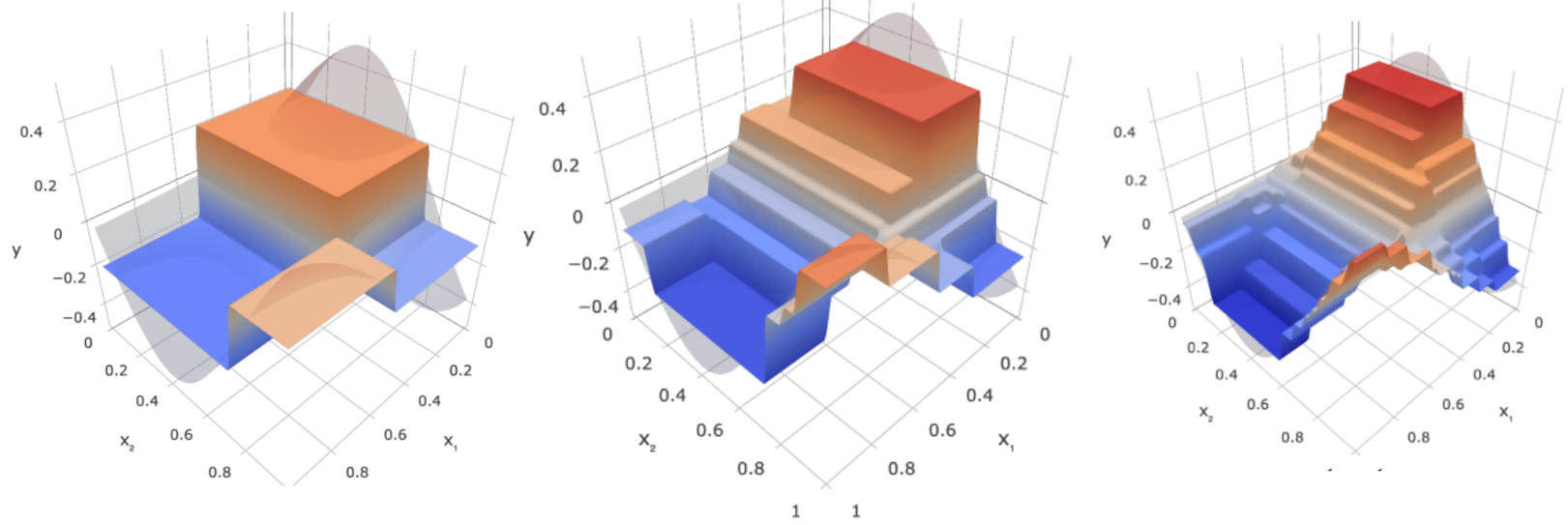
Boosting

- Make a sequence of predictors
  - next predictor corrects mistakes of the previous one
- Gradient Boosting
  - train next predictor on the difference between previous predictor and the actual value
  - result is a sum of shallow trees
- XGBoost smart implementation of the GB approach
  - parameters: max # of trees, single tree depth

Generic Model: Decision Trees

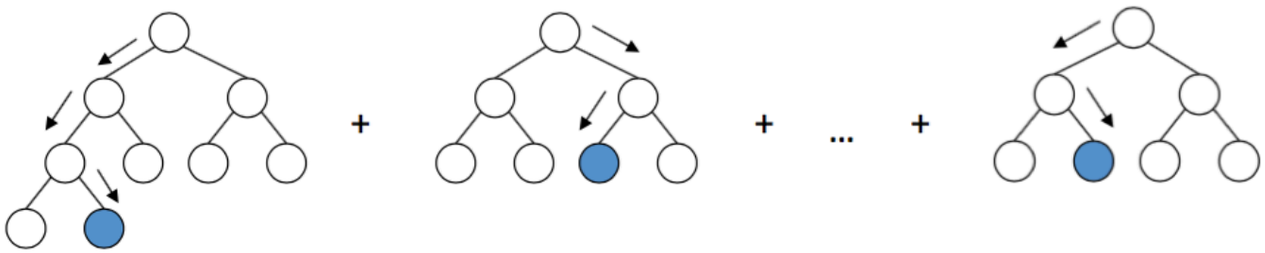
Deep DT

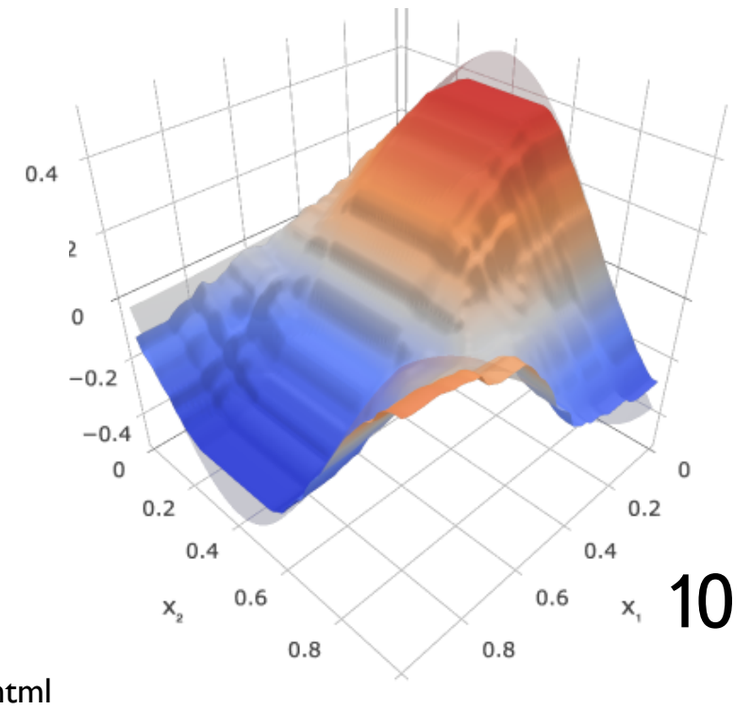




depth=2,4,6 DT

Shallow Boosted DT





100× depth=3 BDT

http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

BDT vs NN

- Both have similar performance
  - as after meaningful training both represent the dataset properties
    - actual winner depends on data specifics
- NN are more flexible for building models beyond classification problems
  - special architecture (connections)
  - special additions to loss training function

Outlook

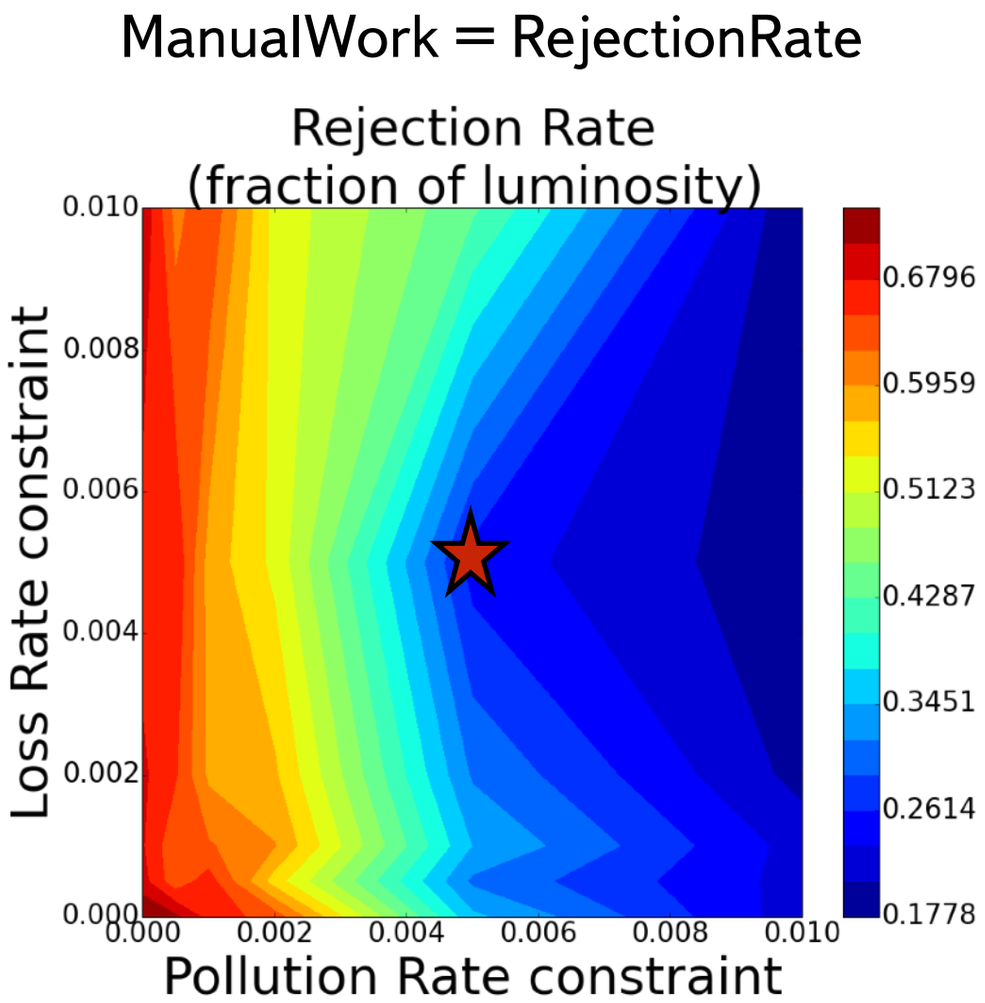
- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
  - few examples
  - HEP data
  - industrial ML
  - HEP success stories
- Concluding remarks

