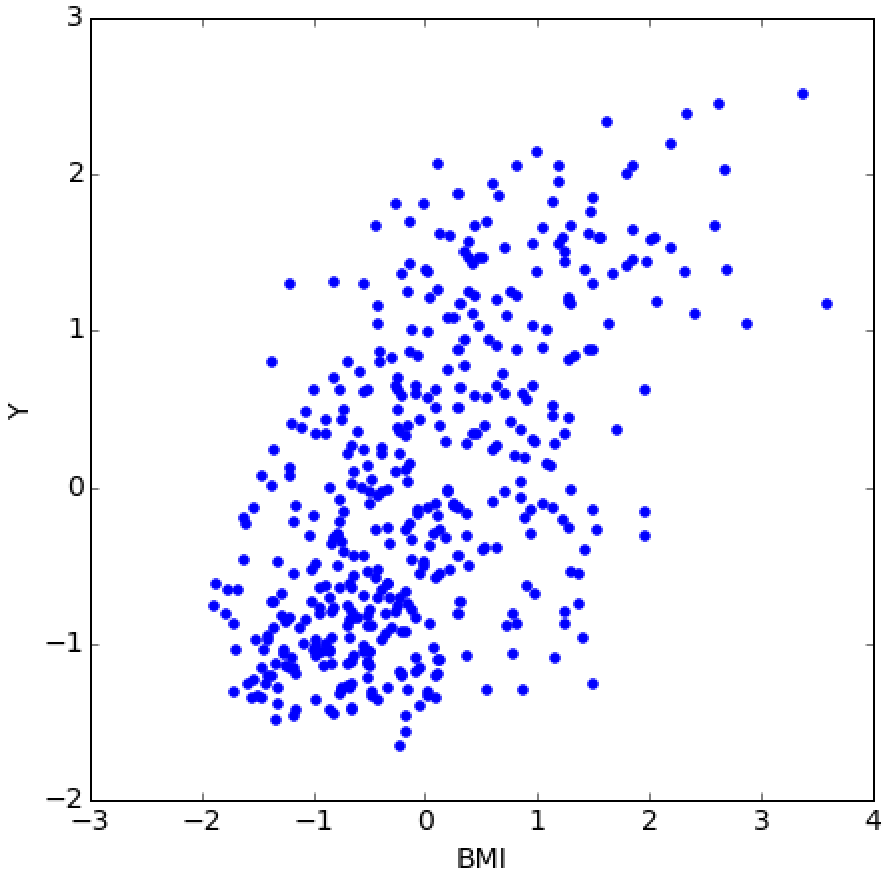
linear regression



We need to make

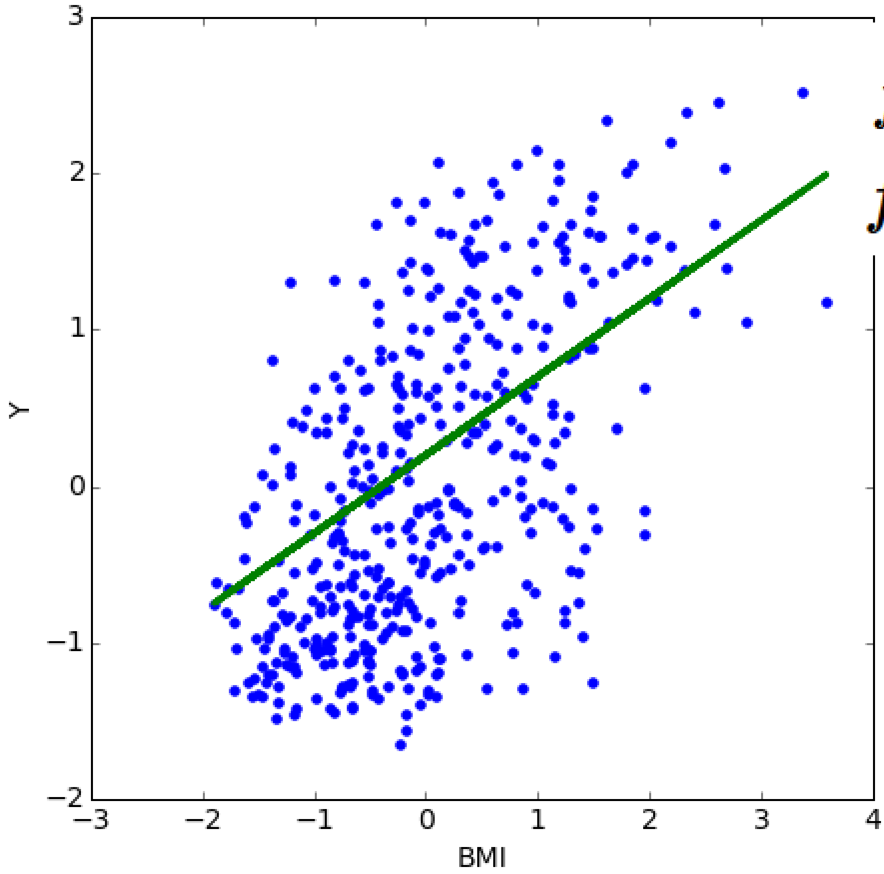
assumptions *linear relationship*

about the

model *linear model*

that generated the data.

linear regression

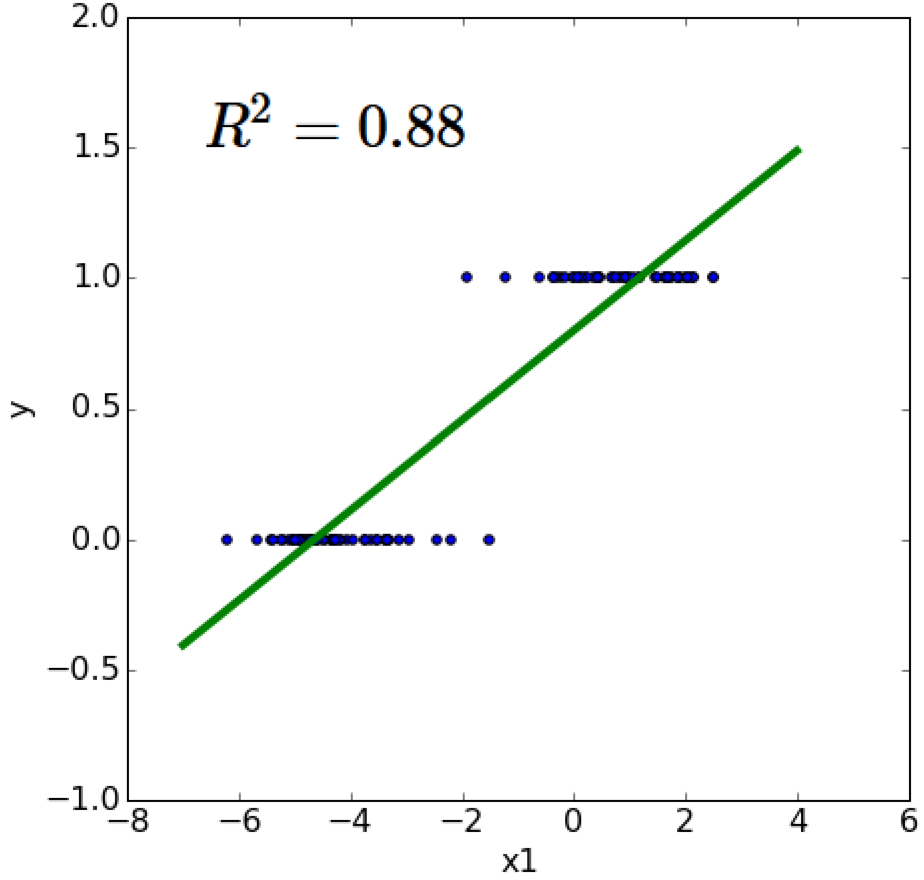


$$f(x) = 0.5 \cdot x + 0.2 = ax + b$$

$$f(x, \theta) = \theta_0 \cdot \theta_1 x$$

- *a* and *b* are model parameters
- *a* is the slope to the line (direction)
- *b* is the intercept or bias (position)

logistic regression



We need to make

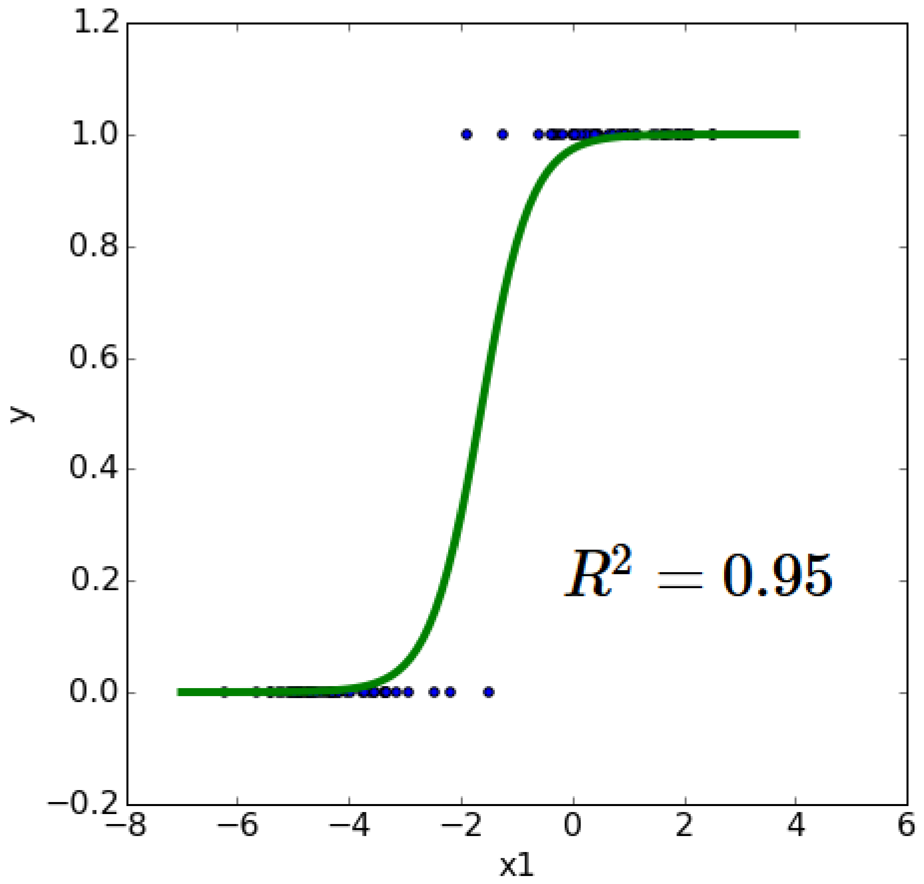
assumptions *linear relationship*

about the

model *linear model*

that generated the data.

logistic regression



We need to make

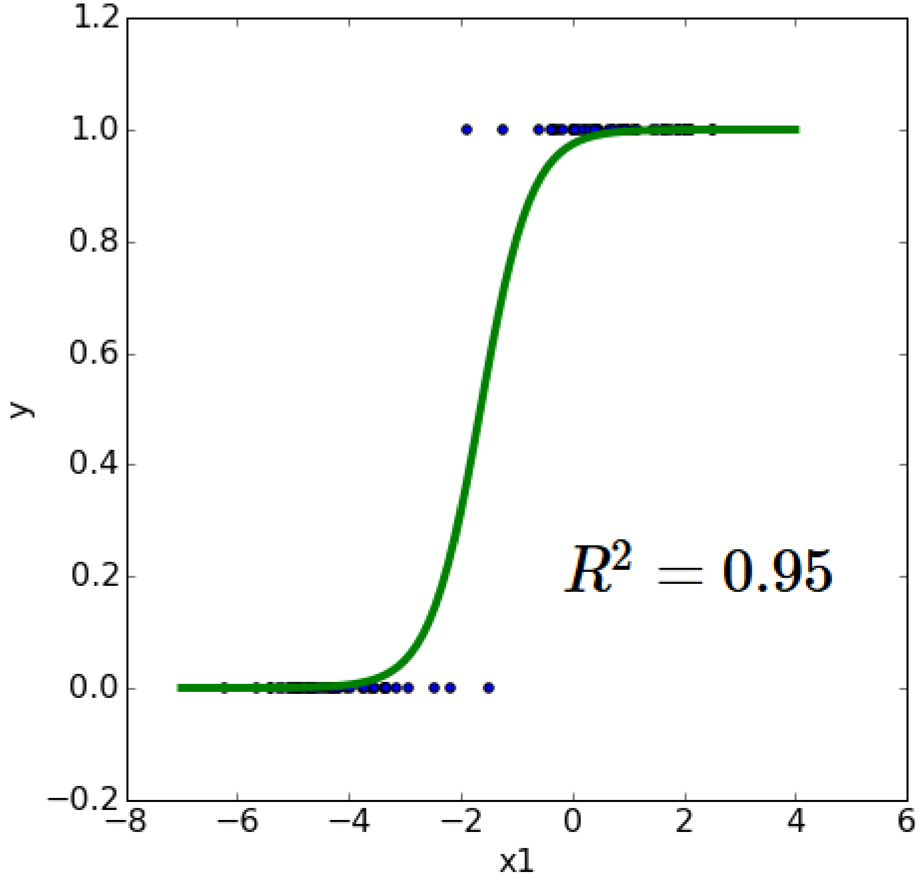
assumptions *linearly separable*

about the

model *logistic model*

that generated the data.

logistic regression: logistic model



$$f(x, \theta) = g(\theta_0 + \theta_1 x_1)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

logistic regression: cost function

$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)},\theta)) + (1 - y^{(i)}) log(1 - f(x^{(i)},\theta))]$$

We know that $y^{(i)}$ is either 0 or 1. If $y^{(i)} = 1$ then the cost function $J(\theta)$ is incremented by

$-log(f(x^{(i)},\theta))$.

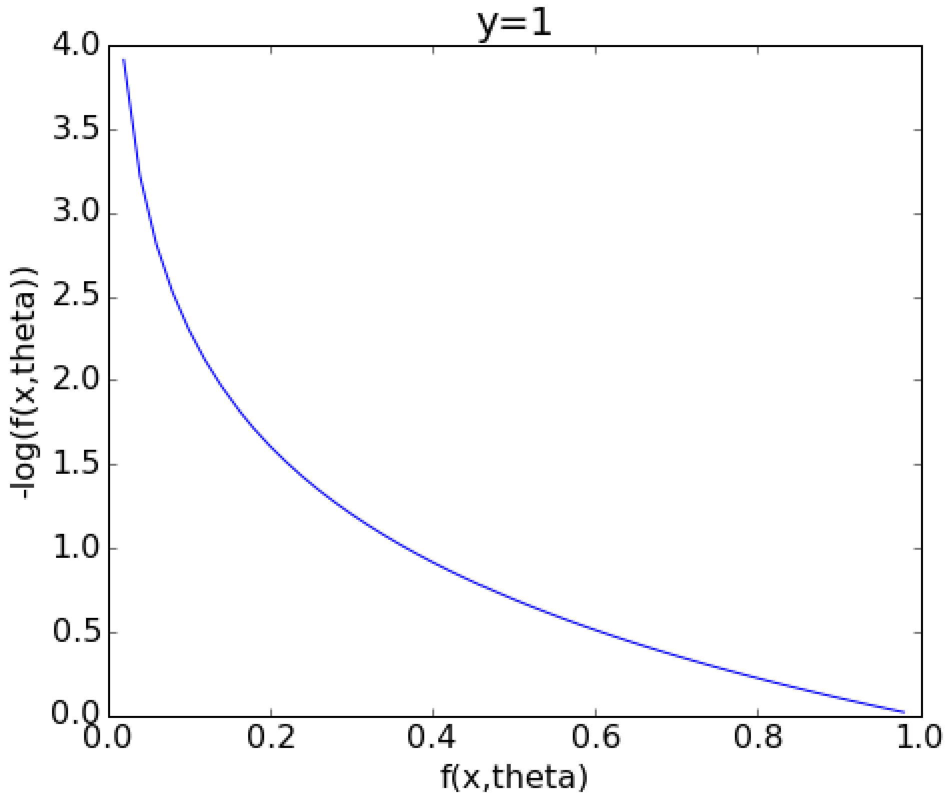
Similarly, if $y^{(i)} = 0$ then the cost function $J(\theta)$ is incremented by

