

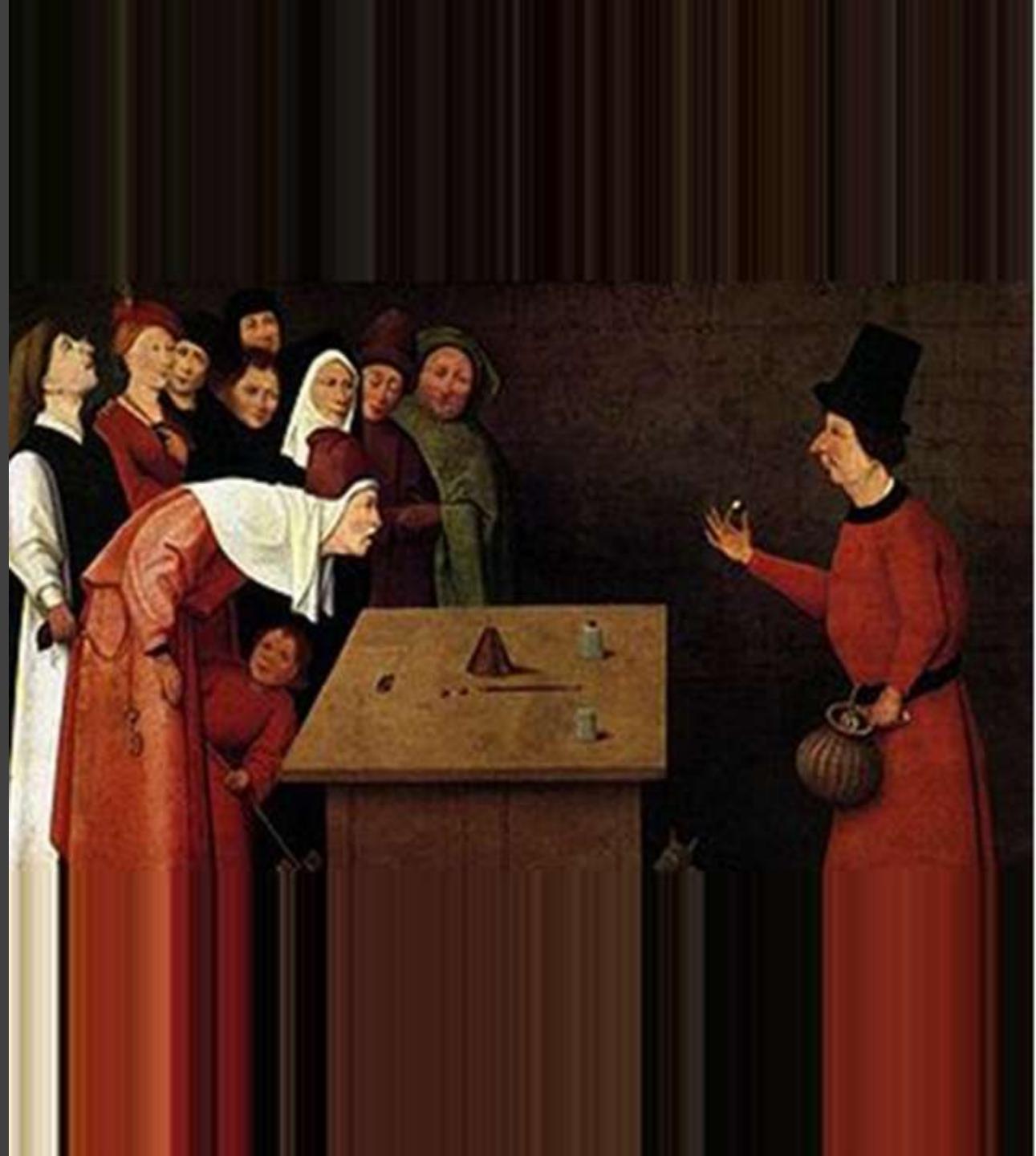
Basics of Pattern Recognitions

Dmitry Ryabokon, github.com/dryabokon

Content

- 1.1 Bayesian approach
 - 1.2. Approaches for ML
 - 1.3. Classifying the independent binary features
 - 1.4. Naive Bayesian classifier
 - 1.5. Gaussian classifier
 - 1.6. Linear discrimination
 - 1.7. Adaboost
 - 1.8. ROC-Curve
 - 1.9. Benchmarking the Classifiers
 - 1.10. Unsupervised learning
 - 1.11. Non-Bayesian approach
 - 1.12. Labeling
 - 1.13. Object detection
 - 1.14. Detection of patterns and anomalies
 - 1.15. Stereo Vision
 - 2.1. CNNs out of box
 - 2.2. Basic operations
 - 2.3. Feature extraction
 - 2.4. Transfer learning
 - 2.5. Fine Tuning
 - 2.6. Benchmarking the CNNs
 - 2.7. Reconstruct features back into images
-
- 3.1 Assignments

1.1. Bayesian approach



Bayesian approach

Definitions

$x \in X$ feature

$k \in K$ state (hidden)

$p(x, k)$ joint probability

$W : K \times K \rightarrow R$ penalty function

$q : X \rightarrow K$ decision strategy

$$Risk(q) = \sum_{k \in K} \sum_{x \in X} p(x, k) \cdot W(k, q(x))$$

Bayesian approach

The problem

$x \in X$ feature

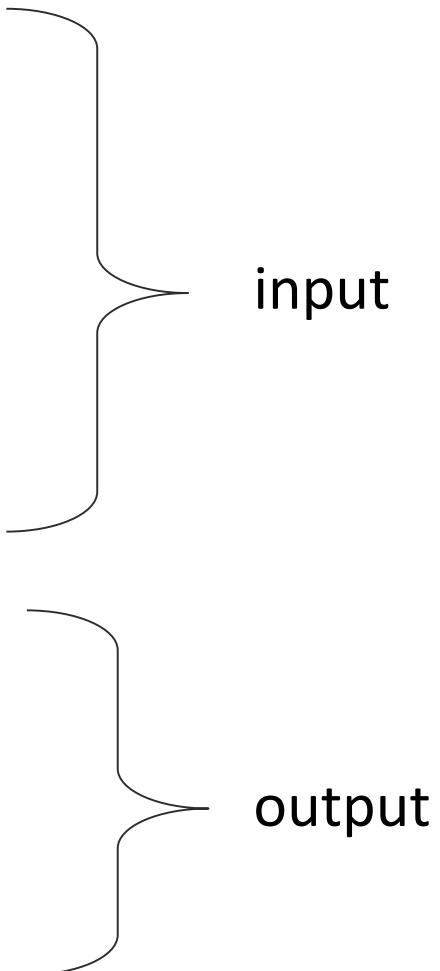
$k \in K$ state (hidden)

$p(x, k)$ joint probability

$W : K \times K \rightarrow R$ penalty function

$q : X \rightarrow K$ decision strategy

$$q^*(x) = \arg \min_{q(x)} Risk(q(x)) = \arg \min_{q(x)} \sum_{k \in K} \sum_{x \in X} p(x, k) \cdot W(k, q(x))$$



Bayesian approach

Some basics

$$p(x, k) = p(k) \cdot p(x|k) = p(x) \cdot p(k|x) \quad \text{joint probability}$$

$$p(x) = \sum_k p(k) \cdot p(x|k) \quad \text{law of total probability}$$

$$p(k|x) = \frac{p(k) \cdot p(x|k)}{\sum_{k'} p(k') \cdot p(x|k')} \quad \text{Bayes' theorem}$$

$p(k)$ prior probability

$p(k|x)$ posterior probability

$p(x|k)$ conditional probability of x , assuming k

$p(x)$ probability of x

Bayesian approach

Example

$$X = \{8, 9, 10, 11, 12\} \quad \text{time}$$

$$\mathcal{K} = \{\text{1-revision}, \text{2-free_ride}\}$$

$$p(k=1|x=8) = 10\%$$

$$p(k=2|x=8) = 90\%$$

$$W(k=1, k'=1) = 5 \quad W(k=2, k'=1) = 100$$

$$W(k=1, k'=2) = 5 \quad W(k=2, k'=2) = 0$$



Bayesian approach

Flexibility to not make a decision

$$W(k, k') = \begin{cases} 0 & \text{when } k = k' \\ 1 & \text{when } k \neq k' \\ \varepsilon & \text{when } k = \text{reject} \end{cases}$$

$$\begin{aligned} Risk(\text{reject}) &= \sum_{k \in K} p(k|x) \cdot W(k, \text{reject}) = \\ &= \sum_{k \in K} p(k|x) \cdot \varepsilon = \varepsilon \end{aligned}$$



Bayesian approach

Flexibility to not make a decision

$$\varepsilon = 0$$

if $\min_k Risk(k) < \varepsilon$

then $k^* = \arg \min_k \sum_{k' \in K} p(k|x) \cdot W(k, k')$

else reject

$$\varepsilon = \infty$$

if $\min_k Risk(k) < \varepsilon$

then $k^* = \arg \min_k \sum_{k' \in K} p(k|x) \cdot W(k, k')$

else ~~reject~~

Bayesian approach

Decision strategies for different penalty functions

$$w(k, k') = |k - k'| \quad \text{median}$$

$$w(k, k') = (k - k')^2 \quad \text{average}$$

$$w(k, k') = \begin{cases} 0 & k = k' \\ 1 & k \neq k' \end{cases} \quad \text{the most probable sample}$$

$$w(k, k') = \begin{cases} 0 & |k - k'| < \Delta \\ 1 & |k - k'| \geq \Delta \end{cases} \quad \text{center of the most probable section } \Delta$$

1.2. Approaches for ML



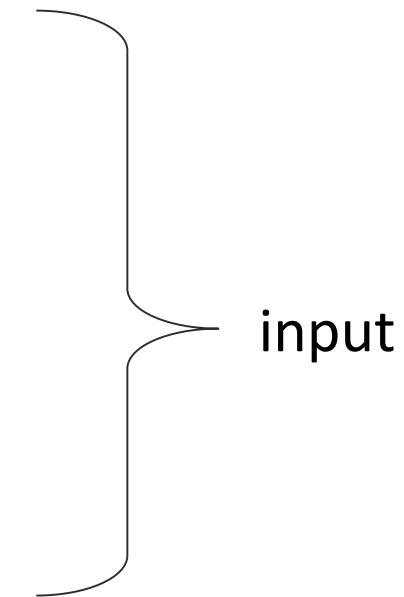
Approaches for ML

Maximum Likelihood

x_1, x_2, \dots, x_N features and
 k_1, k_2, \dots, k_N states

$k \in K = \{1, 2, \dots, K\}$ domain for k

$p(x|k) = f(x, \mu_k)$ The model is parametrized by
 $\mu_1, \mu_2, \dots, \mu_K$



$$(\mu_1, \mu_2, \dots, \mu_K)^* = \arg \max_{\mu_1, \mu_2, \dots, \mu_K} \prod_{i=1}^N p(x_i, k_i)$$

output

Approaches for ML

Maximum Likelihood

$$\begin{aligned} \arg \max_{\mu_1, \mu_2, \dots, \mu_K} \prod_{i=1}^N p(x_i, k_i) &= \arg \max_{\mu_1, \mu_2, \dots, \mu_K} \left(\prod_{i=1}^N p(k_i) \cdot p(x_i | k_i) \right) = \arg \max_{\mu_1, \mu_2, \dots, \mu_K} \left(\prod_{i=1}^N p(x_i | k_i) \right) = \\ &= \arg \max_{\mu_1, \mu_2, \dots, \mu_K} \left(\prod_{x \in X_1} p(x|1) \times \prod_{x \in X_2} p(x|2) \times \prod_{x \in X_K} p(x|K) \right) \end{aligned}$$

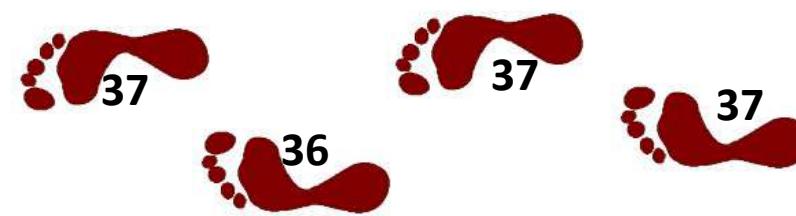
$$\mu_k^* = \arg \max_{\mu} \prod_{x \in X_k} f(x, \mu)$$

Approaches for ML

Maximum Likelihood: example



$$p(x|k=1) = f(x, \mu_1) \cong \exp\left\{ - (x - \mu_1)^2 \right\}$$



$$p(x|k=2) = f(x, \mu_2) \cong \exp\left\{ - (x - \mu_2)^2 \right\}$$

$$(\mu_1, \mu_2) = \arg \max \prod_{i=1}^7 p(x_i, k_i)$$

$$\mu_1 = \arg \max_{\mu} \exp\left\{ -(45 - \mu)^2 - (46 - \mu)^2 - (45 - \mu)^2 \right\}$$

Approaches for ML

Maximum Likelihood: example



$$\mu_1 = \arg \max_{\mu} \exp \left\{ - (45 - \mu)^2 - (46 - \mu)^2 - (45 - \mu)^2 \right\}$$

$$\mu_1 = \arg \min_{\mu} \left\{ (45 - \mu)^2 + (46 - \mu)^2 + (45 - \mu)^2 \right\}$$

$$\mu_1 = \frac{45 + 46 + 45}{3} = \langle x \rangle_{X_1}$$

Approaches for ML

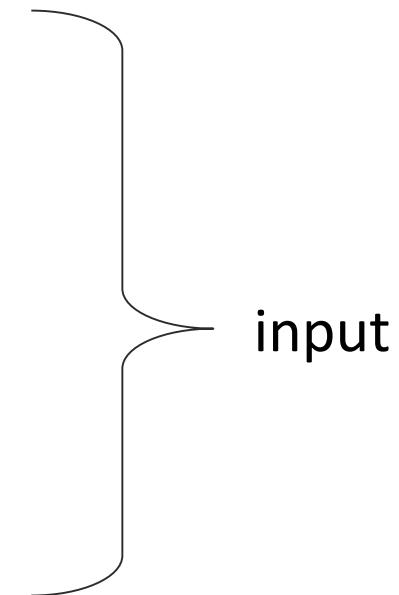
Bias in training data

x_1, x_2, \dots, x_N features and
 k_1, k_2, \dots, k_N states

$k \in K = \{1, 2, \dots, K\}$ domain for k

$p(x|k) = f(x, \mu_k)$ The model is parametrized by
 $\mu_1, \mu_2, \dots, \mu_K$

$$\mu_1^* = \arg \max_{\mu} \left\{ \min_{x \in X_1} p(x|k=1, \mu) \right\}$$



output

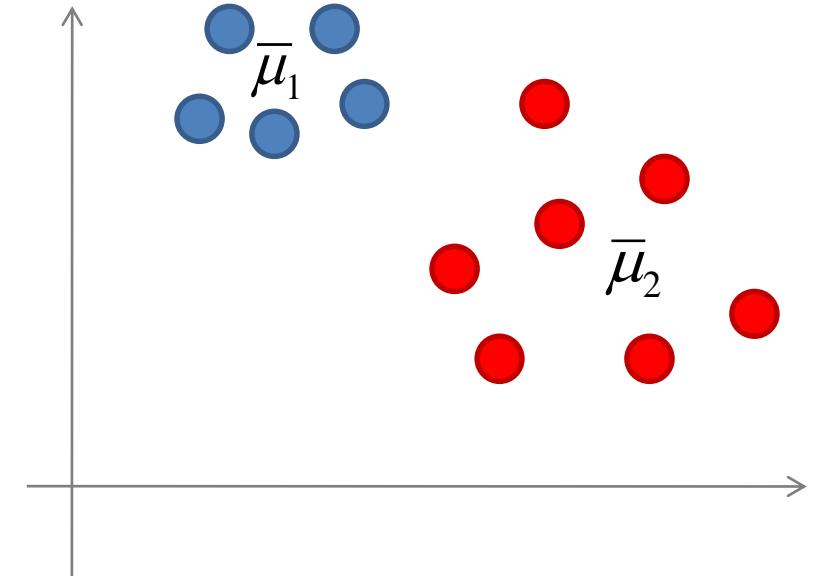
Approaches for ML

Bias in training data: example

$$p(x|k=1) = N(\bar{\mu}_1, 1)$$

$$p(x|k=2) = N(\bar{\mu}_2, 1)$$

$$\bar{\mu}_1^* = \arg \max_{\bar{\mu}} \left\{ \min_{x \in X_1} p(x|k=1, \bar{\mu}) \right\}$$



Approaches for ML

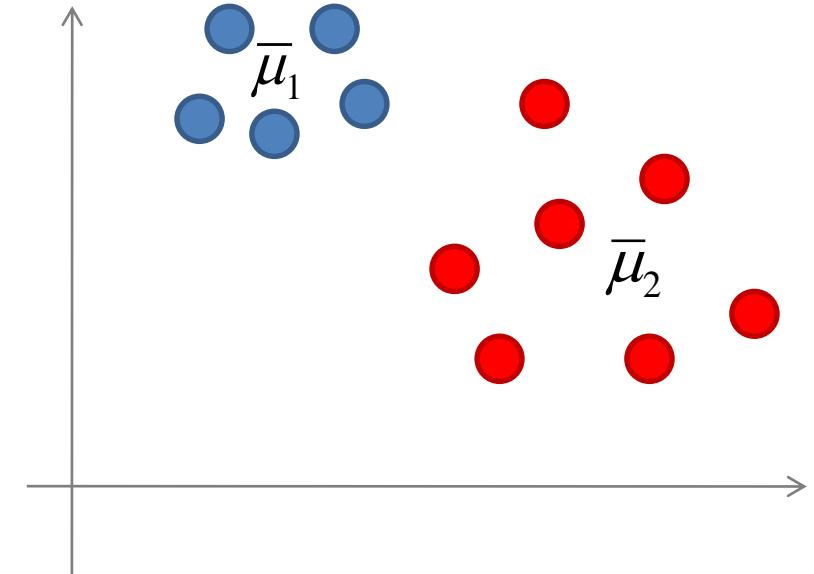
Bias in training data: example

$$p(x|k=1) = N(\bar{\mu}_1, 1)$$

$$p(x|k=2) = N(\bar{\mu}_2, 1)$$

$$\bar{\mu}_1^* = \arg \max_{\bar{\mu}} \left\{ \min_{x \in X_1} p(x|k=1, \bar{\mu}) \right\}$$

center of the circle that
holds all the points from X1



Approaches for ML

ERM: Empirical risk minimization

x_1, x_2, \dots, x_N features and

k_1, k_2, \dots, k_N states

$k \in K = \{1, 2, \dots, K\}$ domain for k

$W(k, k')$ penalty function

$q(x, \bar{\mu}) \in K$ decision strategy is parametrized by μ

Approaches for ML

ERM: Empirical risk minimization

$q(x, \bar{\mu}) \in K$ decision strategy is parametrized by μ

$$Risk(q(\bar{\mu})) = \sum_{x \in X} \sum_{k \in K} p(x, k) \cdot W(q(x, \bar{\mu}), k)$$

$$\cong \sum_{i=1}^N p(x_i, k_i) \cdot W(q(x_i, \bar{\mu}), k_i) = \sum_{i=1}^N W(q(x_i, \bar{\mu}), k_i)$$

$$\mu^* = \arg \min_{\mu} Risk q(\mu)$$

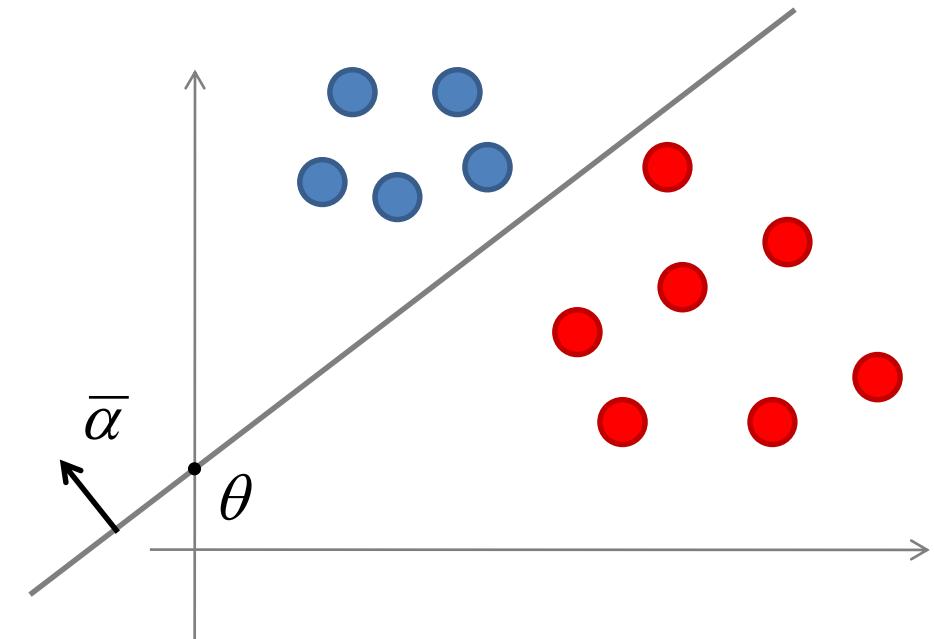
Approaches for ML

ERM: Empirical risk minimization

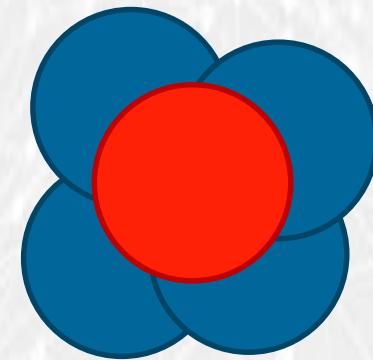
$$q(\bar{x}, \bar{\alpha}, \theta) = \begin{cases} -1 & \text{when } (\bar{x}, \bar{\alpha}) < \theta \\ +1 & \text{when } (\bar{x}, \bar{\alpha}) > \theta \end{cases}$$

$$W(k, k') = \begin{cases} 0 & k = k' \\ 1 & k \neq k' \end{cases}$$

$$\text{Risk}(q(\bar{\alpha}, \theta)) = \text{Number of errors}$$



1.3. Classifying the independent binary features



Classifying the independent binary features

The problem

$\bar{x} = (x_1, x_2, \dots, x_N)$ feature – a set of independent binary values

$$p(\bar{x}|k) = p(x_1|k) \cdot p(x_2|k) \cdot \dots \cdot p(x_N|k)$$

$k \in K = \{1, 2\}$ Few states are possible

$$\frac{p(\bar{x}|k=1)}{p(\bar{x}|k=2)} \geq \theta \quad \text{Decision strategy}$$

1
2

Classifying the independent binary features

The problem

$$\log \frac{p(\bar{x}|k=1)}{p(\bar{x}|k=2)} = \log \frac{p(x_1|k=1) \cdot \dots \cdot p(x_n|k=1)}{p(x_1|k=2) \cdot \dots \cdot p(x_n|k=2)} = \log \frac{p(x_1|k=1)}{p(x_1|k=2)} + \dots + \log \frac{p(x_n|k=1)}{p(x_n|k=2)} =$$

$$x_1 \cdot \log \frac{p(x_1 = 1|k=1) \cdot p(x_1 = 0|k=2)}{p(x_1 = 1|k=2) \cdot p(x_1 = 0|k=1)} + \log \frac{p(x_1 = 0|k=1)}{p(x_1 = 0|k=2)} +$$

⋮

$$x_N \cdot \log \frac{p(x_n = 1|k=1) \cdot p(x_n = 0|k=2)}{p(x_n = 1|k=2) \cdot p(x_n = 0|k=1)} + \log \frac{p(x_n = 0|k=1)}{p(x_n = 0|k=2)}$$

Classifying the independent binary features

Decision rule

$$\sum_{i=1}^N x_i \cdot \log \frac{p(x_i = 1 | k = 1) \cdot p(x_i = 0 | k = 2)}{p(x_i = 1 | k = 2) \cdot p(x_i = 0 | k = 1)} + \log \frac{p(x_i = 0 | k = 1)}{p(x_i = 0 | k = 2)} \begin{matrix} > \\ \leq \end{matrix} \log \theta$$

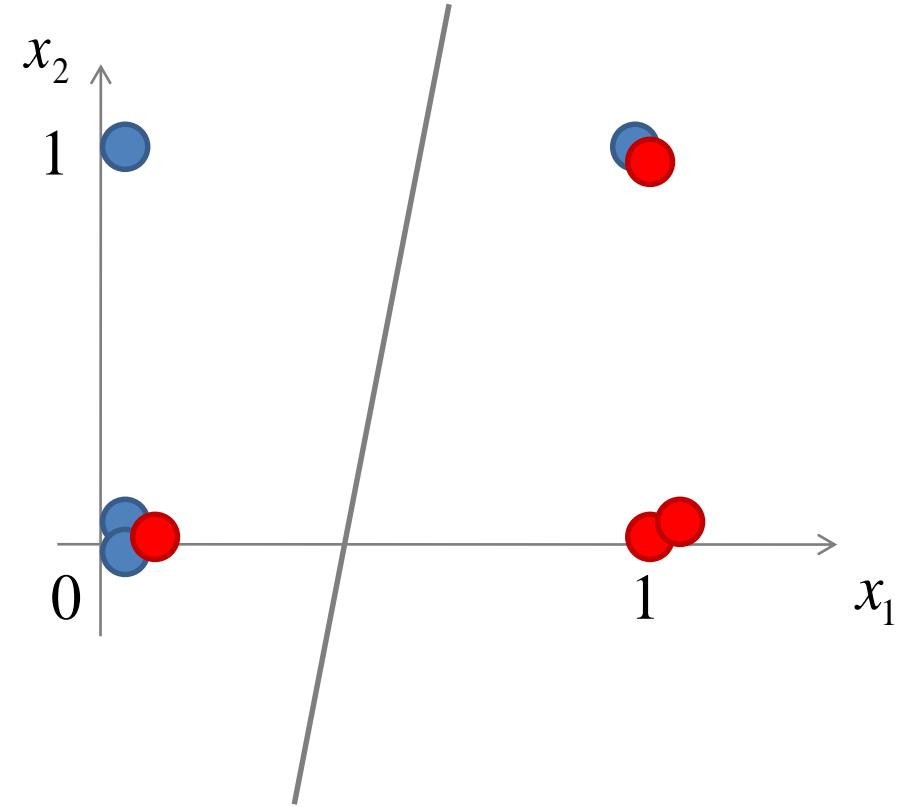
$$\sum_{i=1}^N x_i \cdot a_i \begin{matrix} > \\ \leq \end{matrix} \theta'$$

Classifying the independent binary features

Example

$$\sum_{i=1}^N x_i \cdot a_i \leq \theta'$$

1
≤
2



Classifying the independent binary features

Example

$$p(x_2 | \text{red}) \quad p(x_2 | \text{blue})$$

1/4

3/4

2/4

2/4



1

1

x_1

1/4

3/4

$$p(x_1 | \text{red})$$

$$p(x_1 | \text{blue})$$

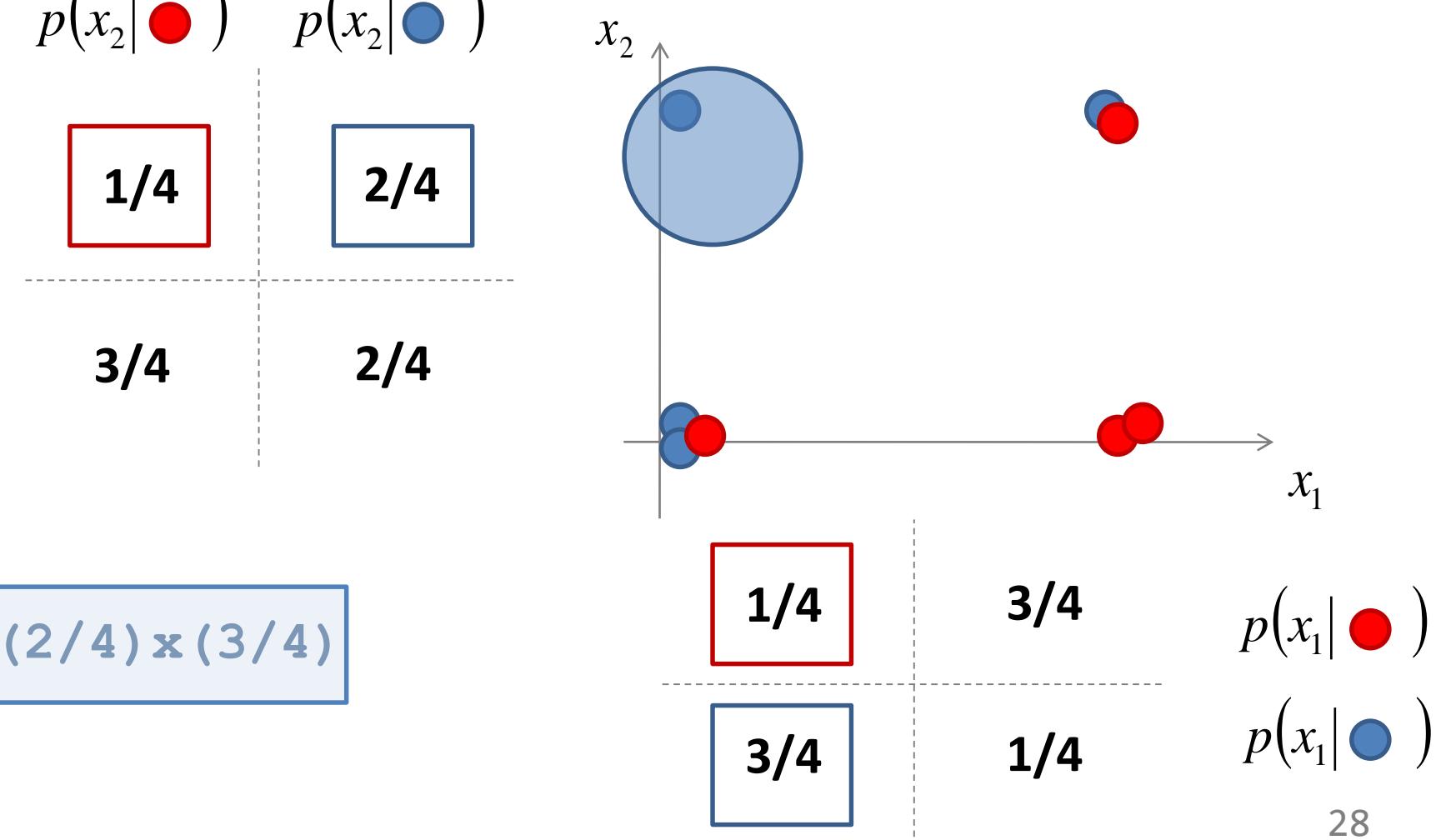
Classifying the independent binary features