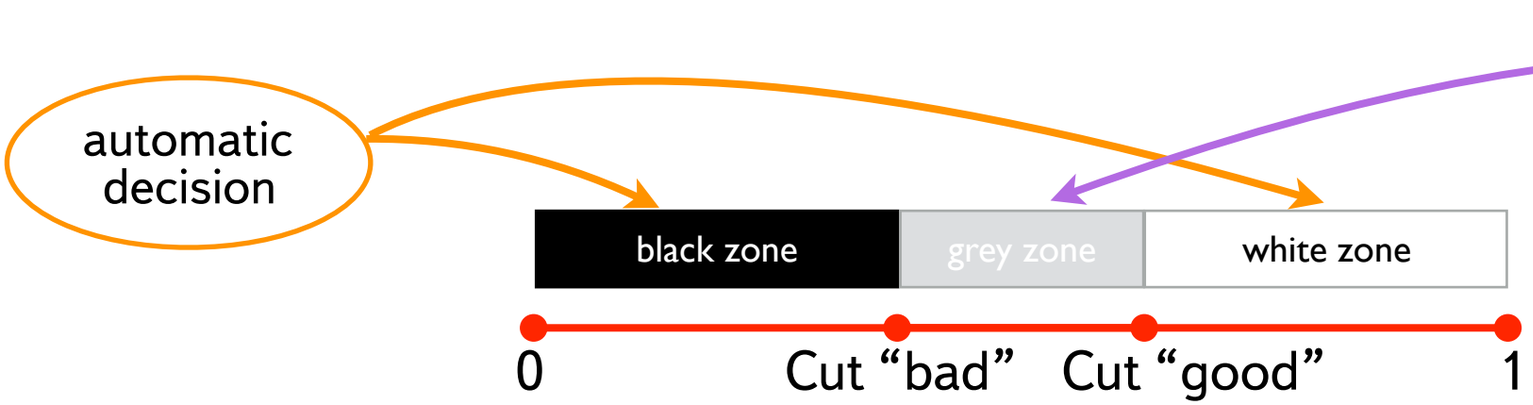
Types of Learning

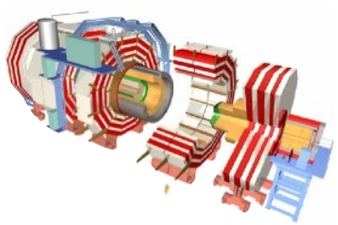
- There is well labelled data
  - we want to learn how to produce labels for more data like this
    - automation of routine
      - supervised learning
- There is labelled data, and unlabelled data which are assumed to be similar
  - we want to derive labels
    - propagation of knowledge
      - supervised learning
- There is unlabelled data with some properties
  - we want to infer some properties of this data
    - inferring new knowledge
      - unsupervised learning

(Automation of) Data Quality Monitoring

- Data quality monitoring is boring and time consuming work
  - Extremely important to guarantee solid physics results
  - Different properties of high level physics objects are analysed
    - photons, muons, calorimeter jets, combined (particle flow) jets
      - kinematics, vertices distribution
        - ~2500 features in total
- Build classifier to decide if data are good or bad



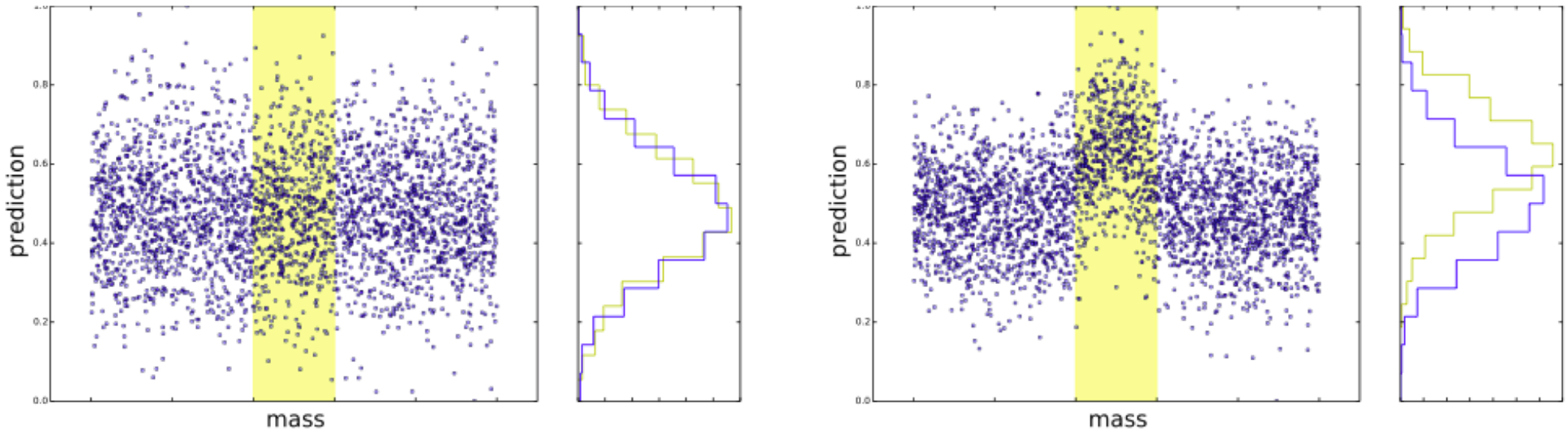




Caveat: how good the expert is?

Propagation of Knowledge

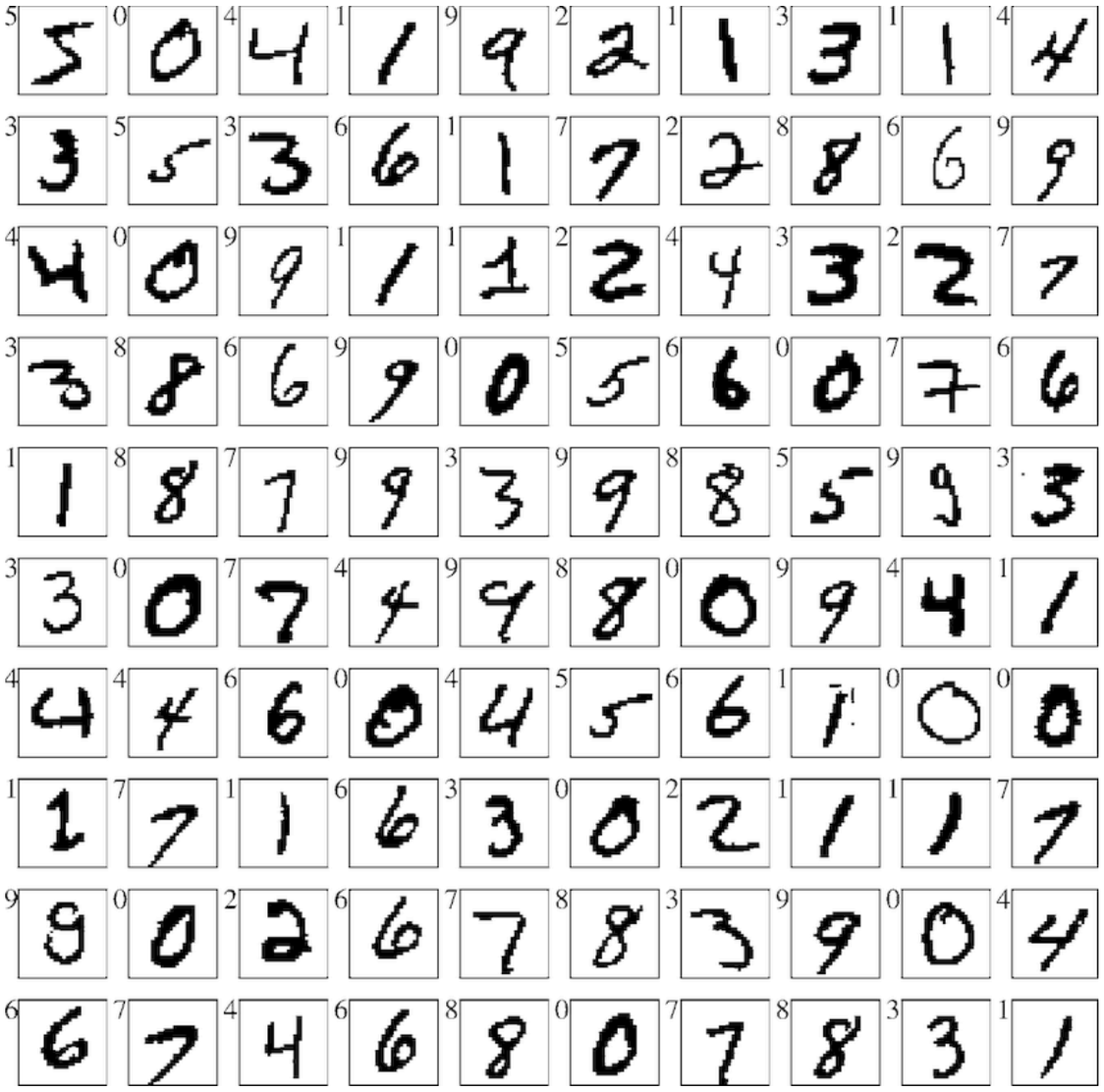
- Most common use case for HEP data analysis (MVA)
  - Train on MC or MC&DATA mixture
    - apply to data