$-log(1 - f(x^{(i)},\theta))$.

logistic regression: cost function

We know that $y^{(i)}$ is either 0 or 1. If $y^{(i)} = 1$ then the cost function $J(\theta)$ is incremented by

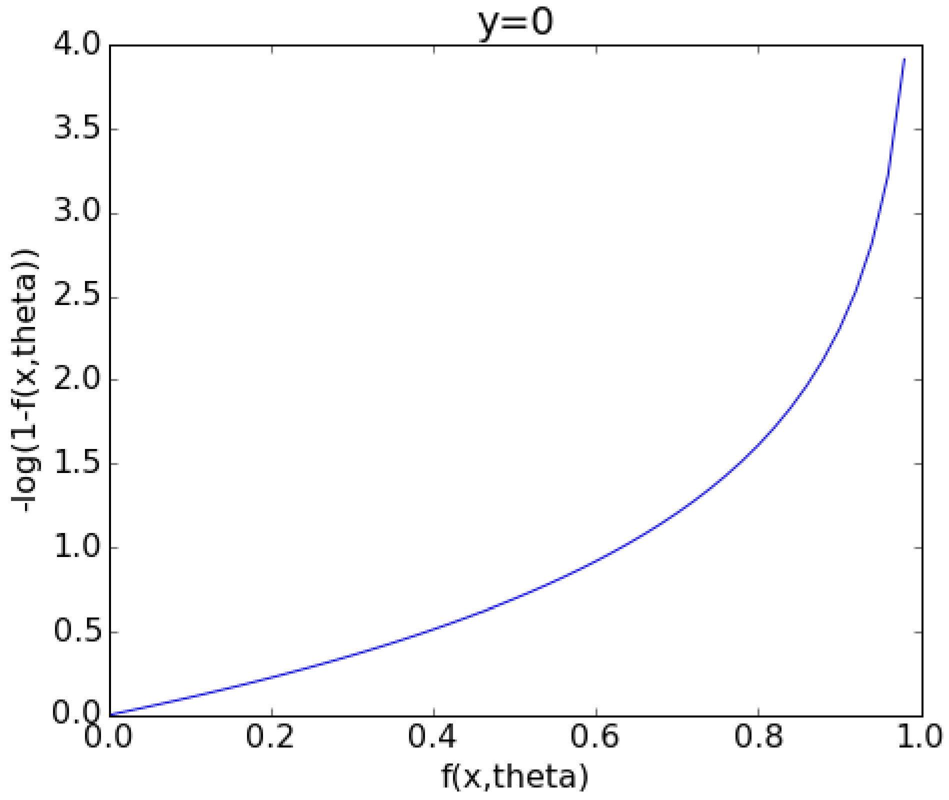
$-log(f(x^{(i)}, \theta))$.



logistic regression: cost function

Similarly, if $y^{(i)} = 0$ then the cost function $J(\theta)$ is incremented by

$$-log(1 - f(x^{(i)}, \theta)).$$



logistic regression

Fit a logistic model

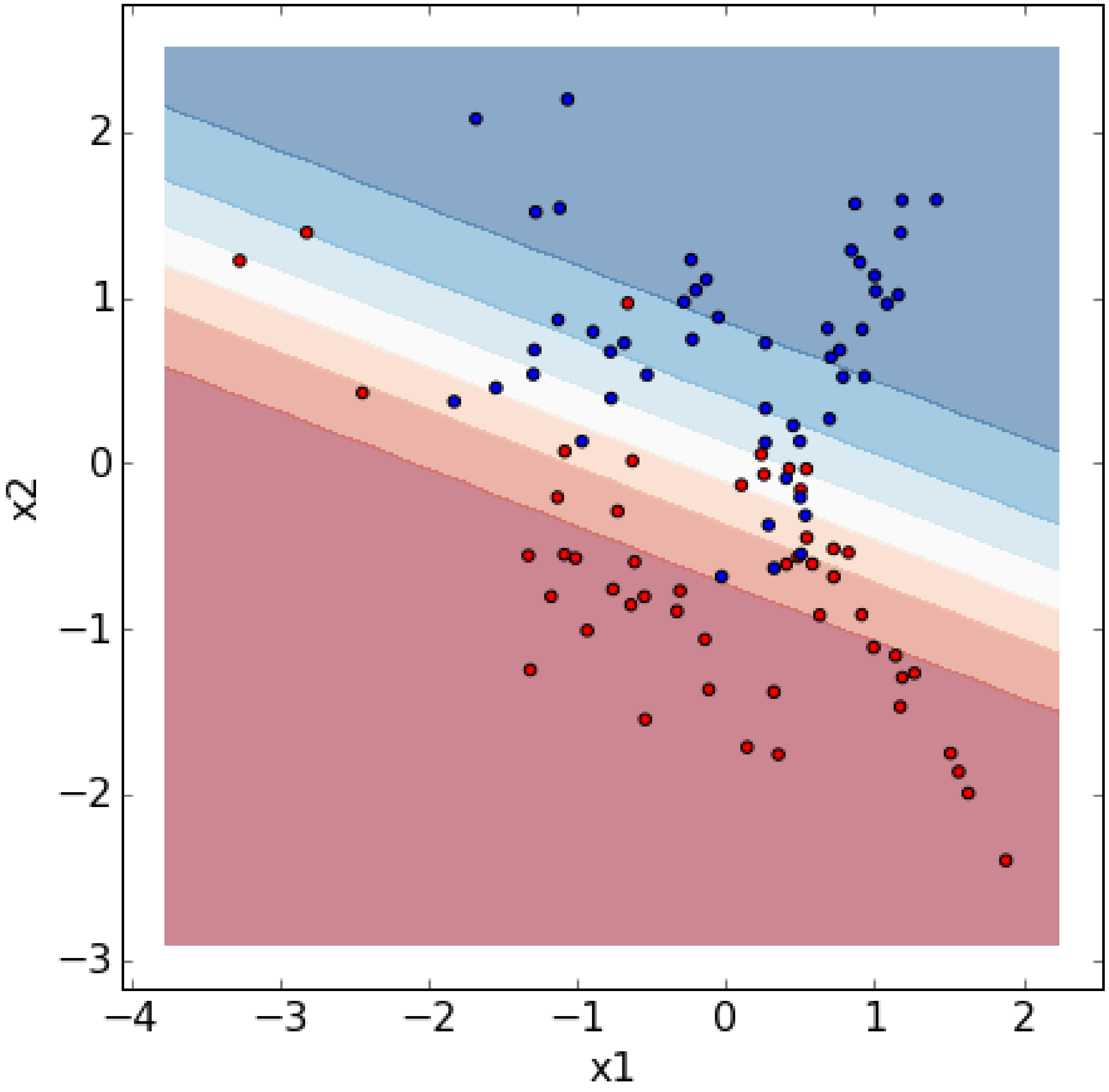
$$f(x,\theta) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m) = g(\theta' x)$$

to the data set such that the cost function

$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)},\theta)) + (1-y^{(i)}) log(1-f(x^{(i)},\theta))]$$

is minimal using gradient descent

$$\theta_j := \theta_j - \alpha \frac{1}{n}\sum_{i=1}^{n}(f(x^{(i)},\theta) - y^{(i)}) x_j^{(i)}$$

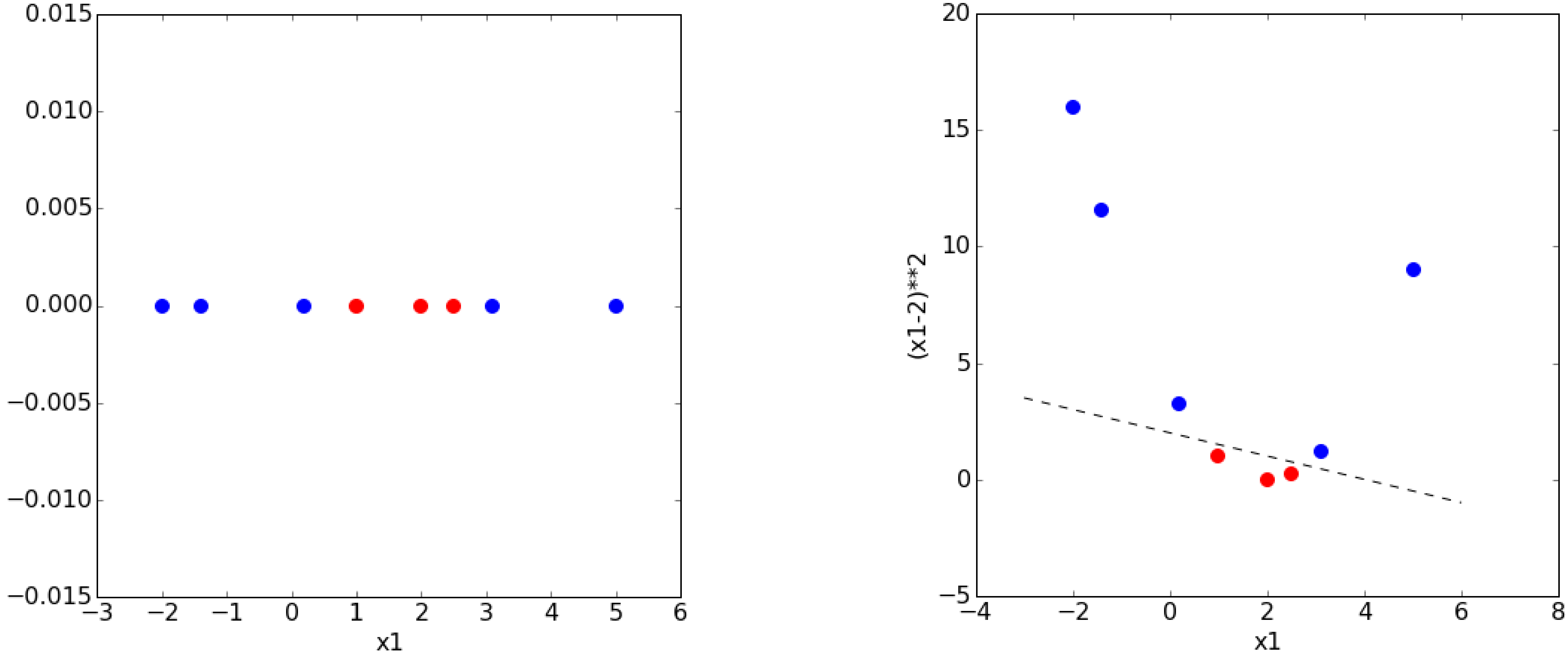


x2
x1
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2

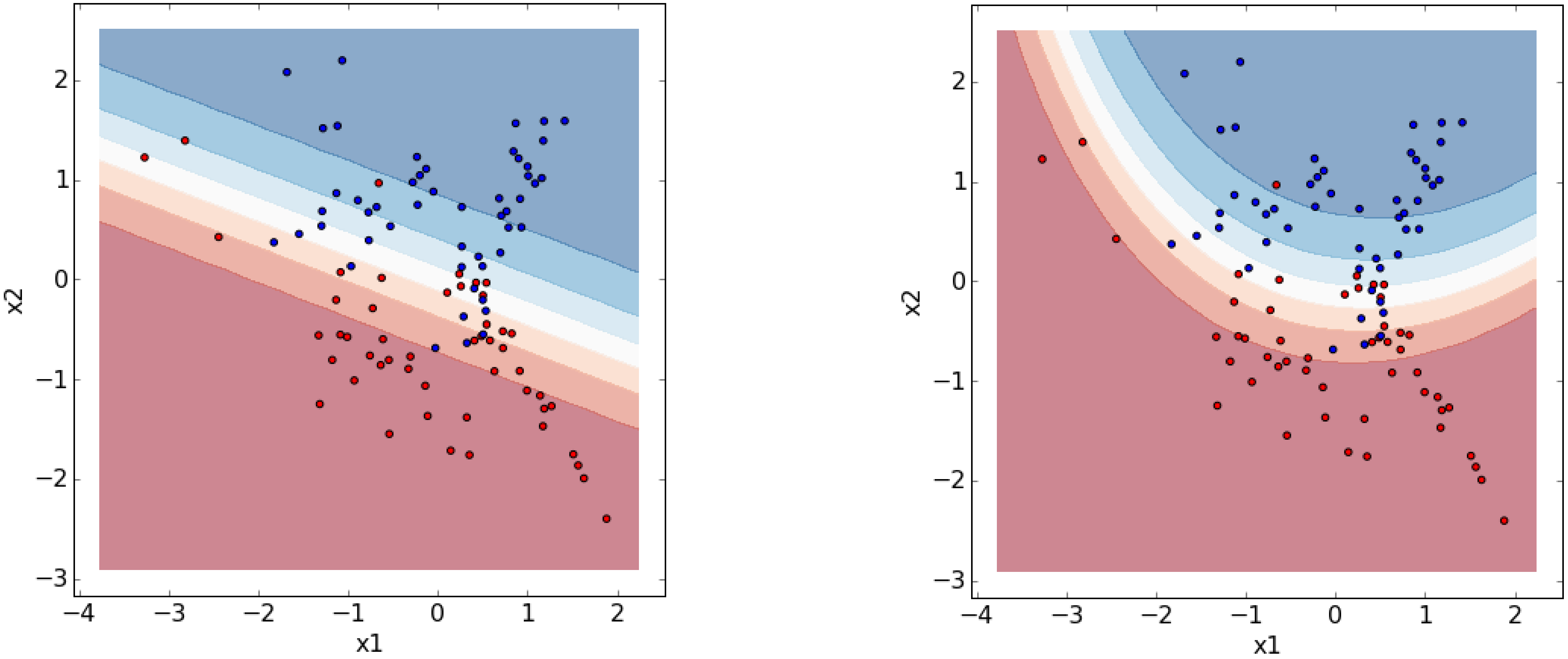
Let’s get real!

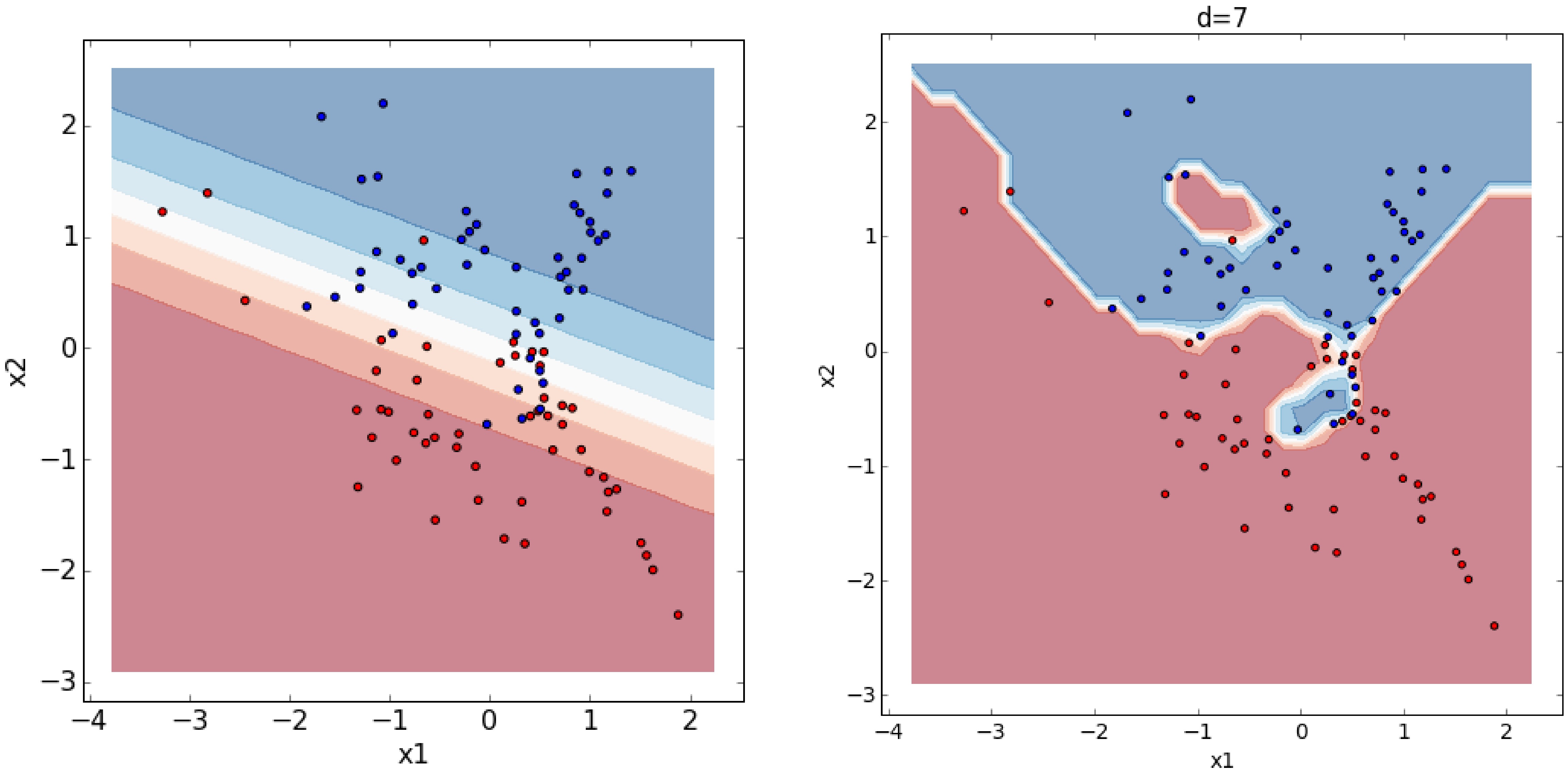
7. **Logistic regression**
8. **Non-linear transformations**