Example

$$p(x_2 | \text{red}) \quad p(x_2 | \text{blue})$$

$1/4$	$2/4$
$3/4$	$2/4$

$$(1/4) \times (1/4) < (2/4) \times (3/4)$$



Classifying the independent binary features

Example

$$p(x_2 | \text{●}) \quad p(x_2 | \text{○})$$

$$\boxed{1/4}$$

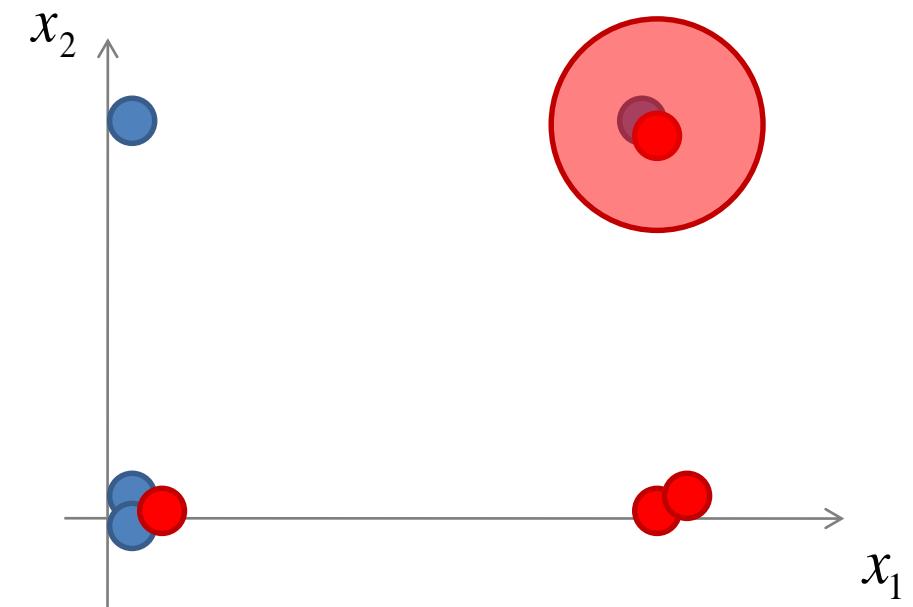
$$\boxed{2/4}$$

$$3/4$$

$$2/4$$

$$\boxed{(1/4) \times (3/4)}$$

$$> (2/4) \times (1/4)$$



$$1/4$$

$$3/4$$

$$\boxed{3/4}$$

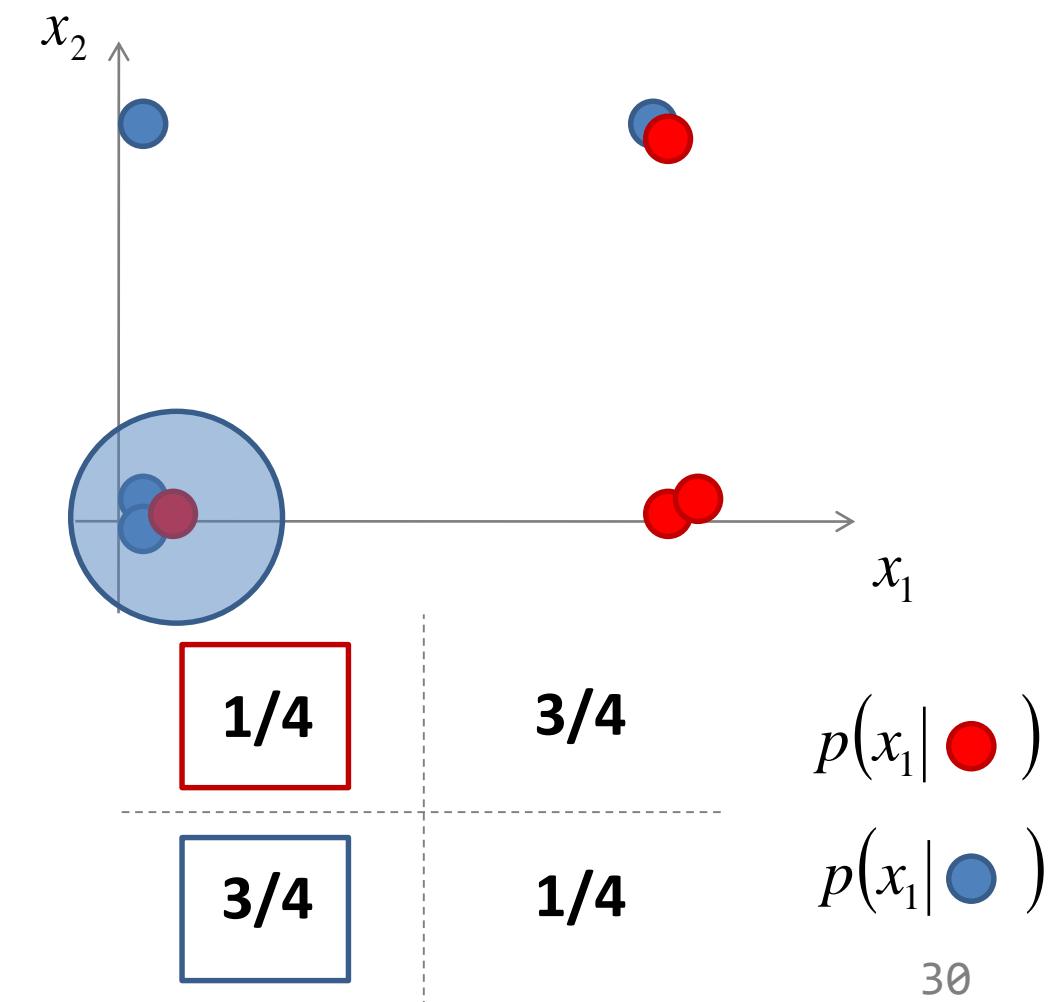
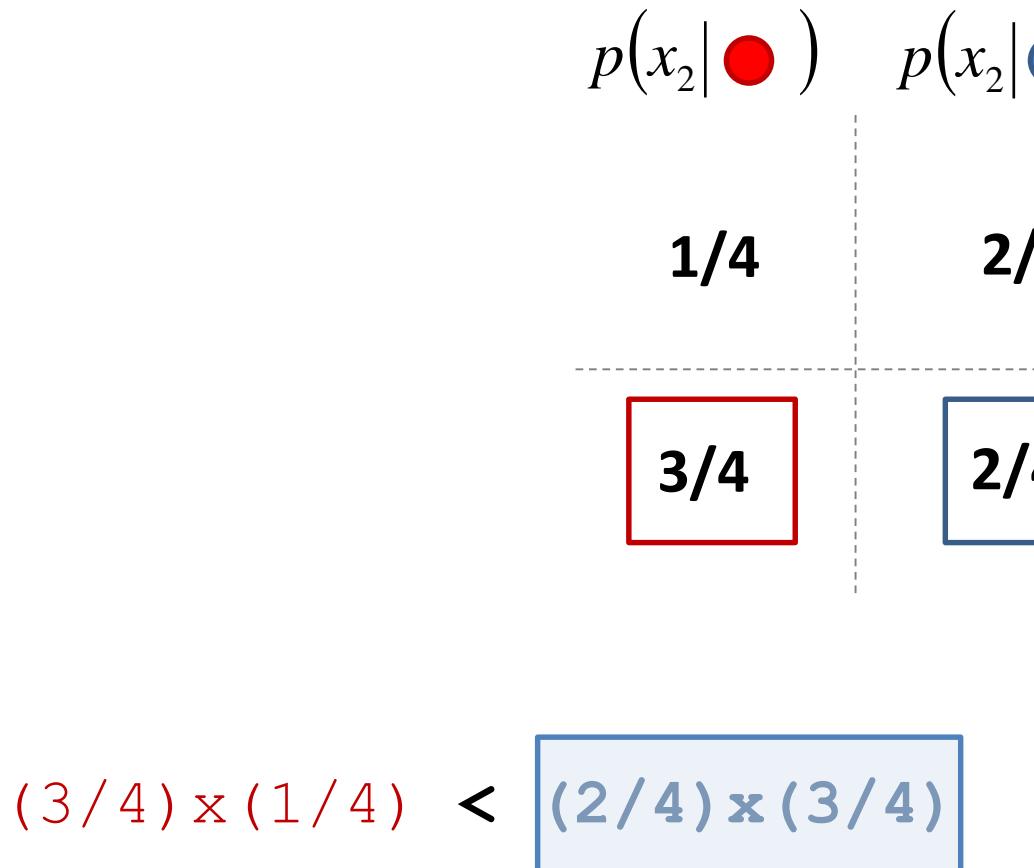
$$\boxed{1/4}$$

$$p(x_1 | \text{●})$$

$$p(x_1 | \text{○})$$

Classifying the independent binary features

Example



Classifying the independent binary features

Example

$$p(x_2 | \text{●}) \quad p(x_2 | \text{○})$$

1/4

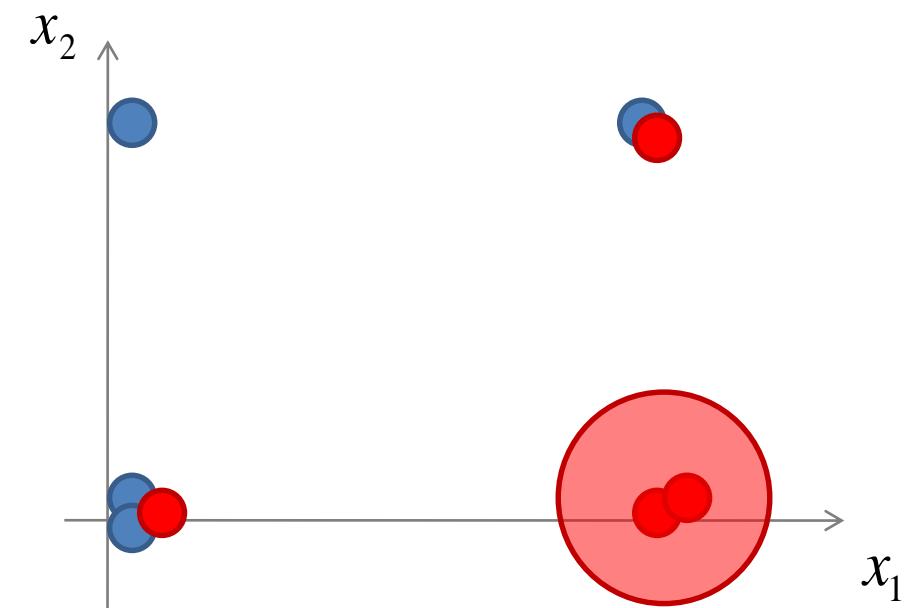
2/4

3/4

2/4

(3/4) x (3/4)

> (2/4) x (1/4)



1/4

3/4

3/4

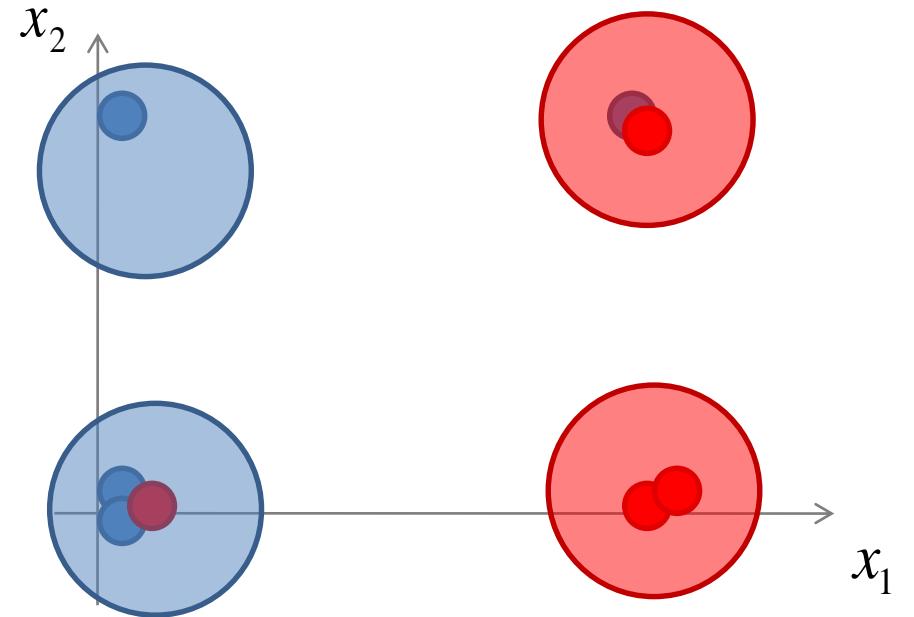
1/4

$$p(x_1 | \text{●})$$

$$p(x_1 | \text{○})$$

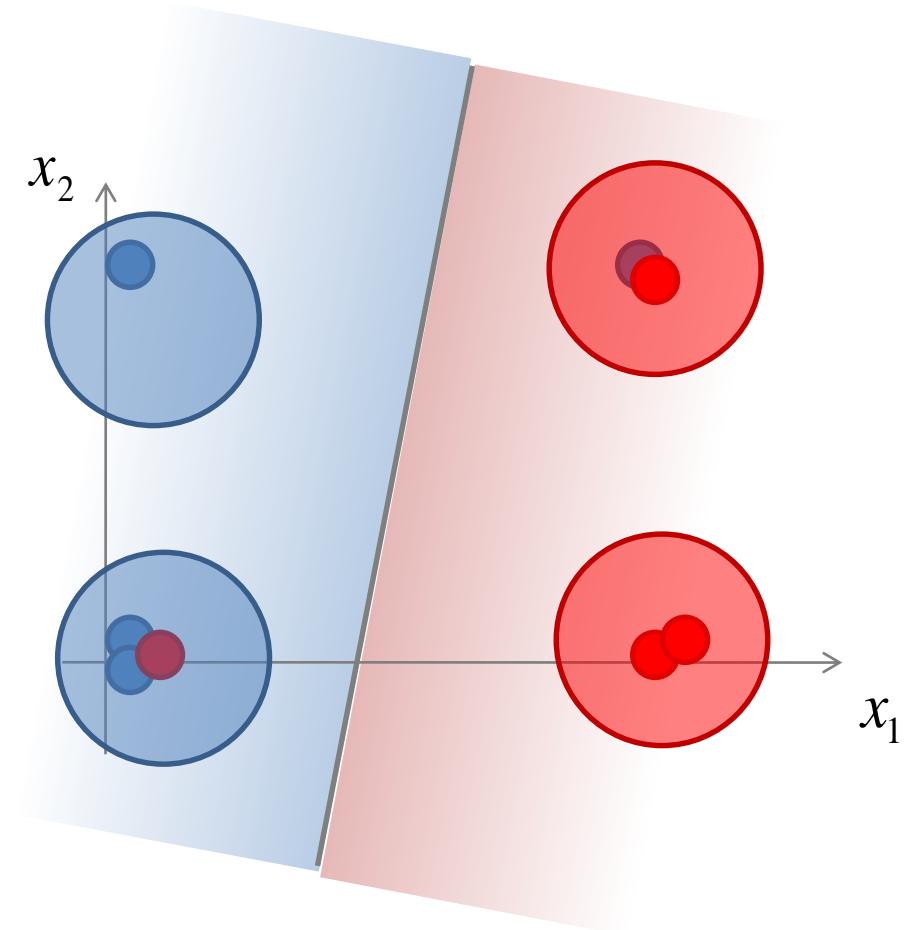
Classifying the independent binary features

Example



Classifying the independent binary features

Example



Classifying the independent binary features

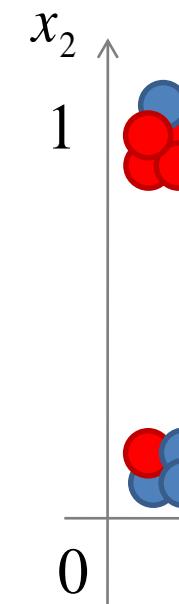
Another example

$$p(x_2 | \text{red}) \quad p(x_2 | \text{blue})$$

5/9

4/9

4/8



5/9

4/8

4/8

4/8

$$p(x_1 | \text{red})$$

$$p(x_1 | \text{blue})$$

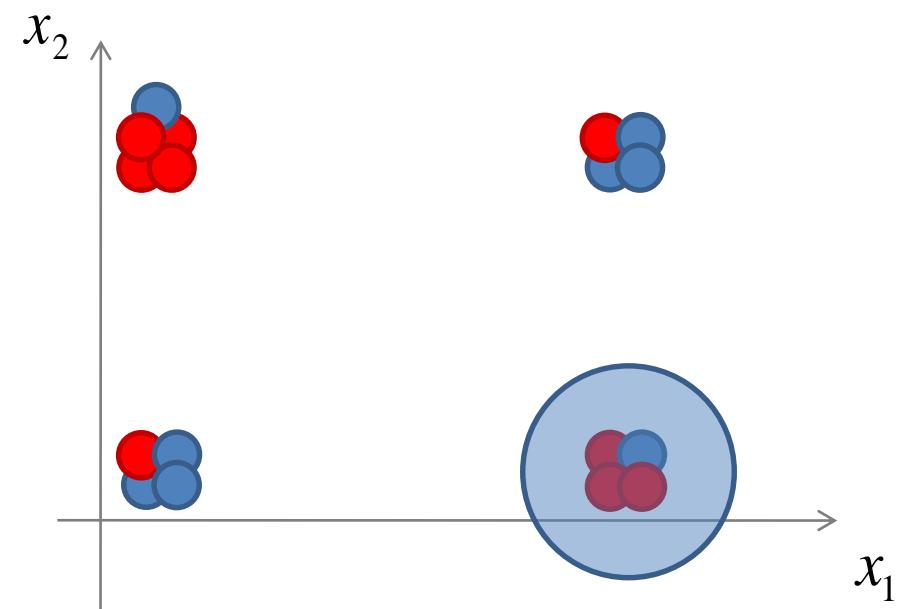
Classifying the independent binary features

Another example

$$p(x_2 | \text{red}) \quad p(x_2 | \text{blue})$$



$$(4/9) \times (4/9) < \boxed{(4/8) \times (4/8)}$$

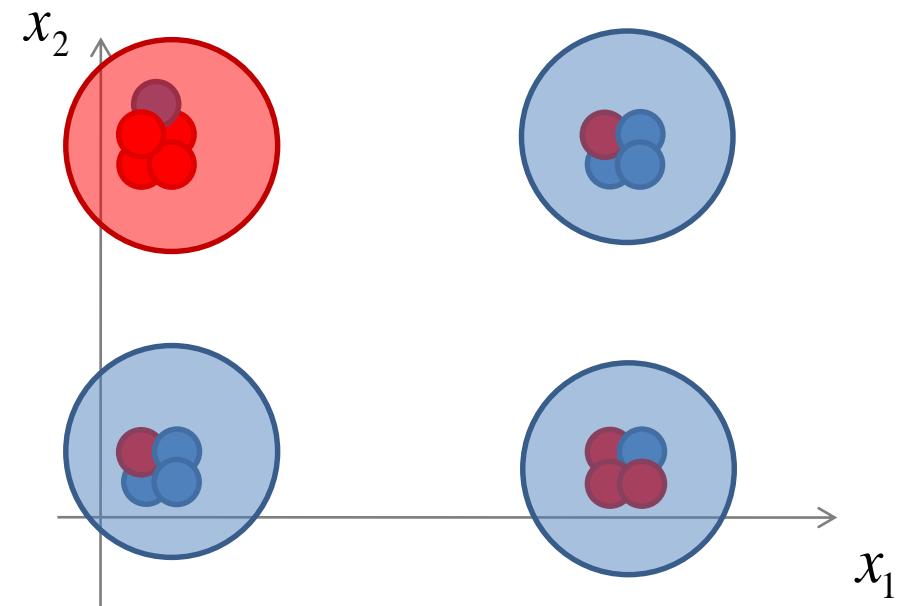


$$p(x_1 | \text{red})$$

$$p(x_1 | \text{blue})$$

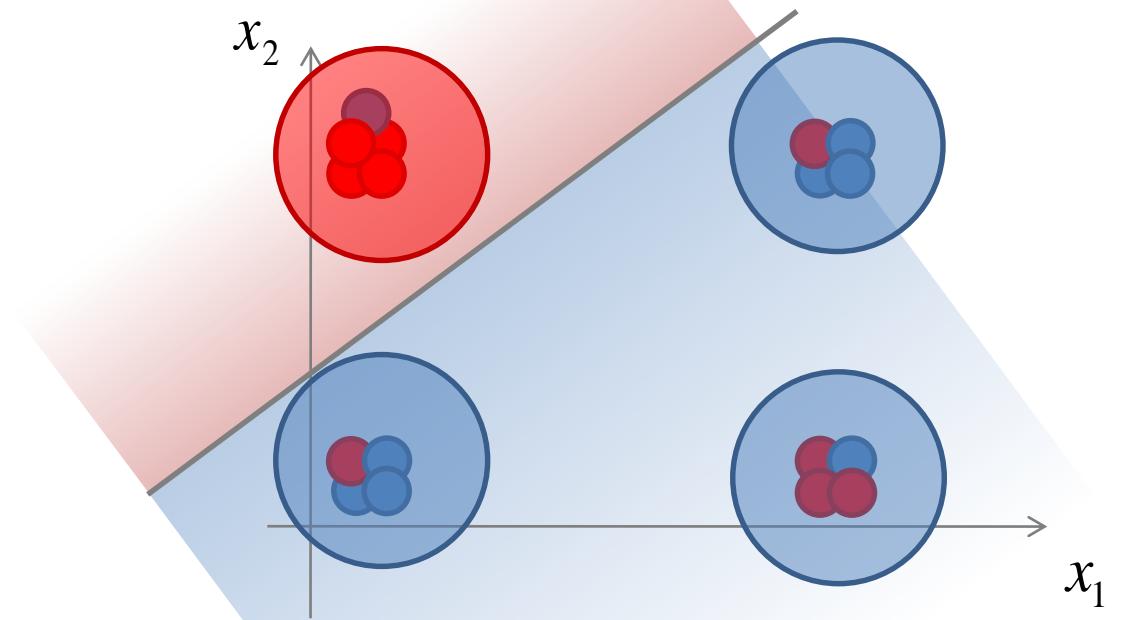
Classifying the independent binary features

Another example



Classifying the independent binary features

Another example



1.4. Naive Bayesian classifier



Naive Bayesian classifier

The problem

$\bar{x} = (x_1, x_2, \dots, x_N)$ feature

$$p(\bar{x}|k) = p(x_1|k) \cdot p(x_2|k) \cdot \dots \cdot p(x_N|k)$$

$k \in K = \{1, 2\}$ few states are possible

$$\frac{p(\bar{x}|k=1)}{p(\bar{x}|k=2)} \geq \theta \quad \text{decision strategy}$$

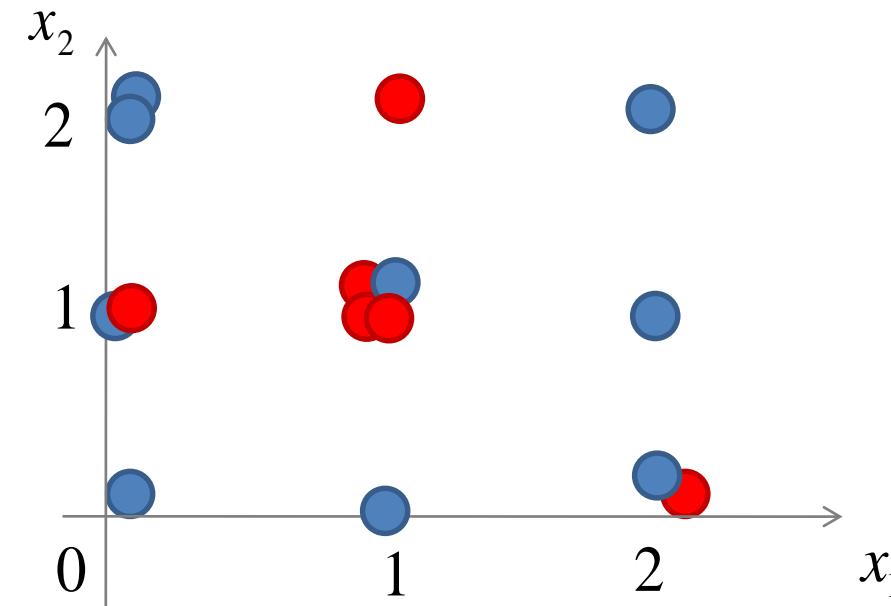
1
2

Naive Bayesian classifier

Example

$$p(x_2 | \text{red}) \quad p(x_2 | \text{blue})$$

$1/6$	$3/9$
$4/6$	$3/9$
$1/6$	$3/9$



$1/6$	$4/6$	$1/6$	$p(x_1 \text{red})$
$4/9$	$2/9$	$3/9$	$p(x_1 \text{blue})$

Naive Bayesian classifier

Example

$$p(x_2 | \text{red}) \quad p(x_2 | \text{blue})$$

1/6

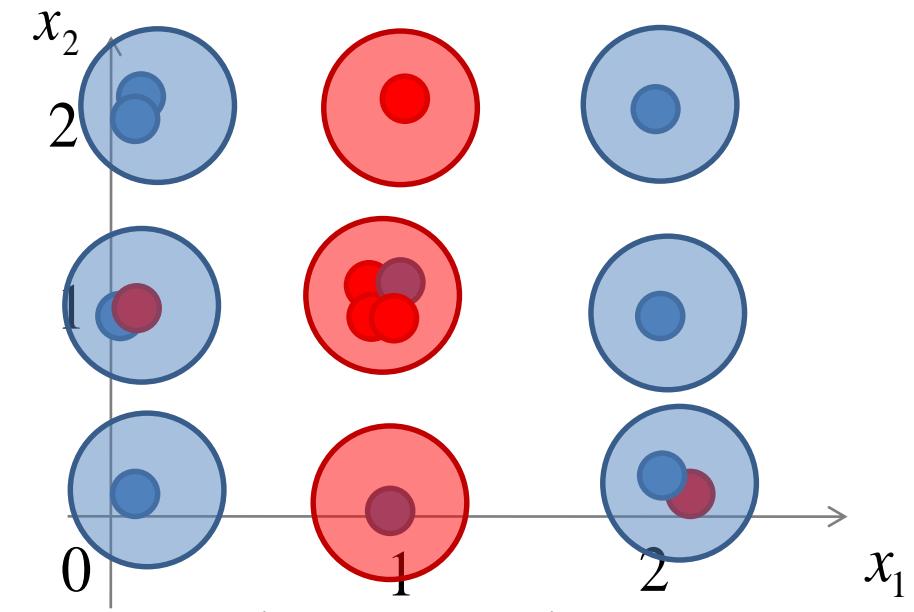
4/6

1/6

3/9

3/9

3/9



1/6

4/9

4/6

2/9

1/6

3/9

$$p(x_1 | \text{red})$$

$$p(x_1 | \text{blue})$$

1.5. Gaussian classifier

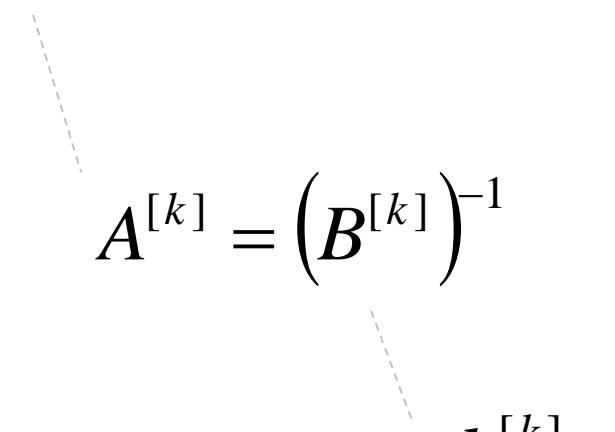


Gaussian classifier

Definitions

$$\bar{x} = (x_1, x_2, \dots, x_N) \text{ feature}$$

$$p(\bar{x}|k) \cong \exp\left(-0.5 \cdot \sum_{i=1}^n \sum_{j=1}^n a_{i,j}^{[k]} (x_i - \mu_i^{[k]})(x_j - \mu_j^{[k]})\right)$$