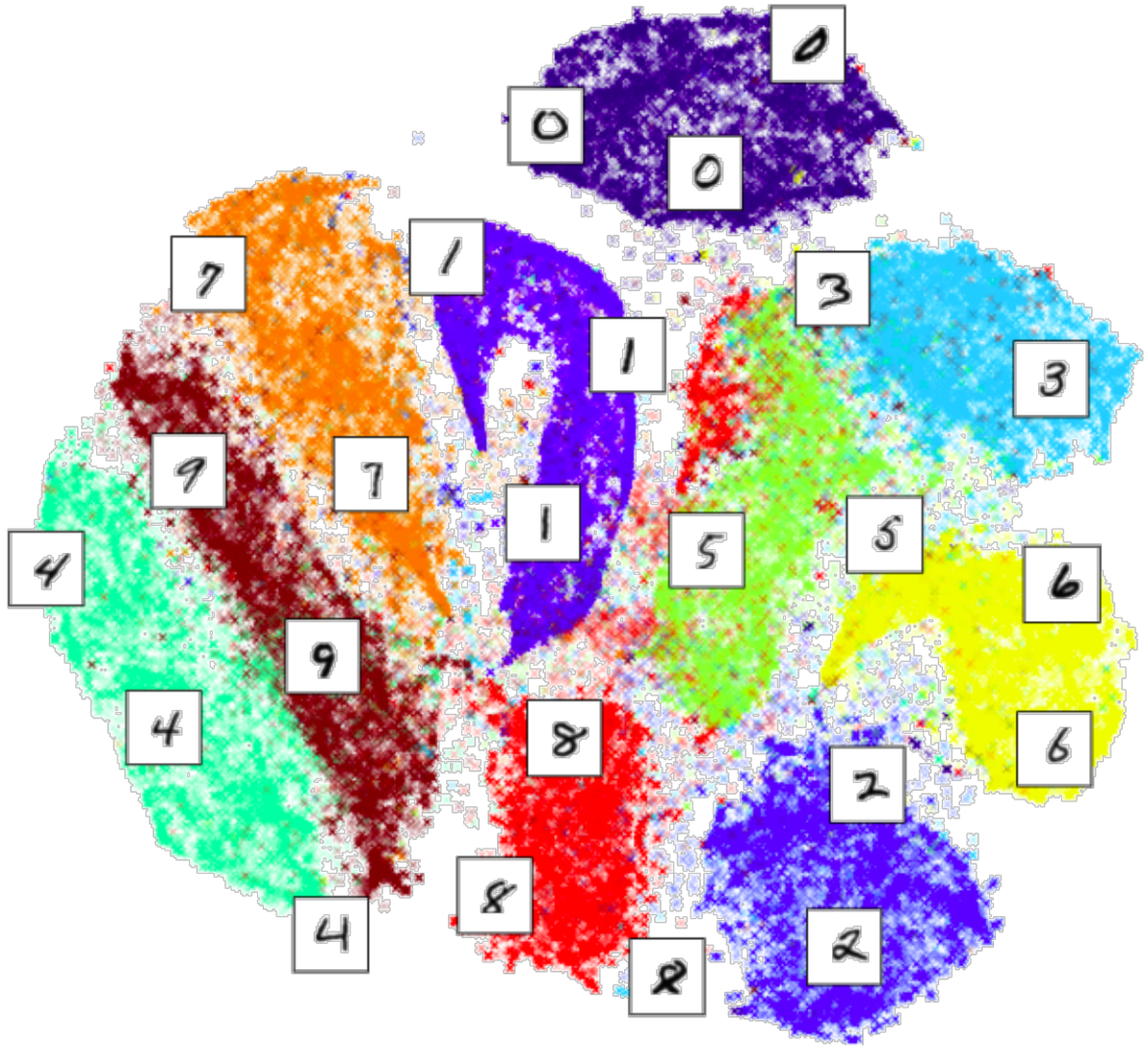


  - Need to account for (minor) difference between MC and data
    - different approaches: data doping, domain adaptation etc.
      - either approach requires control sample to test the difference
      - no ML tools to propagate difference in samples into systematics in classifier

Caveat: how to determine systematics due to sample difference?

Unsupervised Learning: Dimensionality Reduction



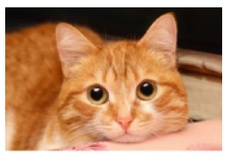


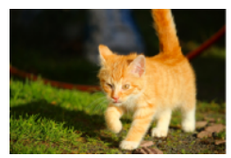
- Reduce 28x28=784D space to 2D
  - t-SNE algorithm effectively separates classes

Generative Models

- Learn properties of the given dataset
  - determine domain where objects of the dataset are concentrated
  - probability distributions in the domain
- Then can generate random new but similar objects

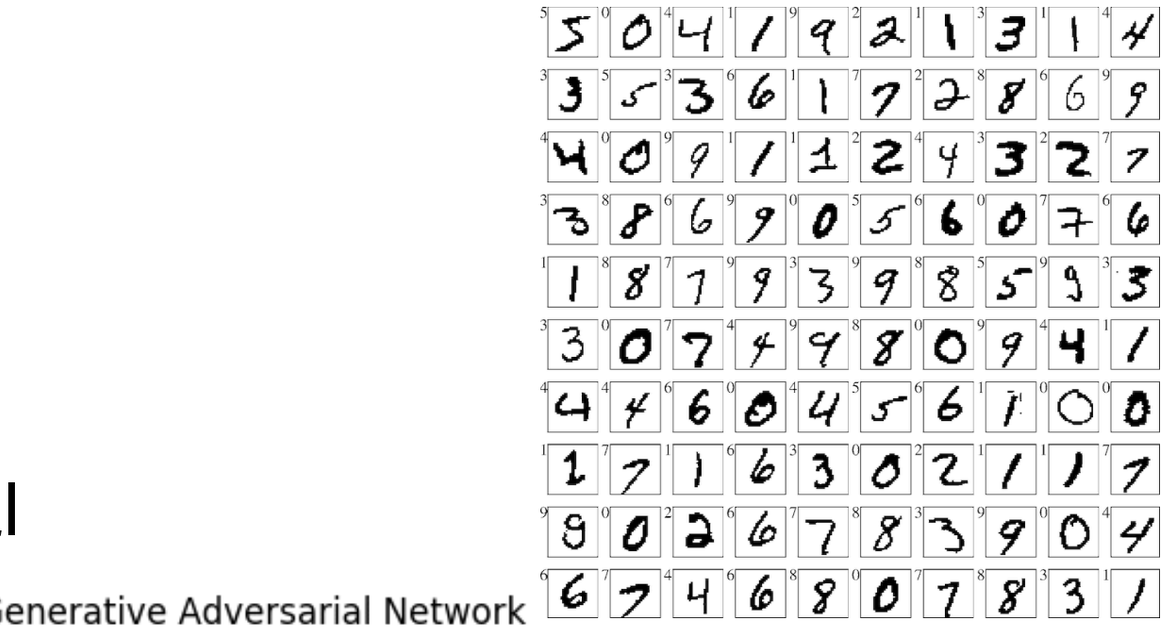
  - Unconditional
    - domain for all objects (e.g. “cat on the image”)
  - Conditional
    - objects are accompanied by features
      - different domains for different conditions (e.g. “red cat”)