Polynomial transformation



non-linear logistic regression





d=7
x1
x2

regularized logistic regression

$$f(x,\theta) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m)$$

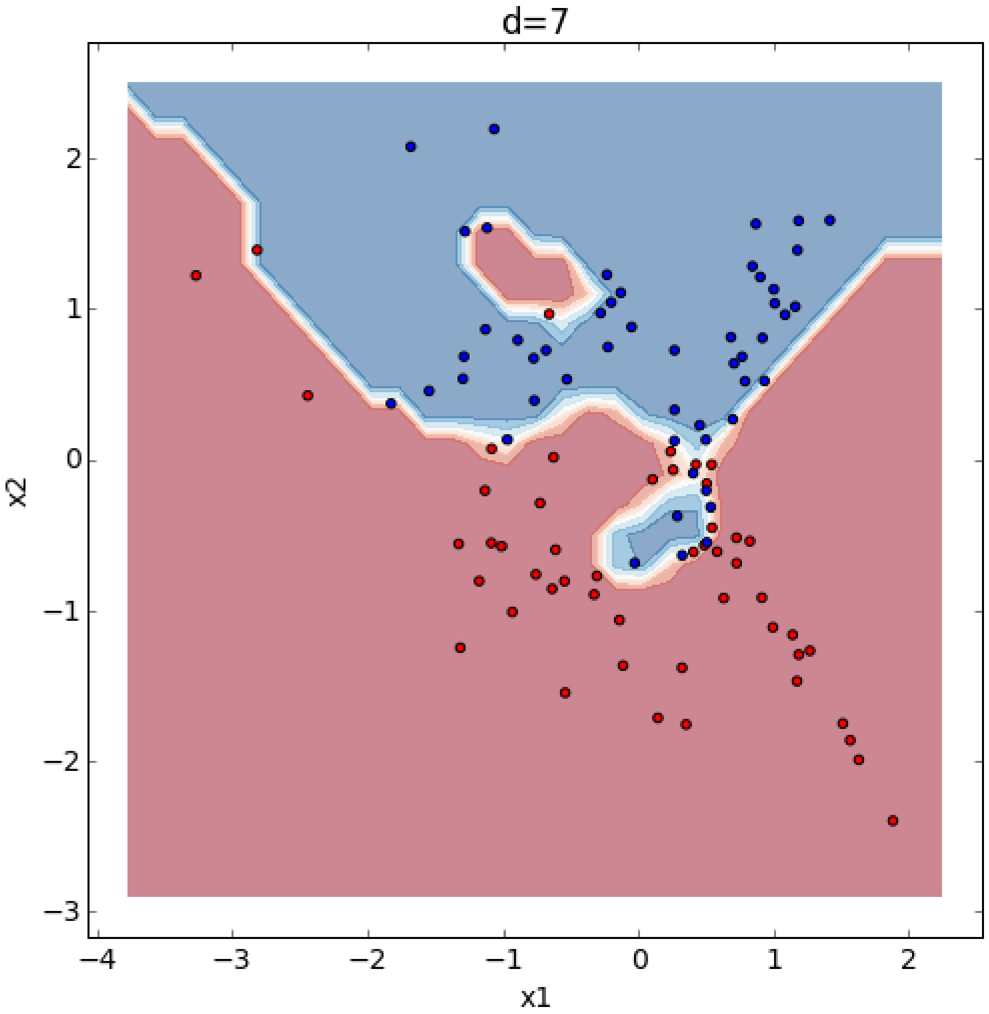
cost function

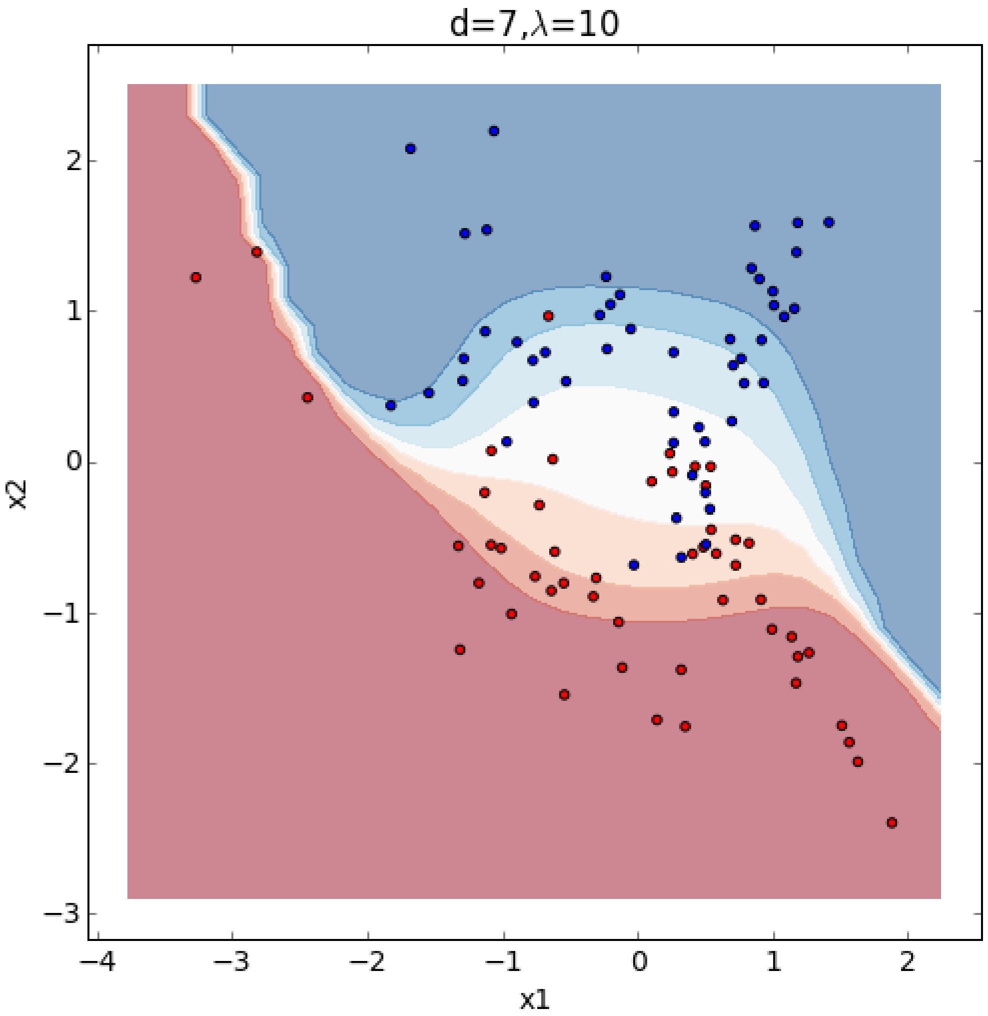
$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)},\theta)) + (1-y^{(i)}) log(1-f(x^{(i)},\theta))]$$

$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)},\theta)) + (1-y^{(i)}) log(1-f(x^{(i)},\theta))] + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

regularized cost function

regularized logistic regression





support vector machines

Fit a linear model

$$f(x,\theta) = \theta' x$$

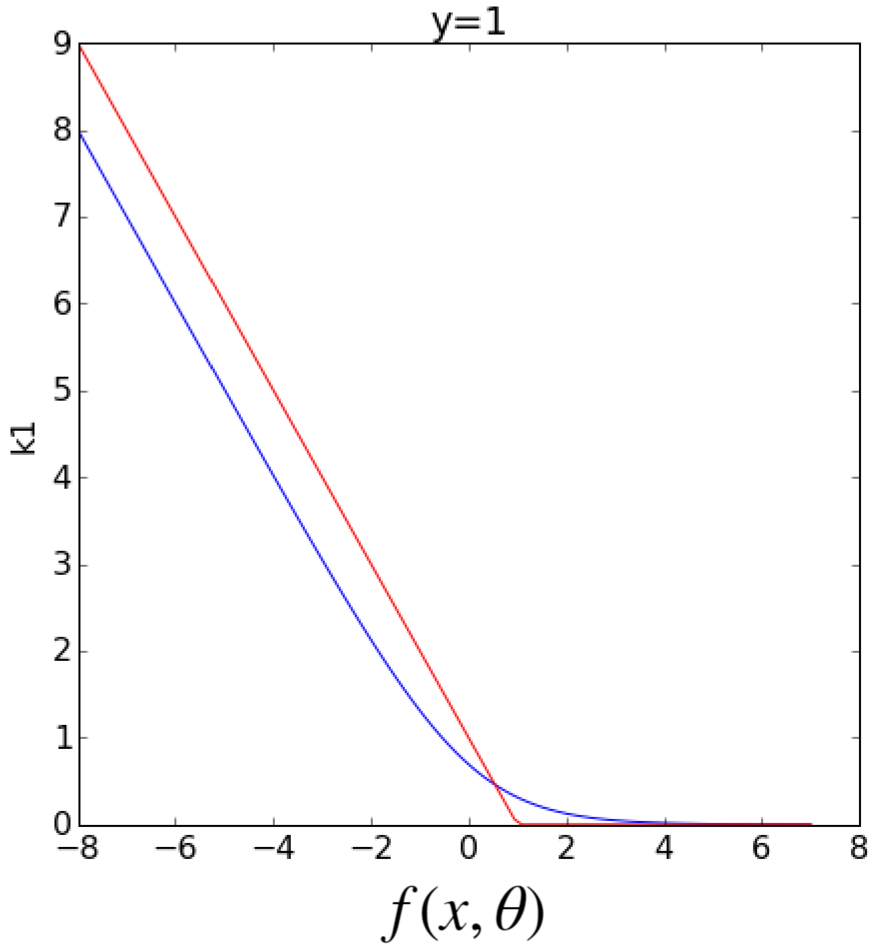
such that

$$J(\theta) = \left[C \sum_{i=1}^{n} y^{(i)} k_1(\theta' x^{(i)}) + (1 - y^{(i)}) k_0(\theta' x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

with $k_1(\theta' x) = max(0, 1 - \theta' x)$ and $k_0(\theta' x) = max(0, 1 + \theta' x)$

is minimized.

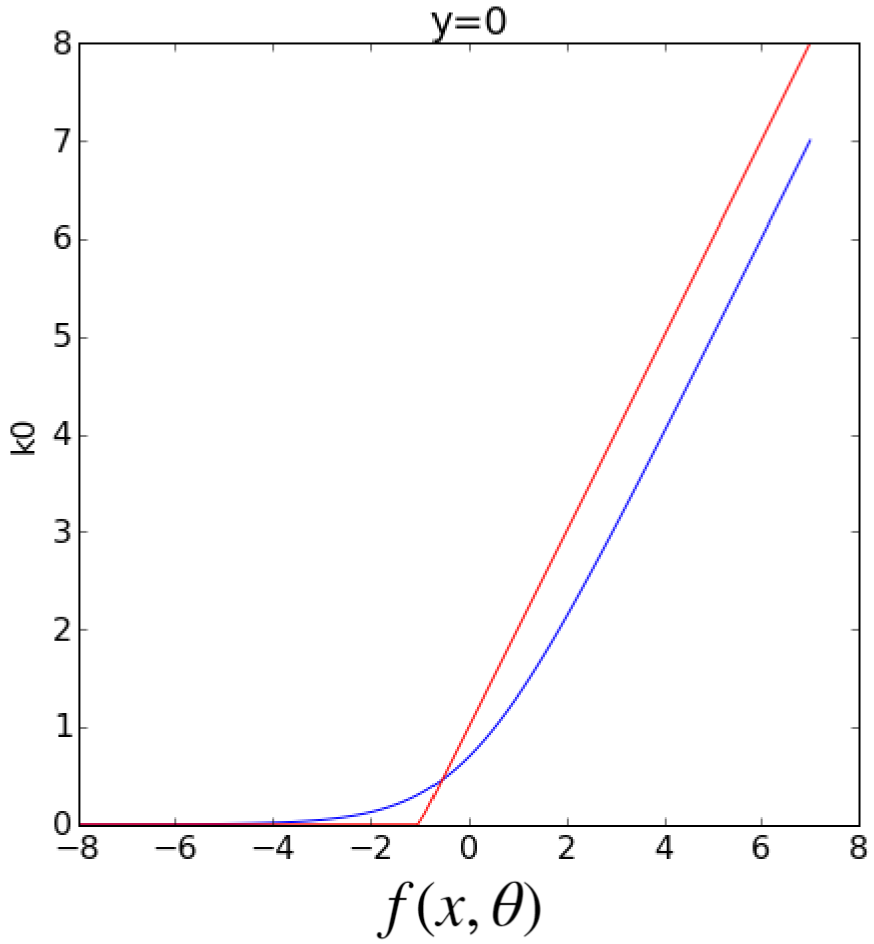
support vector machines



- replace cost function by piecewise linear function
- if $y = 1$ then the contribution to the cost is

$$k_1(f(x,\theta)) = max(0, 1 - f(x,\theta))$$

support vector machines



- replace cost function by piecewise linear function

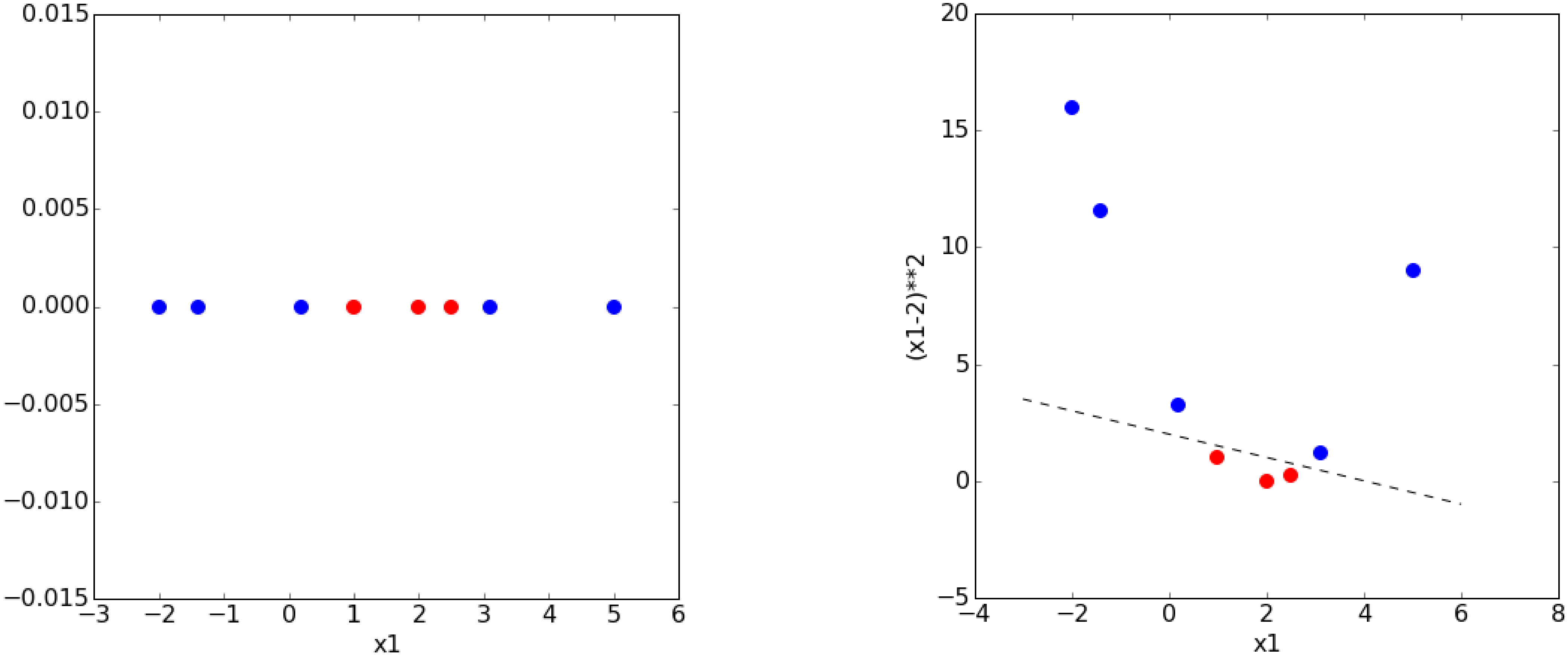
- if y = 1 then the contribution to the cost is

$$k_1(f(x,\theta)) = max(0, 1 - f(x,\theta))$$

- if y = 0 then the contribution to the cost is

$$k_0(f(x,\theta)) = max(0, 1 + f(x,\theta))$$

kernel support vector machines



kernel support vector machines

SVMs can also be formulated as a linear function of the samples (dual form) instead of the features as

$$f(x, \theta) = \sum_{i=1}^{n} \theta_i (x \cdot x^{(i)}) + \theta_0$$

that can be reformulated as a non-linear function using what is know as a kernel function