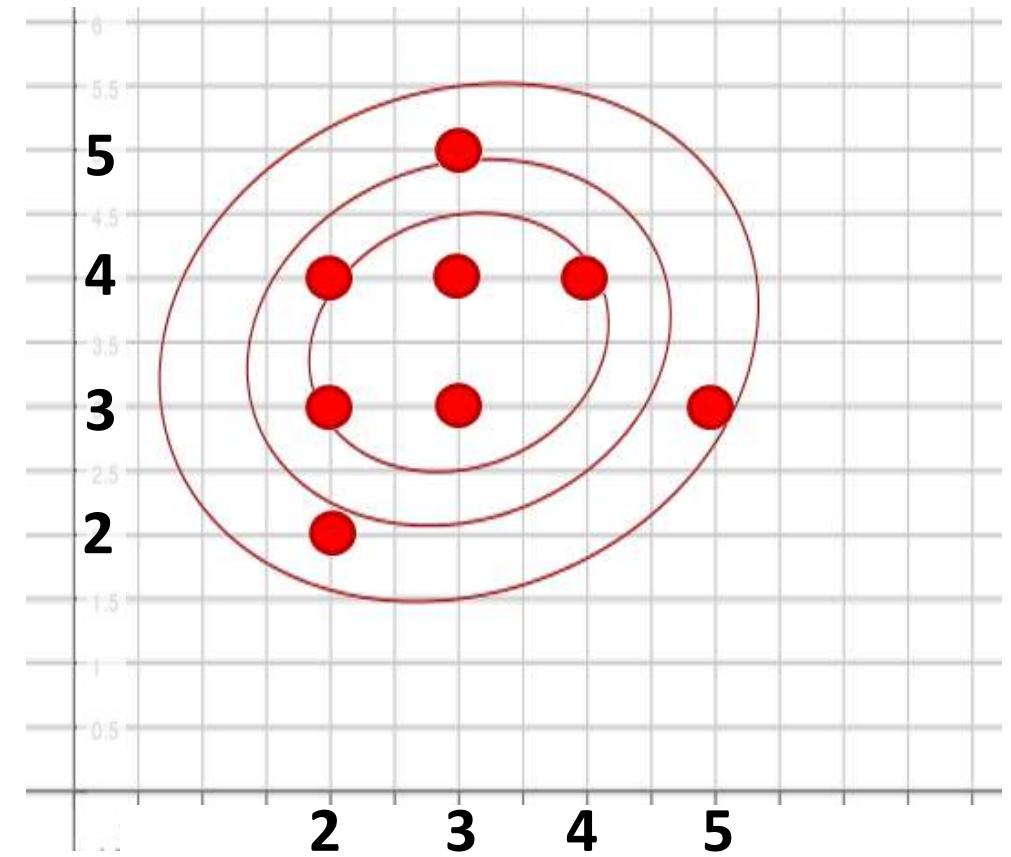


$$A^{[k]} = (B^{[k]})^{-1}$$
$$\mu_i^{[k]} = M.O.(x_i|k)$$
$$b_{ij}^{[k]} = M.O.(x_i - \mu_i^{[k]})(x_j - \mu_j^{[k]})$$

Gaussian classifier

Example

$$p(\bar{x} | \bullet) \approx \exp \left(-0.5 \cdot \sum_{i=1}^n \sum_{j=1}^n a_{i,j} (x_i - \mu_i)(x_j - \mu_j) \right)$$



Gaussian classifier

$$\mu_1 = MO(x_1 | \bullet) = 3$$

$$\mu_2 = MO(x_2 | \bullet) = 3,5$$

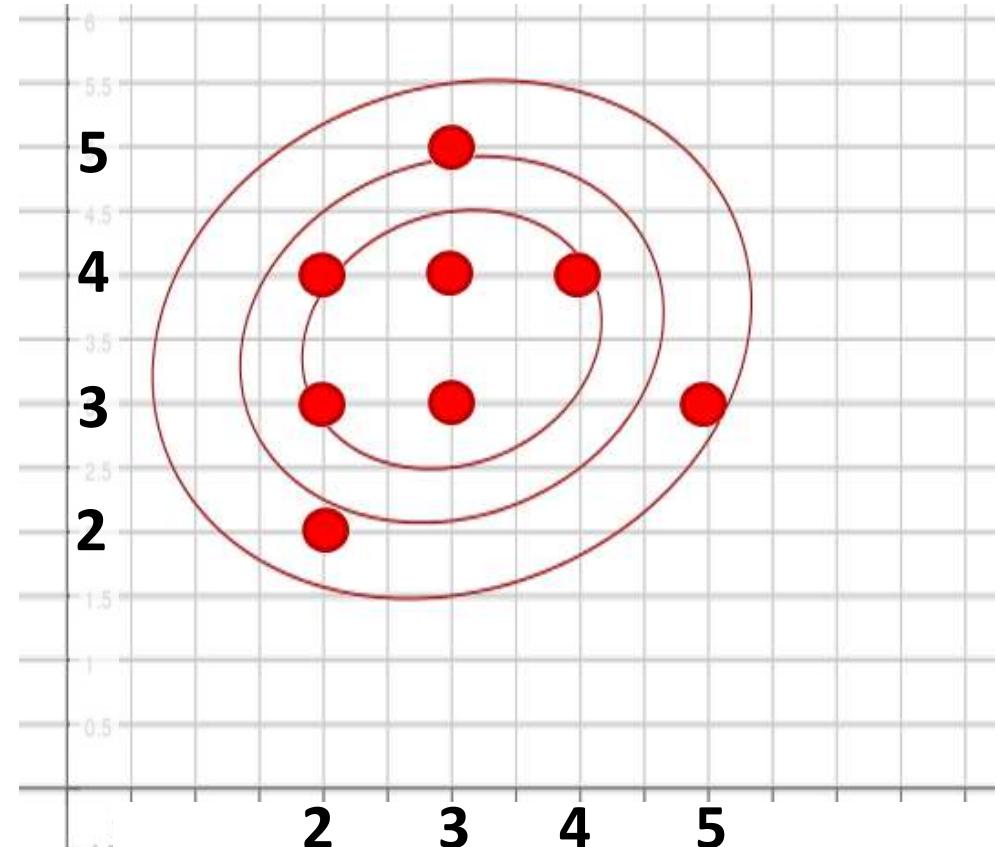
$$b_{11} = MO\left(x_1 - \mu_1 \right) \cdot \left(x_1 - \mu_1 \right) = \frac{(1+1+1+0+0+0+1+4)}{8} = 1$$

$$b_{22} = MO\left(x_2 - \mu_2 \right) \cdot \left(x_2 - \mu_2 \right) = \frac{3}{4}$$

$$b_{12} = b_{21} = MO\left(x_1 - \mu_1 \right) \cdot \left(x_2 - \mu_2 \right) = \frac{1}{8}$$

$$B = \begin{bmatrix} 1 & 1/8 \\ 1/8 & 3/4 \end{bmatrix}^{-1} = \frac{64}{47} \cdot \begin{bmatrix} 3/4 & -1/8 \\ -1/8 & 1 \end{bmatrix} = A$$

$$p(\bar{x} | \bullet) \approx \exp\left(\frac{64}{47} \cdot \left(-\frac{3}{4} \cdot (x_1 - 3)^2 + \frac{1}{4}(x_1 - 3)(x_2 - 3,5) - \frac{1}{1}(x_2 - 3,5)^2\right)\right)$$



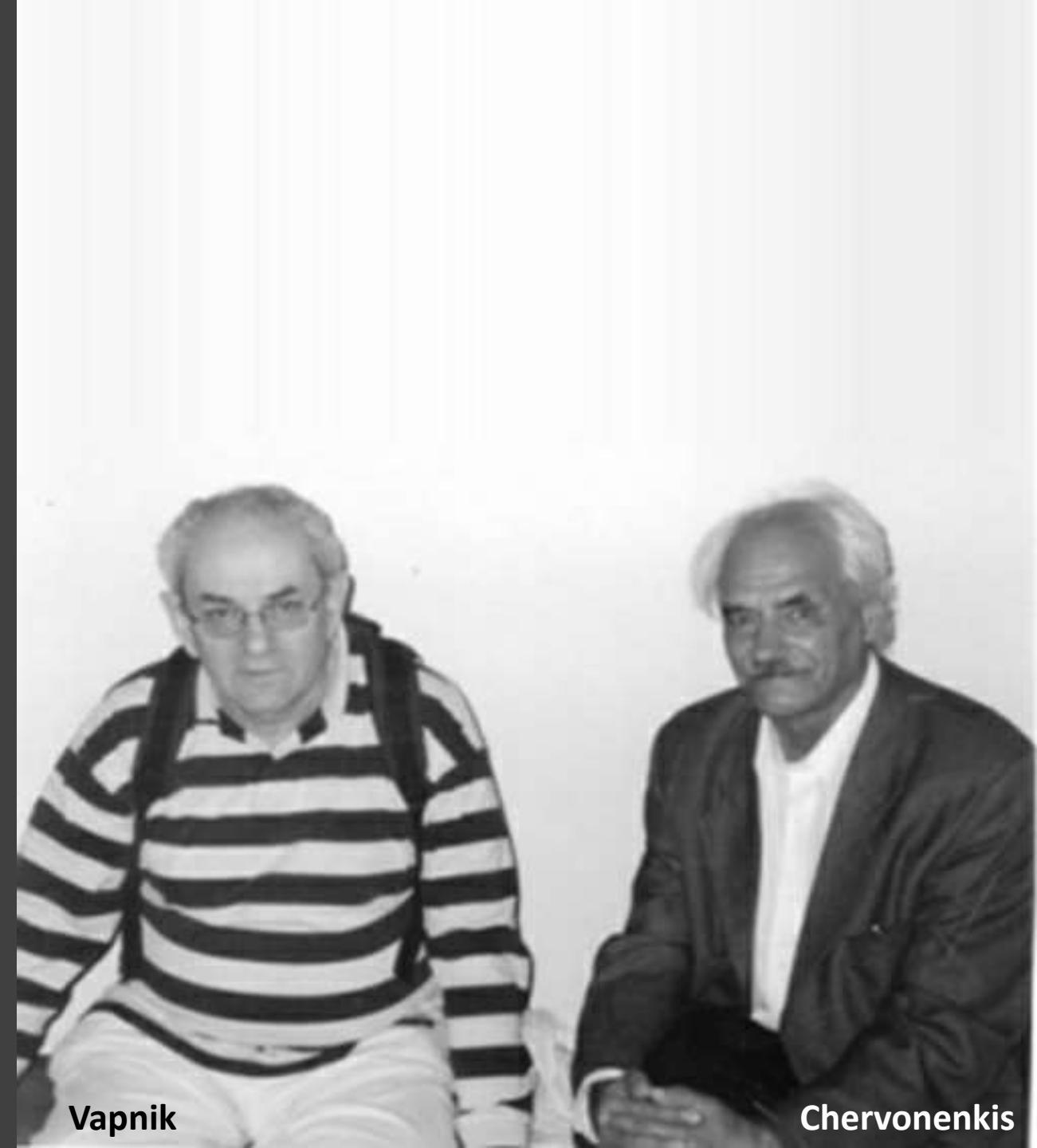
Gaussian classifier

Decision strategy

$$\log \frac{p(\bar{x}|k=1)}{p(\bar{x}|k=2)} = \frac{\sum_{i=1}^n \sum_{j=1}^n a_{i,j}^{[1]} (x_i - \mu_i^{[1]})(x_j - \mu_j^{[1]})}{\sum_{i=1}^n \sum_{j=1}^n a_{i,j}^{[2]} (x_i - \mu_i^{[2]})(x_j - \mu_j^{[2]})}$$

$$\sum_i \sum_j \alpha_{ij} \cdot x_i x_j + \sum_i \beta_i \cdot x_i \begin{matrix} \geq \\ \leq \end{matrix} \theta$$

1.6. Linear discrimination



Vapnik

Chervonenkis

Linear discrimination

Perceptron

input $\left\{ \begin{array}{l} X = \{x_1, x_2, \dots, x_r\} \\ X' = \{x'_1, x'_2, \dots, x'_s\} \end{array} \right.$

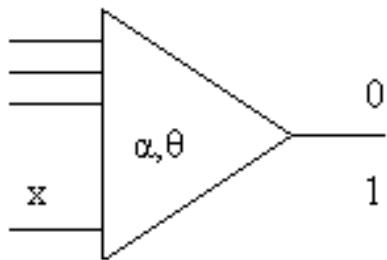
output $\left\{ \begin{array}{l} \forall x \in X \quad (x, \alpha) = \sum_{i=1}^n x_i \cdot \alpha_i > \theta \\ \forall x' \in X' \quad (x', \alpha) = \sum_{i=1}^n x'_i \cdot \alpha_i < \theta \end{array} \right.$



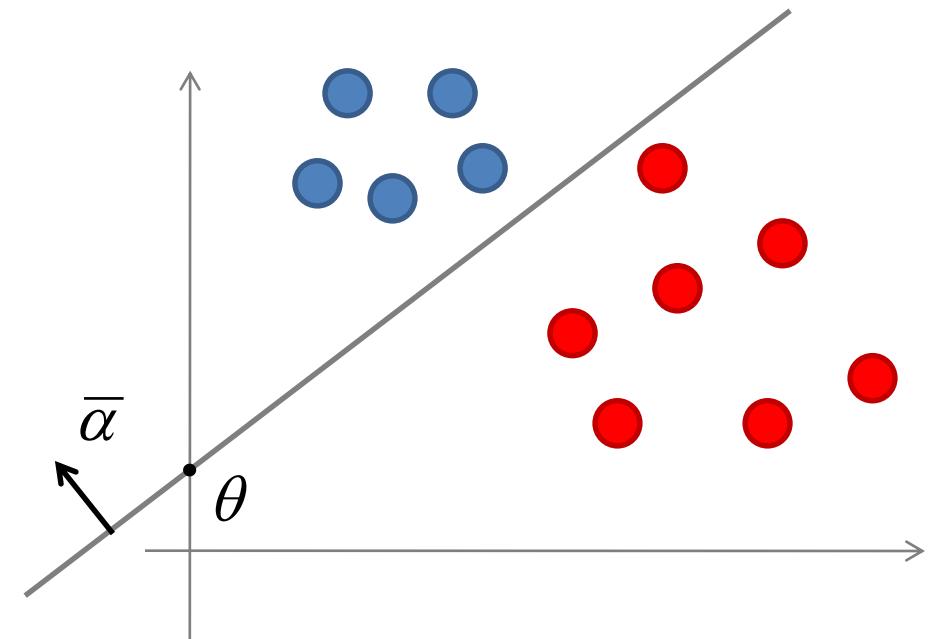
Rosenblatt

Linear discrimination

Perceptron



$$\text{output } \left\{ \begin{array}{ll} \forall x \in X & (x, \alpha) = \sum_{i=1}^n x_i \cdot \alpha_i > \theta \\ \alpha \in R^n & \\ \theta & \end{array} \right.$$
$$\forall x' \in X' \quad (x', \alpha) = \sum_{i=1}^n x'_i \cdot \alpha_i < \theta$$



Linear discrimination

Perceptron

$$\begin{cases} \forall x \in X \quad \sum_{i=1}^n x_i \cdot \alpha_i > \theta \\ \forall x' \in X' \quad \sum_{i=1}^n x'_i \cdot \alpha_i < \theta \end{cases} = \begin{cases} \forall x \in X \quad \sum_{i=1}^n x_i \cdot \alpha_i + 1 \cdot \alpha_{n+1} > 0 \\ \forall x' \in X' \quad \sum_{i=1}^n x'_i \cdot \alpha_i + 1 \cdot \alpha_{n+1} < 0 \end{cases}$$

Linear discrimination

Perceptron: tuning

$$t = 0$$

$$\alpha_t = 0$$

while (sets are not separated by hyperplane)

{

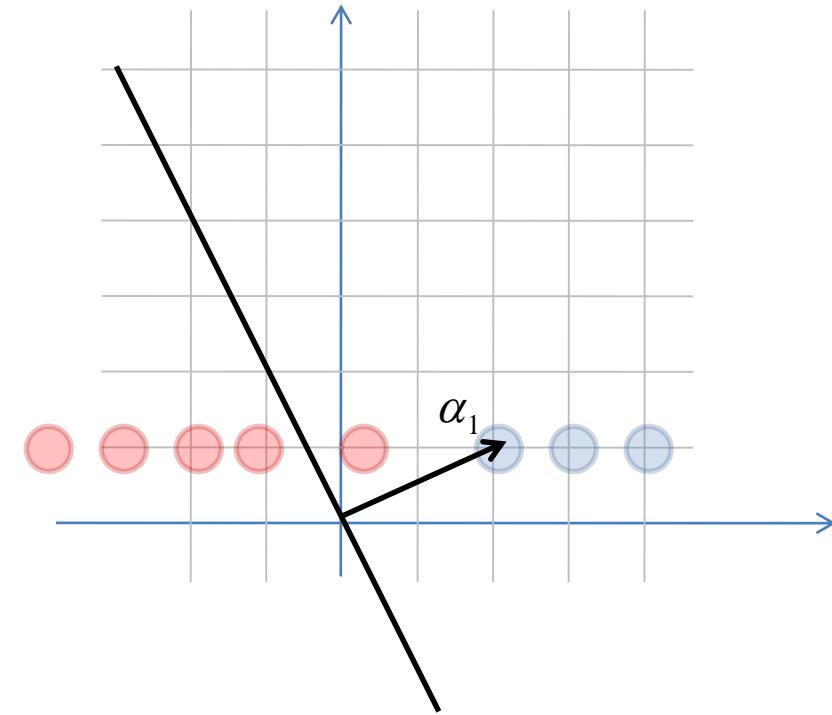
$$\text{if } (\exists x \in X | (x, \alpha_t) < 0) \quad \alpha_{t+1} = a_t + x;$$

$$\text{if } (\exists x' \in X' | (x', \alpha_t) > 0) \quad \alpha_{t+1} = a_t - x';$$

}

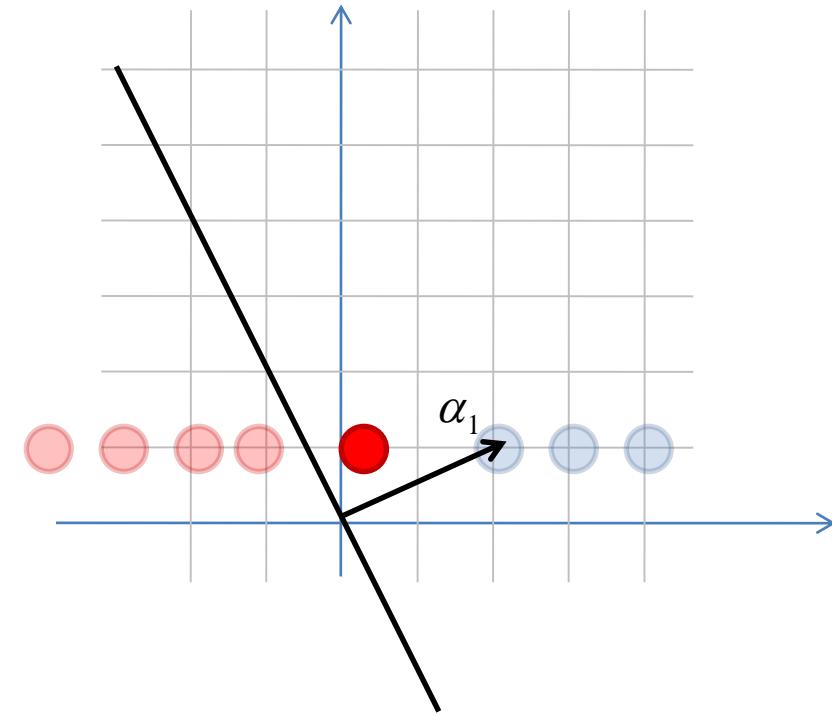
Linear discrimination

Perceptron: tuning



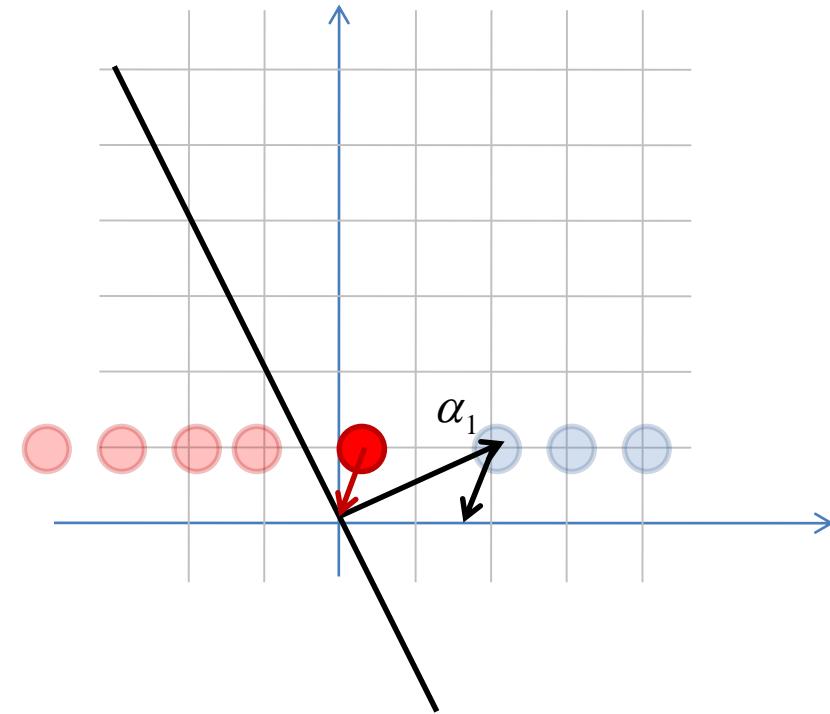
Linear discrimination

Perceptron: tuning



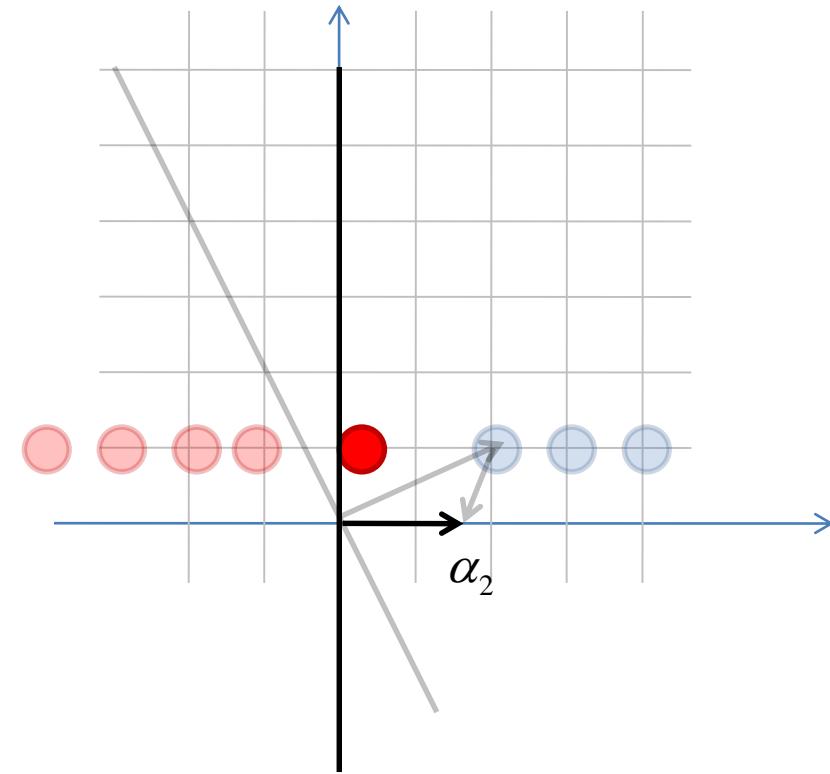
Linear discrimination

Perceptron: tuning



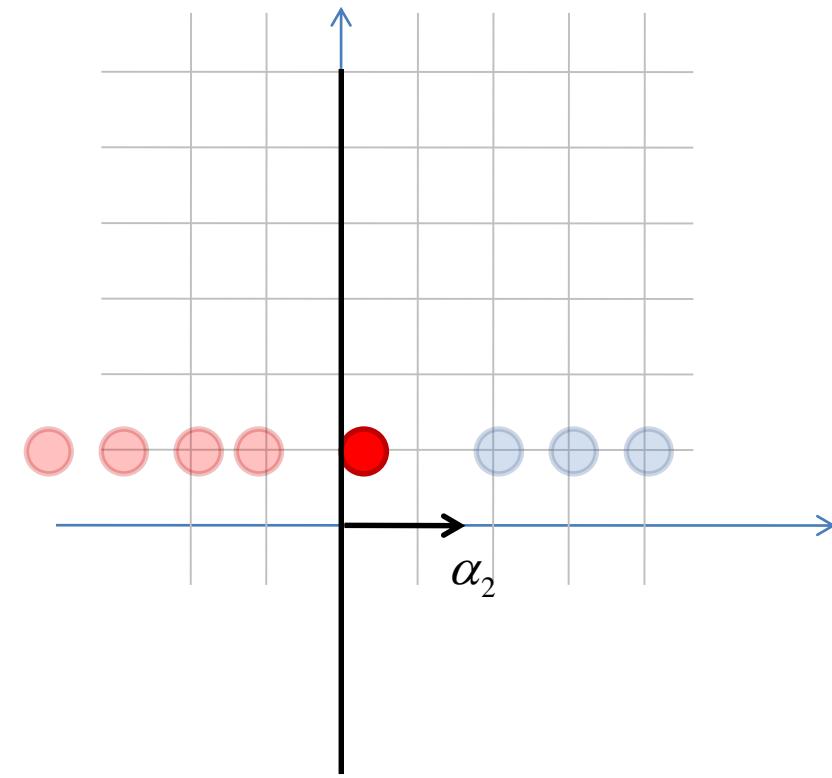
Linear discrimination

Perceptron: tuning



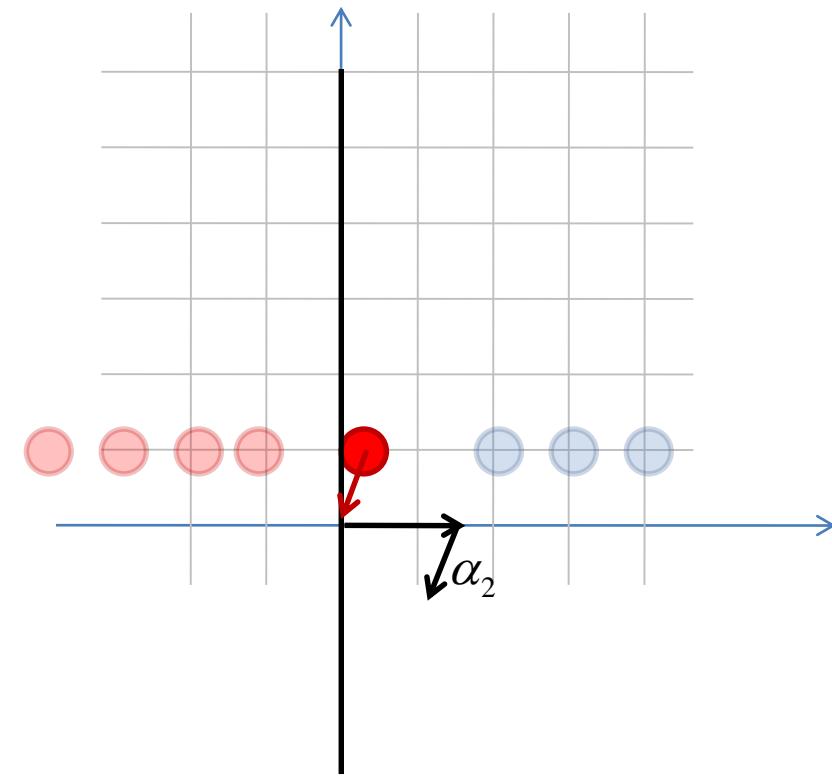
Linear discrimination

Perceptron: tuning



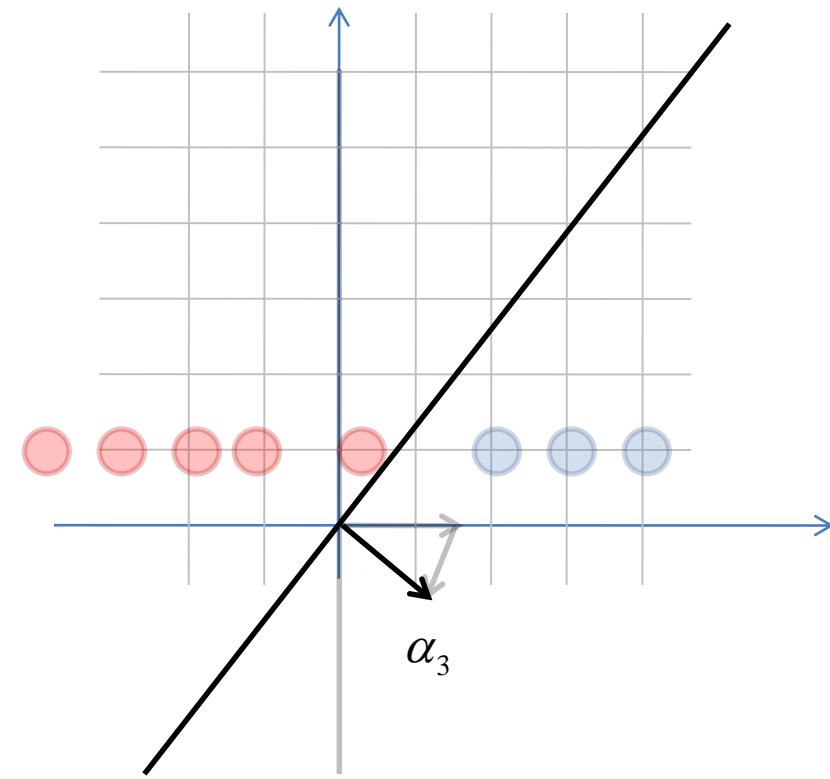
Linear discrimination

Perceptron: tuning



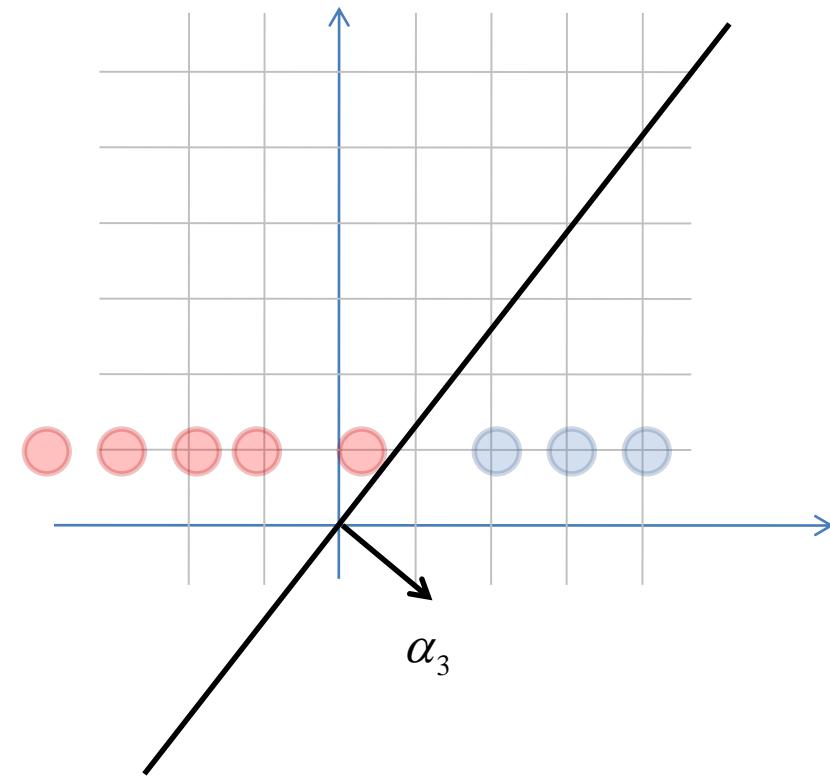
Linear discrimination

Perceptron: tuning



Linear discrimination

Perceptron: tuning



Linear discrimination

Perceptron: convergence

$$\max_{\substack{x \in X \\ x \in X'}} \|x\| = D \quad \|\alpha^*\| = 1$$

$$\begin{cases} \forall x \in X \quad (x, \alpha^*) \geq \varepsilon > 0 \\ \forall x' \in X' \quad (x', \alpha^*) \leq -\varepsilon < 0 \end{cases}$$