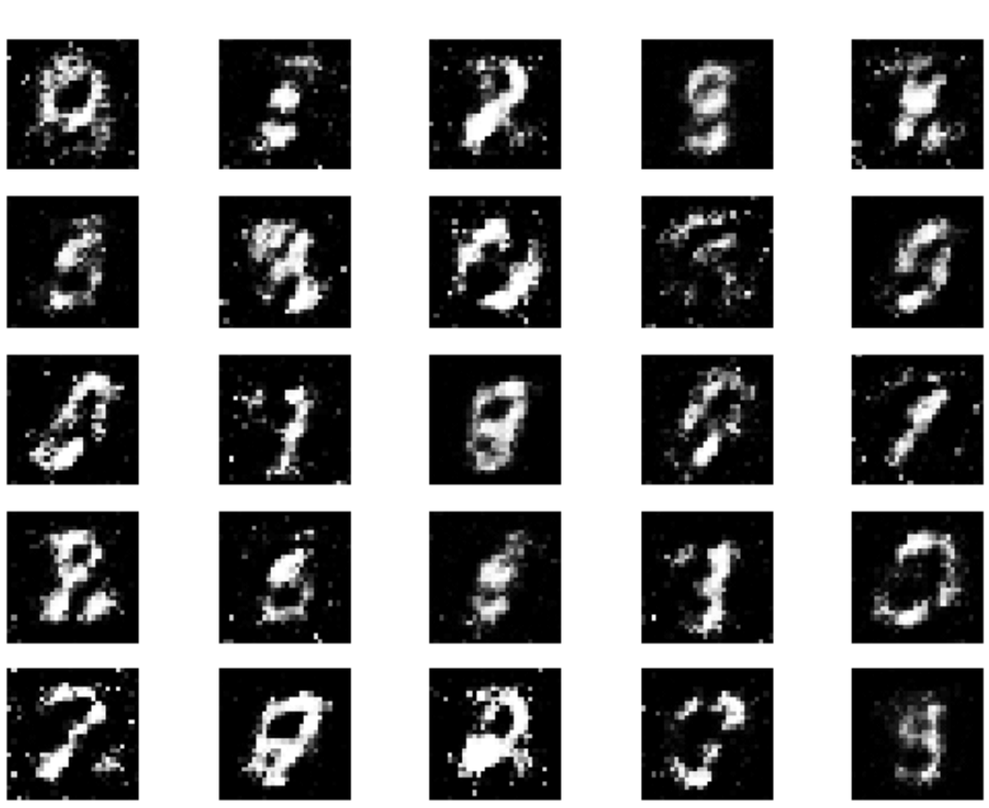
Baseline Generator

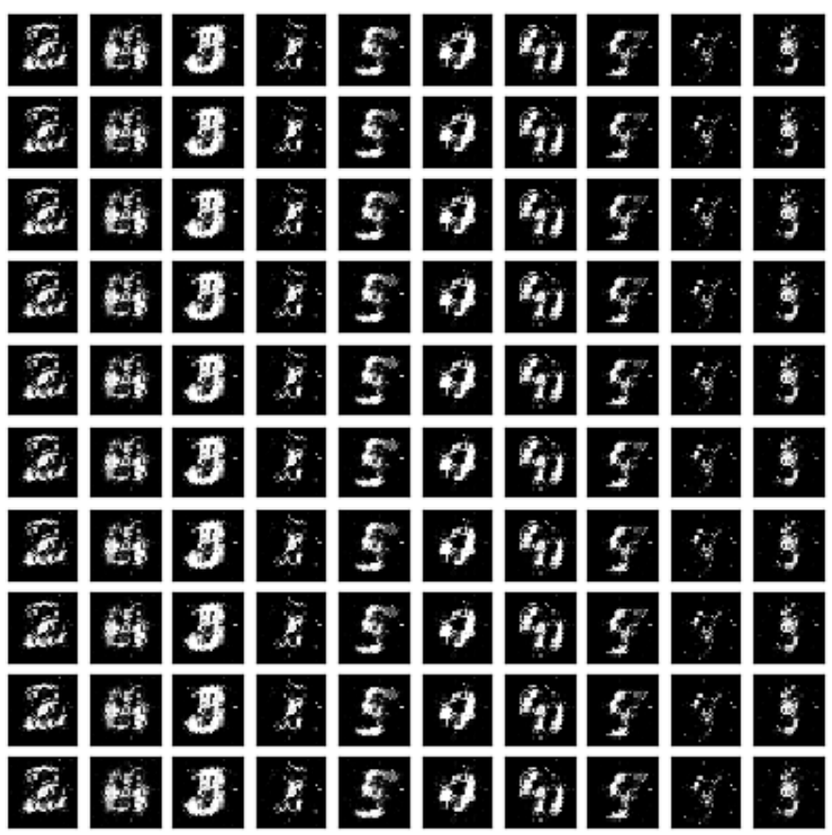
unconditional



Generative Adversarial Network

conditional



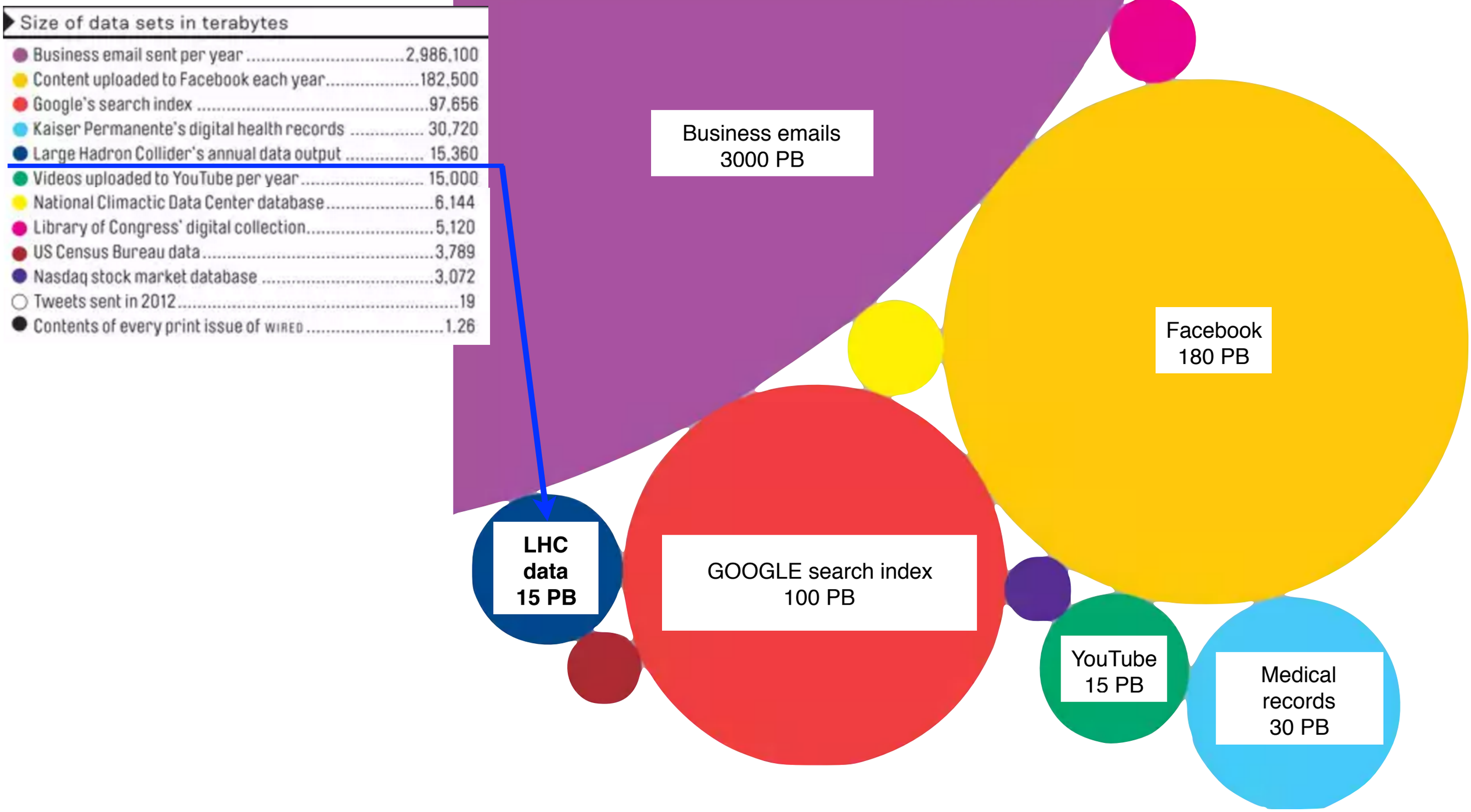


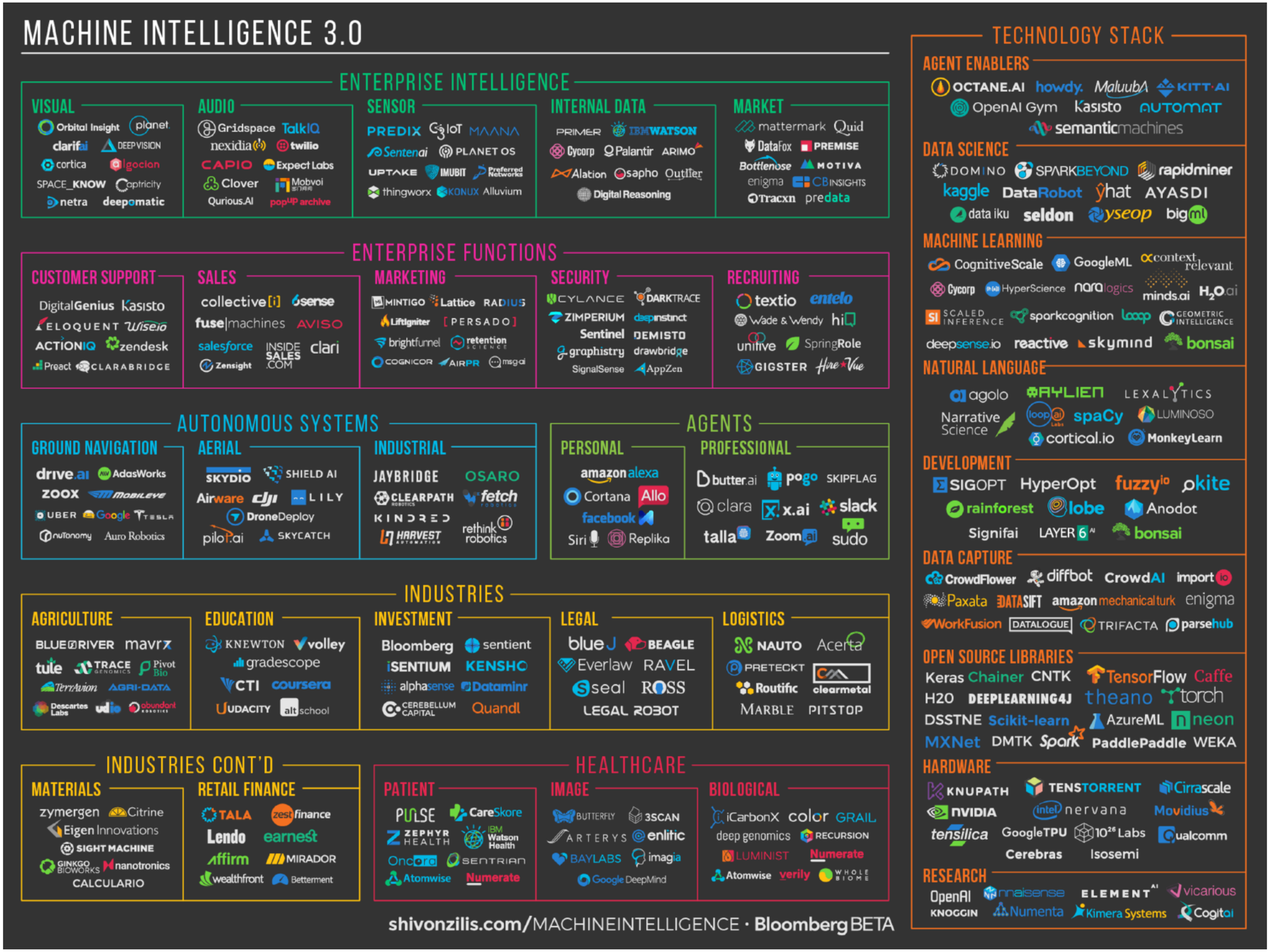
Epoch 22

Big Data Big Players

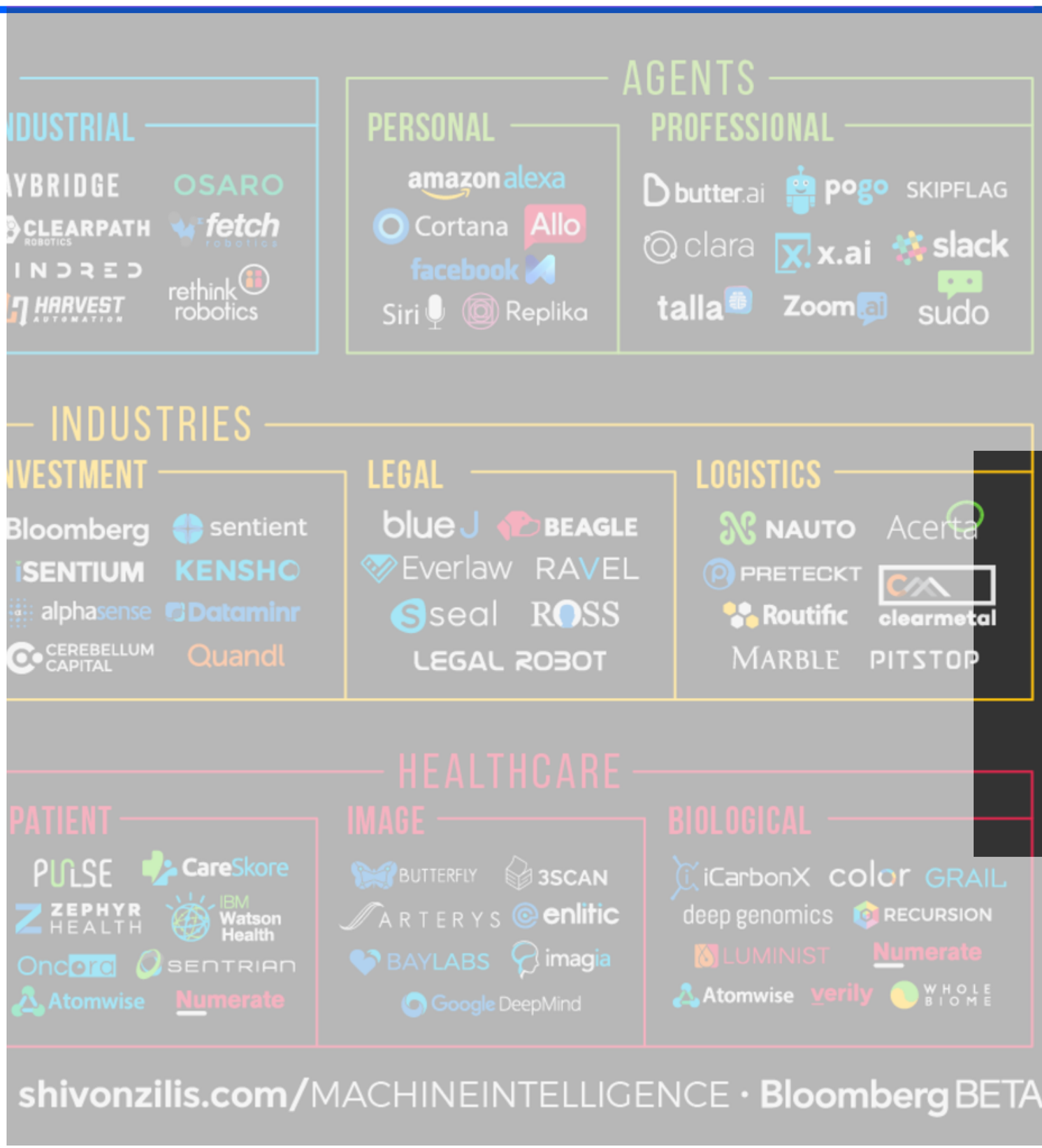
data collected in 2012

адаптировано из http://www.wired.com/2013/04/bigdata/




MACHINE INTELLIGENCE 3.0
ENTERPRISE INTELLIGENCE
VISUAL
Orbital Insight
planet
clarifai
DEEP VISION
cortica
algocian
SPACE_KNOW
captricity
netra
deepomatic
AUDIO
Gridspace
TalkIQ
nexidia
twilio
CAPIO
Expect Labs
Clover
Mobvoi
Qurious.AI
popUP archive
SENSOR
PREDIX
IoT
MAANA
Sentenai
PLANET OS
UPTAKE
IMUBIT
Preferred Networks
thingworx
KONUX
Alluvium
INTERNAL DATA
PRIMER
IBM WATSON
Cycorp
Palantir
ARIMO
Alation
sapho
Outlier
Digital Reasoning
MARKET
mattermark
Quid
DataFox
PREMISE
Bottlenose
MOTIVA
enigma
CB INSIGHTS
Tracxn
predata
ENTERPRISE FUNCTIONS
CUSTOMER SUPPORT
DigitalGenius
Kasisto
ELOQUENT
Wisedo
ACTIONIQ
zendesk
Preact
CLARABRIDGE
SALES
collective[i]
6sense
fuse|machines
AVISO
salesforce
INSIDE SALES .COM
clari
Zensight
MARKETING
MINTIGO
Lattice
RADIUS
LiftIgniter
[PERSADO]
brightfunnel
retention SCIENCE
COGNICOR
AIRPR
msg.ai
SECURITY
CYLANCE
DARKTRACE
ZIMPERIUM
deepinstinct
Sentinel
DEMISTO
graphistry
drawbridge