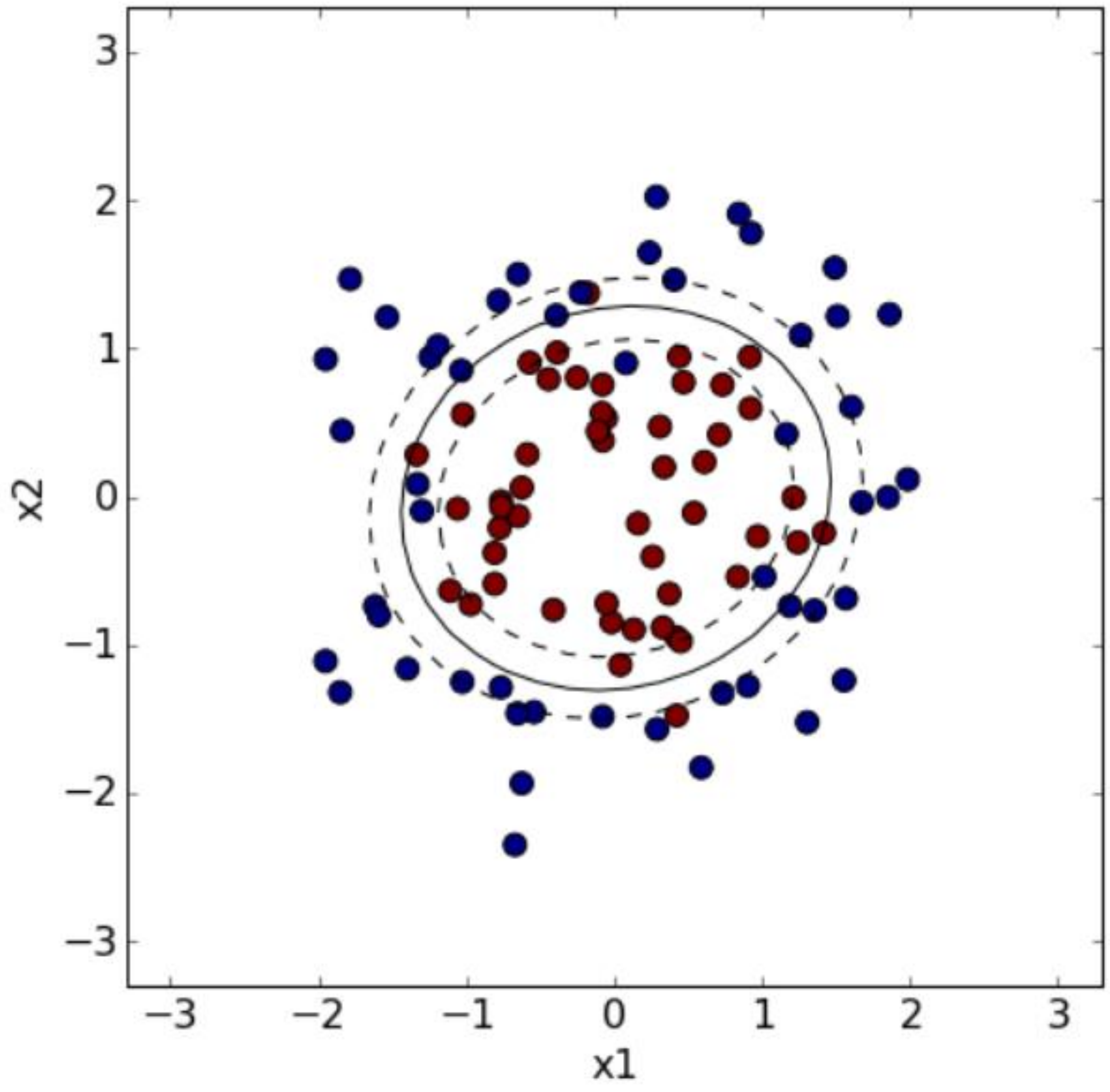
$$K(x^{(i)}, x^{(j)})$$

to become

$$f(x, \theta) = \sum_{i=1}^{n} \theta_i K(x, x^{(i)}) + \theta_0$$

The data points $x^{(i)}$ for which $\theta_i > 0$ are called the support vectors.

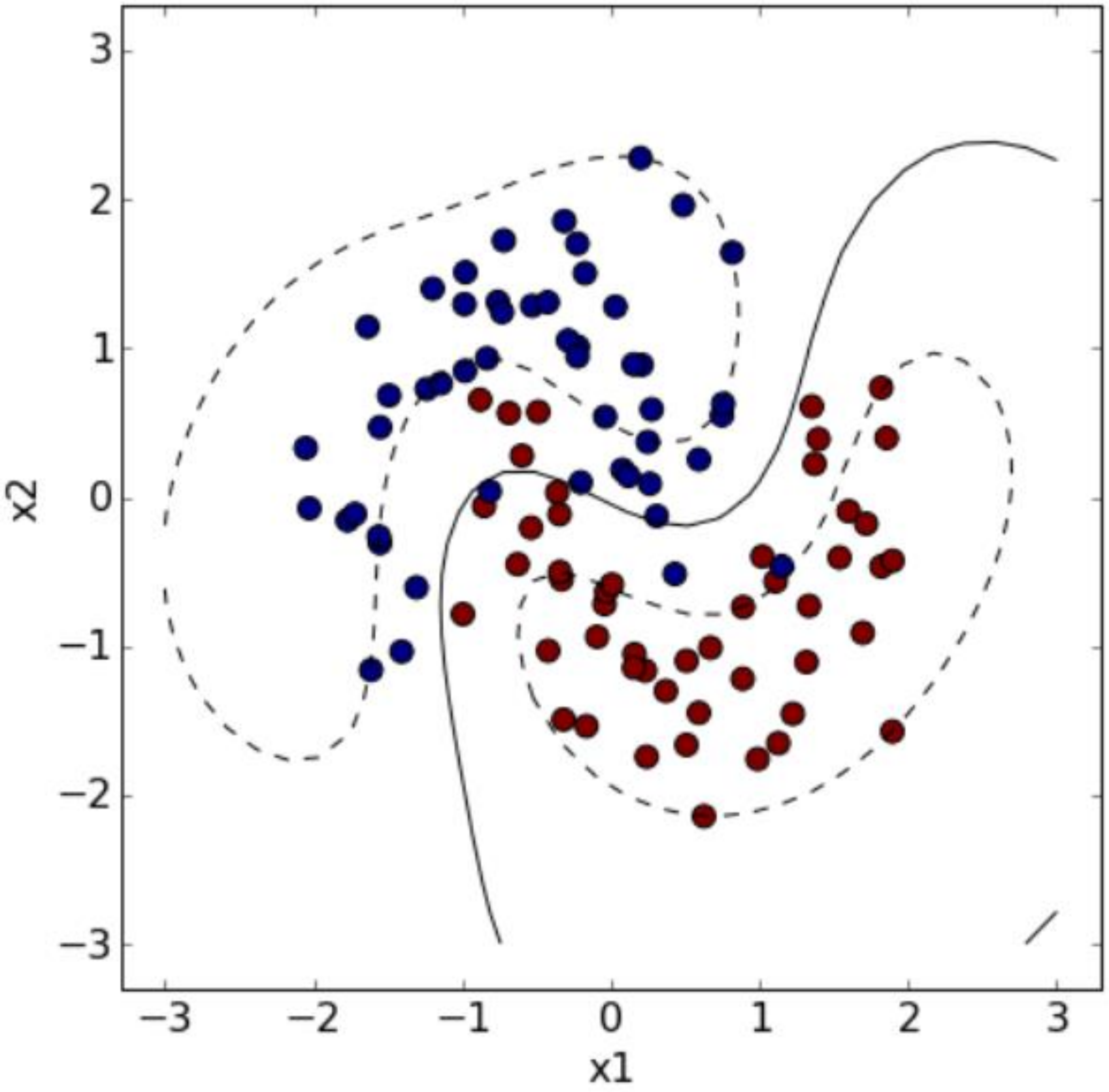
kernel support vector machines



$$f(x,\theta) = \sum_{i=1}^{n} \theta_i K(x, x^{(i)}) + \theta_0$$

$$K(x^{(i)}, x^{(j)}) = (x^{(i)} \cdot x^{(j)} + c)^d$$

kernel support vector machines



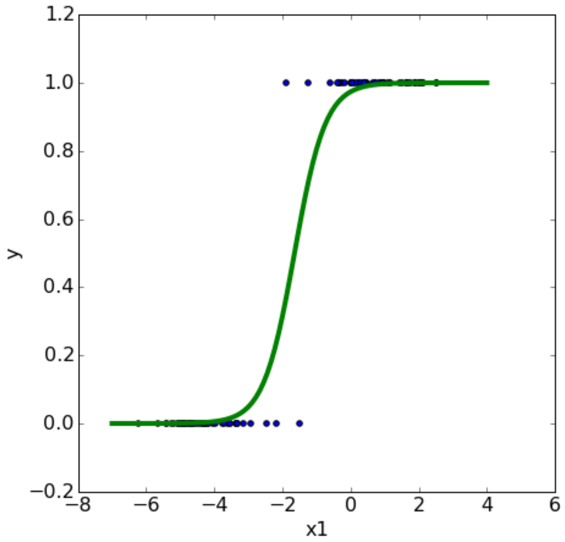
$$f(x,\theta) = \sum_{i=1}^{n} \theta_i K(x, x^{(i)}) + \theta_0$$

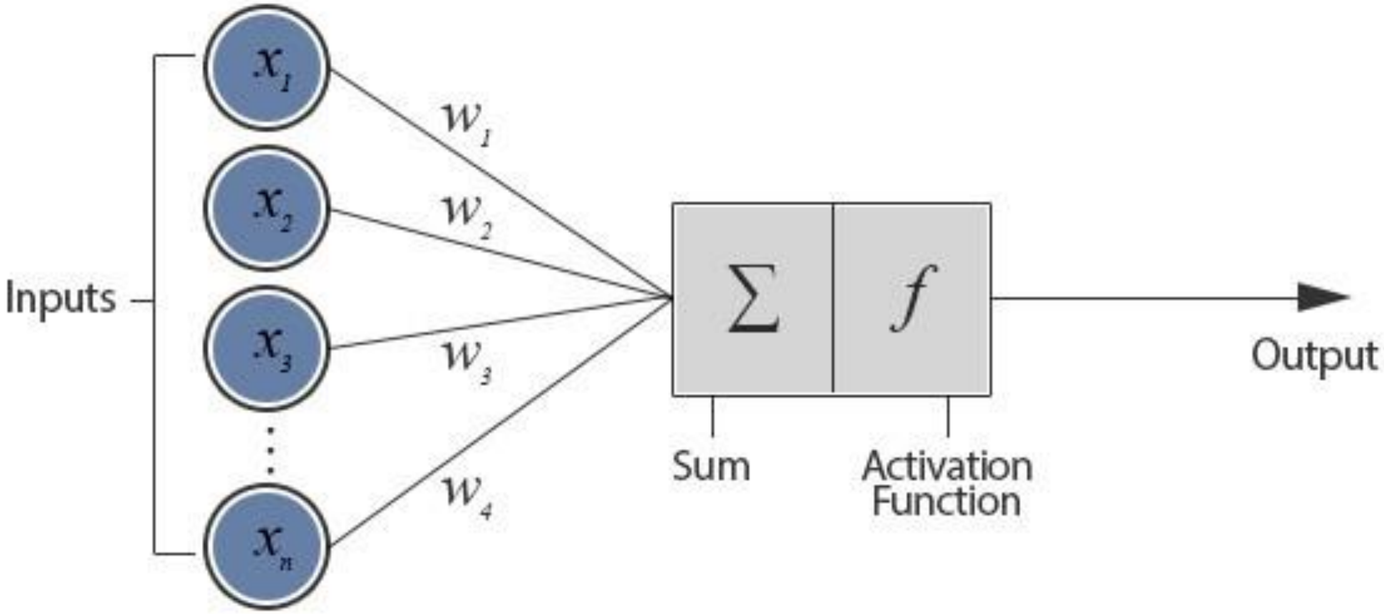
$$K(x^{(i)}, x^{(j)}) = \exp\left[-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right]$$

Let’s get real!