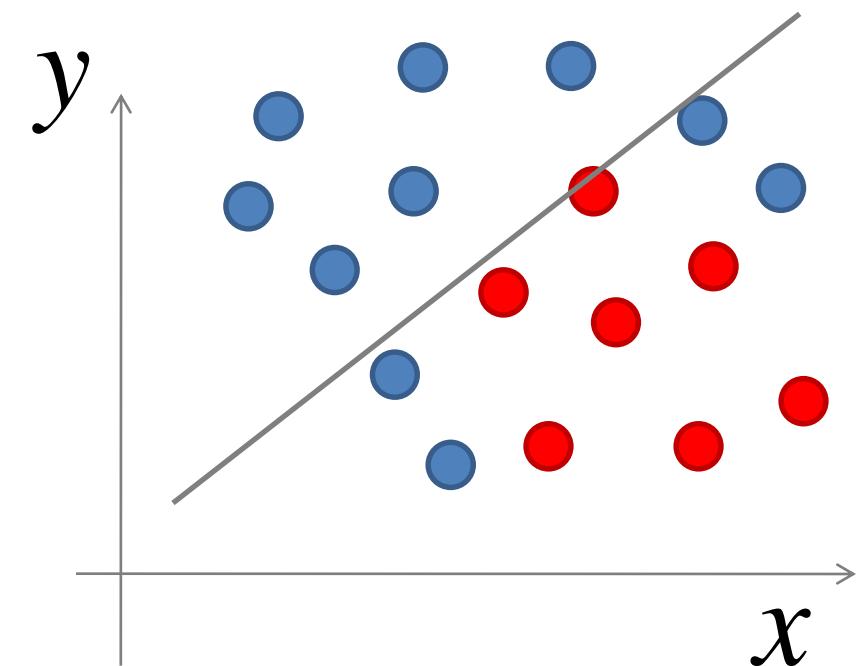
$$t \leq \frac{D^2}{\varepsilon}$$

Linear discrimination

Transformation of the feature space

$$Ax + By \geq C$$

a line in \mathbb{R}^2



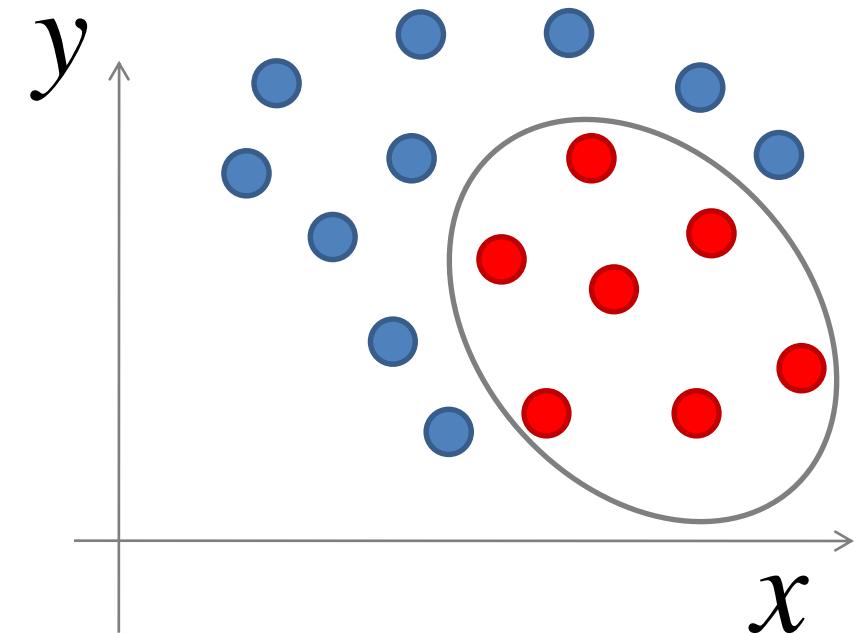
Linear discrimination

Transformation of the feature space

$$Ax^2 + Bxy + Cy^2 + Dx + Ey \geq F$$

hyperplane in \mathbb{R}^5

elliptic curve in \mathbb{R}^2



Linear discrimination

SVM: support vector machine

$$\begin{cases} \forall x \in X \quad (x, \alpha) = \sum_{i=1}^n x_i \cdot \alpha_i > \theta \\ \forall x' \in X' \quad (x', \alpha) = \sum_{i=1}^n x'_i \cdot \alpha_i < \theta \end{cases}$$

$$(\alpha, \alpha) \rightarrow \min$$

Separate by the
widest band

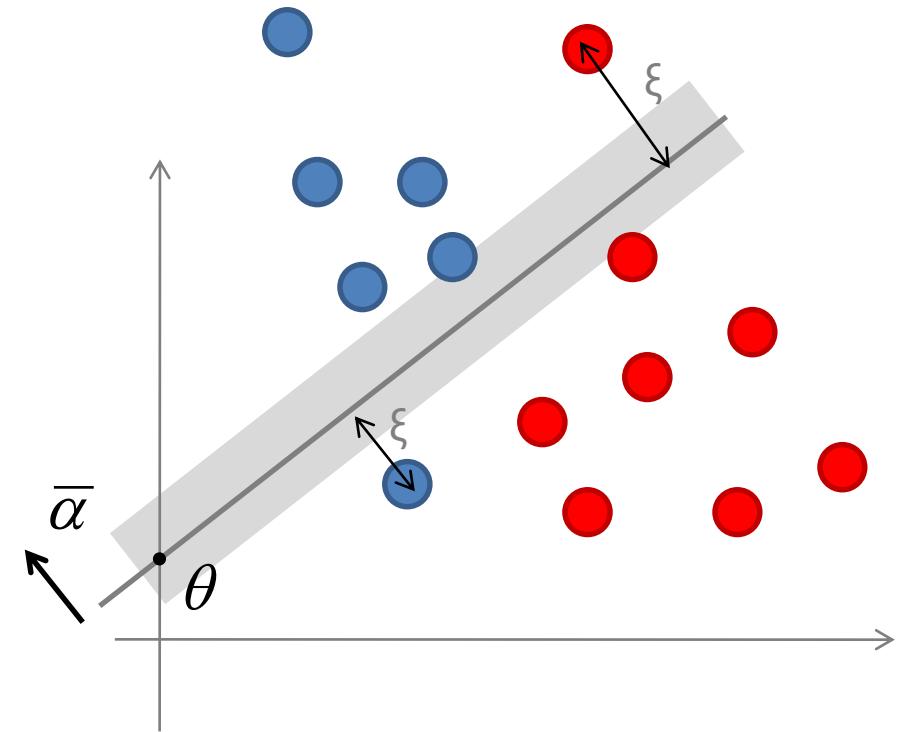
Linear discrimination

SVM: support vector machine

$$\begin{cases} \forall x \in X \quad (x, \alpha) = \sum_{i=1}^n x_i \cdot \alpha_i > \theta - \xi(x) \\ \forall x' \in X' \quad (x', \alpha) = \sum_{i=1}^n x'_i \cdot \alpha_i < \theta + \xi(x') \end{cases}$$

$$(\alpha, \alpha) + const \cdot \sum_{X, X'} \xi(x) \rightarrow \min$$

Separate by the
widest band



1.7. Adaboost



Robert Schapire



Yoav Freund

Adaboost

Definitions

$x_1, x_2, x_i, \dots, x_N$ Training features

$k_1, k_2, k_i, \dots, k_N = \{-1, 1\}$ Training targets

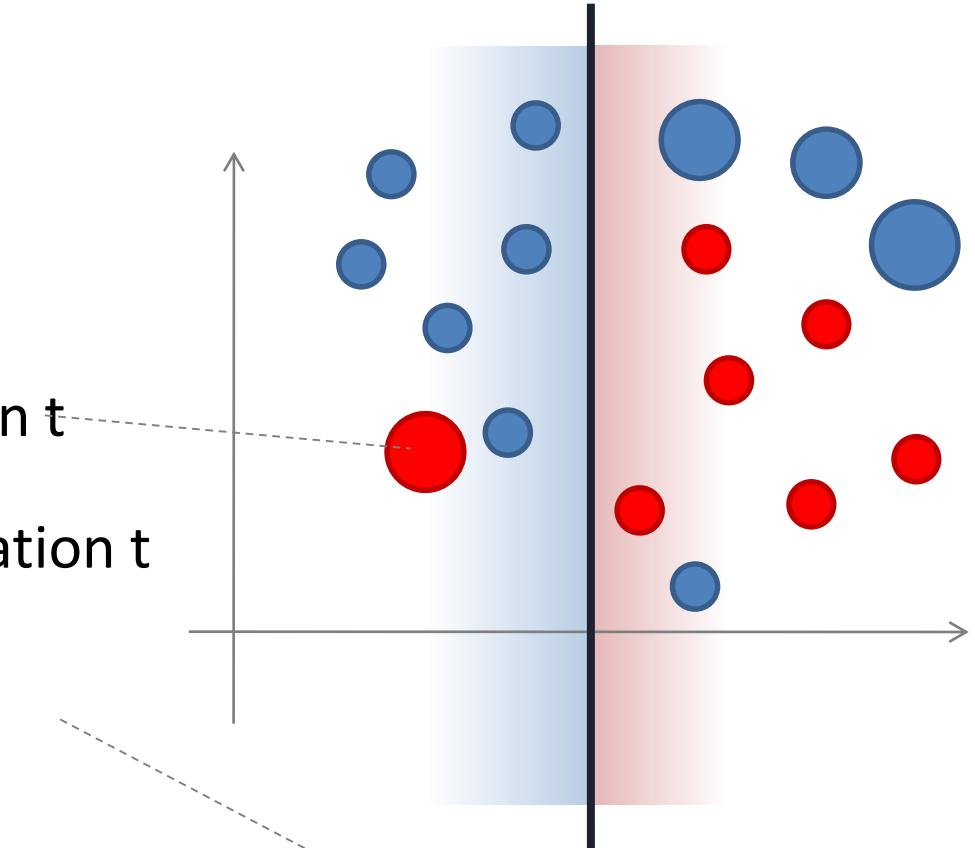
$t = 1, 2, 3, \dots$ Iteration number

$W_t(x_i)$ Weight of the element on iteration t

α_t Weight of a weak classifier on iteration t

$h_t(x)$ Weak classifier on iteration t

$$k(x) \approx \sum_t \alpha_t \cdot h_t(x)$$



$$h(x) = \{x > \theta ? 0 : 1\}$$

Adaboost

Algorithm

Weak classifier to minimize
the weighted error

$$h_t(x) = \arg \min_h \sum_i W_t(i) \cdot [k_i \neq h(x_i)]$$

Update weights

$$W_{t+1}(i) = W_t(i) \cdot \exp\{-\alpha_t \cdot k_i \cdot h_t(x_i)\} = W_t(i) \times \begin{cases} \sqrt{\frac{1 - r_t}{1 + r_t}} & \text{if } h_t(x) = k \\ \sqrt{\frac{1 + r_t}{1 - r_t}} & \text{if } h_t(x) \neq k \end{cases}$$

Calculate the weight od classifier

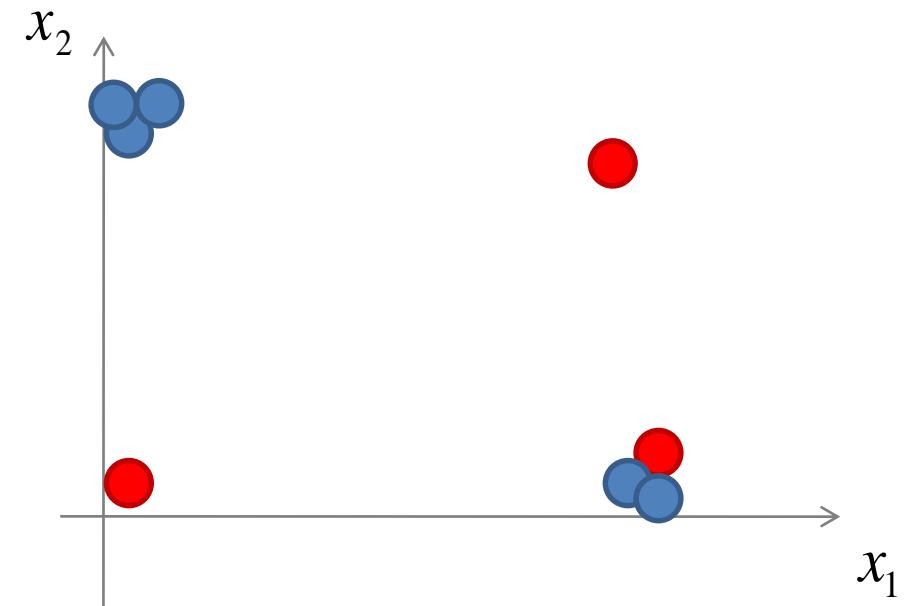
$$\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right)$$

$$r_t = \sum_i W_t(i) \cdot h(x_i) \cdot k_i / \sum_i W_t(i)$$

Adaboost

Example: step 1

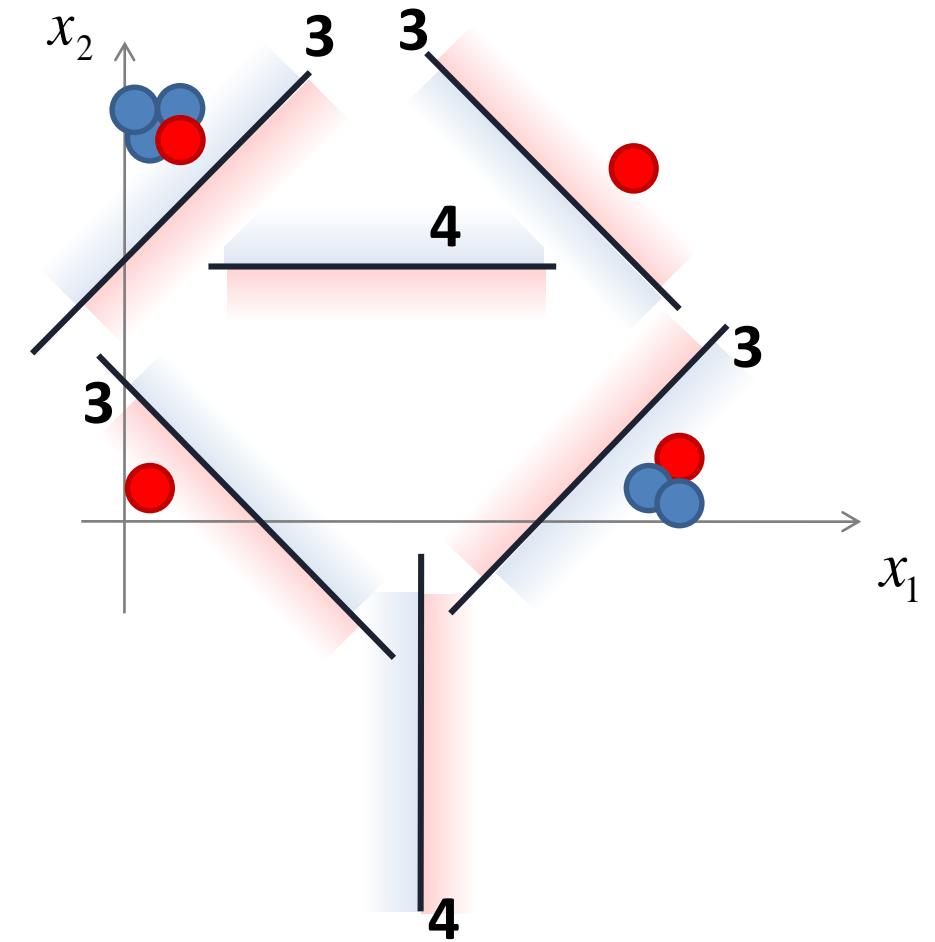
$$h_{t=1}(\bar{x}) =$$



Adaboost

Example: step 1

$$h_{t=1}(\bar{x}) =$$

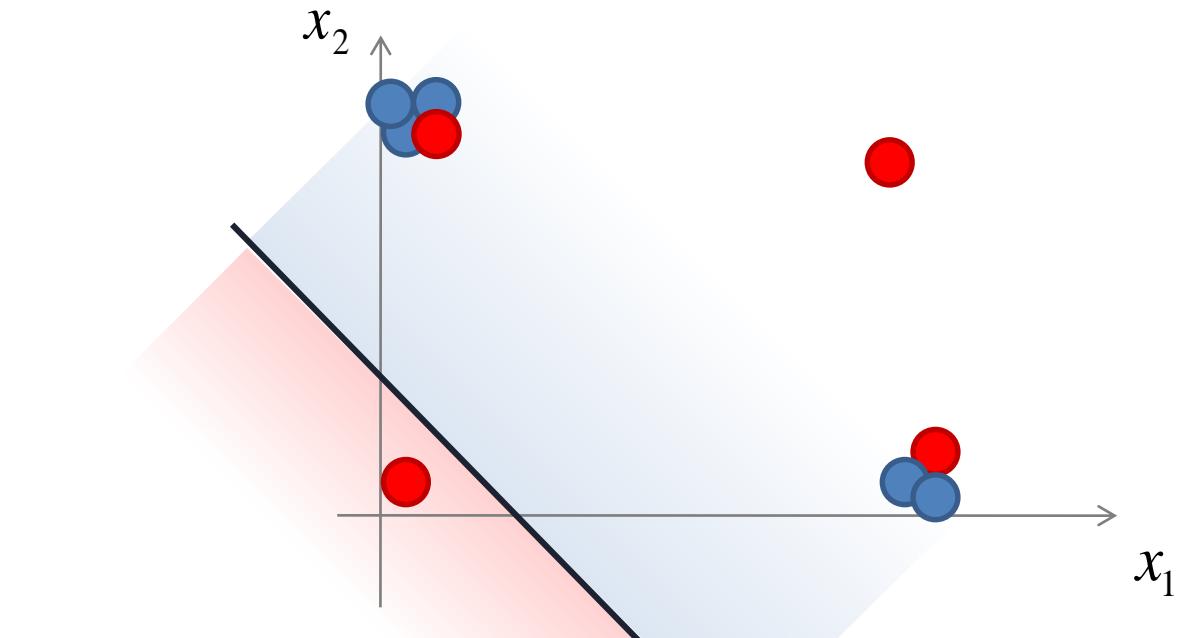


Adaboost

Example: step 1

$$h_{t=1}(\bar{x}) = \{x_1 + x_2 < 0.5? \text{ } \textcolor{red}{\bullet} : \textcolor{blue}{\bullet}\}$$

$$r_{t=1} = 0.33 \quad \alpha_{t=1} = 0.34$$



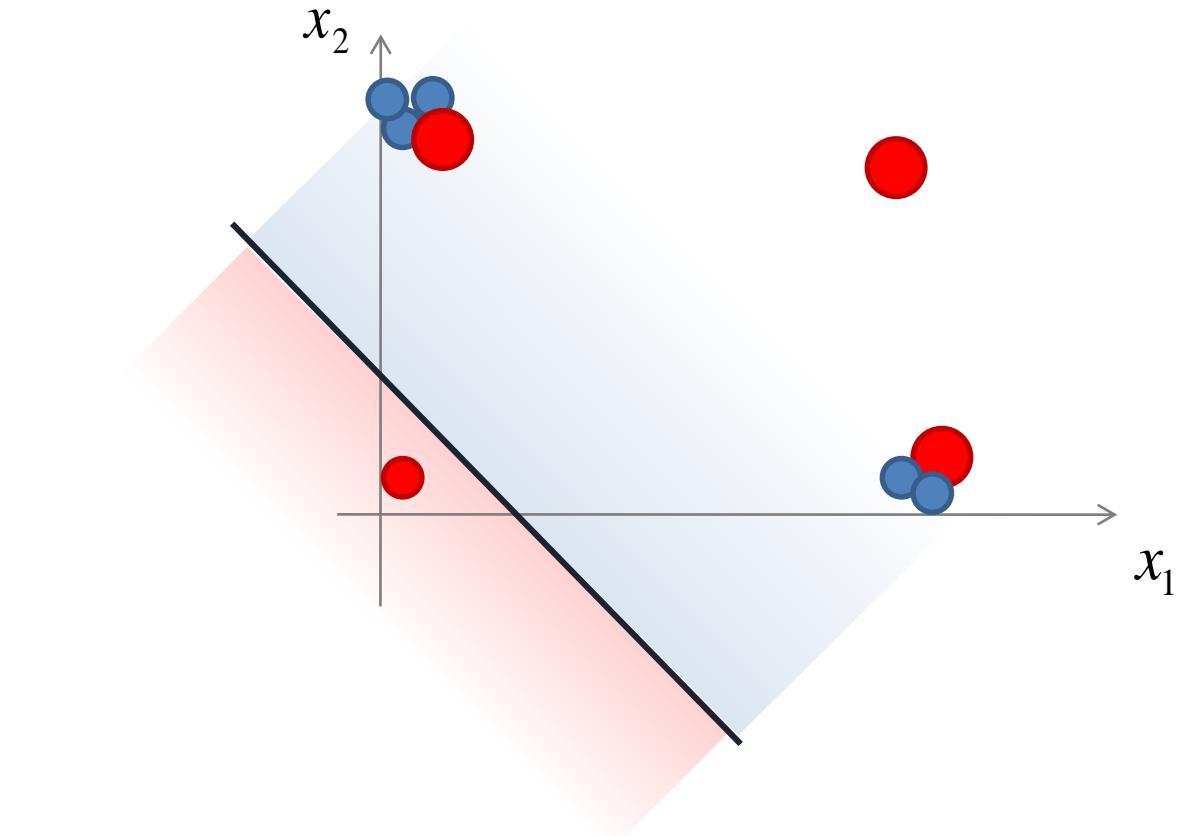
Adaboost

Example: step 1

$$h_{t=1}(\bar{x}) = \{x_1 + x_2 < 0.5? \text{ } \textcolor{red}{\bullet} : \textcolor{blue}{\bullet}\}$$

$$r_{t=1} = 0.33 \quad \alpha_{t=1} = 0.34$$

$$W_{t=2}(i) = W_1(i) \cdot \exp\{-\alpha_1 \cdot k_i \cdot h_1(x_i)\}$$



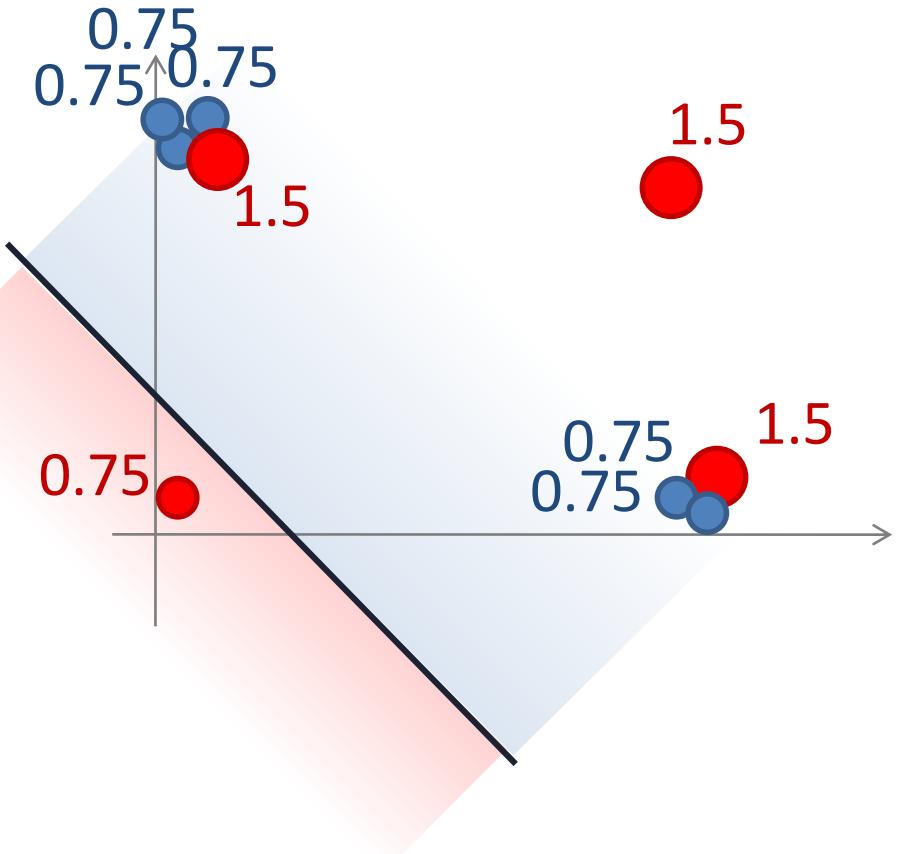
Adaboost

Example: step 1

$$h_{t=1}(\bar{x}) = \{x_1 + x_2 < 0.5? \text{ } \textcolor{red}{\bullet} : \textcolor{blue}{\bullet}\}$$

$$r_{t=1} = 0.33 \quad \alpha_{t=1} = 0.34$$

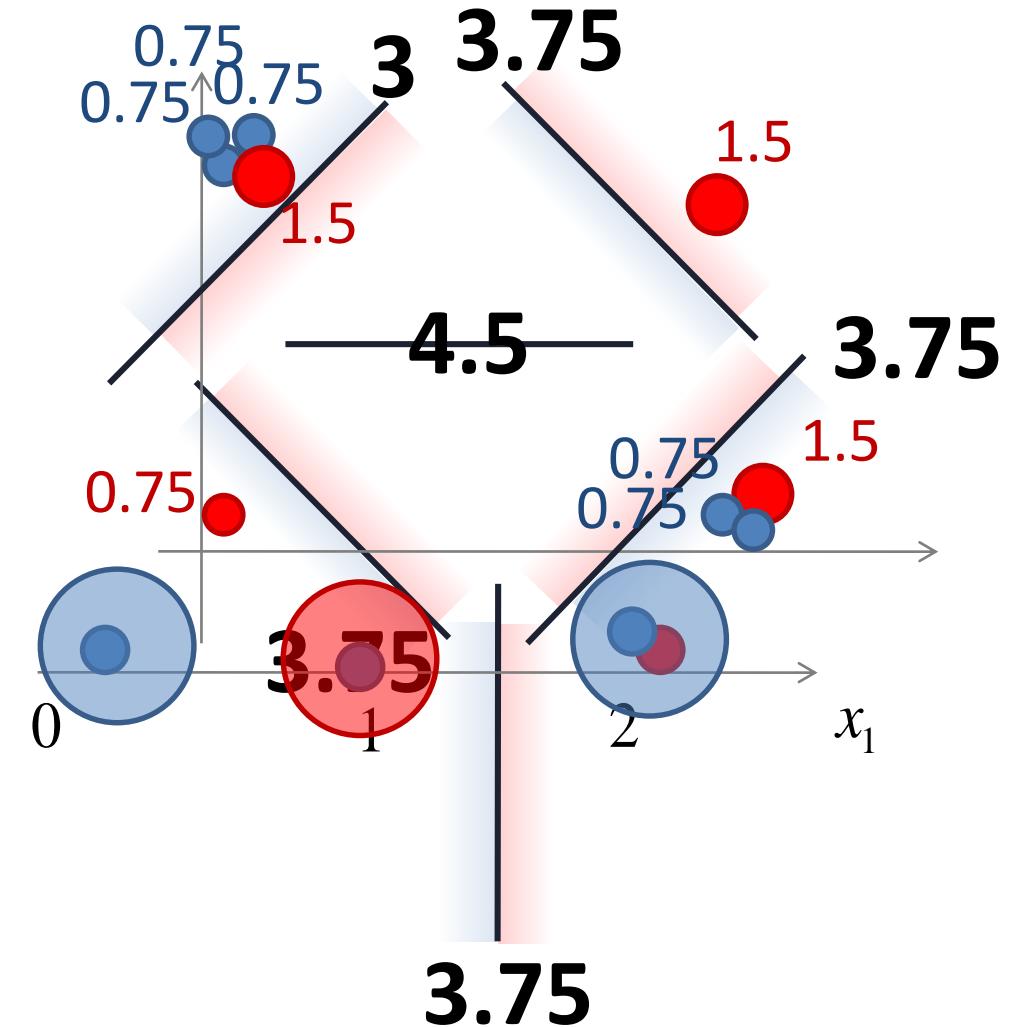
$$W_{t=2}(i) = W_1(i) \cdot \exp\{-\alpha_1 \cdot k_i \cdot h_1(x_i)\}$$