SignalSense
AppZen
RECRUITING
textio
entelo
Wade & Wendy
hi
unitive
SpringRole
GIGSTER
HireVue
AUTONOMOUS SYSTEMS
GROUND NAVIGATION
drive.ai
AdasWorks
ZOOX
MOBILEYE
UBER
Google
TESLA
nuTonomy
Auro Robotics
AERIAL
SKYDIO
SHIELD AI
Airware
dji
LILY
DroneDeploy
pilot.ai
SKYCATCH
INDUSTRIAL
JAYBRIDGE
OSARO
CLEARPATH
fetch
KINDRED
HARVEST
rethink robotics
AGENTS
PERSONAL
amazon alexa
Cortana
Allo
facebook
Siri
Replika
PROFESSIONAL
butter.ai
pogo
SKIPFLAG
clara
x.ai
slack
talla
Zoom.ai
sudo
INDUSTRIES
AGRICULTURE
BLUE RIVER
mavrx
tule
TRACE GENOMICS
Pivot Bio
TerrAvion
AGRI-DATA
Descartes Labs
udio
abundant
EDUCATION
KNEWTON
volley
gradescope
CTI
coursera
UDACITY
altschool
INVESTMENT
Bloomberg
sentient
iSENTIUM
KENSHO
alphasense
Dataminr
CEREBELLUM CAPITAL
Quandl
LEGAL
blueJ
BEAGLE
Everlaw
RAVEL
seal
ROSS
LEGAL ROBOT
LOGISTICS
NAUTO
Acerta
PRETECKT
Routific
clearmetal
MARBLE
PITSTOP
INDUSTRIES CONT'D
MATERIALS
zymergen
Citrine
Eigen Innovations
SIGHT MACHINE
GINKGO BIOWORKS
nanotronics
CALCULARIO
RETAIL FINANCE
TALA
zestfinance
Lendo
earnest
affirm
MIRADOR
wealthfront
Betterment
HEALTHCARE
PATIENT
PULSE
CareSkore
ZEPHYR HEALTH
IBM Watson Health
Oncora
SENTRIAN
Atomwise
Numerate
IMAGE
BUTTERFLY
3SCAN
ARTERYS
enlitic
BAYLABS
imagia
Google DeepMind
BIOLOGICAL
iCarbonX
color
GRAIL
deep genomics
RECURSION
LUMINIST
Numerate