9. Support Vector Machines

$$f(x,\theta) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m) = g(\theta' x)$$



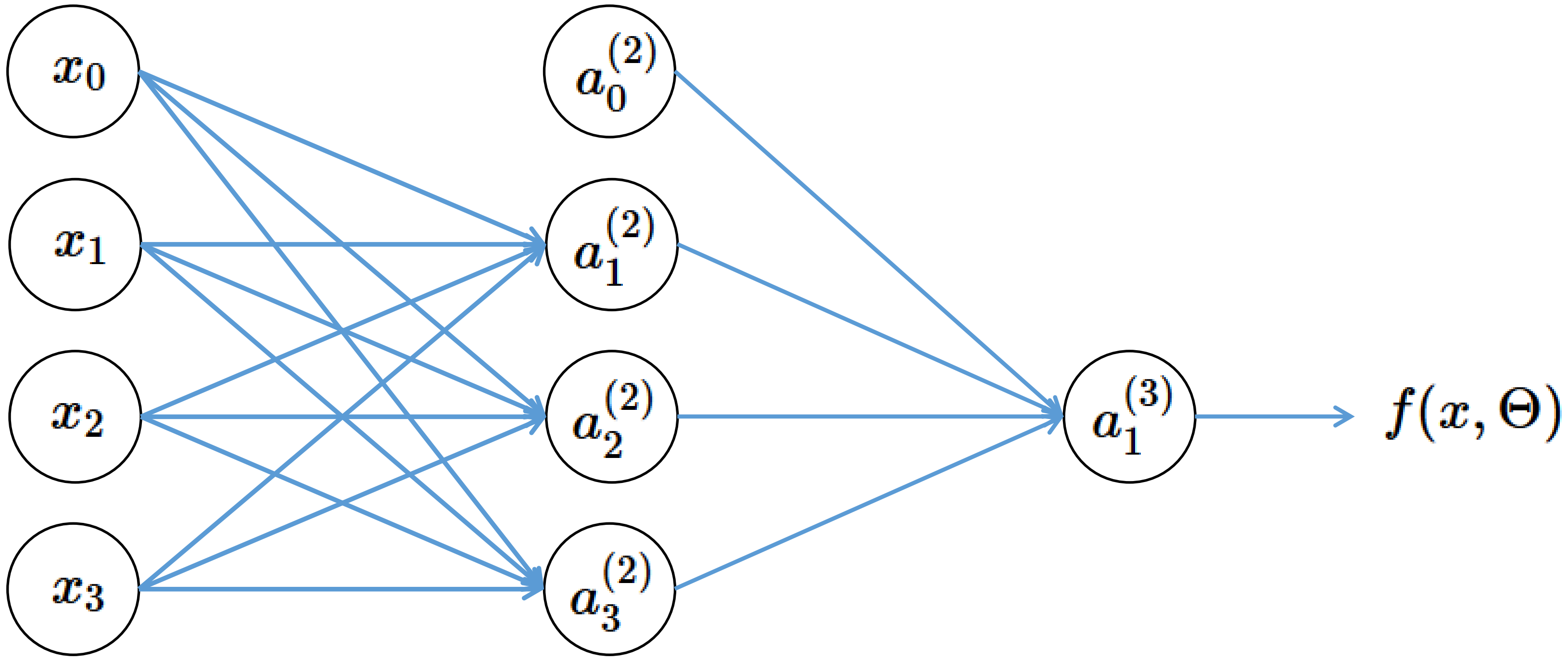


Model: $$f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



Cost function logistic regression:

$$J(\theta) = -[\frac{1}{m}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)},\theta) + (1-y^{(i)})log(1-f(x^{(i)},\theta))] + \frac{\lambda}{2n}\sum_{j=1}^{n}\theta^2$$

Cost function feedforward neural network:

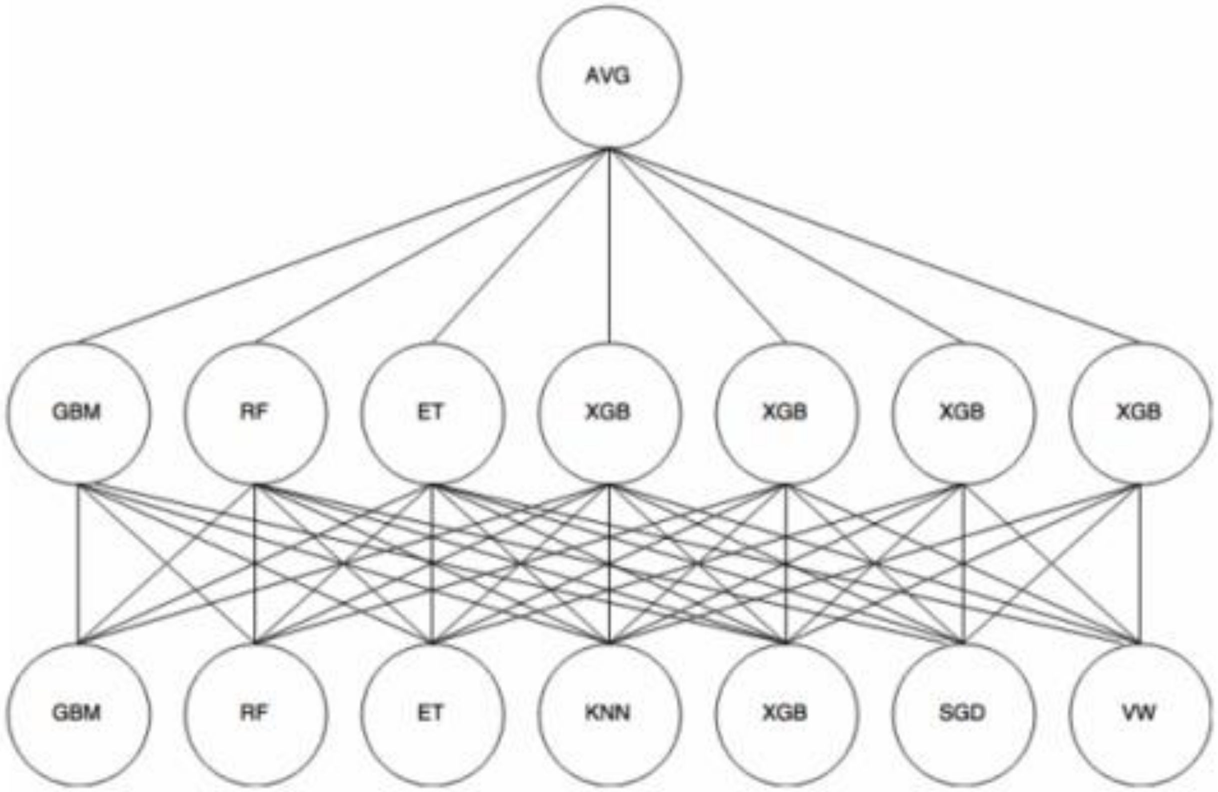
$$J(\theta) = -[\frac{1}{m}\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)} log(f(x^{(i)},\Theta)_k) + (1-y_k^{(i)})log(1-f(x^{(i)},\Theta)_k)] + \frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

https://playground.tensorflow.org/

Let’s get real!

10. Neural Networks

Blending/Stacking



https://mlwave.com/kaggle-ensembling-guide/

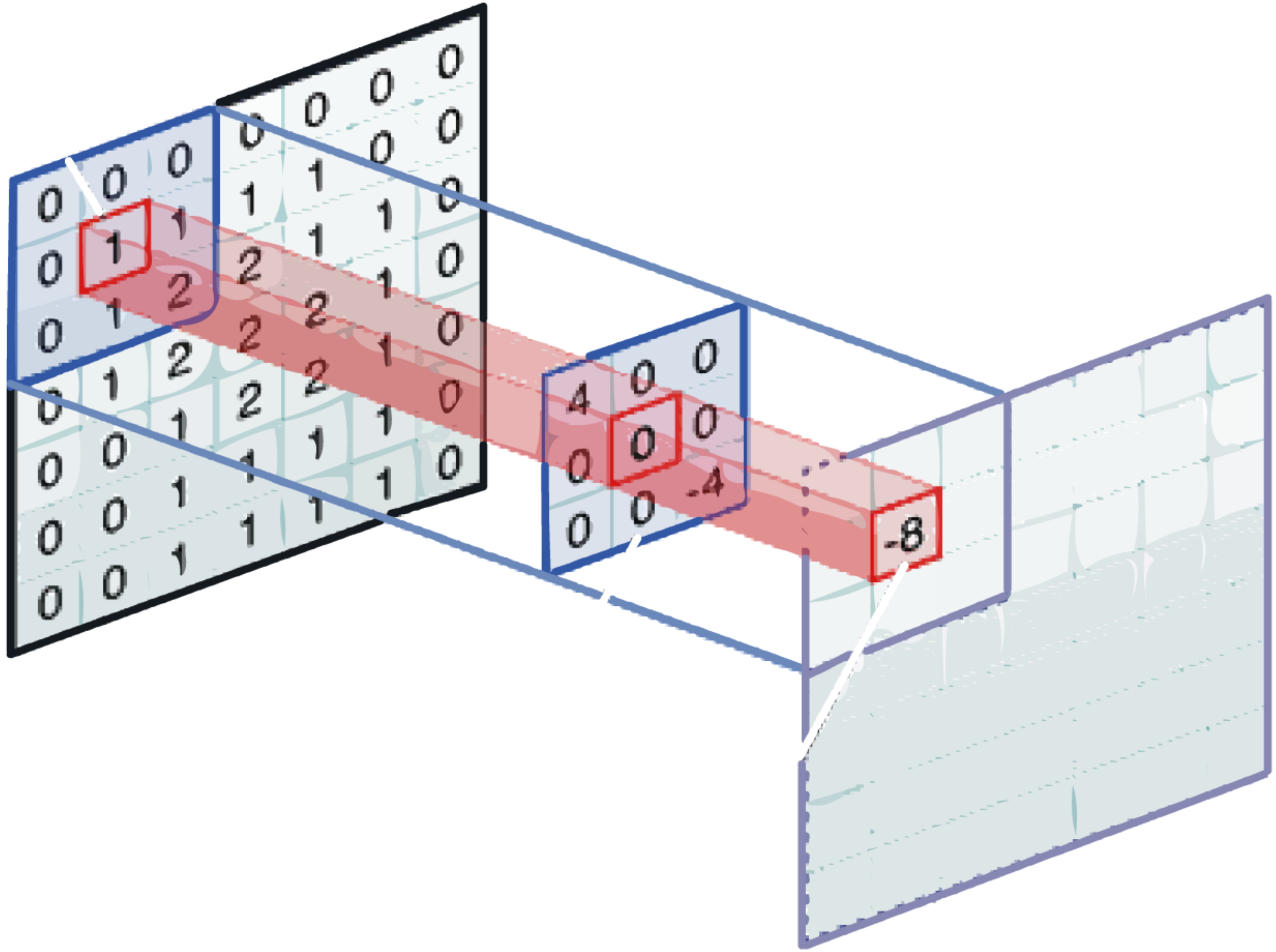
Convolutional Neural Networks

What We See

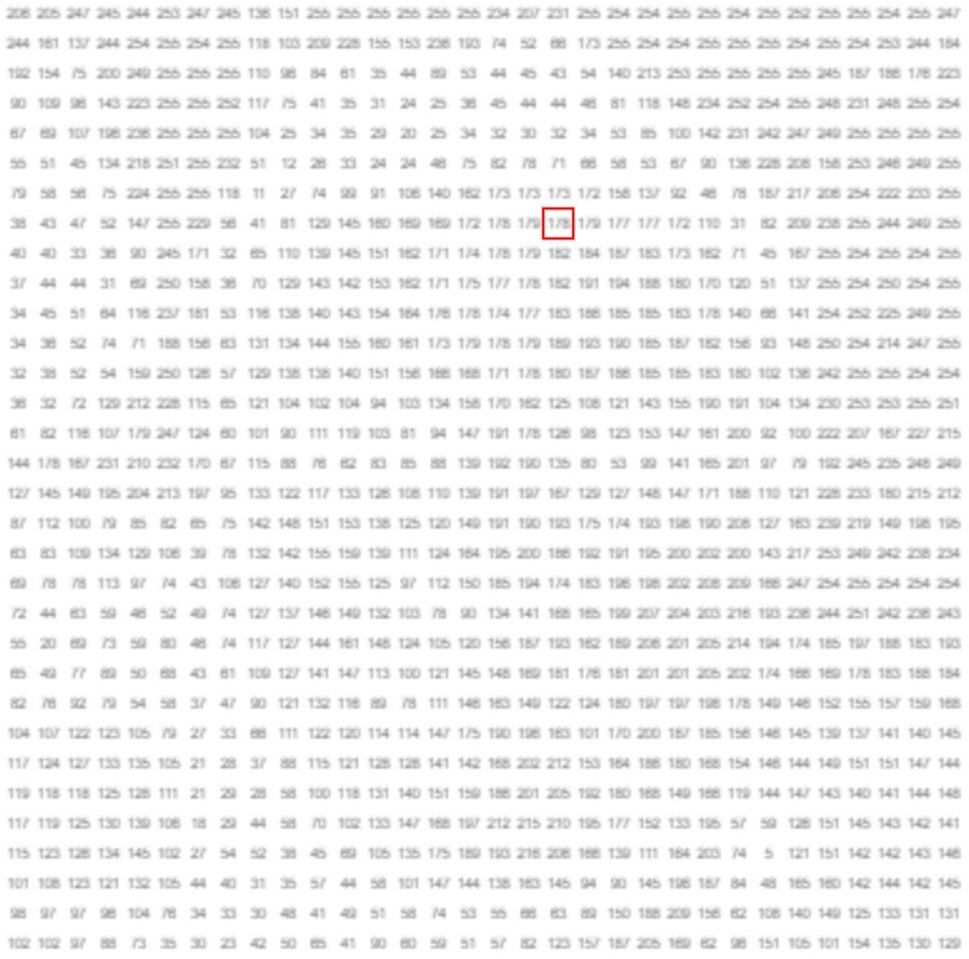
08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