Adaboost

Example: step 2

$$h_{t=2}(\bar{x}) =$$

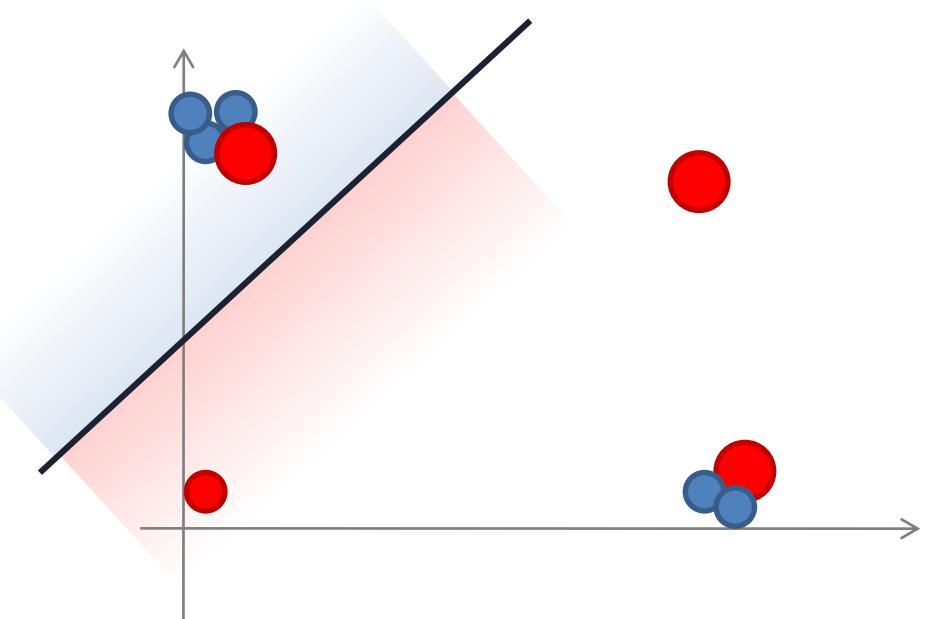


Adaboost

Example: step 2

$$h_{t=2}(\bar{x}) = \{x_1 - x_2 < -0.5? \text{ } \textcolor{red}{\bullet} : \textcolor{blue}{\bullet} \}$$

$$r_{t=2} = 0.33 \quad \alpha_{t=2} = 0.34$$



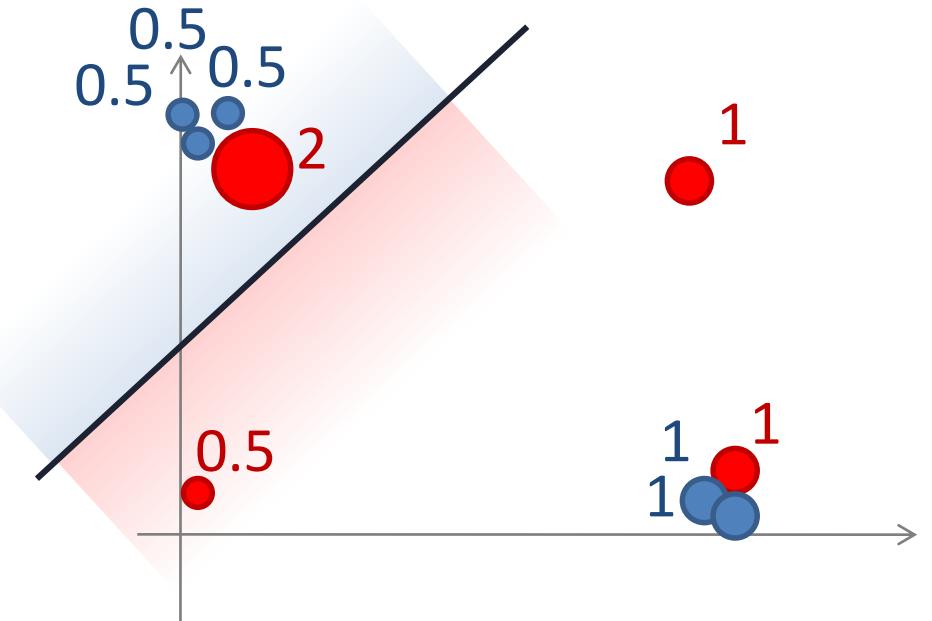
Adaboost

Example: step 2

$$h_{t=2}(\bar{x}) = \{x_1 - x_2 < -0.5? \text{ } \textcolor{red}{\bullet} : \textcolor{blue}{\bullet}\}$$

$$r_{t=2} = 0.33 \quad \alpha_{t=2} = 0.34$$

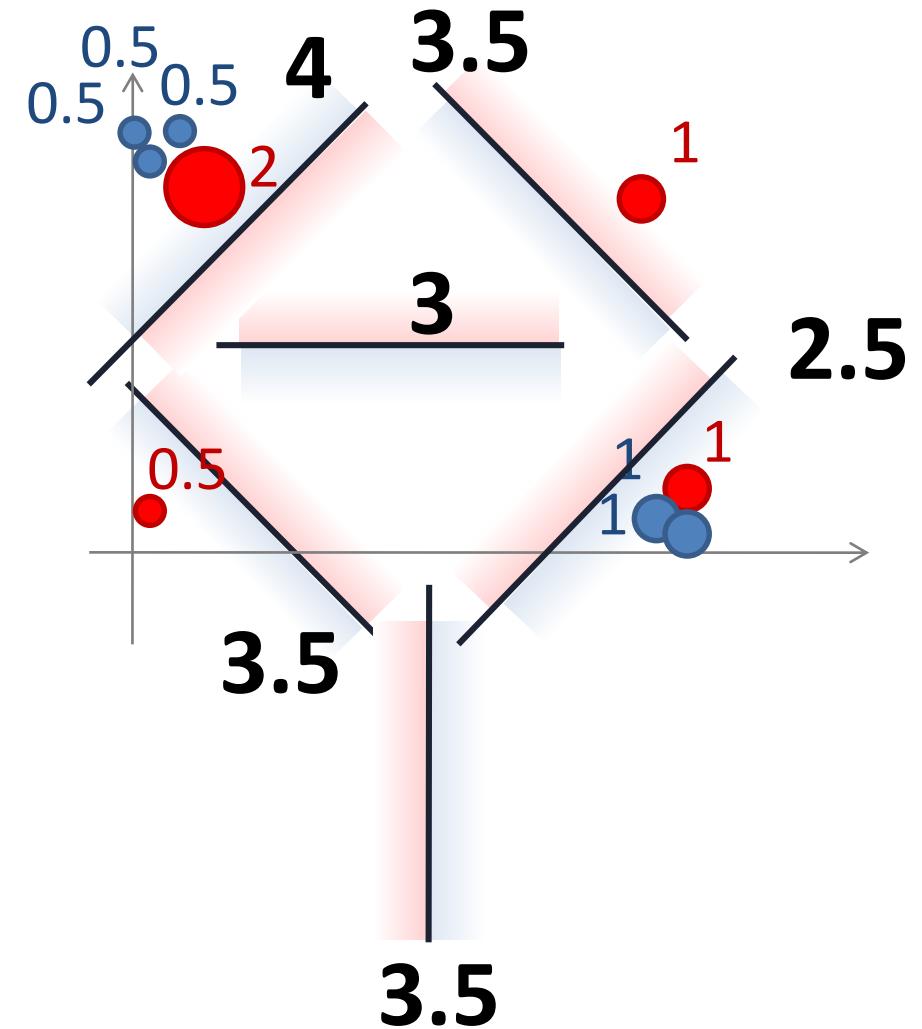
$$W_{t=2}(i) = W_1(i) \cdot \exp\{-\alpha_1 \cdot k_i \cdot h_1(x_i)\}$$



Adaboost

Example: step 3

$$h_{t=3}(\bar{x}) =$$

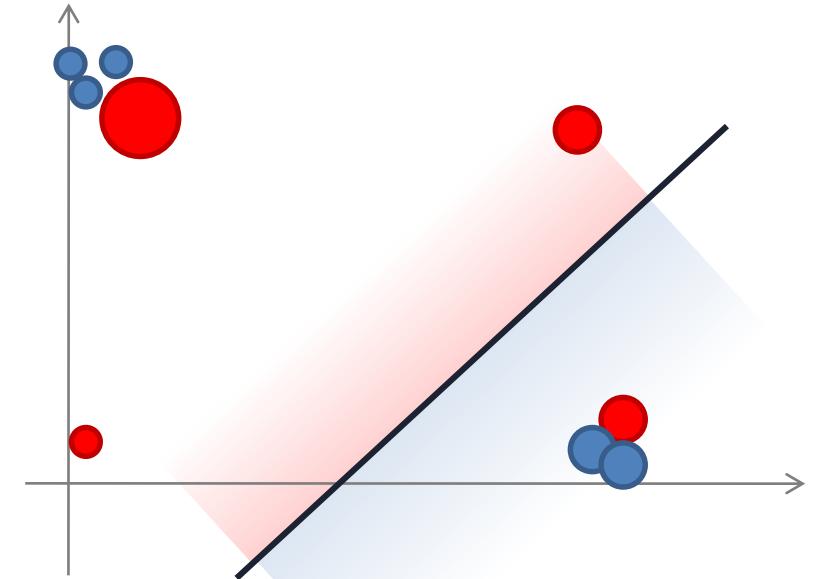


Adaboost

Example: step 3

$$h_{t=3}(\bar{x}) = \{x_1 - x_2 > 0.5? \quad \text{red circle} : \text{blue circle}\}$$

$$r_{t=3} = 0.38 \quad \alpha_{t=3} = 0.39$$



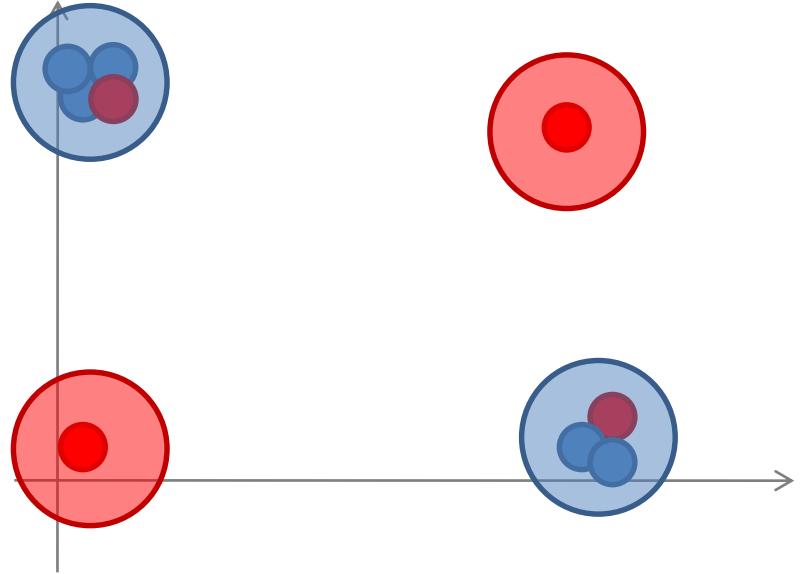
Adaboost

Example: result

$$k(x) \cong \sum_t \alpha_t \cdot h_t(x)$$

$$h_{t=3}$$

$$h_{t=2}$$



$$x \quad (0,0) \quad (1,0) \quad (1,0) \quad (1,0) \quad (0,1) \quad (0,1) \quad (0,1) \quad (0,1) \quad (0,1) \quad (1,1)$$

$$\alpha_1 = 0.35$$

$$h_1 \quad \textcolor{red}{1} \quad -1 \quad -1$$

$$\alpha_2 = 0.35$$

$$h_2 \quad \textcolor{red}{1} \quad \textcolor{red}{1} \quad \textcolor{red}{1} \quad \textcolor{red}{1} \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad 1$$

$$\alpha_3 = 0.39$$

$$h_3 \quad \textcolor{red}{1} \quad -1 \quad -1 \quad -1 \quad \textcolor{red}{1} \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$$

$$k(x) \cong \sum_t \alpha_t \cdot h_t(x)$$

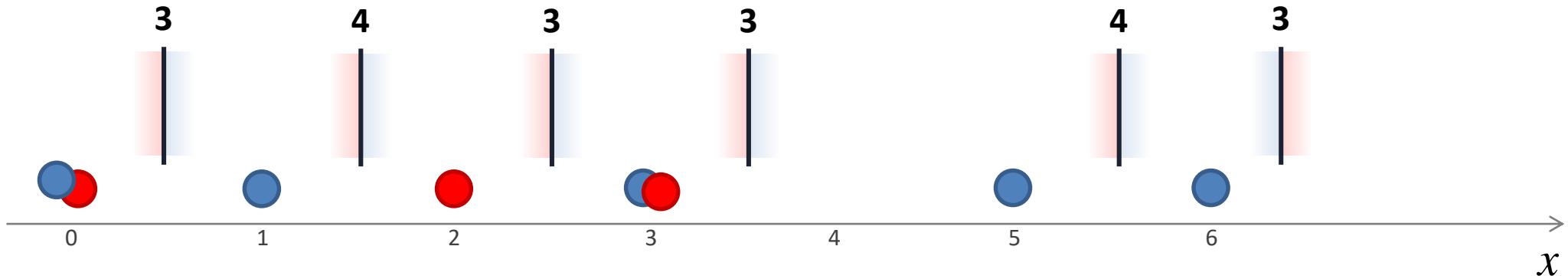
$$1 \quad -1 \quad 1$$

Adaboost

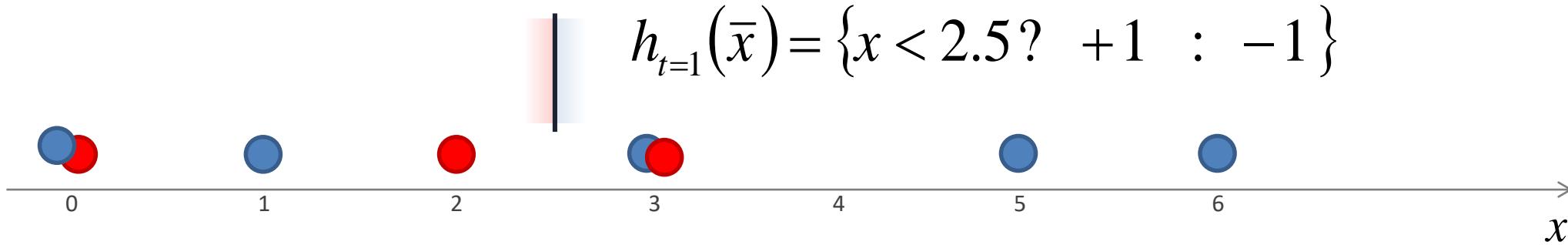
Another example



Adaboost



Adaboost

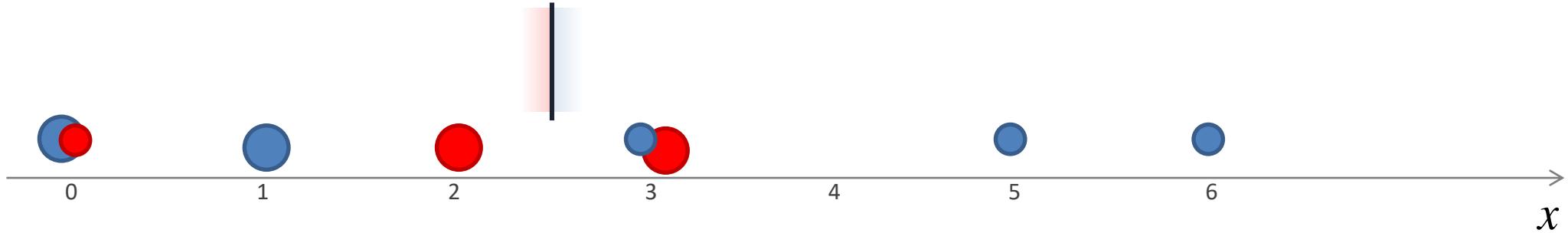


$$r_{t=1} = \frac{1}{\sum_i W_1(i)} \sum_i W_1(i) \cdot h_1(x_i) \cdot k_i = \frac{2}{8}$$

$$\alpha_t = \frac{1}{2} \log \left(\frac{1+r_t}{1-r_t} \right)$$

$$W_{t=2}(i) = W_1(i) \cdot \exp \{-\alpha_1 \cdot k_i \cdot h_1(x_i)\} = W_1(i) \times \begin{cases} \sqrt{\frac{1-r_t}{1+r_t}} & \text{if } h_1(x)=k \\ \sqrt{\frac{1+r_t}{1-r_t}} & \text{if } h_1(x) \neq k \end{cases}$$

Adaboost

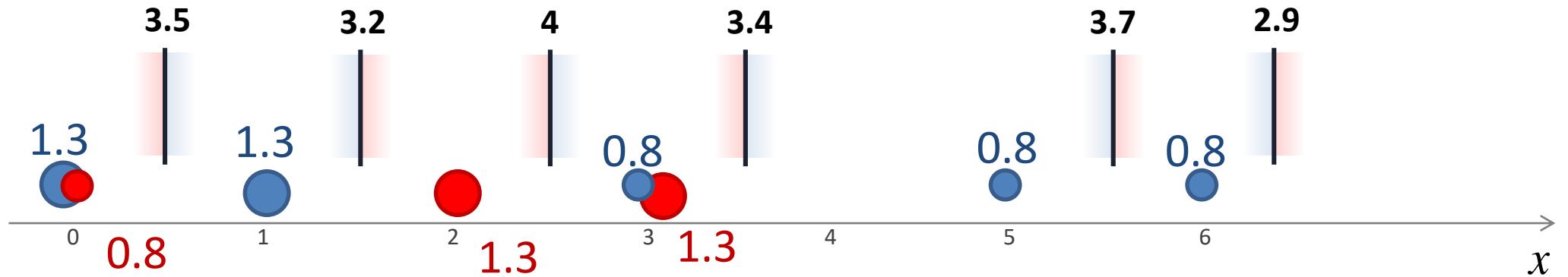


$$r_{t=1} = \frac{1}{\sum_i W_1(i)} \sum_i W_1(i) \cdot h_1(x_i) \cdot k_i = \frac{2}{8}$$

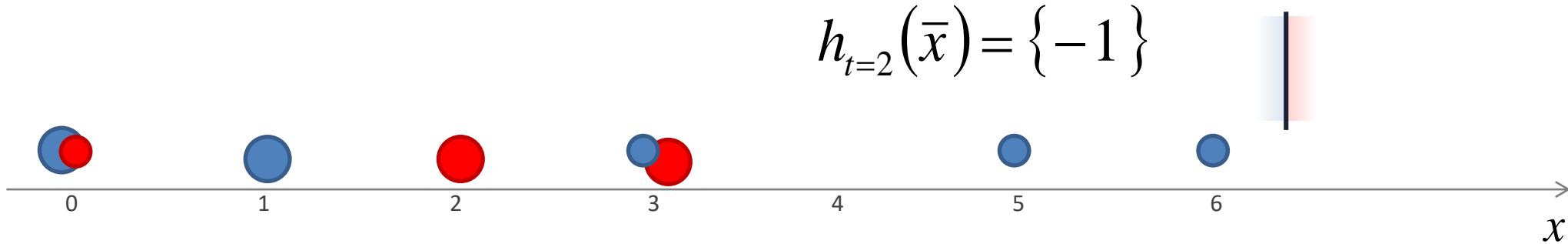
$$\alpha_t = \frac{1}{2} \log \left(\frac{1+r_t}{1-r_t} \right)$$

$$W_{t=2}(i) = W_1(i) \cdot \exp \left\{ -\alpha_1 \cdot k_i \cdot h_1(x_i) \right\} = W_1(i) \times \begin{cases} \sqrt{\frac{3}{5}} & \text{if } h_1(x) = k \\ \sqrt{\frac{5}{3}} & \text{if } h_1(x) \neq k \end{cases}$$

Adaboost



Adaboost

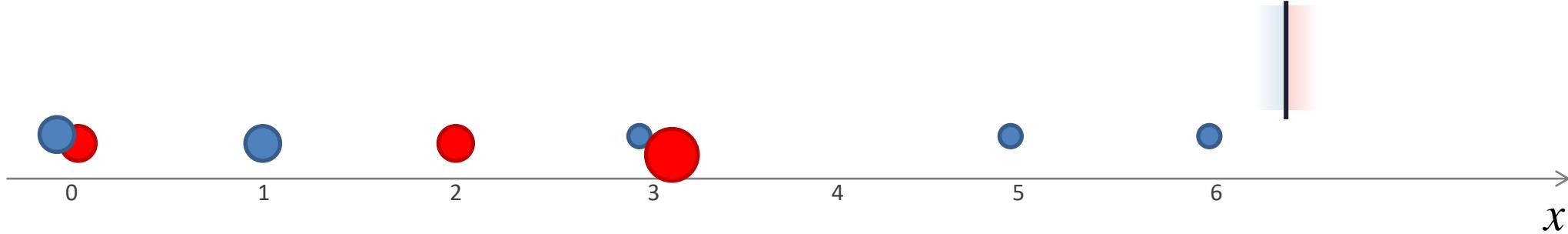


$$W_{t=3}(i) = W_2(i) \times \begin{cases} \sqrt{\frac{1 - r_t}{1 + r_t}} & \text{if } h_2(x) = k \\ \sqrt{\frac{1 + r_t}{1 - r_t}} & \text{if } h_2(x) \neq k \end{cases}$$

$$r_{t=2} = \frac{1}{\sum_i W_2(i)} \sum_i W_2(i) \cdot h_2(x_i) \cdot k_i = 2.9$$

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right)$$

Adaboost

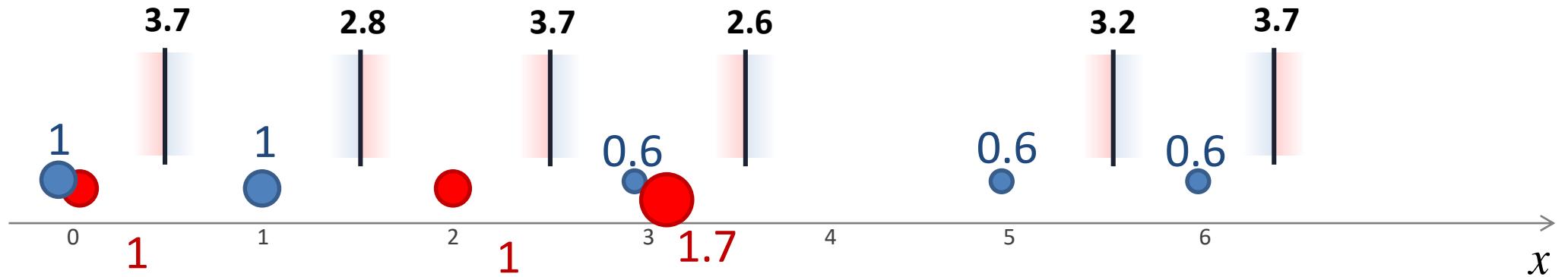


$$W_{t=3}(i) = W_2(i) \times \begin{cases} \sqrt{\frac{1 - r_t}{1 + r_t}} & \text{if } h_2(x) = k \\ \sqrt{\frac{1 + r_t}{1 - r_t}} & \text{if } h_2(x) \neq k \end{cases}$$

$$r_{t=2} = \frac{1}{\sum_i W_2(i)} \sum_i W_2(i) \cdot h_2(x_i) \cdot k_i = 2.9$$

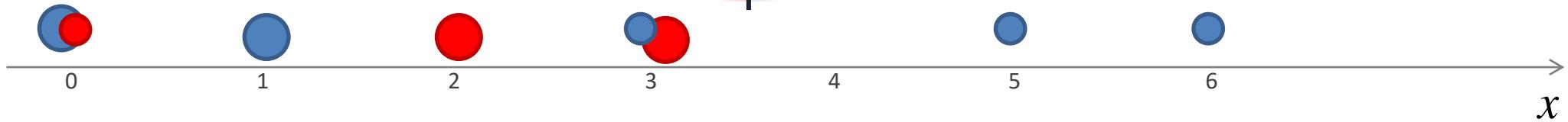
$$\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right)$$