Atomwise
verily
WHOLE BIOME
shivonzilis.com/MACHINEINTELLIGENCE · Bloomberg BETA
TECHNOLOGY STACK
AGENT ENABLERS
OCTANE.AI
howdy.
MaluubA
KITT.AI
OpenAI Gym
Kasisto
AUTOMAT
semanticmachines
DATA SCIENCE
DOMINO
SPARKBEYOND
rapidminer
kaggle
DataRobot
yhat
AYASDI
data iku
seldon
yseop
bigml
MACHINE LEARNING
CognitiveScale
GoogleML
context relevant
Cycorp
HyperScience
narralogics
minds.ai
H2O.ai
SCALED INFERENCE
sparkcognition
Loop
GEOMETRIC INTELLIGENCE
deepsense.io
reactive
skymind
bonsai
NATURAL LANGUAGE
agolo
AYLIEN
LEXALYTICS
Narrative Science
spaCy
LUMINOSO
cortical.io
MonkeyLearn
DEVELOPMENT
SIGOPT
HyperOpt
fuzzyio
okite
rainforest
lobe
Anodot
Signifai
LAYER 6 AI
bonsai
DATA CAPTURE
CrowdFlower
diffbot
CrowdAI
import.io
Paxata
DATASIFT
amazon mechanical turk
enigma
WorkFusion
DATALOGUE
TRIFACTA
parsehub
OPEN SOURCE LIBRARIES
Keras
Chainer
CNTK
TensorFlow
Caffe
H20
DEEPLEARNING4J
theano
torch
DSSTNE
Scikit-learn
AzureML
neon
MXNet
DMTK
Spark
PaddlePaddle
WEKA
HARDWARE
KNUPATH
TENSTORRENT
Cirrascale
NVIDIA
intel nervana
Movidius
tensilica
GoogleTPU
10^26 Labs
Qualcomm
Cerebras
Isosemi
RESEARCH
OpenAI
maisense
ELEMENT AI
vicarious
KNOGGIN
Numenta
Kimera Systems
Cogitai

Domain Relevant for Science



Generative Models in Cosmology

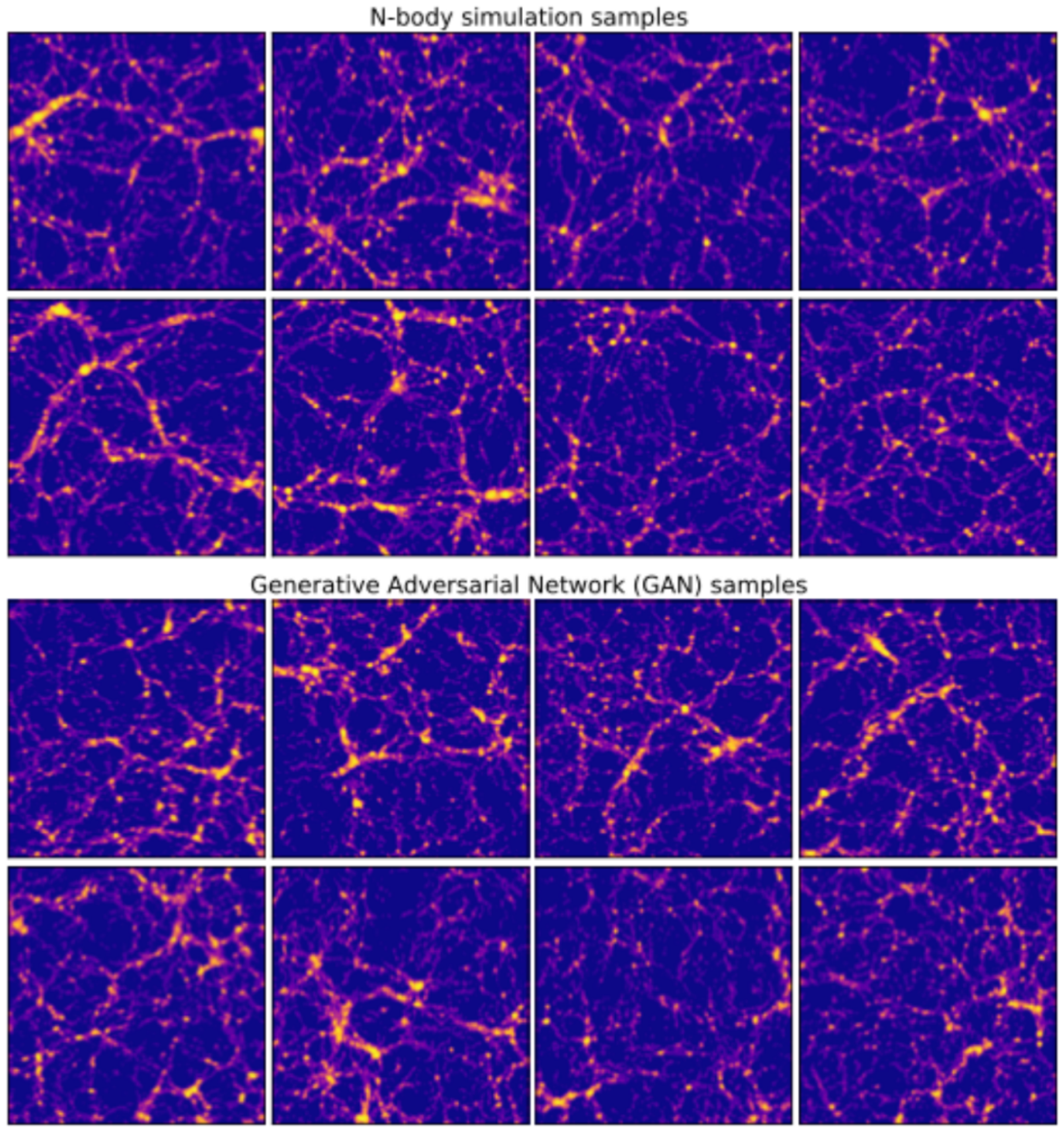
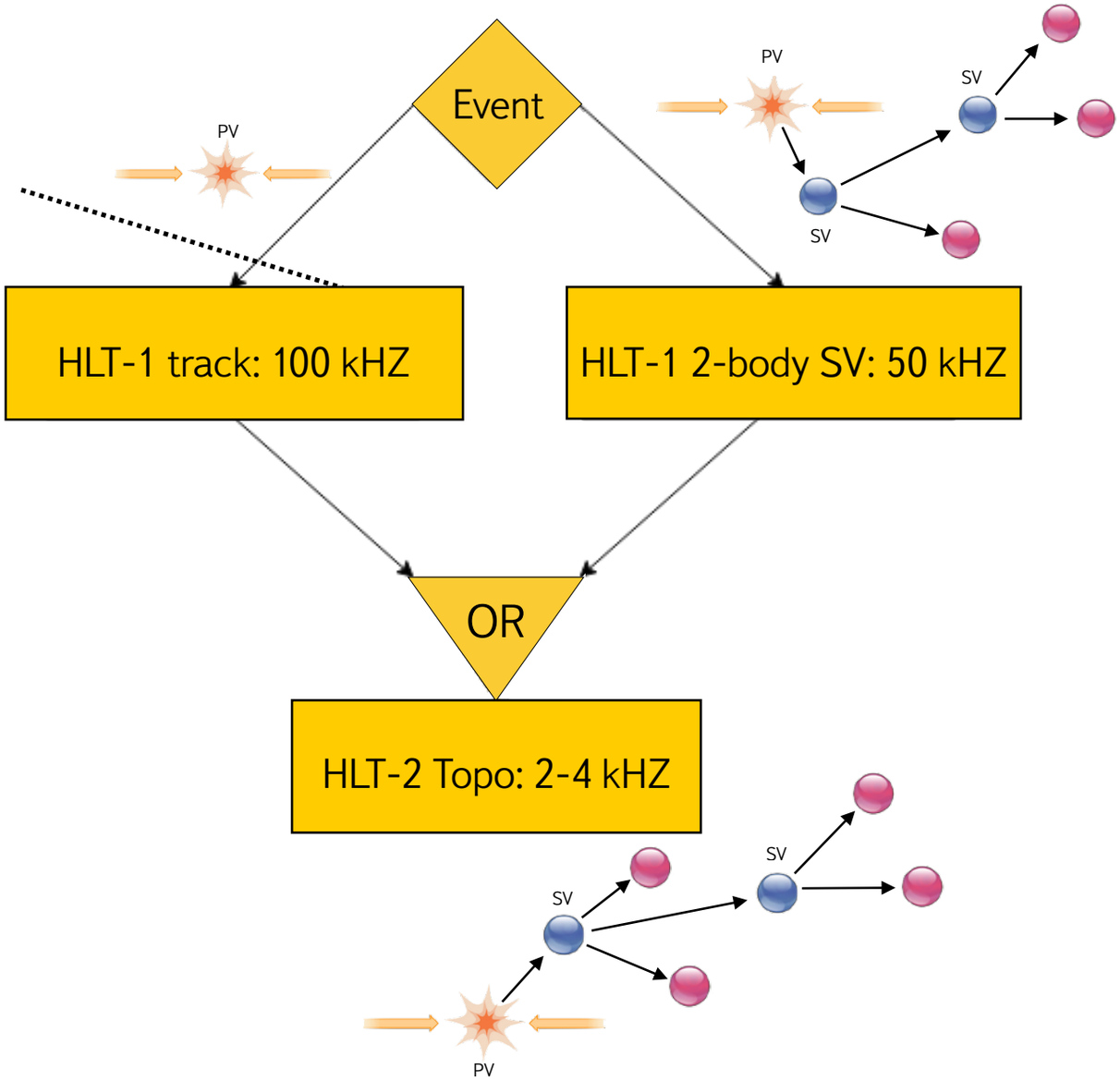