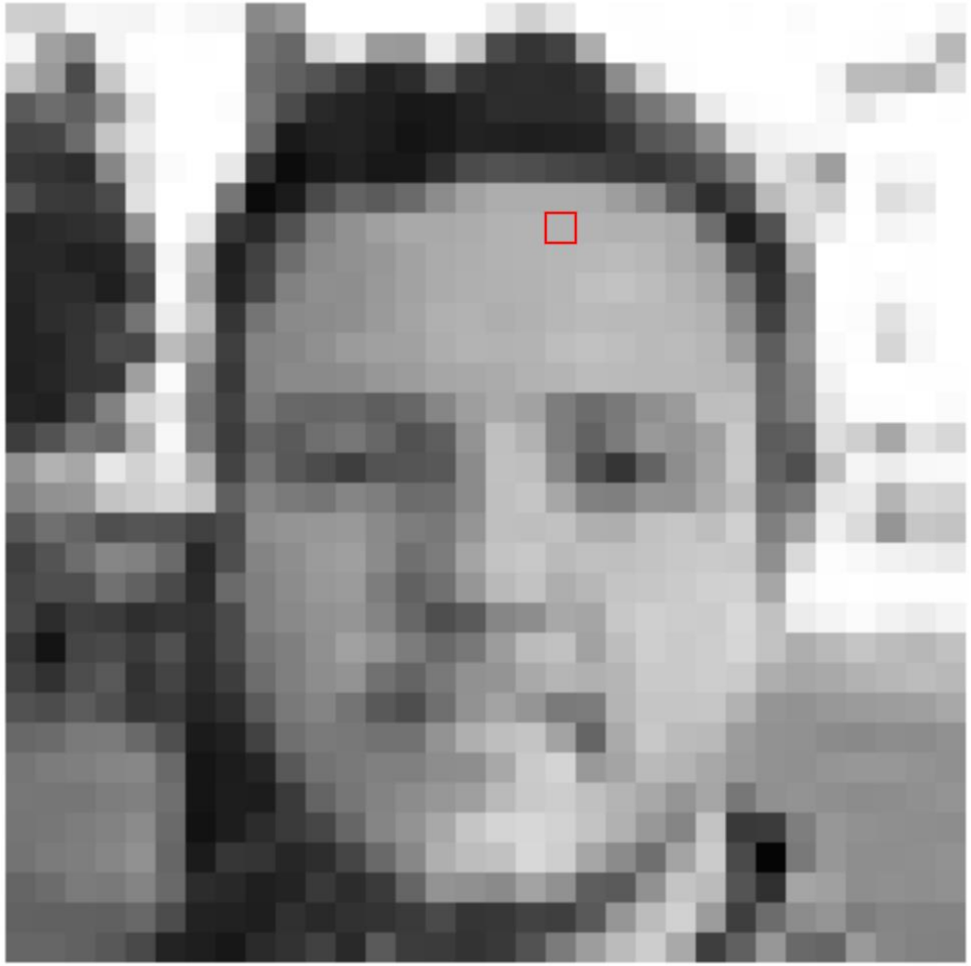
What Computers See

https://github.com/leriomaggio



https://github.com/leriomaggio



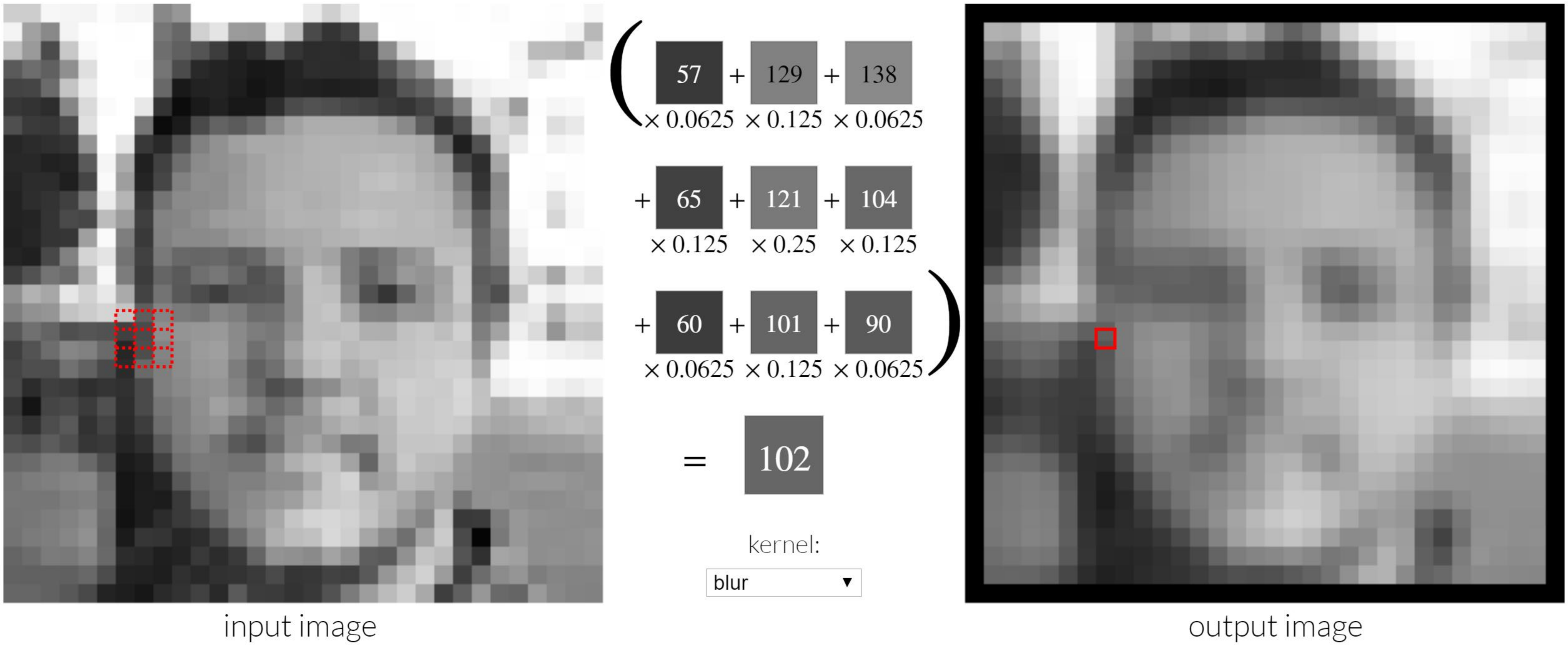


http://setosa.io/ev/image-kernels/

blur

$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$

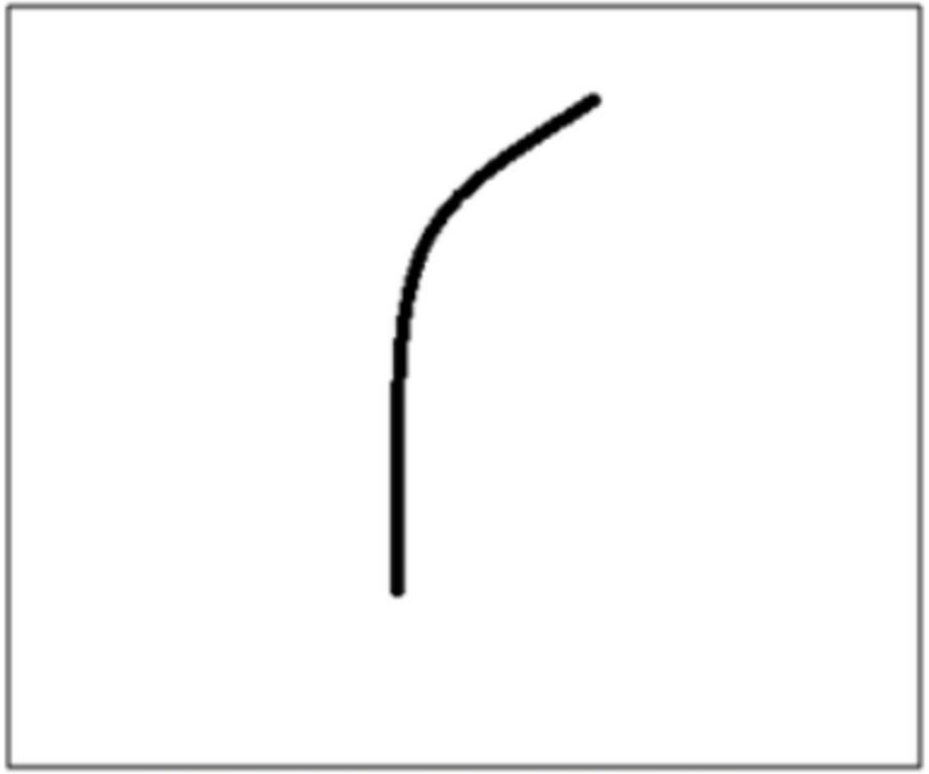
http://setosa.io/ev/image-kernels/



http://setosa.io/ev/image-kernels/

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

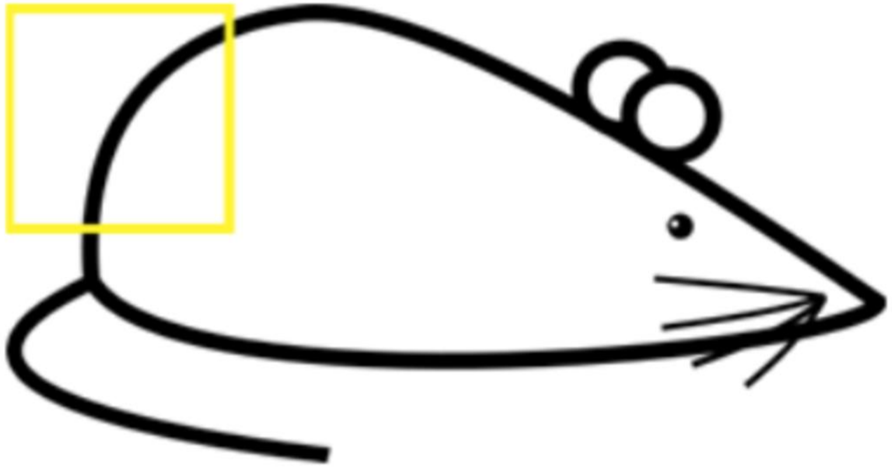
Pixel representation of filter



Visualization of a curve detector filter

https://github.com/leriomaggio

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

https://github.com/leriomaggio

Visualization of the filter on the image

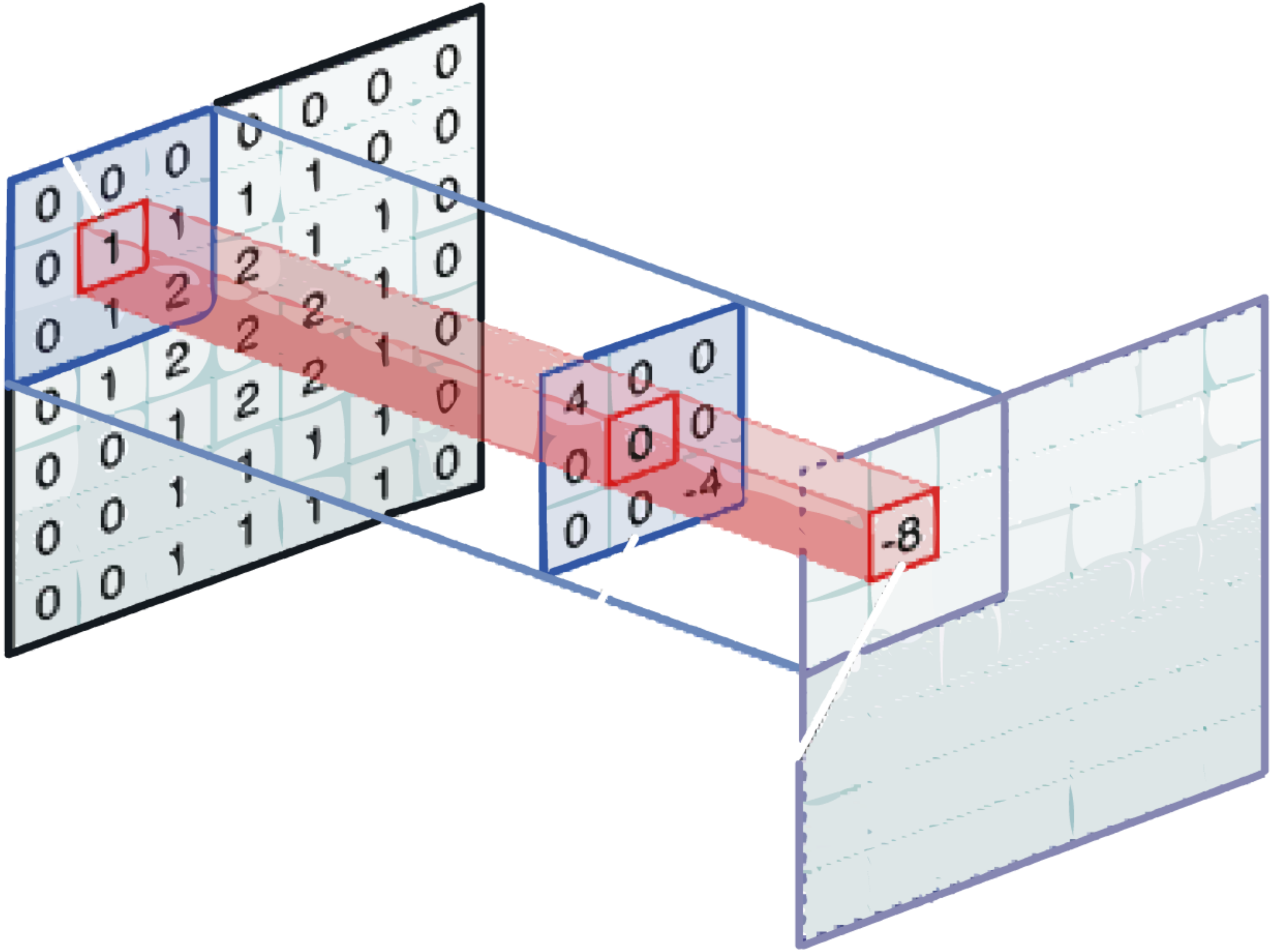
0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

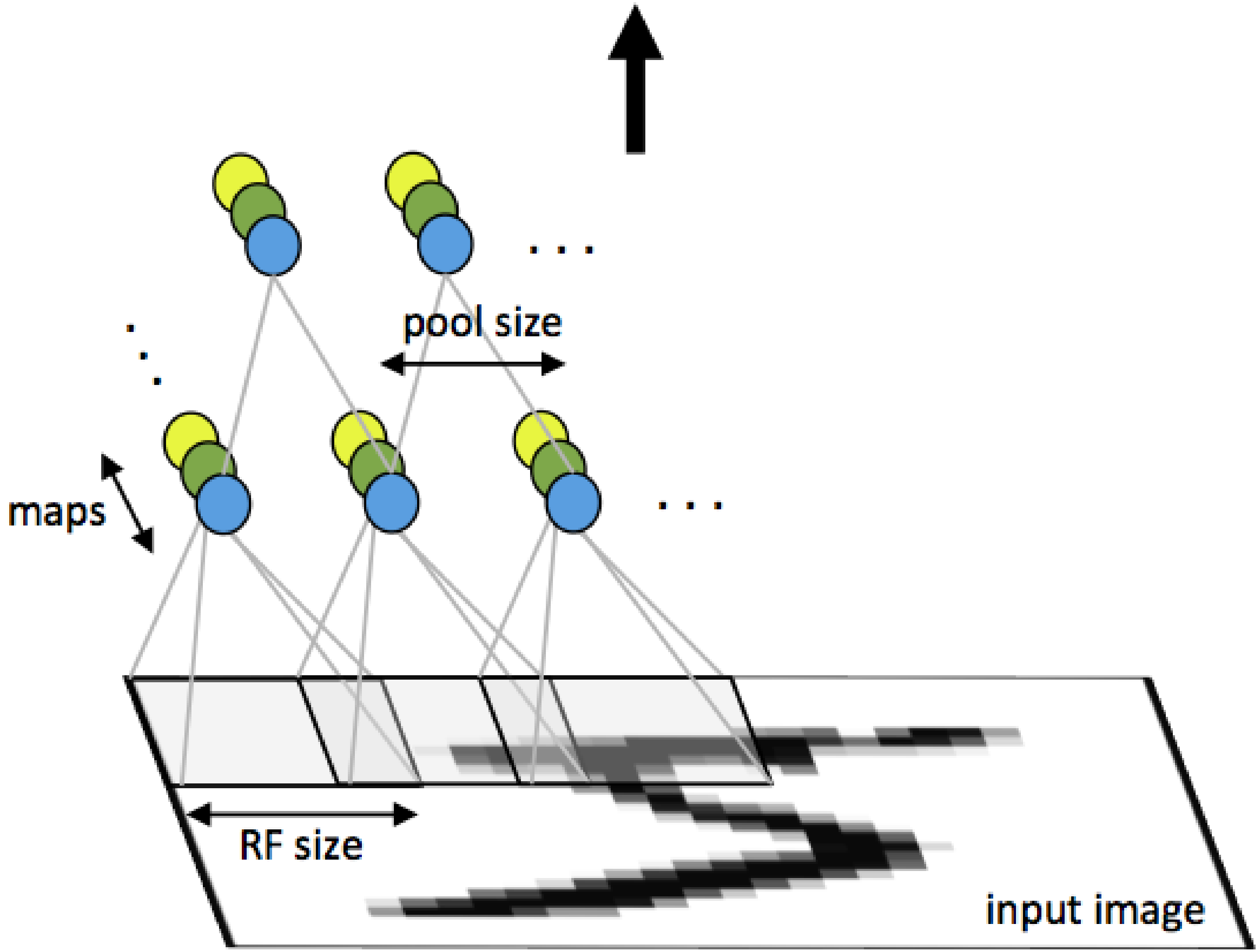
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