Adaboost



Adaboost

$$h_{t=3}(\bar{x}) = \begin{cases} x < 3.5? & +1 \\ & -1 \end{cases}$$



$$\alpha_1 = 0.25$$

$$\alpha_2 = 0.27$$

$$\alpha_3 = 0.33$$

$$1$$

$$1$$

$$1$$

$$-1$$

$$-1$$

$$-1$$

$$1$$

$$1$$

$$1$$

$$-1$$

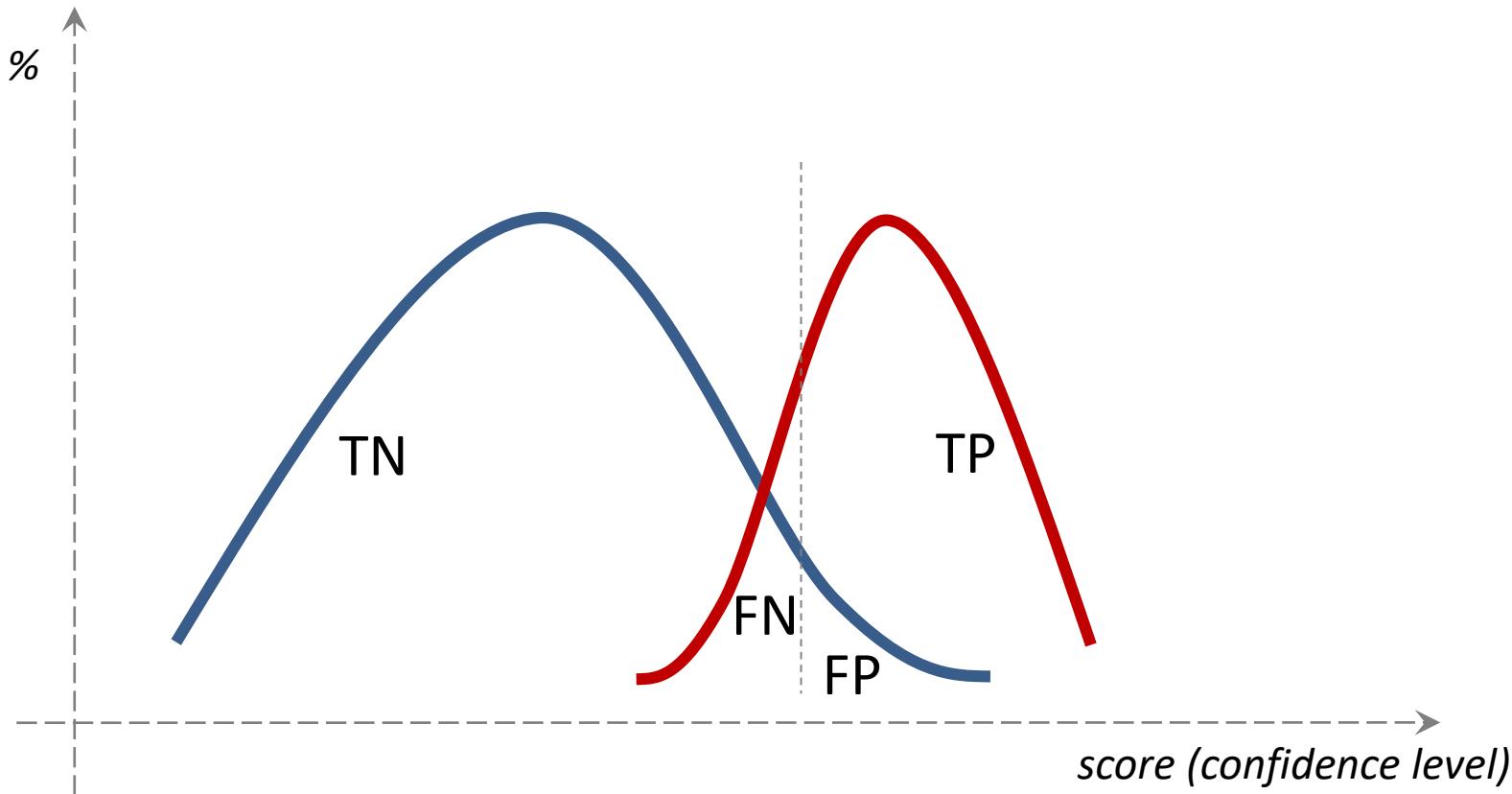
$$-1$$

$$-1$$

1.8. ROC curve

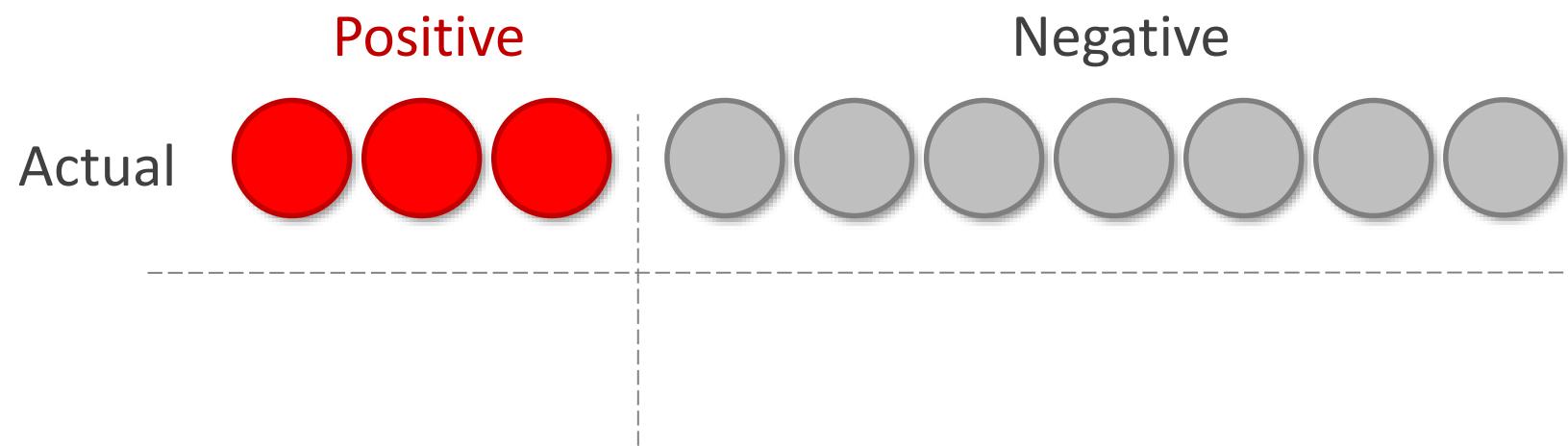


ROC curve



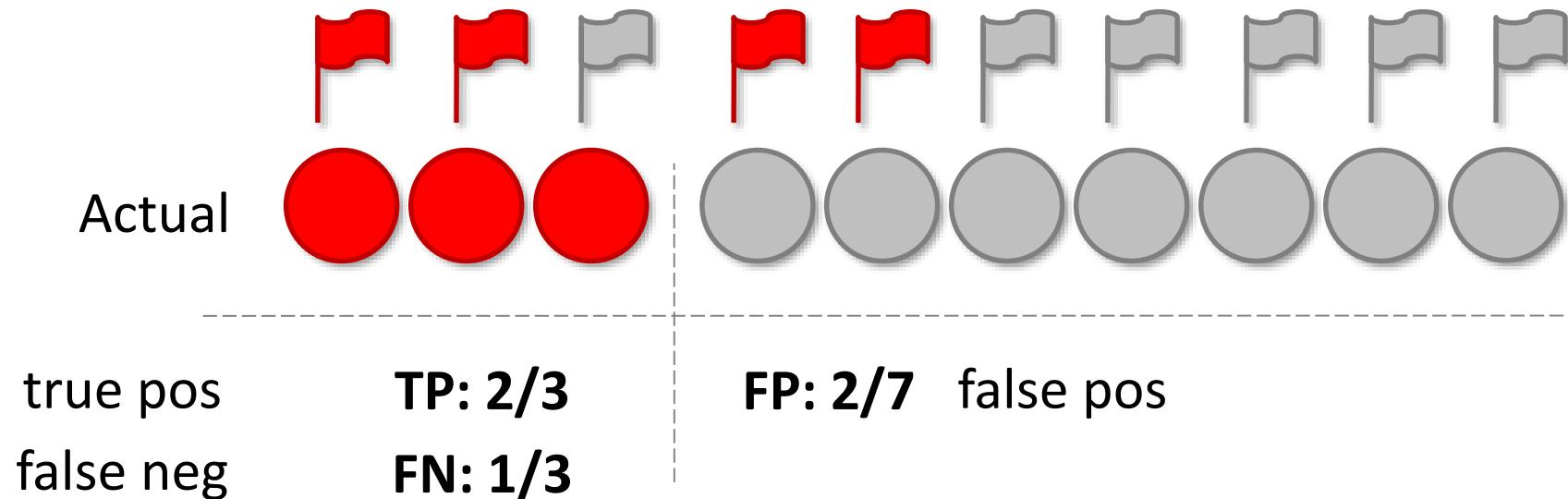
ROC curve

Error types

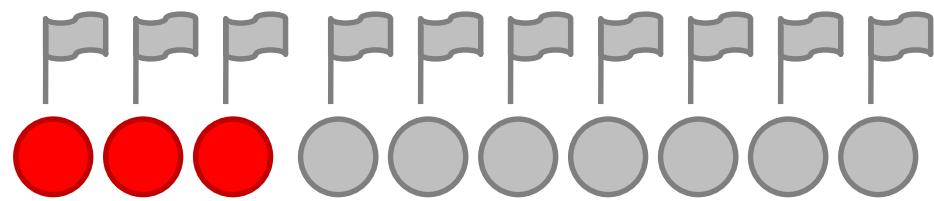


ROC curve

Error types



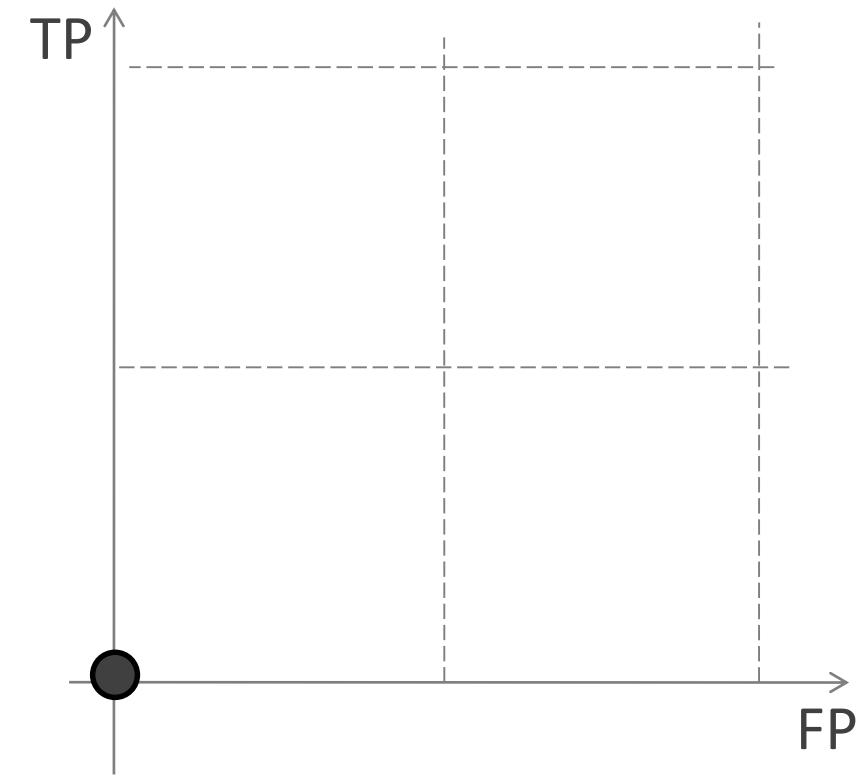
ROC curve



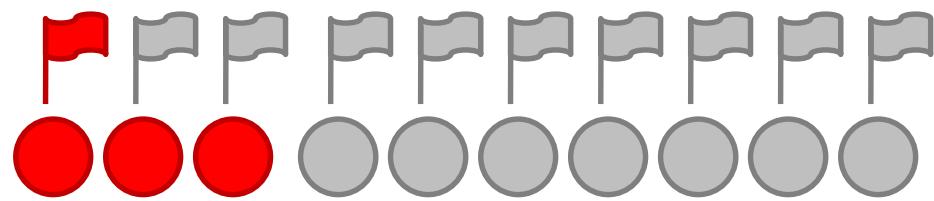
TP: 0/3

FP: 0/7

FN: 3/3



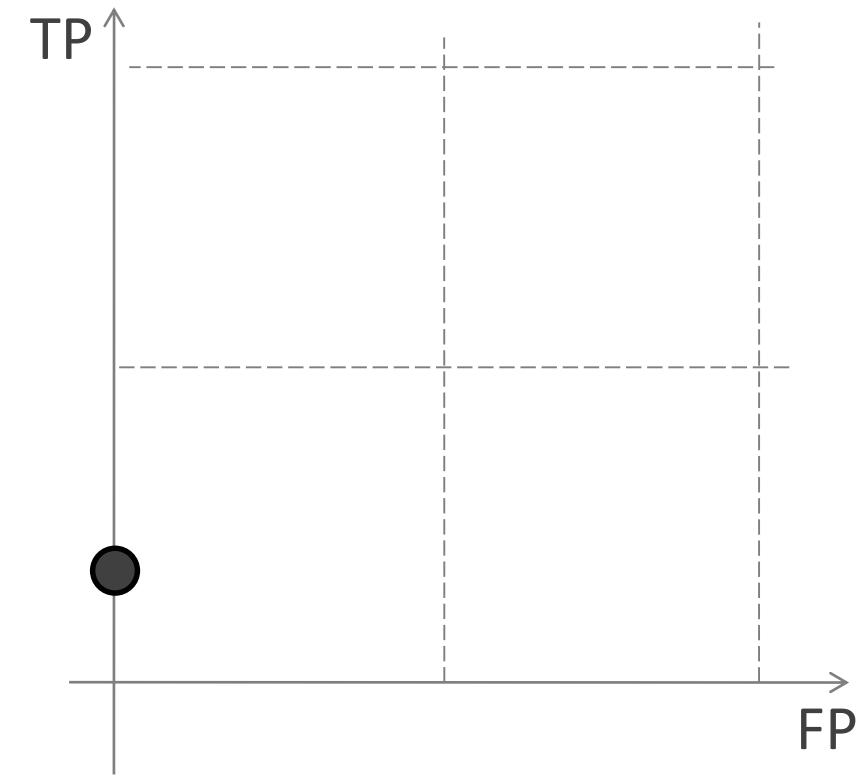
ROC curve



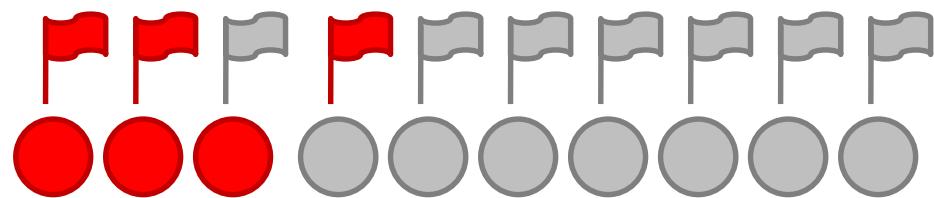
TP: 1/3

FP: 0/7

FN: 2/3



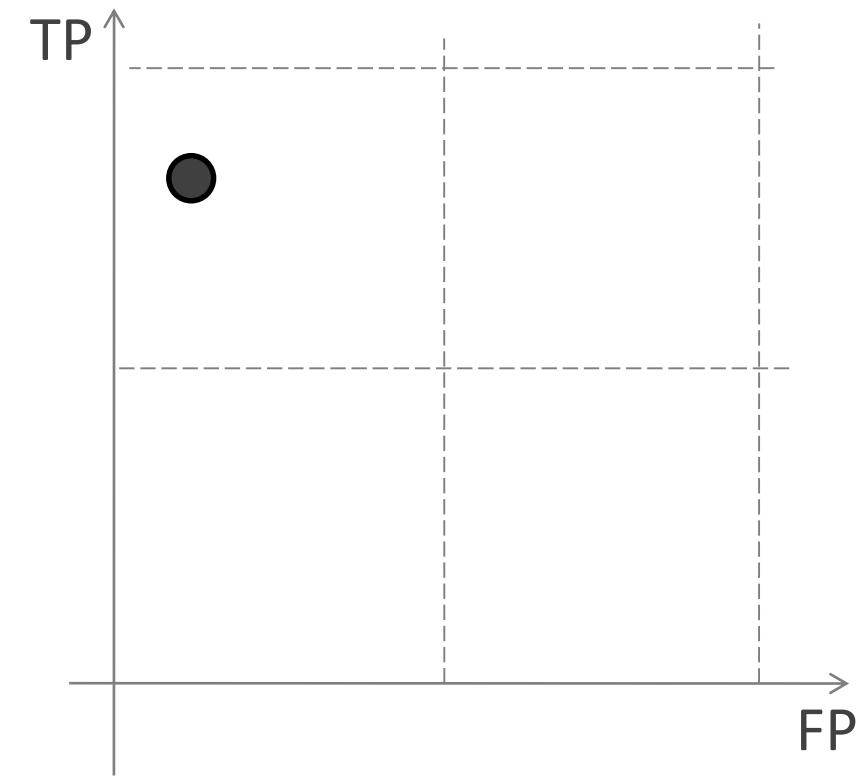
ROC curve



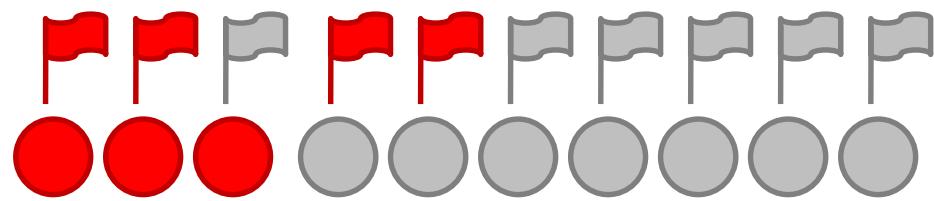
TP: 2/3

FP: 1/7

FN: 1/3



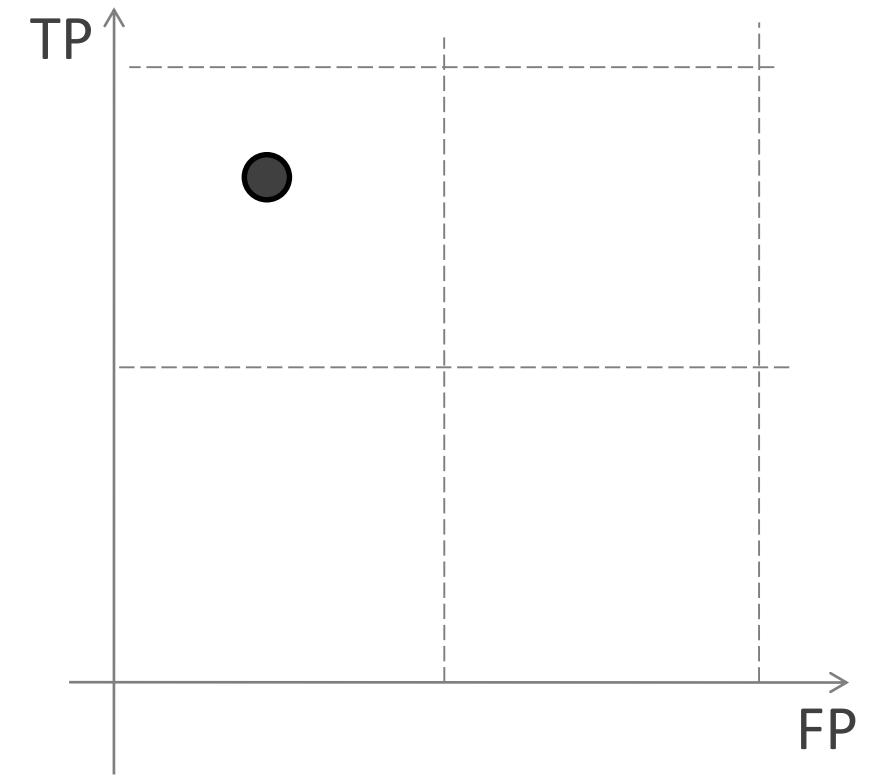
ROC curve



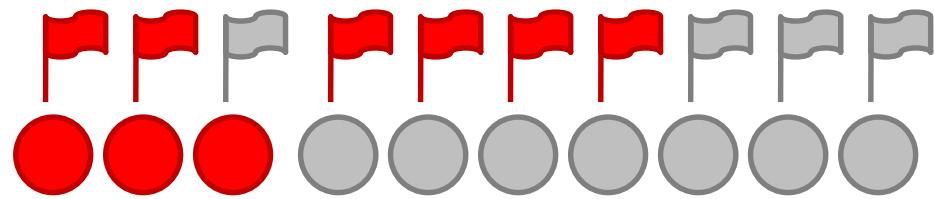
TP: 2/3

FP: 2/7

FN: 1/3



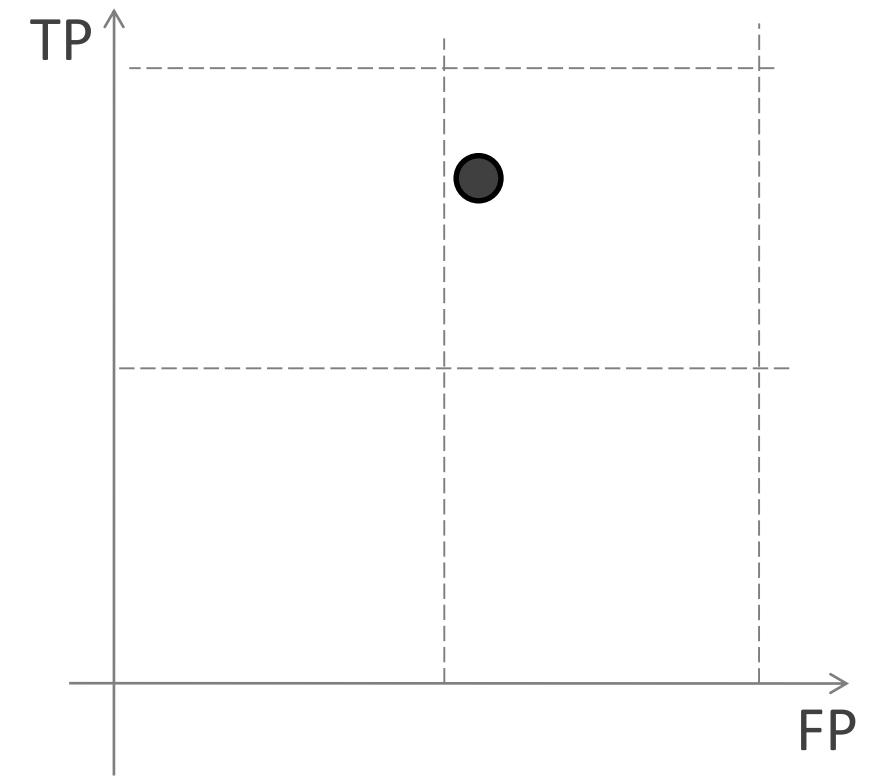
ROC curve



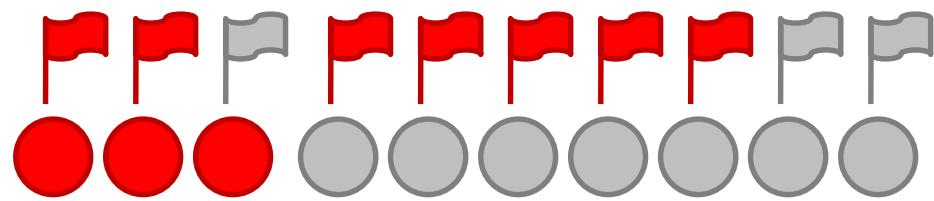
TP: 2/3

FP: 4/7

FN: 1/3



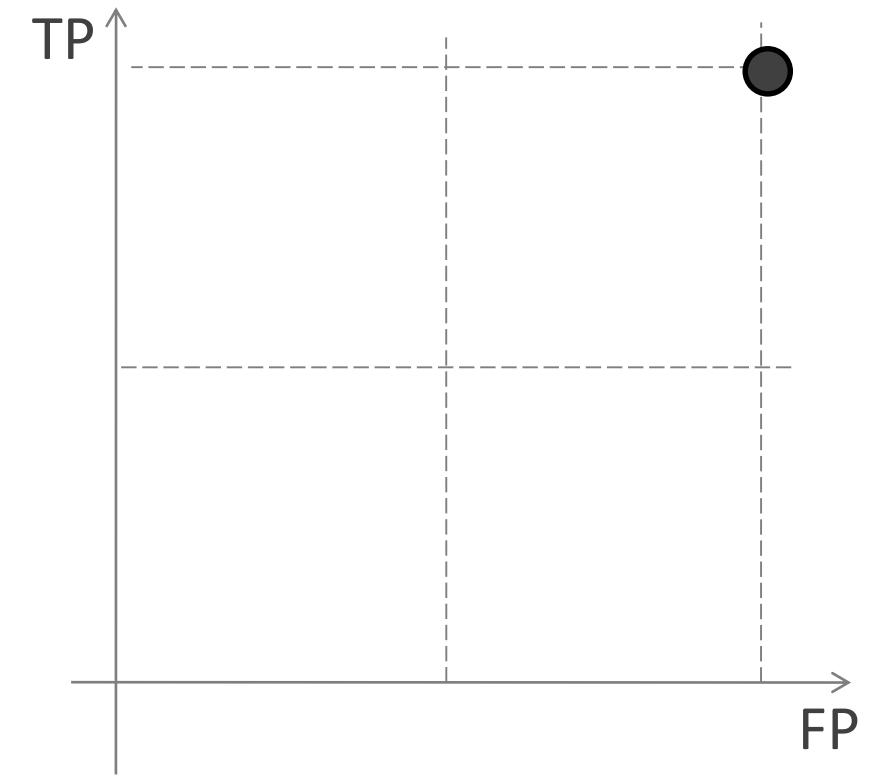
ROC curve



TP: 3/3

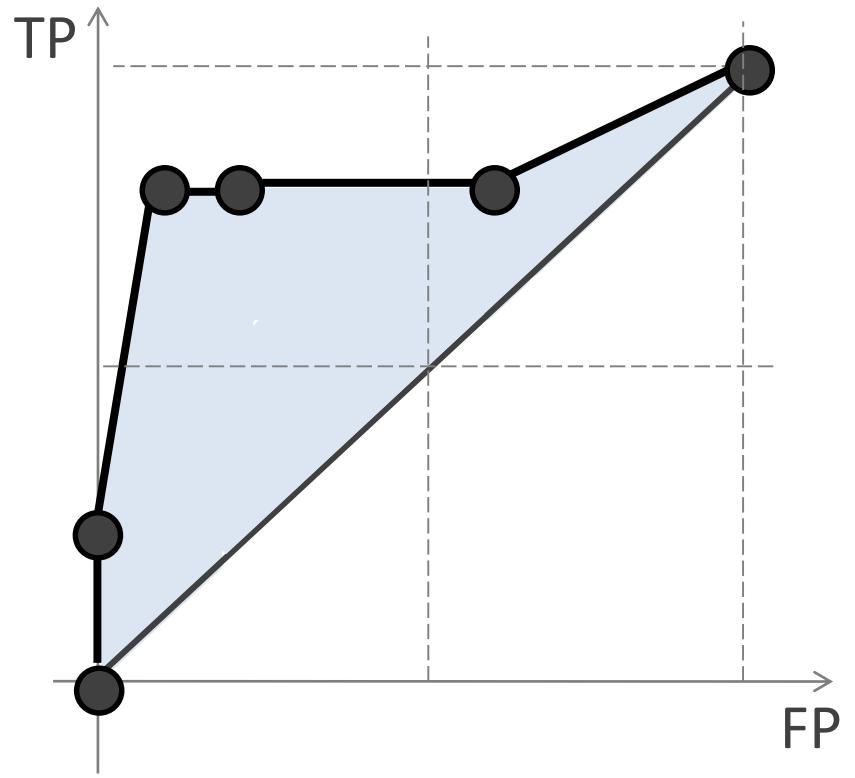
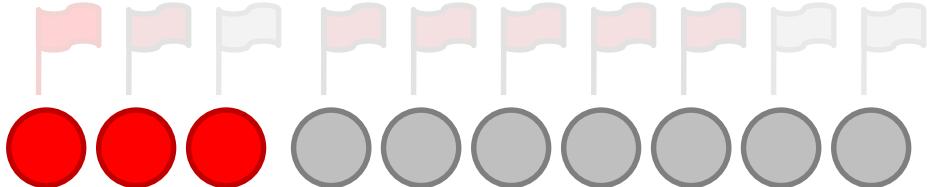
FP: 7/7

FN: 0/3



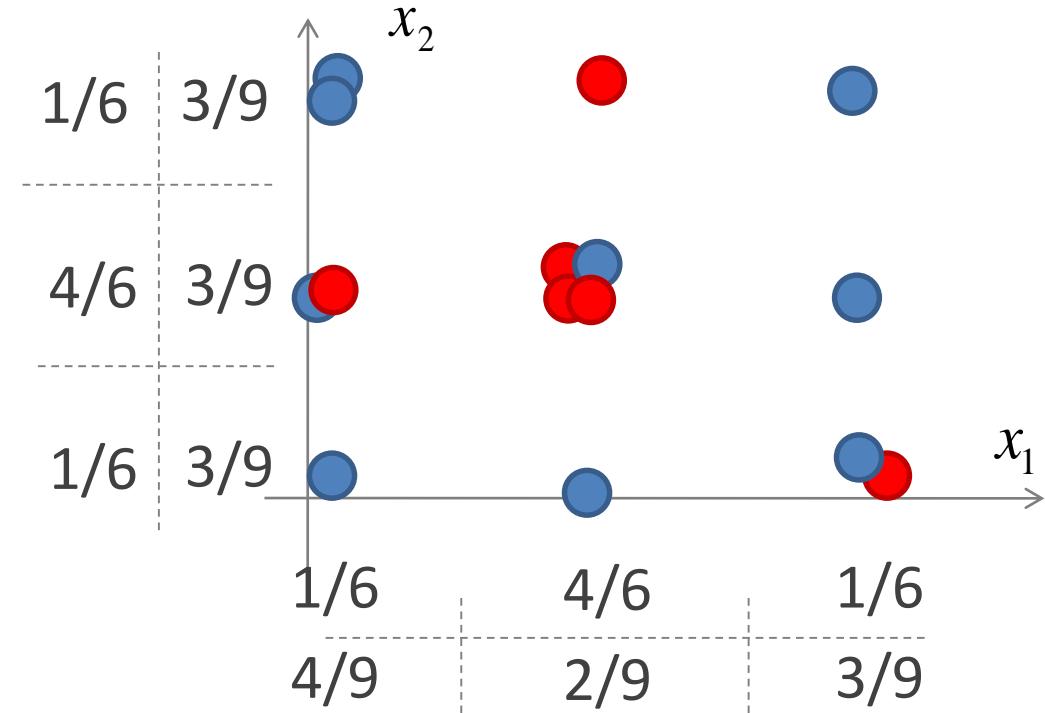
ROC curve

AUC: area under ROC curve



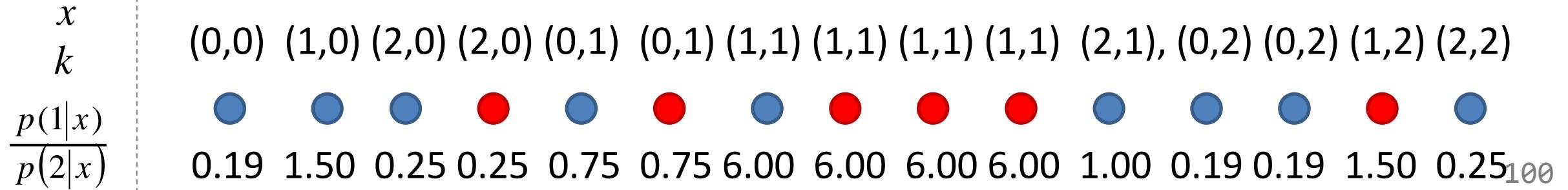
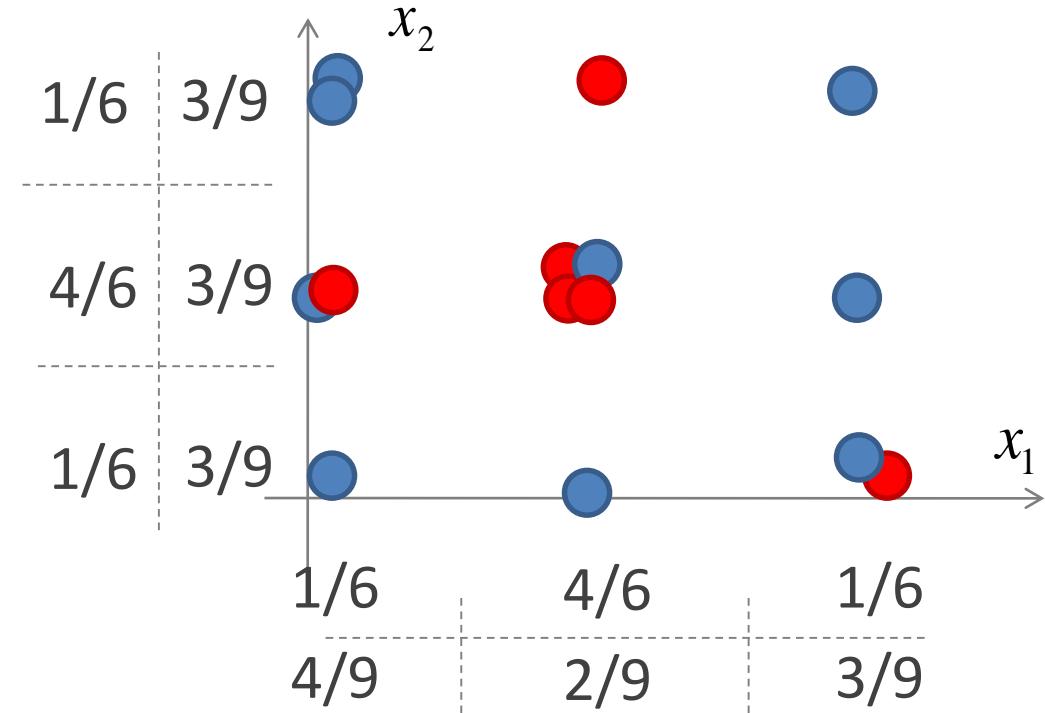
ROC curve