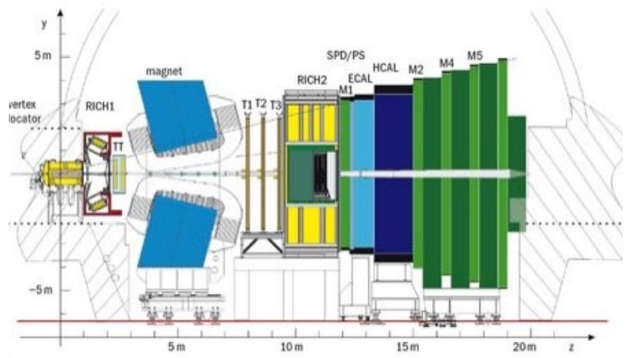
Figure 3: Samples from N-body simulation (top two rows) and from GAN (bottom two rows) for the box size of 100 Mpc. In this figure, transformation S1 with $k = 7$ was applied.

Rodríguez et al,

arXiv:1801.09070

Success Stories: LHCb Trigger



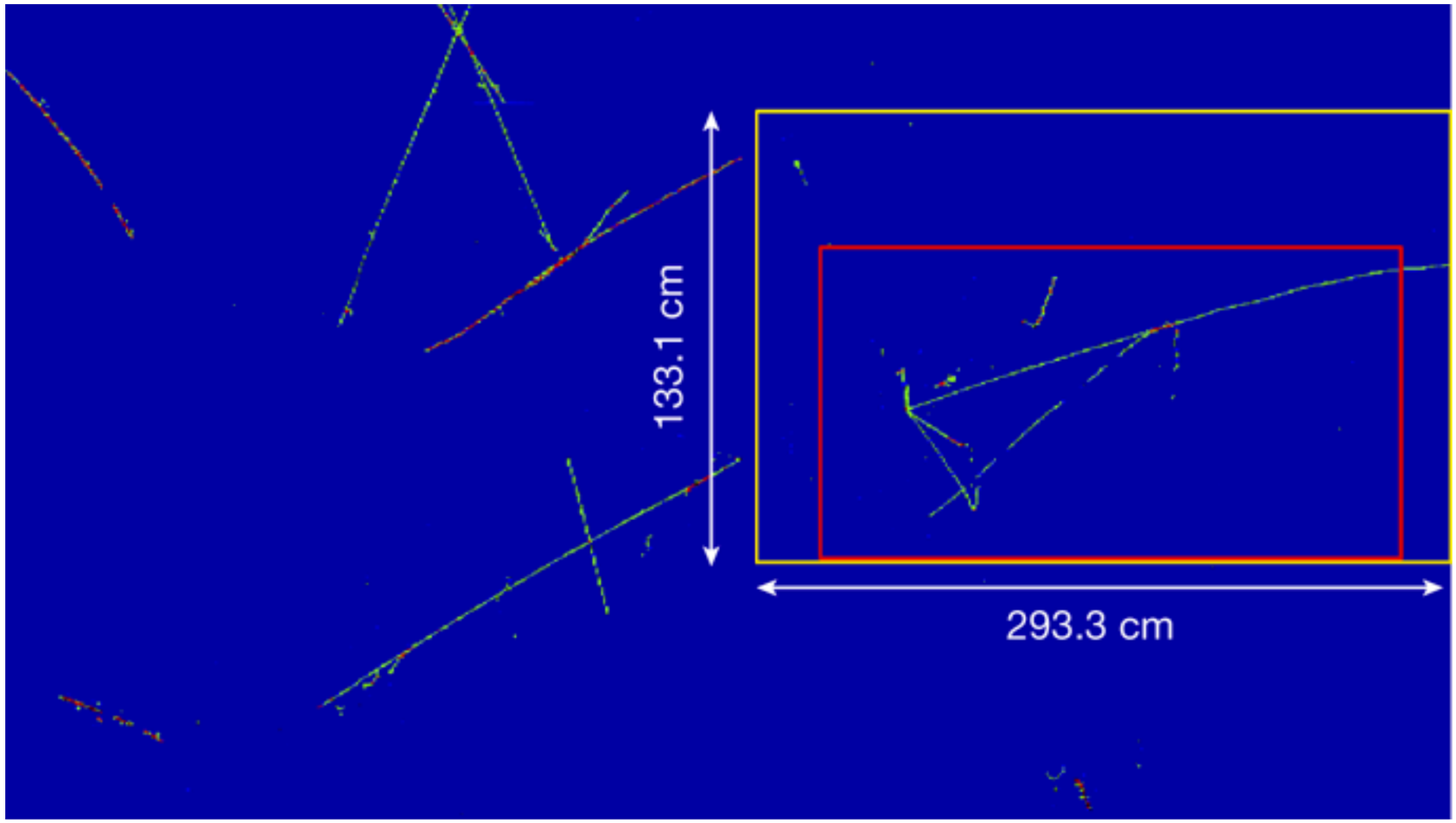


mode	2.5 kHz	4. kHz
$B^0 \to K^*[K^+\pi^-]\mu^+\mu^-$	1.64	1.72
$B^+ \to \pi^+K^-K^+$	1.59	1.65
$B^0_s \to D^-_s[K^+K^-\pi^-]\mu^+\nu_\mu$	1.14	1.47
$B^0_s \to \psi(1S)[\mu^+\mu^-]K^+K^-\pi^+\pi^-$	1.62	1.71
$B^0_s \to D^-_s[K^+K^-\pi^-]\pi^+$	1.46	1.52
$B^0 \to D^+[K^-\pi^+\pi^+]D^-[K^+\pi^-\pi^-]$	1.40	1.86

- Aggressively re-optimised topological trigger for Run II operation
- Gain 10%..70% efficiency for different channels!