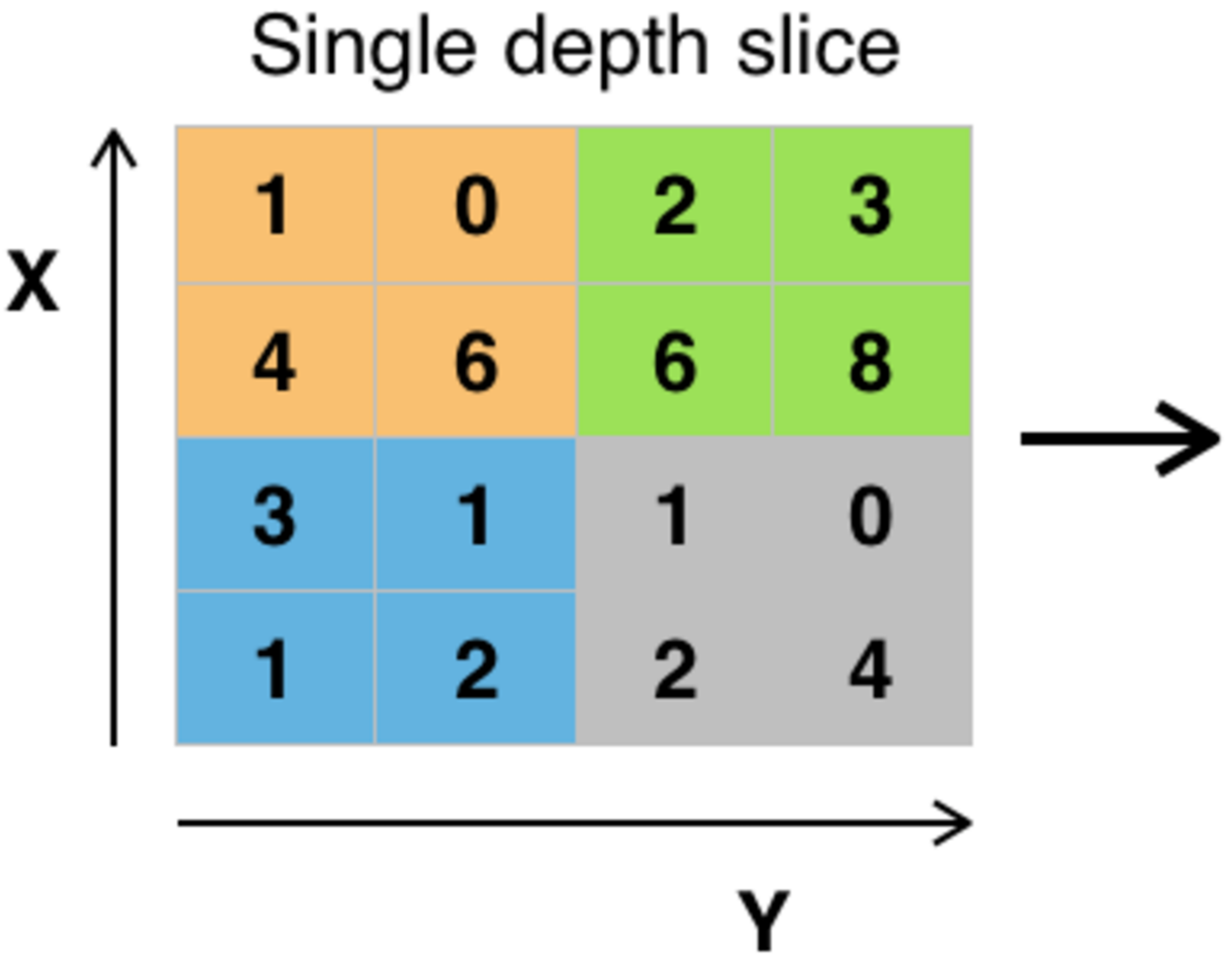
Pixel representation of filter



https://github.com/leriomaggio



https://github.com/leriomaggio



Example of Maxpool with a 2x2 filter and a stride of 2

https://github.com/leriomaggio

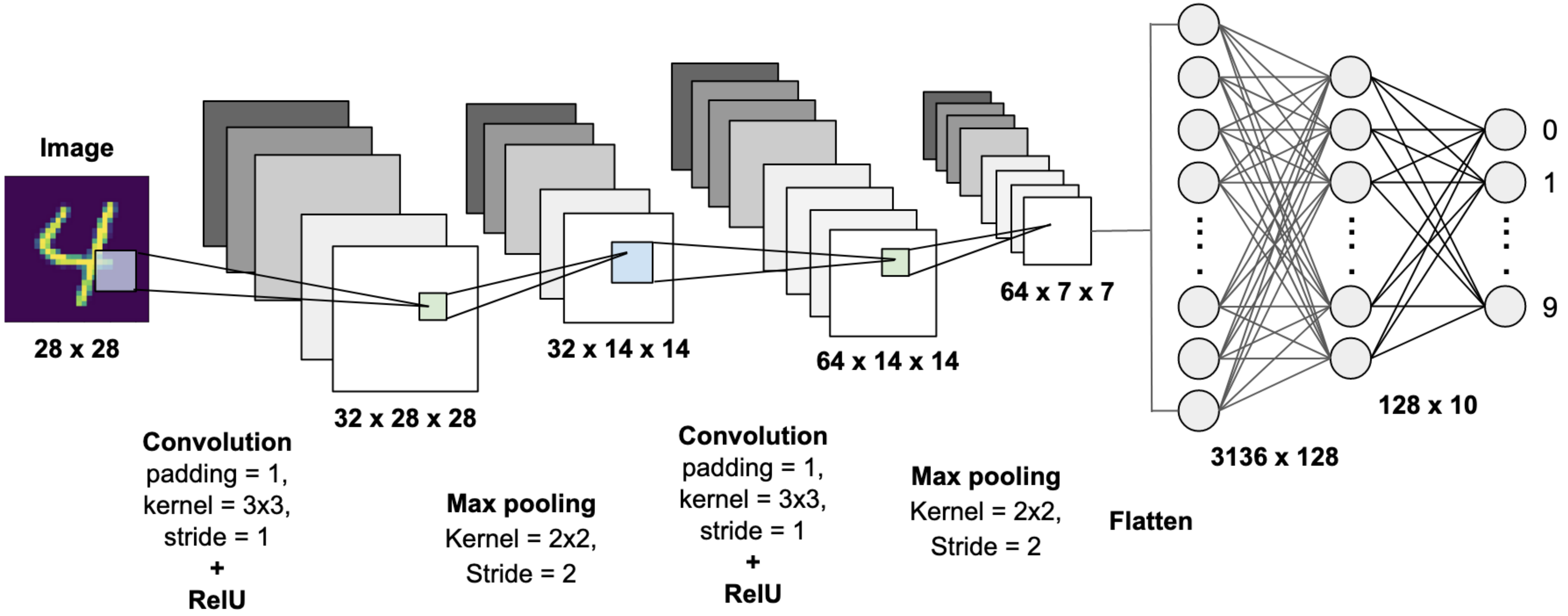
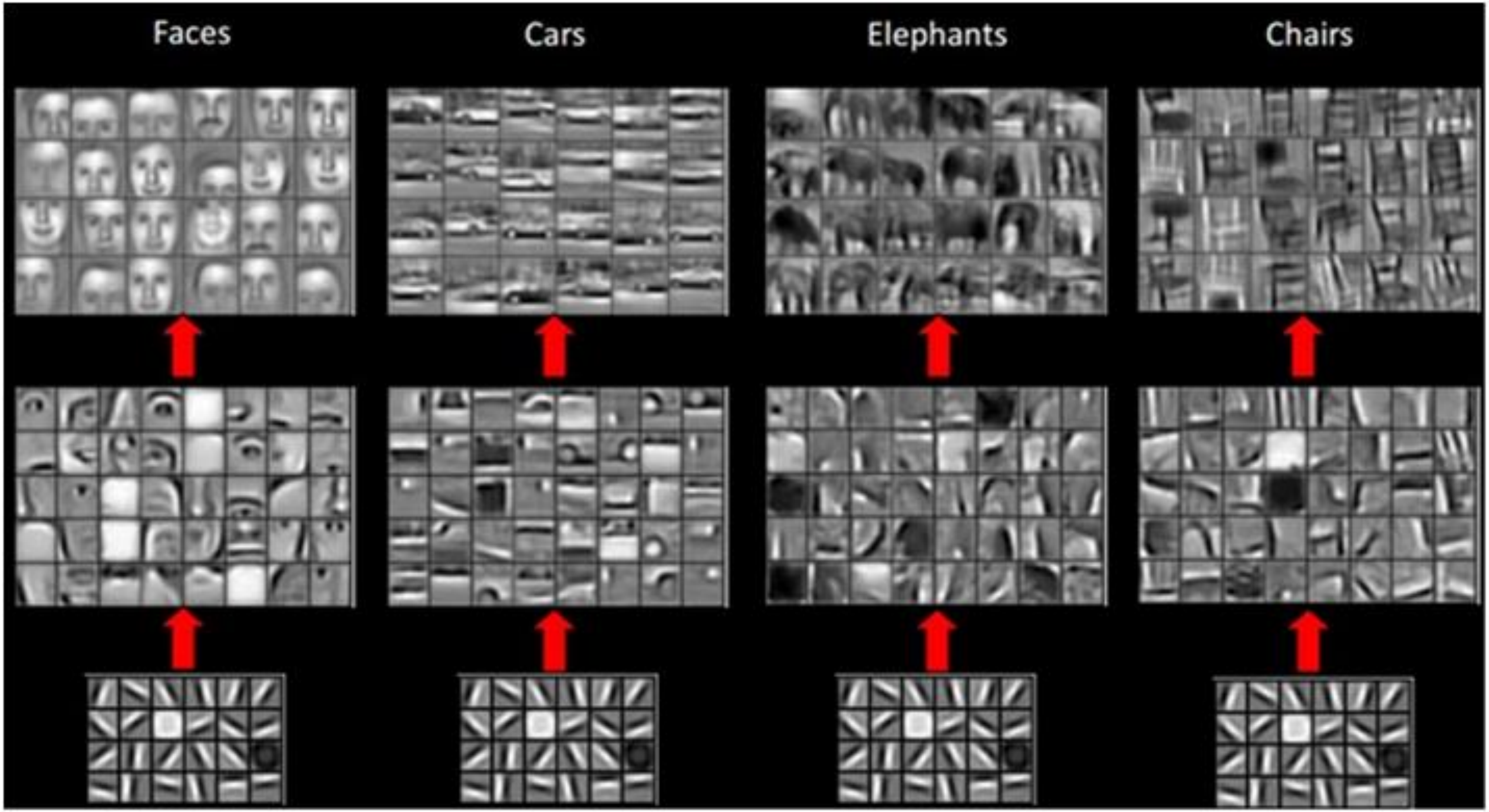
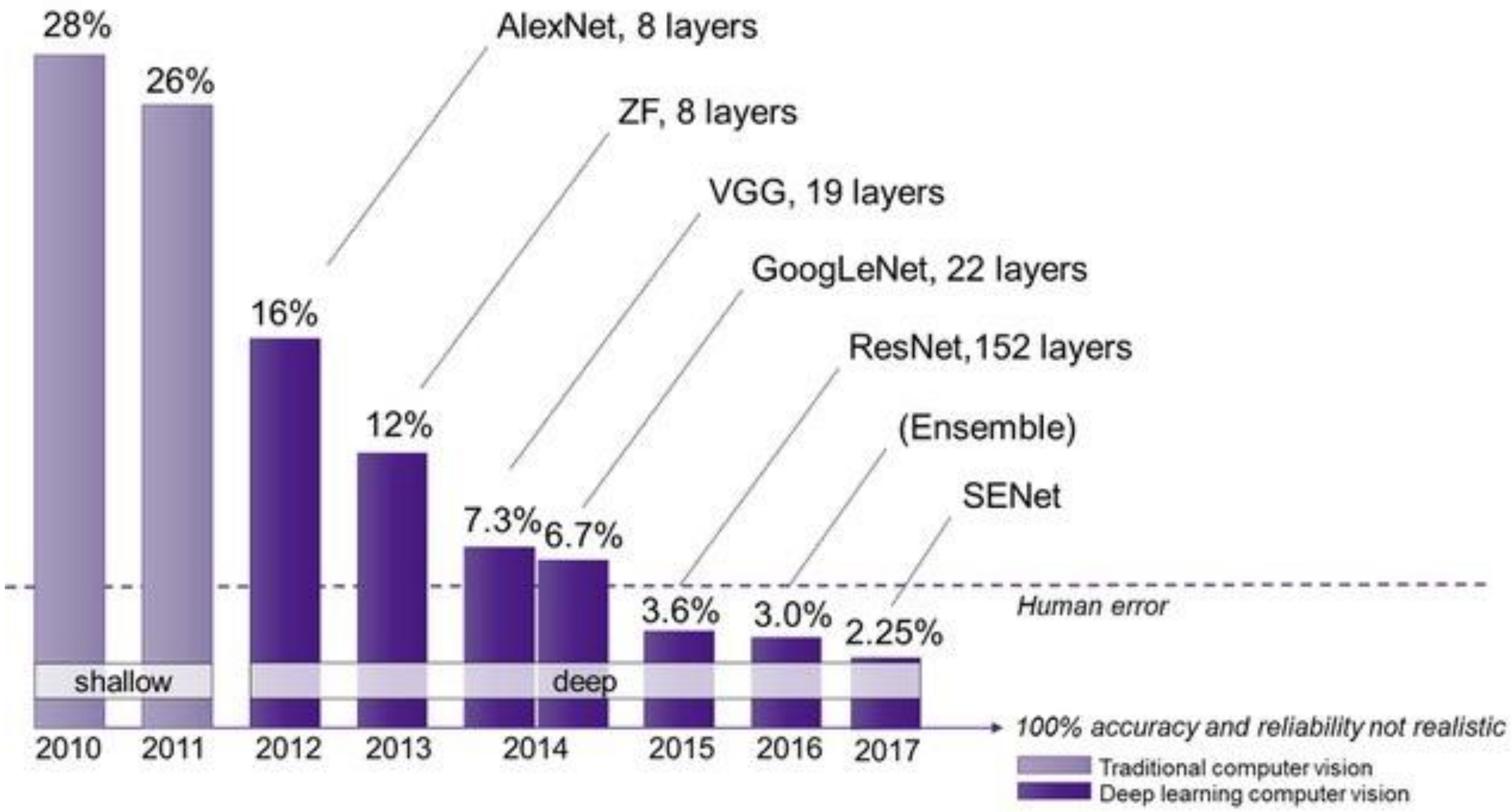


Image
28 x 28
32 x 28 x 28
32 x 14 x 14
64 x 14 x 14
64 x 7 x 7
0
1
9
3136 x 128
128 x 10
Convolution
padding = 1,
kernel = 3x3,
stride = 1
+
RelU
Max pooling
Kernel = 2x2,
Stride = 2
Convolution
padding = 1,
kernel = 3x3,
stride = 1
+
RelU
Max pooling
Kernel = 2x2,
Stride = 2
Flatten


Faces
Cars
Elephants
Chairs

- 14.197.122 images
- 21.841 subcategories
- 27 high-level categories

ImageNet Large Scale Visual Recognition Challenge



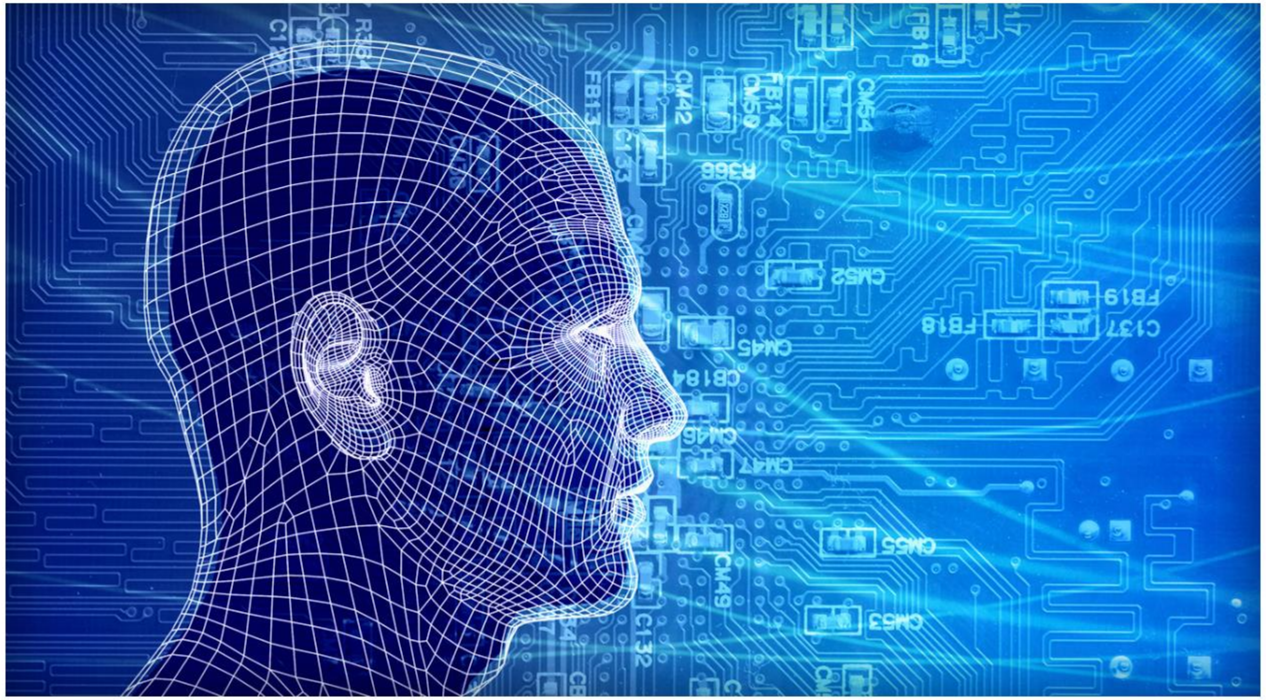
Let’s get real!