Example: Naive Bayesian classifier



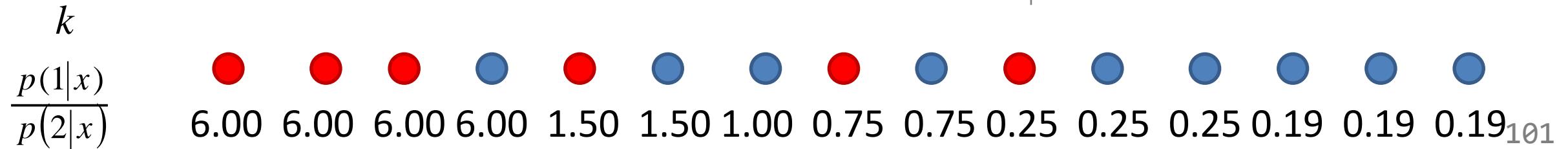
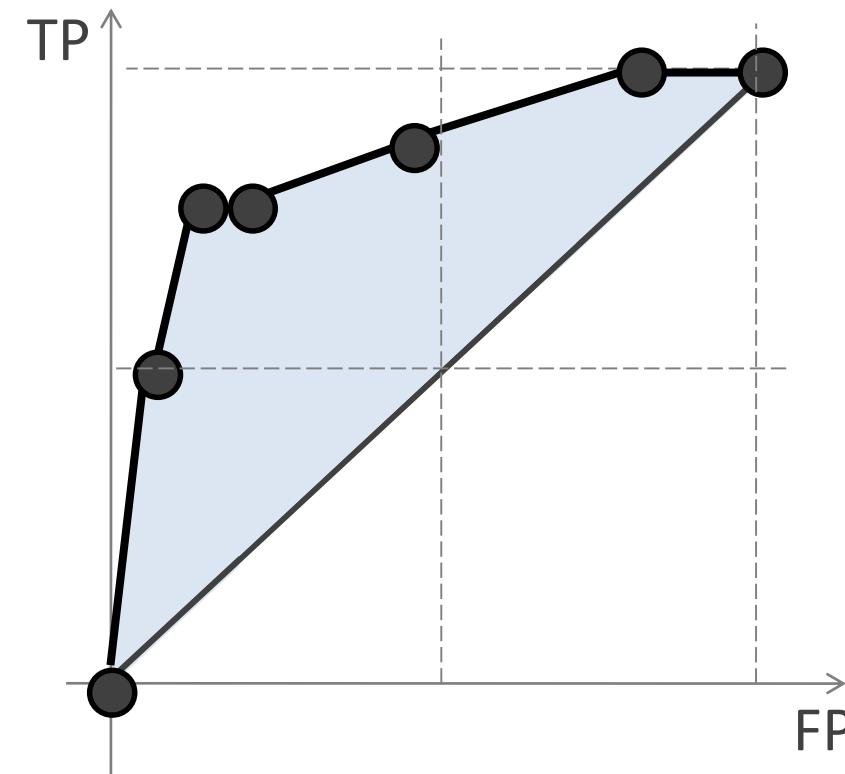
ROC curve

Example: Naive Bayesian classifier



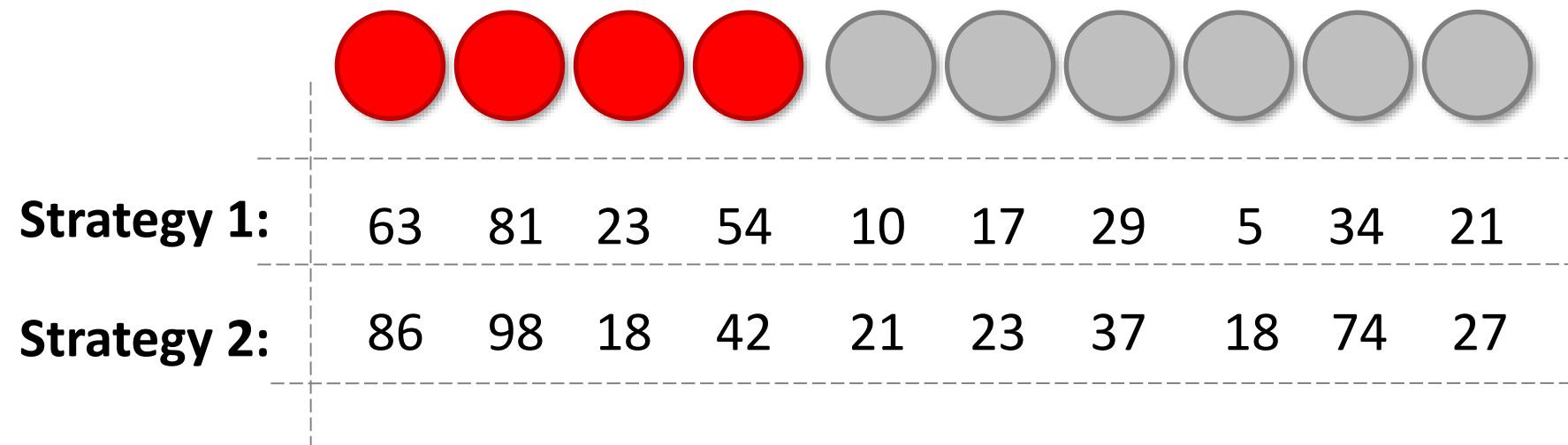
ROC curve

Example: Naive Bayesian classifier



ROC curve

The task: compare accuracy of two strategies



ROC curve

```
def display_roc_curve_from_file(plt,fig,path_scores,caption=' '):

    data = load_mat(path_scores, dtype = numpy.chararray, delim='\t')
    labels = (data[:, 0]).astype('float32')
    scores = data[:, 1:].astype('float32')

    fpr, tpr, thresholds = metrics.roc_curve(labels, scores)
    roc_auc = auc(fpr, tpr)

    plot_tp_fp(plt,fig,tpr,fpr,roc_auc,caption)

    return


def plot_tp_fp(plt,fig,tpr,fpr,roc_auc,caption=' '):

    lw = 2
    plt.plot(fpr, tpr, color='darkgreen', lw=lw, label='AUC = %0.2f' % roc_auc)
    plt.plot([0, 1.05], [0, 1.05], color='lightgray', lw=lw, linestyle='--')
    plt.grid(which='major', color='lightgray', linestyle='--')
    fig.canvas.set_window_title(caption + ('AUC = %0.4f' % roc_auc))
    return
```

1.9. Benchmarking the classifiers

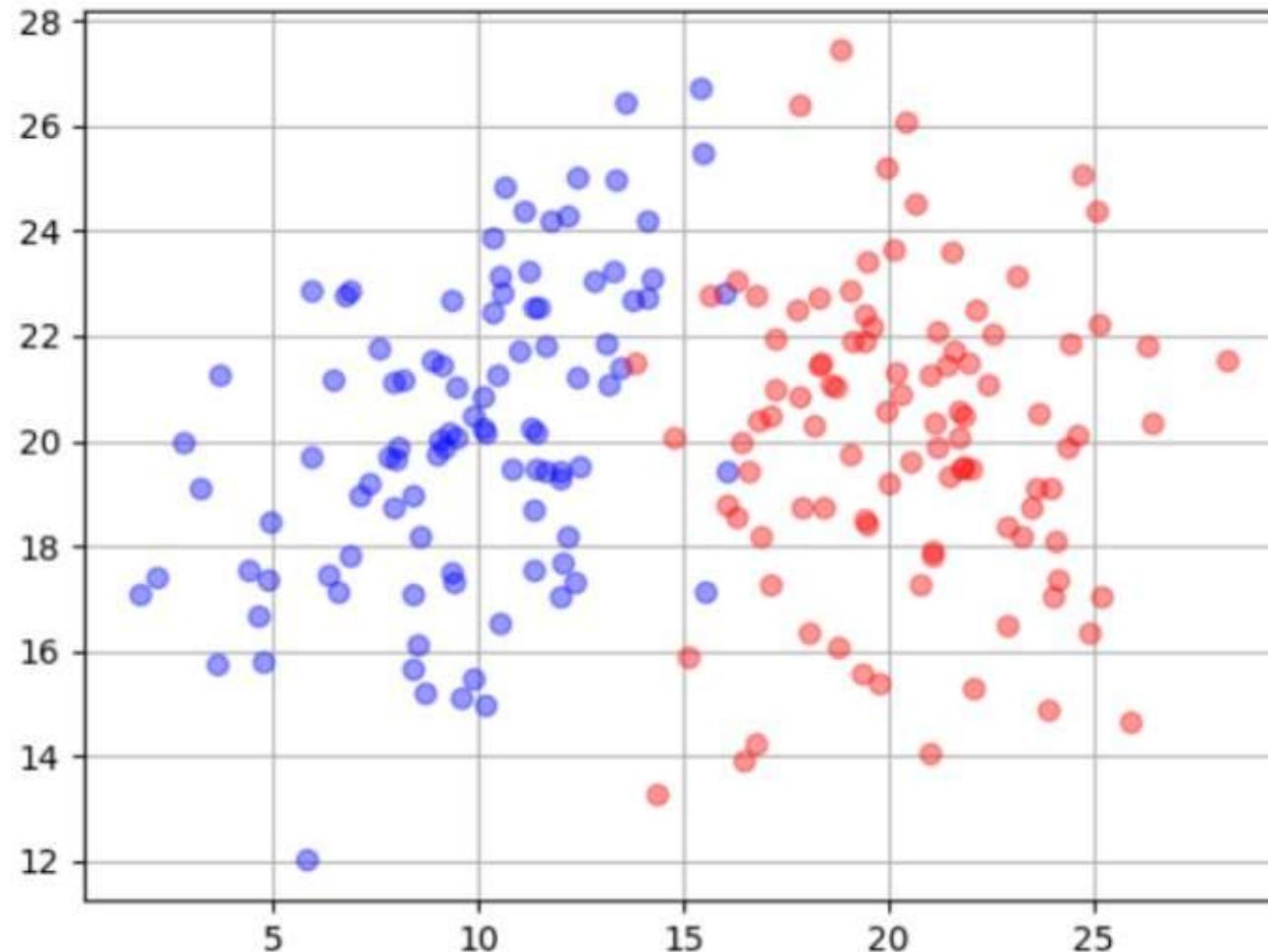


Benchmarking the Classifiers

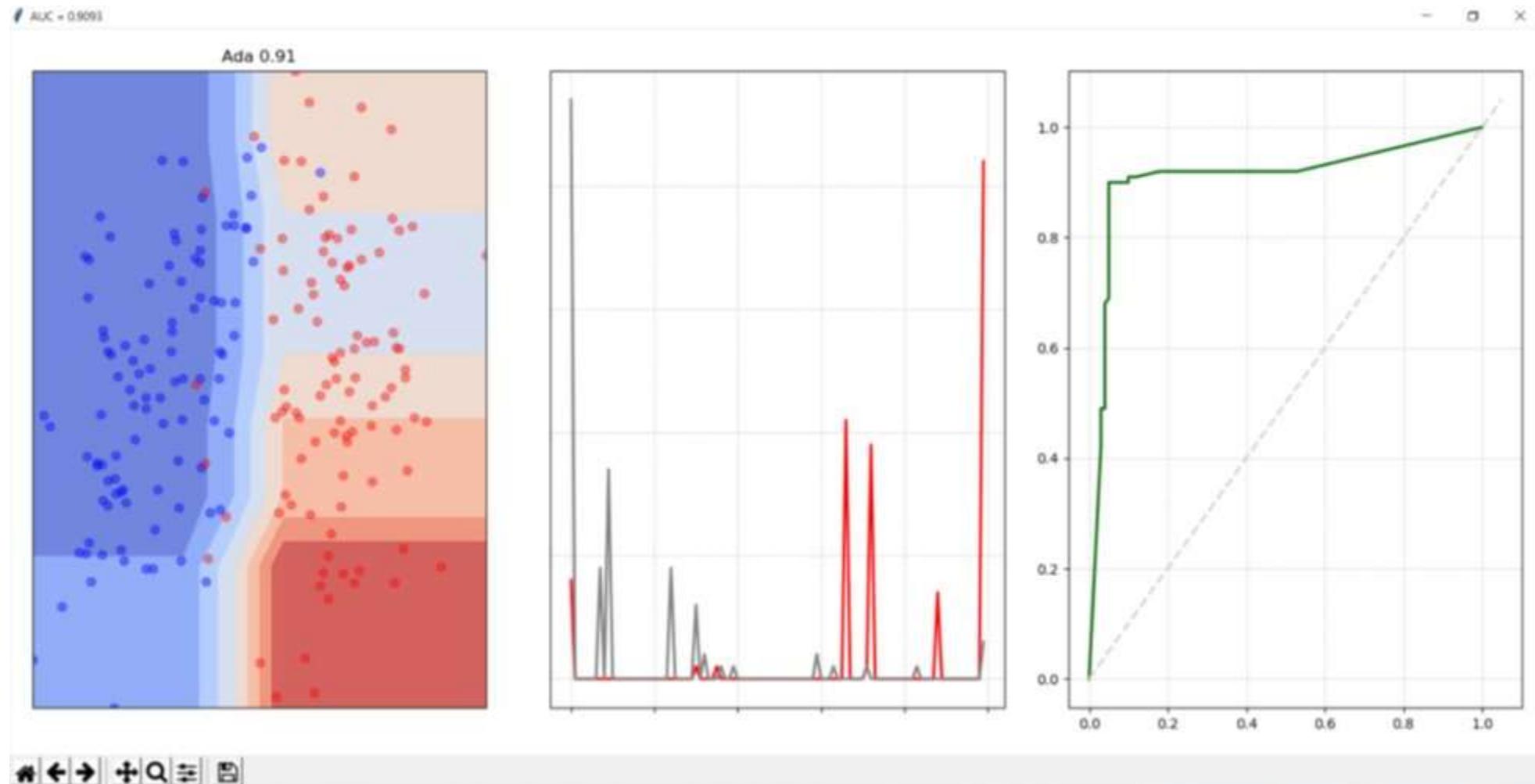
- Regression
- Naive Bayes
- Gaussian
- SVM
- Nearest Neighbors
- Decision Tree
- Random Forest
- AdaBoost
- xgboost
- Neural Net