SS: MicroBooNE Neutrino Selection

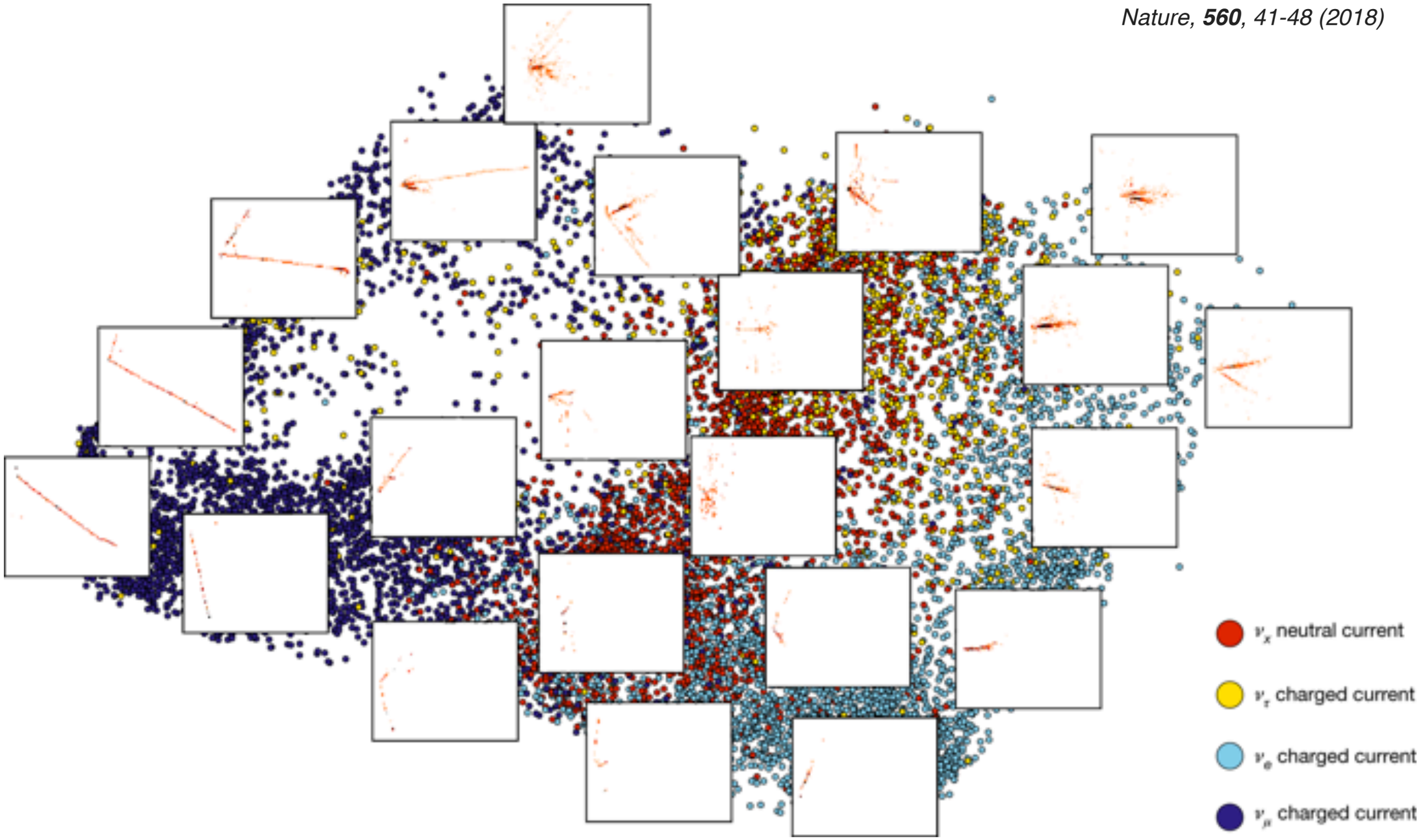
*Nature, **560**, 41-48 (2018)*



- CNN effectively predicts bounding box containing interacting neutrino

SS: NOvA Event Selection

*Nature, **560**, 41-48 (2018)*



- Classification of different types of different neutrino interactions

Outlook

- Models and data
- Formal set of the ML problem
- Practical points
- Logistic regression
- Figures of Merit
- Models optimization
- Neural networks basics
- Neural Networks heuristics
- Decision Trees
- Illustrations
- Concluding remarks

Evolution of HEP x ML Engineering

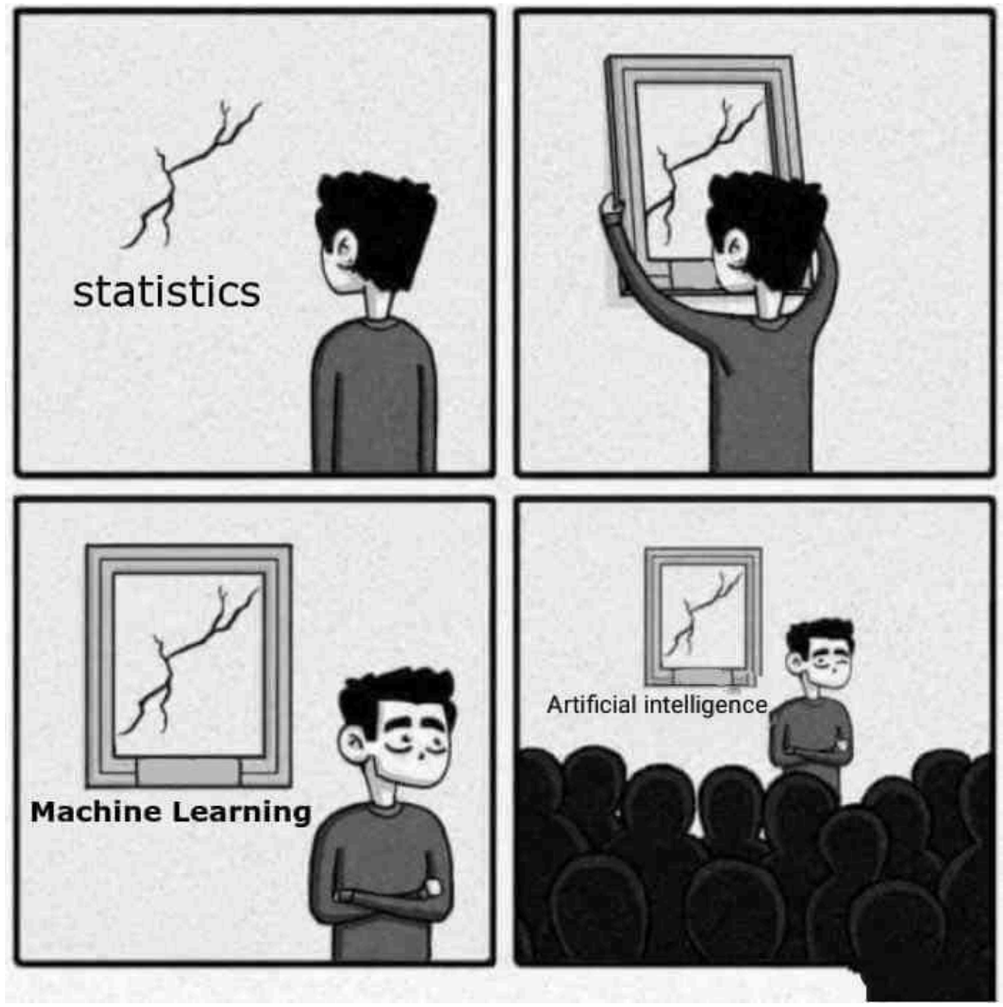
Data Layer		
ROOT Files	ROOT Files	DB / HDFS etc.
Loading Layer		
Ad hoc ROOT ETL logic	Numpy / HDF5 Converters / Loaders	Numpy / HDF5 Converters / Loaders
Training Layer		
TMVA	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
Serving Layer		
Deployment Target (TMVA)	Deployment Target (lwtnn, TensorFlow, TMVA wrappers)	Deployment Target (TensorFlow Serving, SageMaker, etc.)
HEP (Circa 2013)	HEP (Circa 2018)	Industry

11

Luke De Oliveira, 2018

Conclusions

- Machine Learning **is not** just glorified statistics
  - though it is strongly related to it
- Machine Learning is about
  - **representation**
    - art of converting data to the format suitable for processing by algorithms
  - **evaluation**
    - writing a good loss function that effectively converts data into answer of interest
  - **optimization**
    - govern the system to develop in the right direction and converge to the right point



https://towardsdatascience.com/no-machine-learning-is-not-just-glorified-statistics-26d3952234e3