Cats & dogs

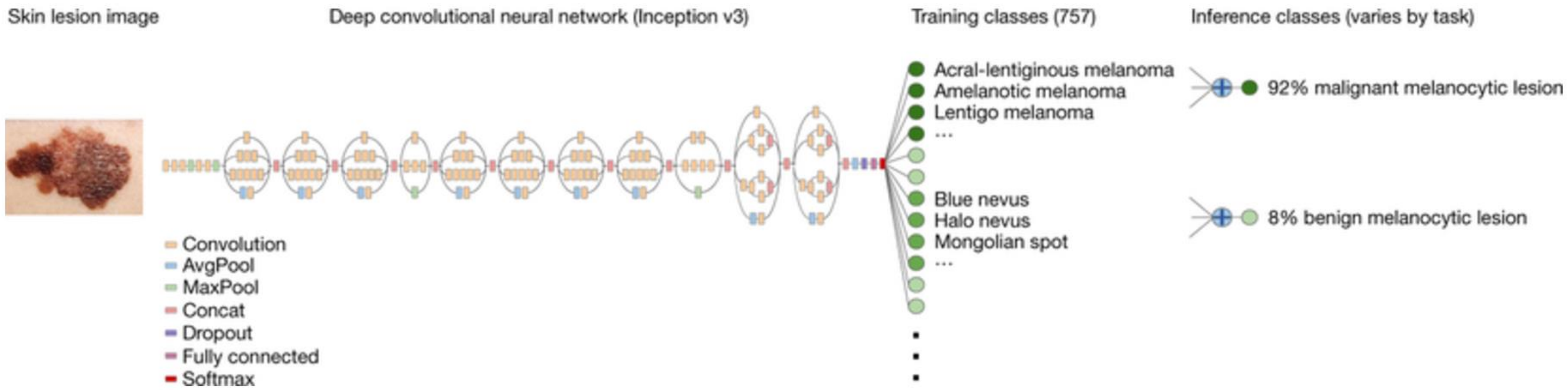
Deep learning algorithm diagnoses skin cancer as well as seasoned dermatologists

By Jessica Hall on January 25, 2017 at 1:25 pm | 16 Comments

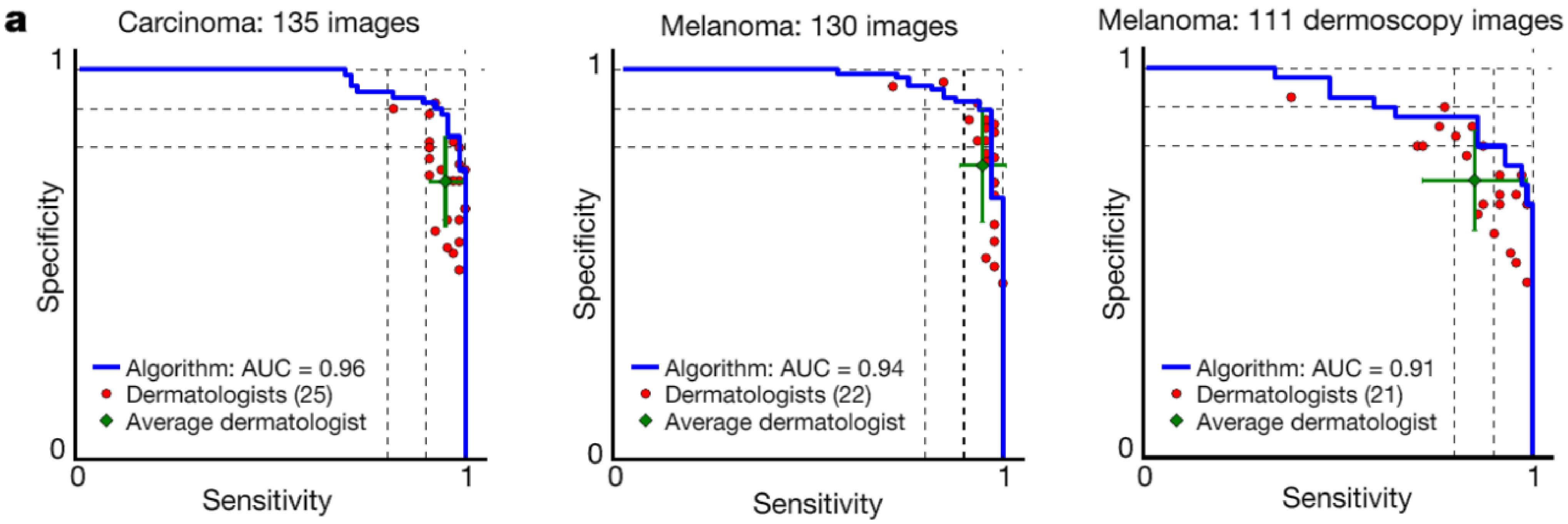
422 shares



Thrun et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature, January 2017 DOI: 10.1038/nature21056



Thrun et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature, January 2017 DOI: 10.1038/nature21056



keratinocyte carcinomas: 65
benign seborrheic keratoses: 75

malignant melanomas: 33
benign nevi: 97

malignant melanomas: 71
benign nevi: 40

Branch: master ▾ models / research / inception /

Create new file | Upload files | Find file | History

ema987 and shlens Added missing python interpreter directive (#6015) Latest commit 47a7d7c on 11 Jan

..

g3doc	Move the research models into a research subfolder (#2430)	2 years ago
inception	Added missing python interpreter directive (#6015)	10 months ago
.gitignore	Move the research models into a research subfolder (#2430)	2 years ago
README.md	Make the redirect to slim notice more prominent	2 years ago
WORKSPACE	Move the research models into a research subfolder (#2430)	2 years ago

README.md

NOTE: For the most part, you will find a newer version of this code at models/research/slim. In particular:

- `inception_train.py` and `imagenet_train.py` should no longer be used. The slim editions for running on multiple GPUs are the current best examples.
- `inception_distributed_train.py` and `imagenet_distributed_train.py` are still valid examples of distributed training.

For performance benchmarking, please see https://www.tensorflow.org/performance/benchmarks.

Inception in TensorFlow

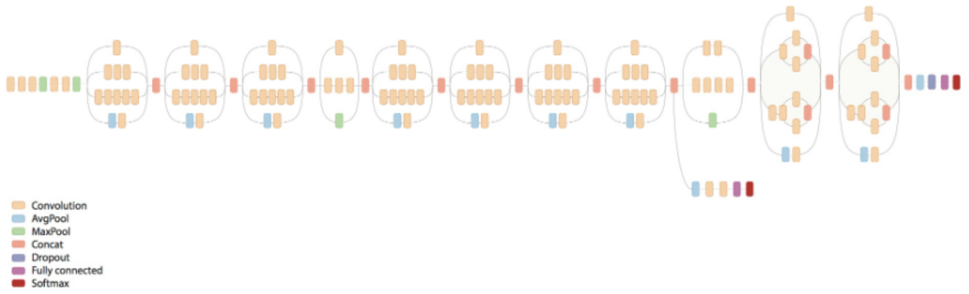
ImageNet is a common academic data set in machine learning for training an image recognition system. Code in this directory demonstrates how to use TensorFlow to train and evaluate a type of convolutional neural network (CNN) on this academic data set. In particular, we demonstrate how to train the Inception v3 architecture as specified in:

Rethinking the Inception Architecture for Computer Vision

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna

http://arxiv.org/abs/1512.00567

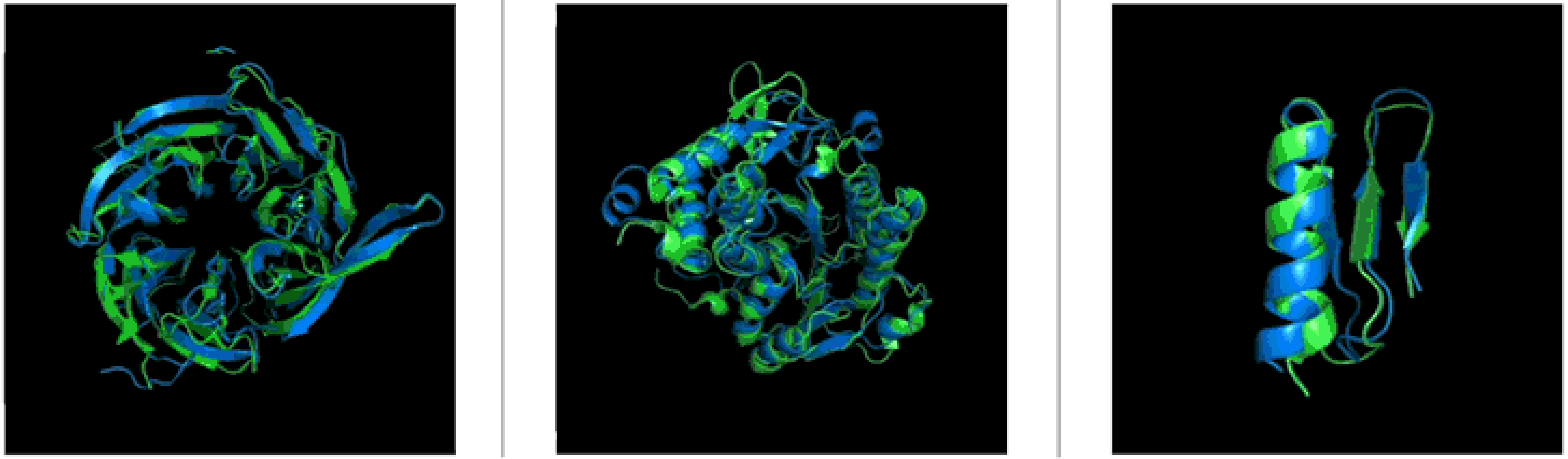
This network achieves 21.2% top-1 and 5.6% top-5 error for single frame evaluation with a computational cost of 5 billion multiply-adds per inference and with using less than 25 million parameters. Below is a visualization of the model architecture.



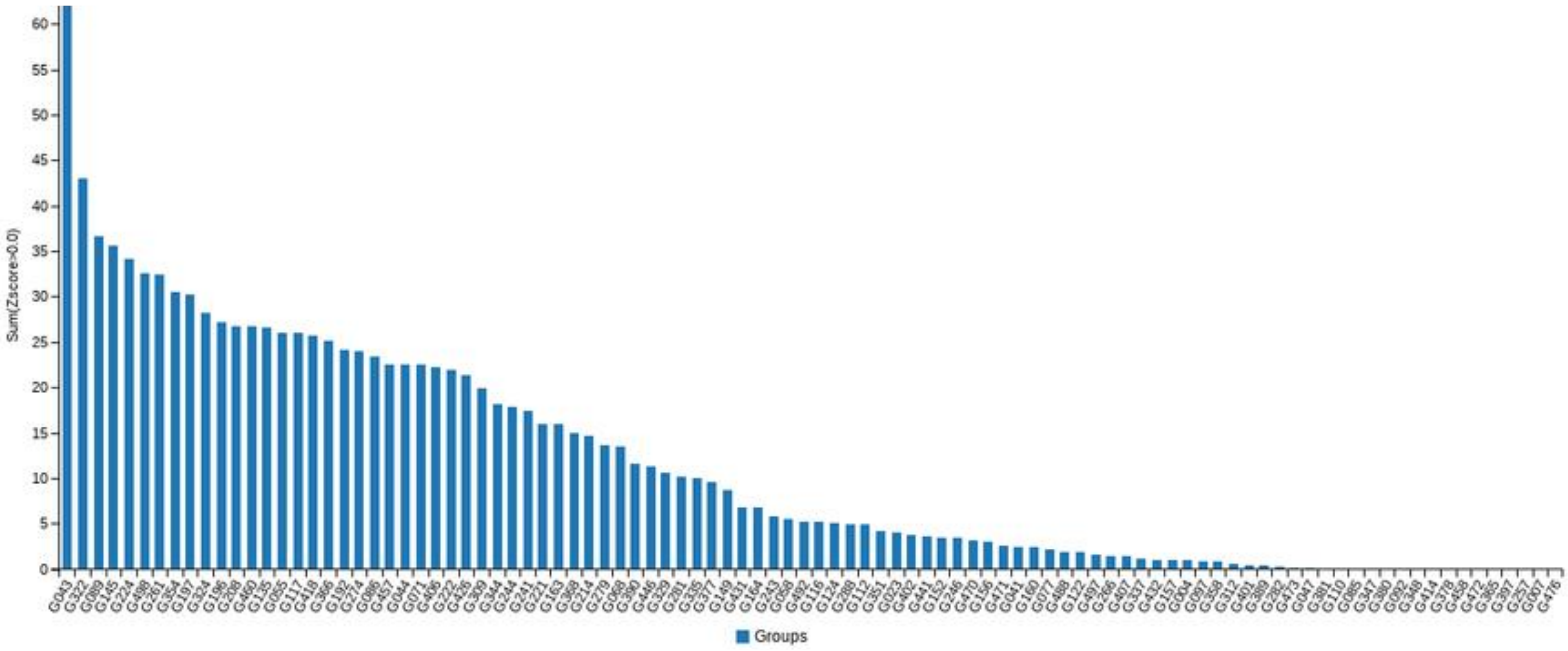
Critical Assessment of Techniques for Protein Structure Prediction

T0954 / 6CVZ | T0965 / 6D2V | T0955 / 5W9F

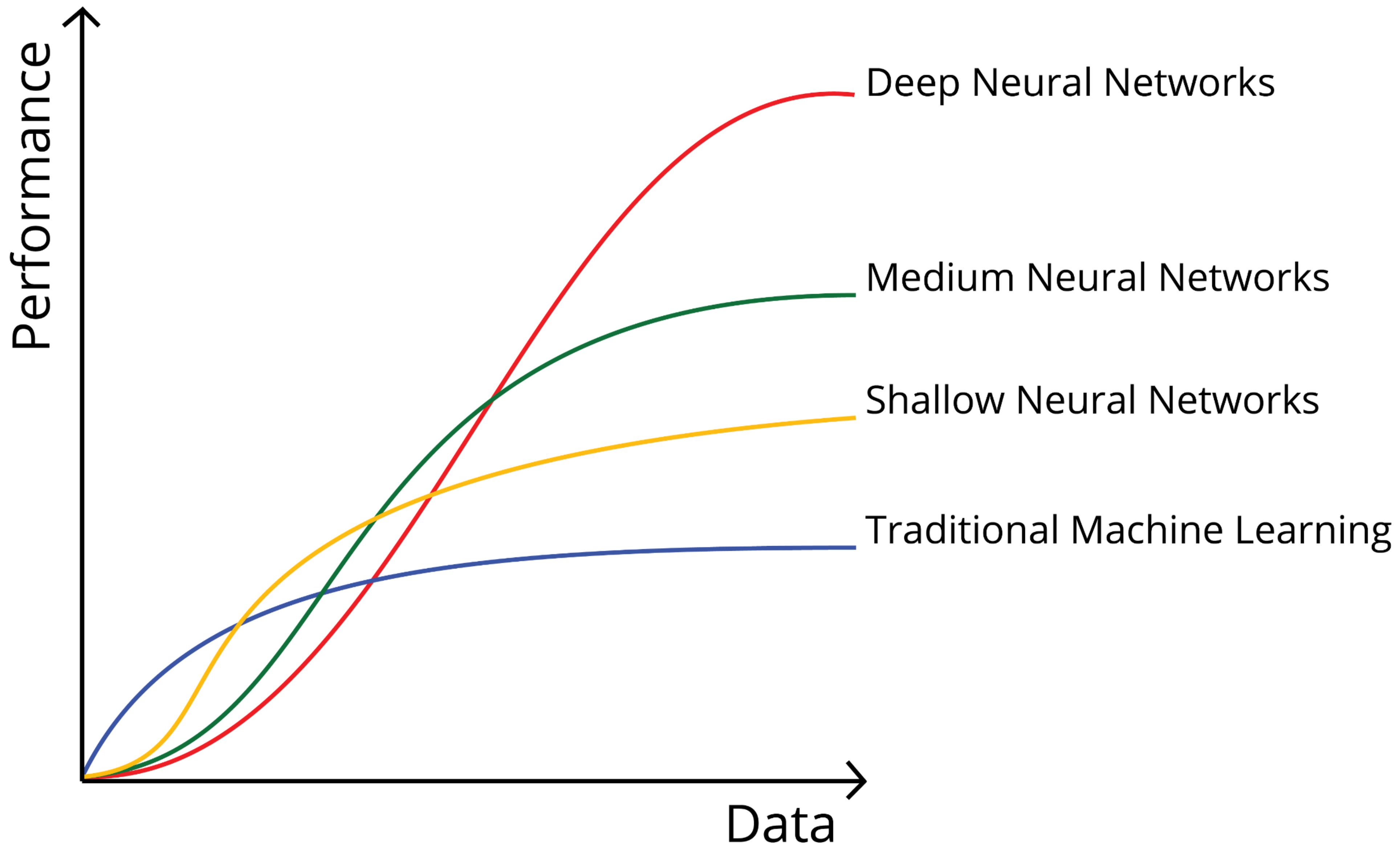
Structures:
Ground truth (green)
Predicted (blue)



deepmind.com/blog/article/alphafold @ CASP13 (2018)



predictioncenter.org


Performance
Deep Neural Networks
Medium Neural Networks
Shallow Neural Networks
Traditional Machine Learning
Data