Benchmarking the Classifiers

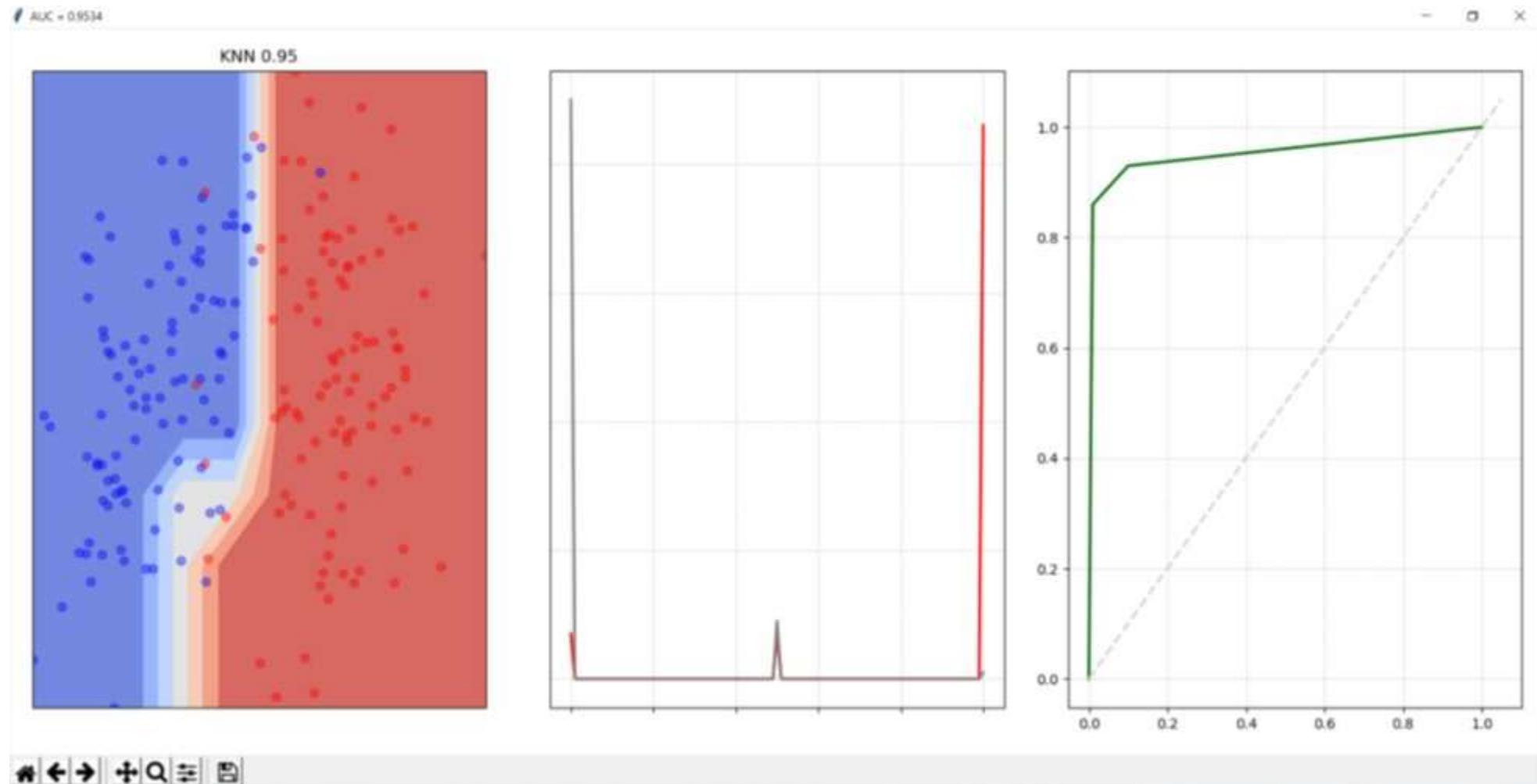
Gaussian distribution



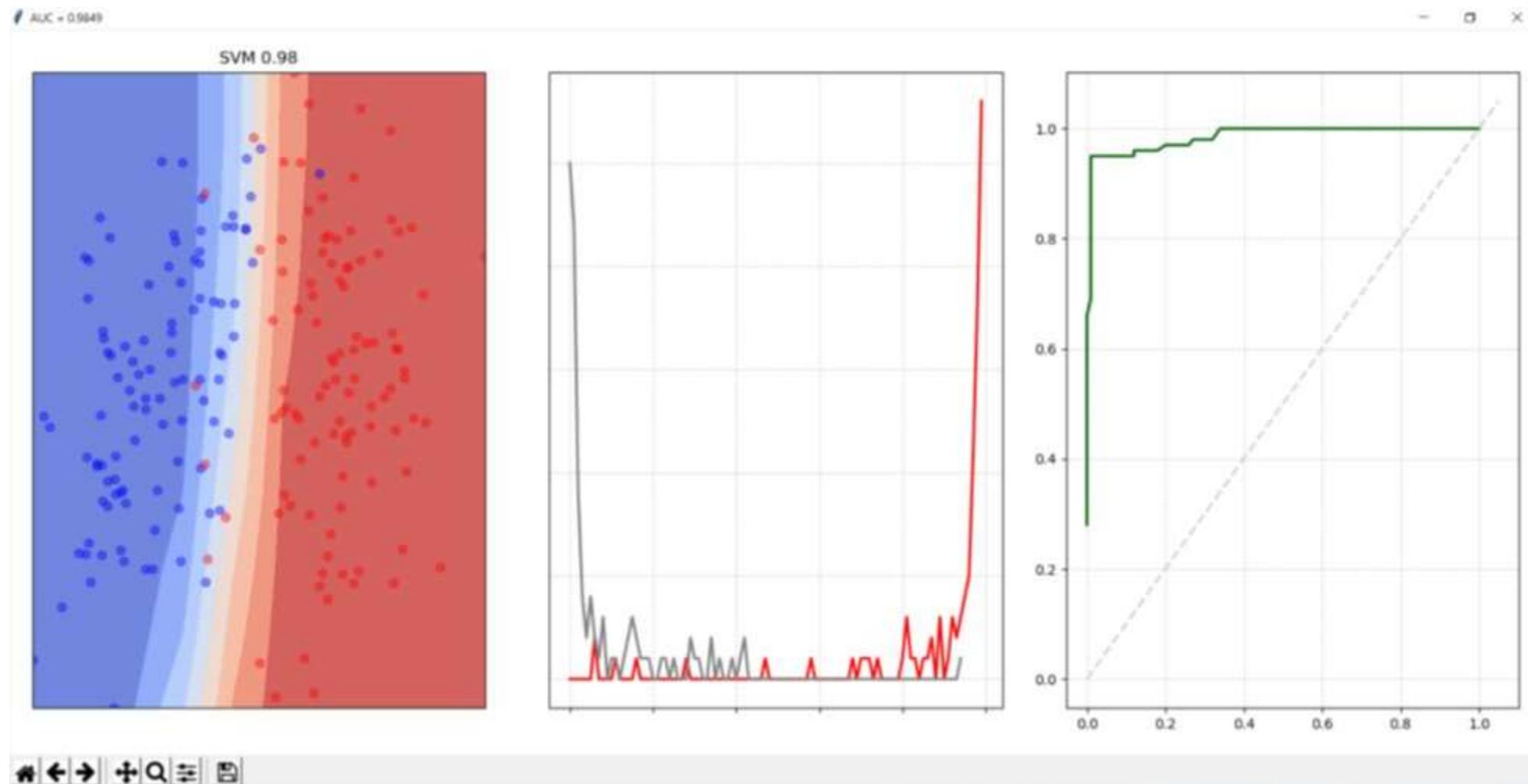
Benchmarking the Classifiers



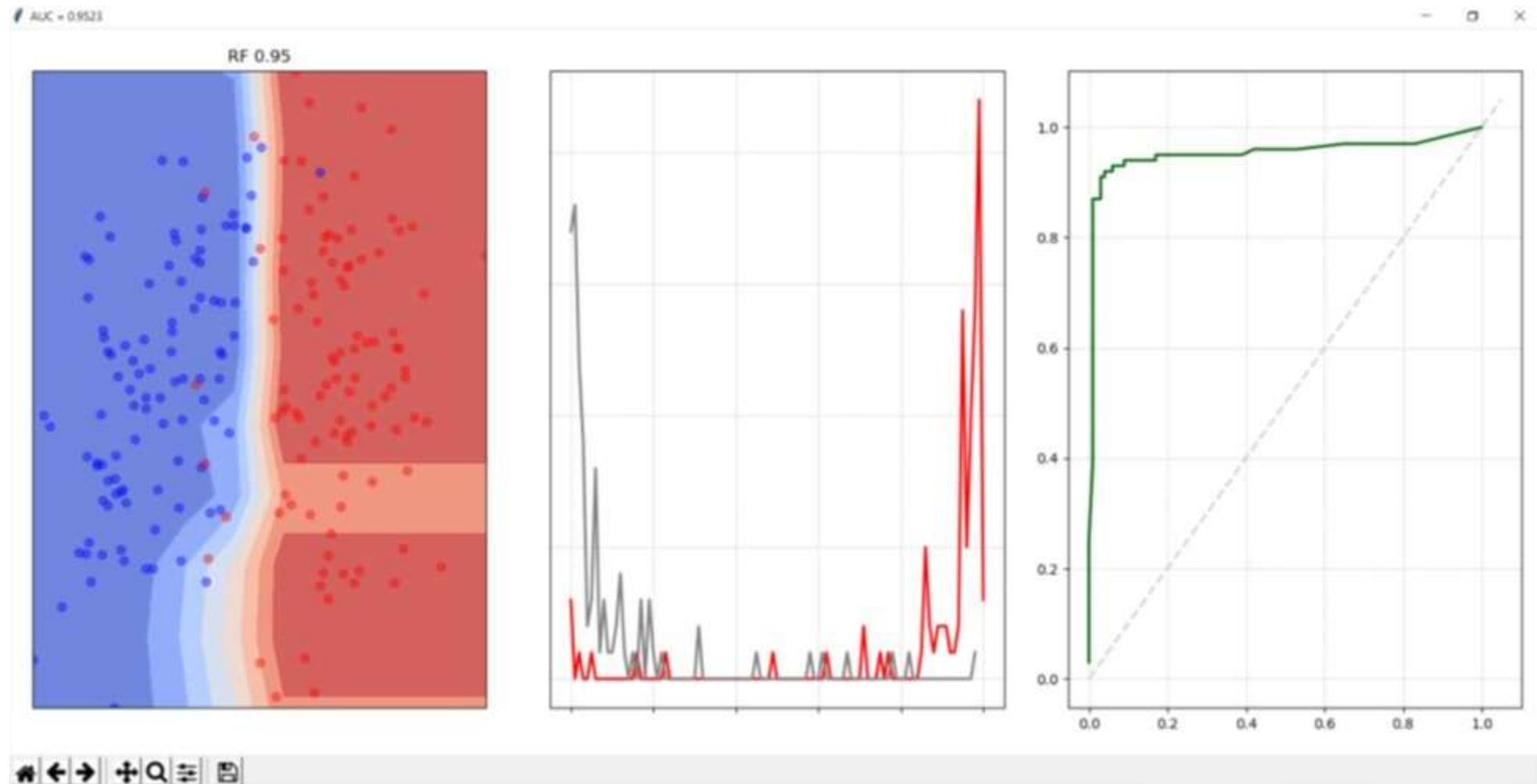
Benchmarking the Classifiers



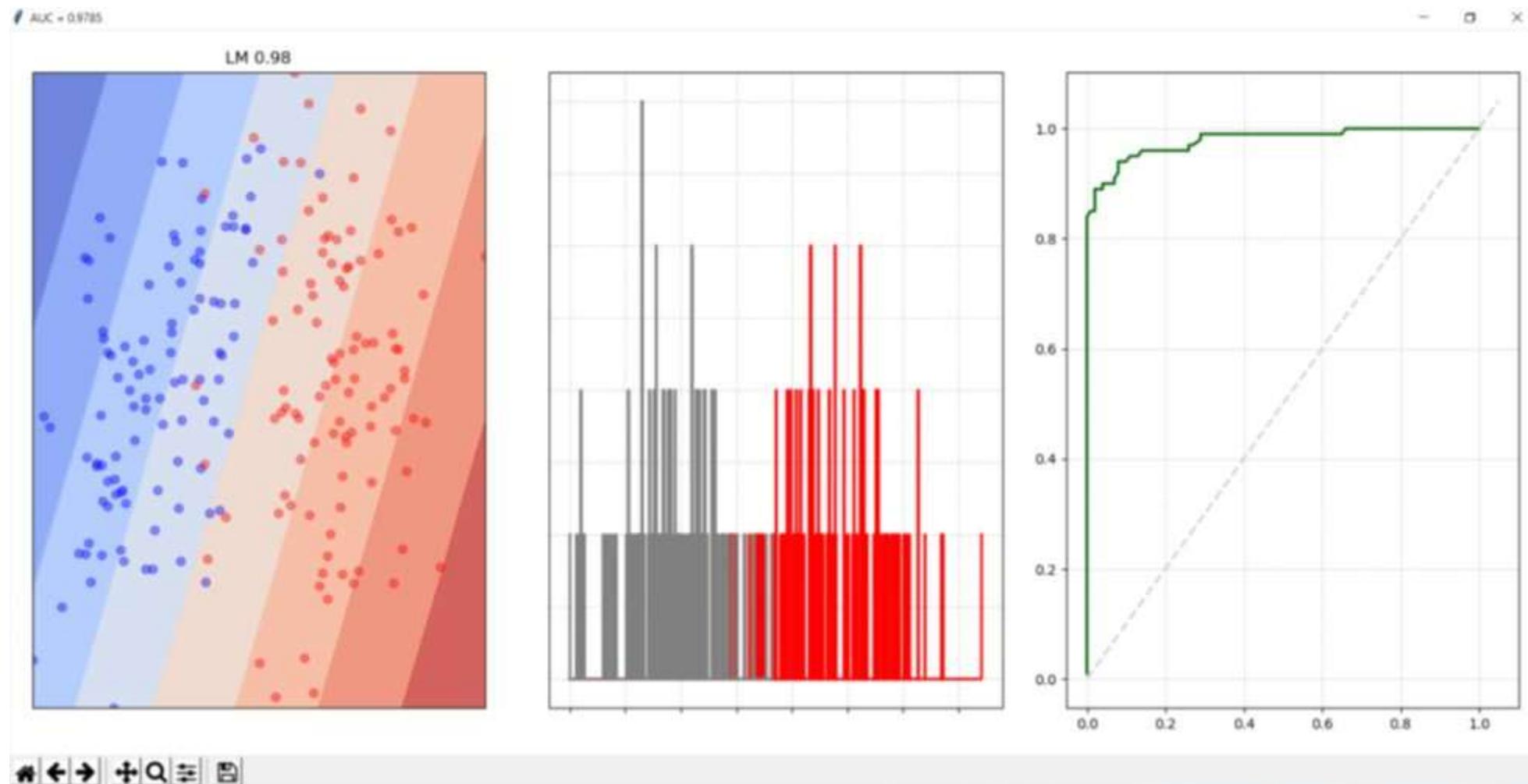
Benchmarking the Classifiers



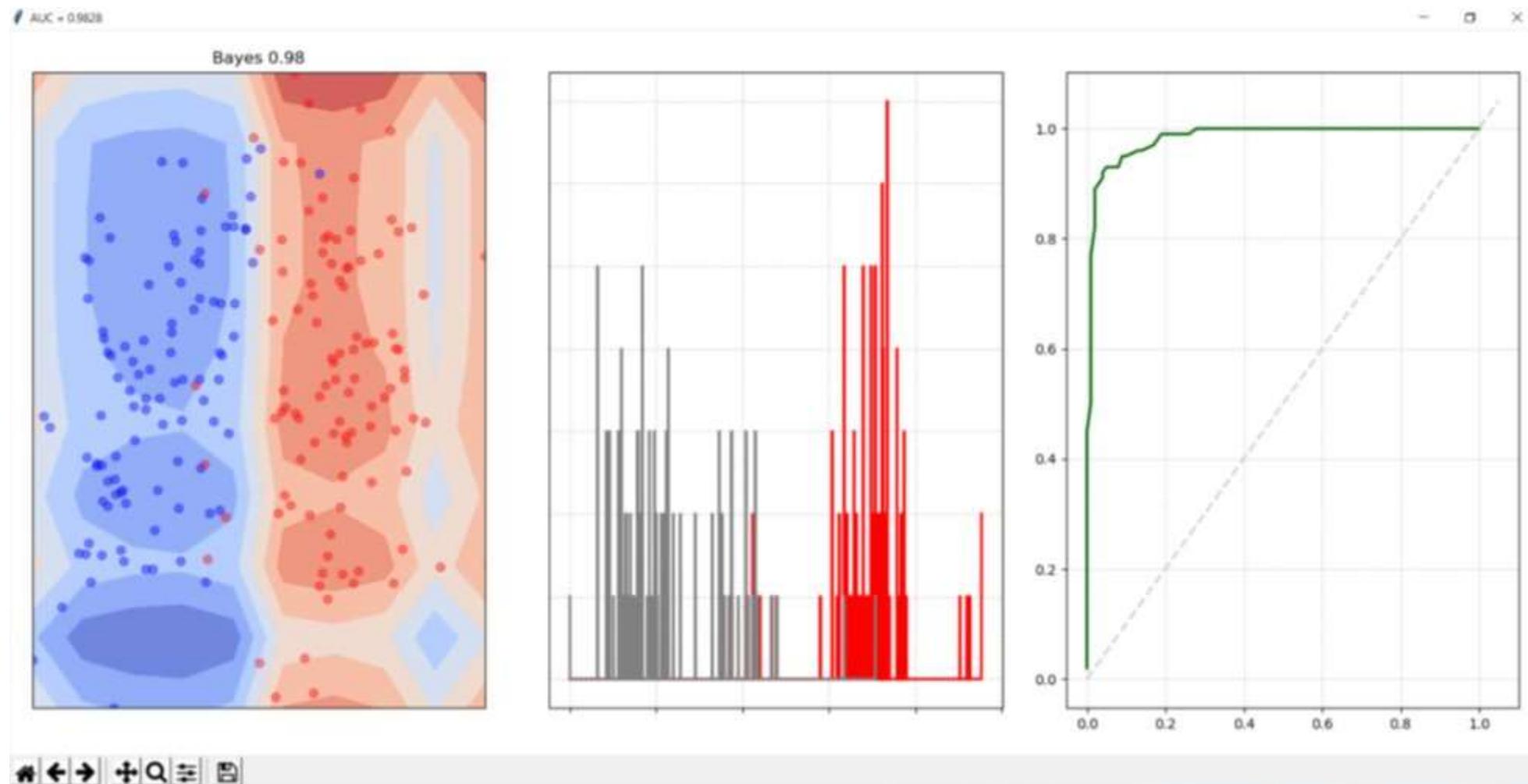
Benchmarking the Classifiers



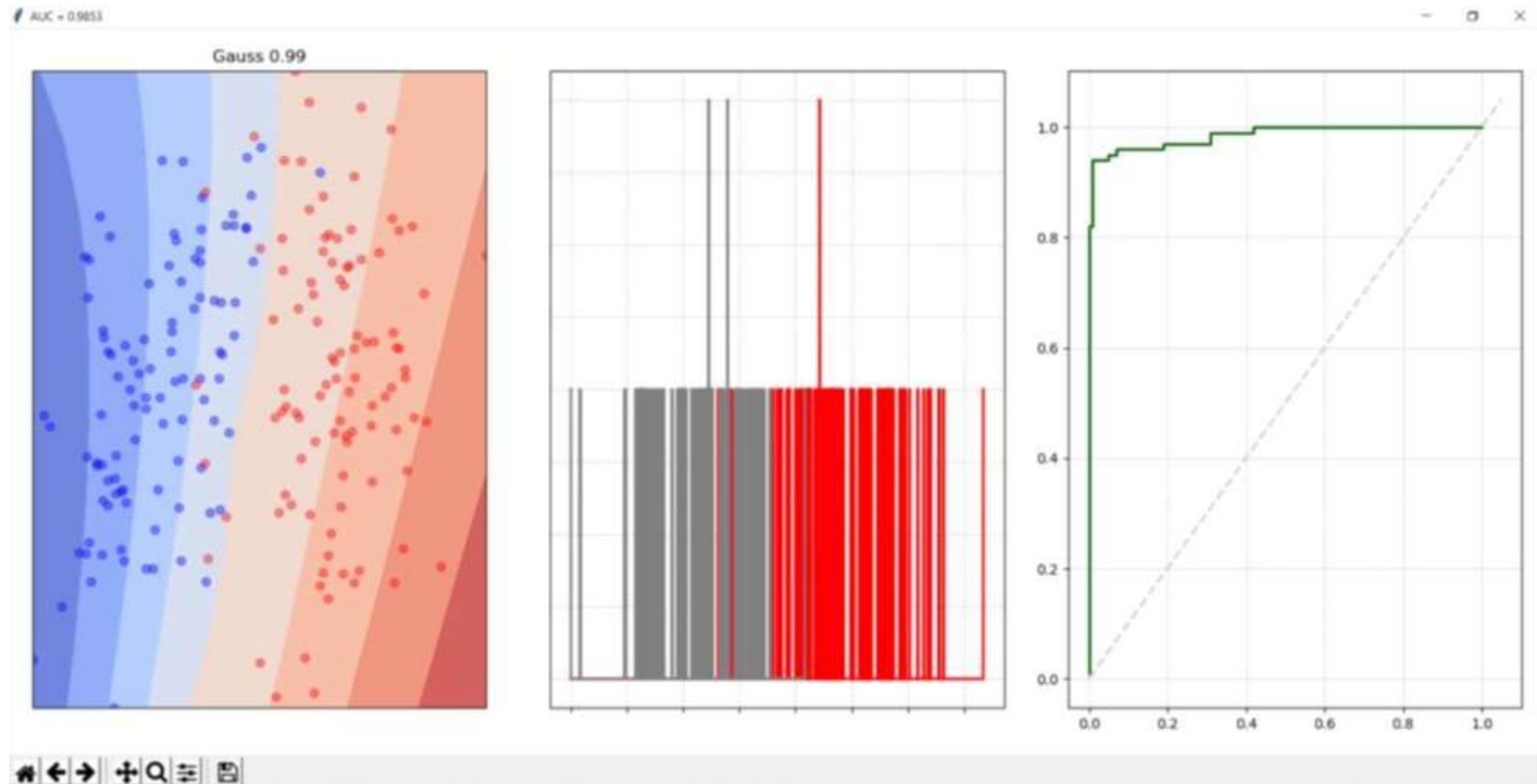
Benchmarking the Classifiers



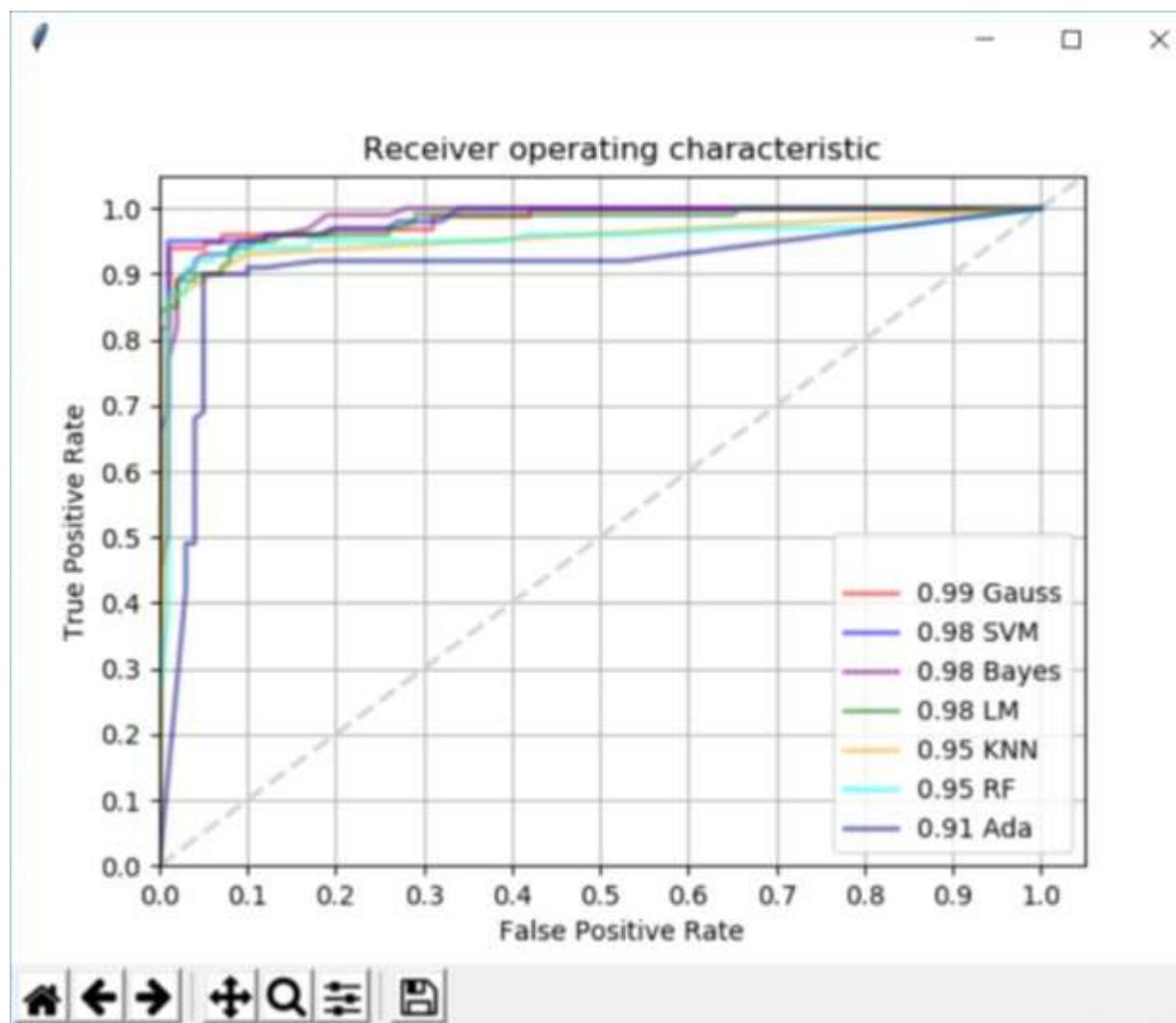
Benchmarking the Classifiers



Benchmarking the Classifiers

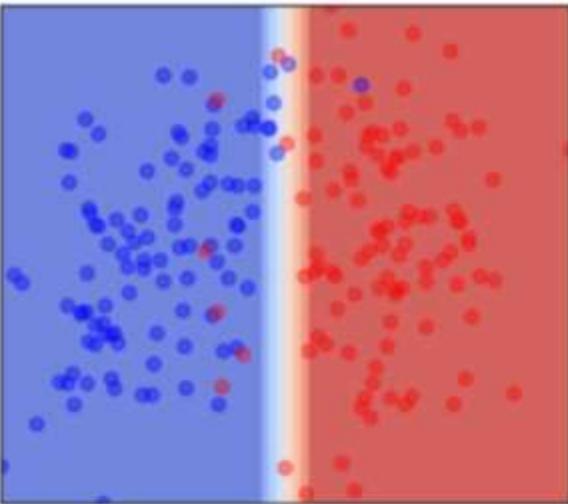


Benchmarking the Classifiers

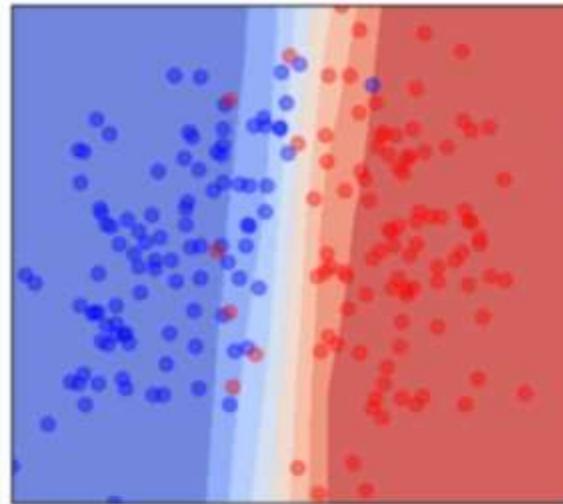


Benchmarking the Classifiers

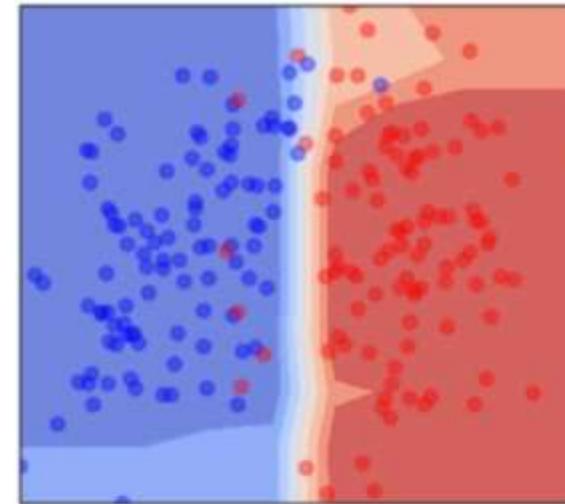
Ada 0.91



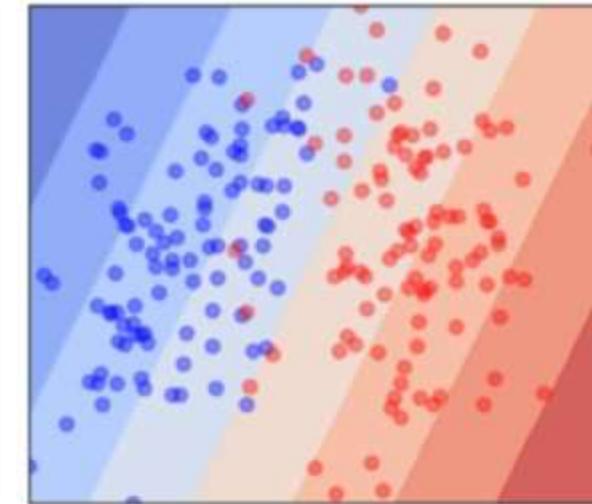
SVM 0.99



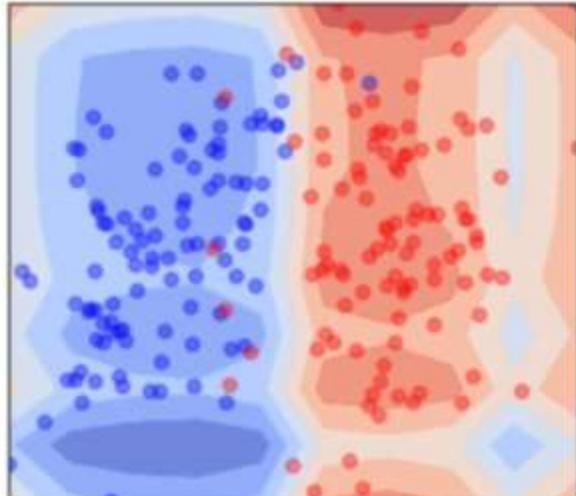
RF 0.95



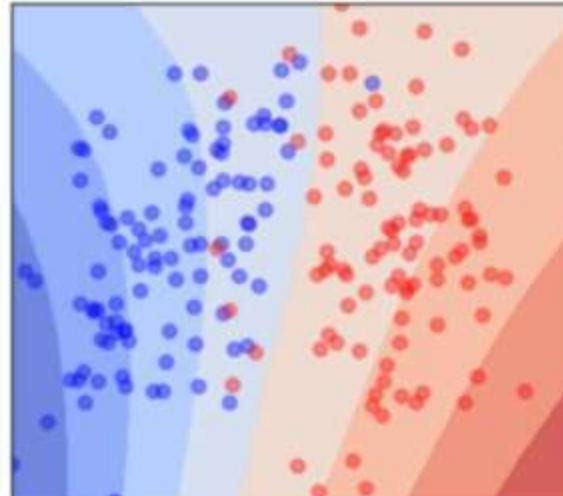
LM 0.98



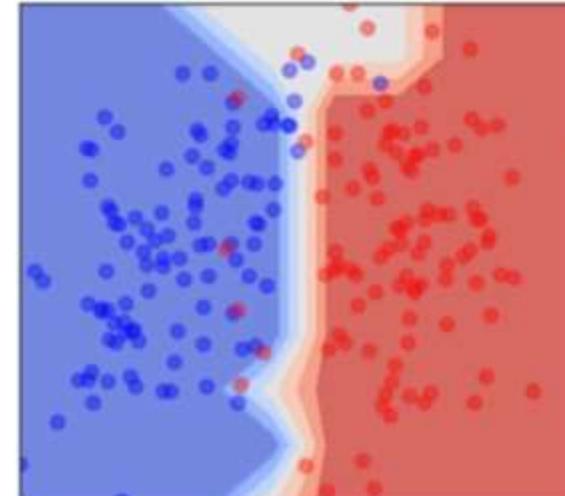
Bayes 0.98



Gauss 0.99

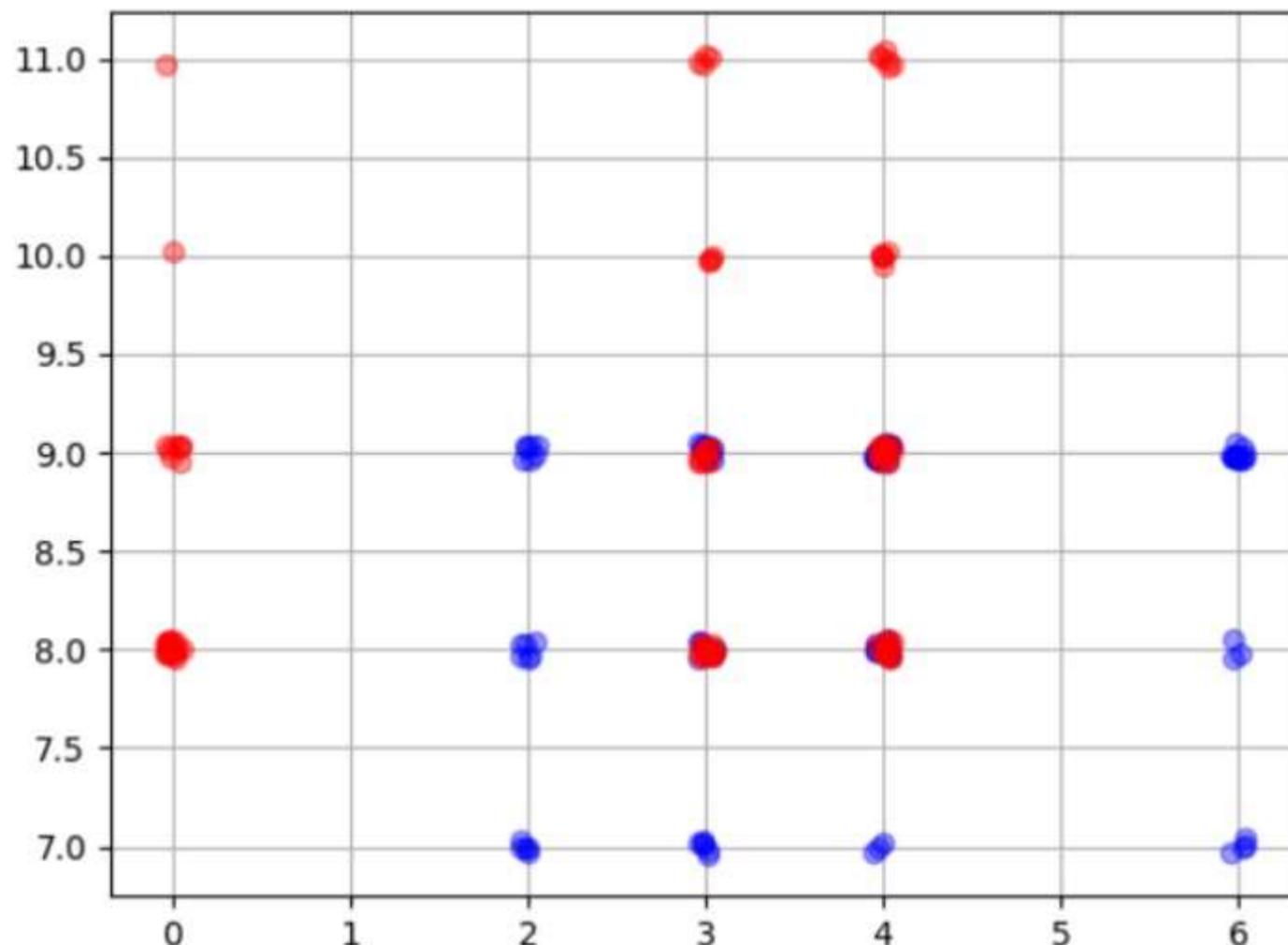


KNN 0.95



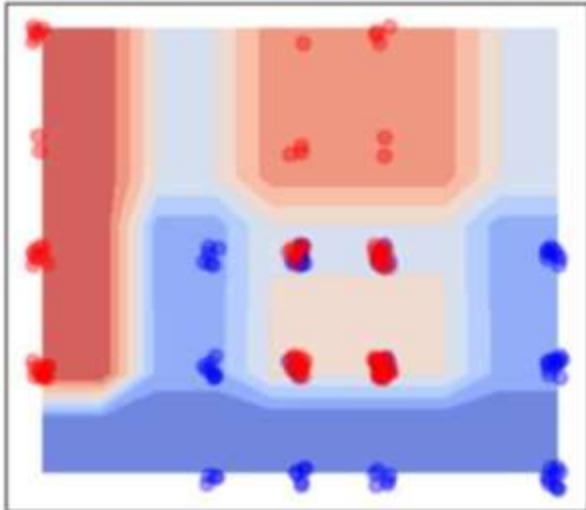
Benchmarking the Classifiers

Non-gaussian distribution

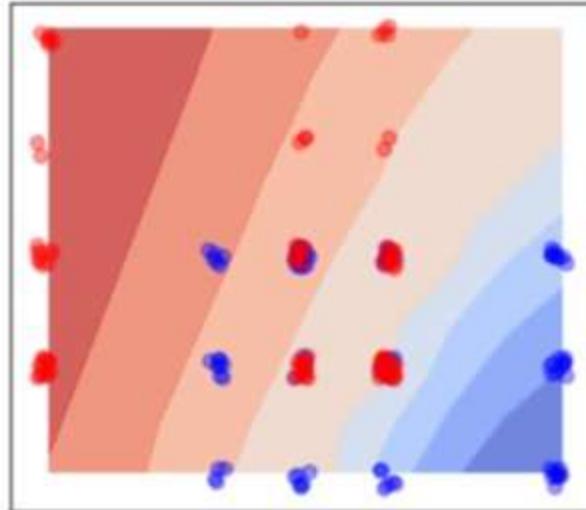


Benchmarking the Classifiers

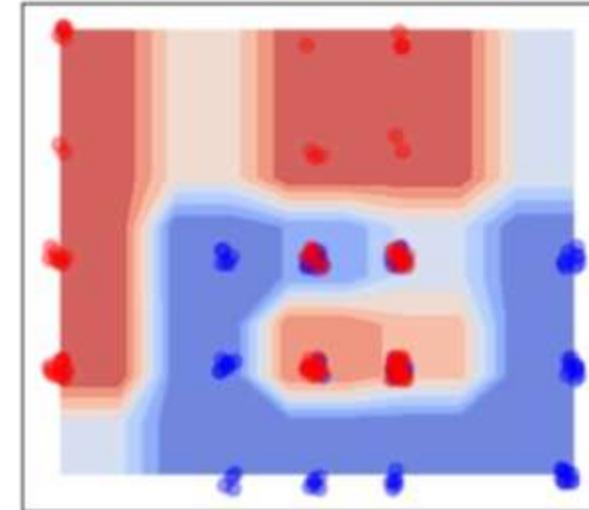
Ada 0.88



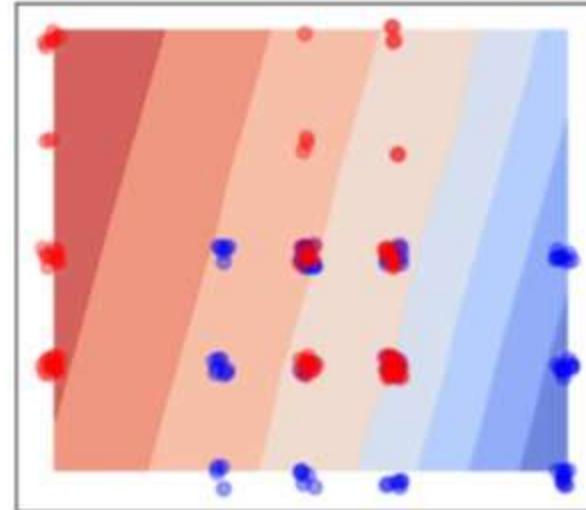
SVM 0.74



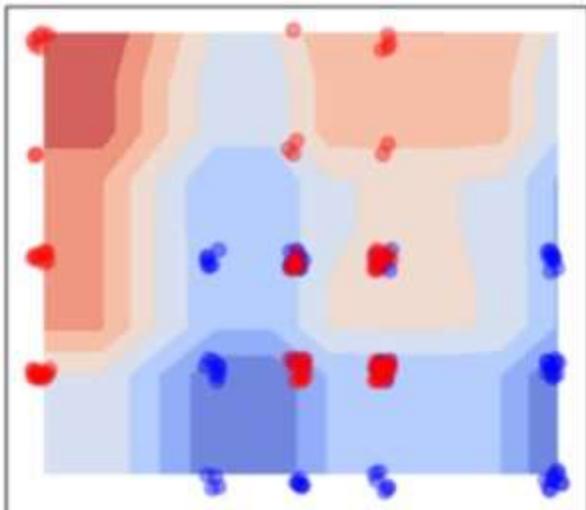
RF 0.88



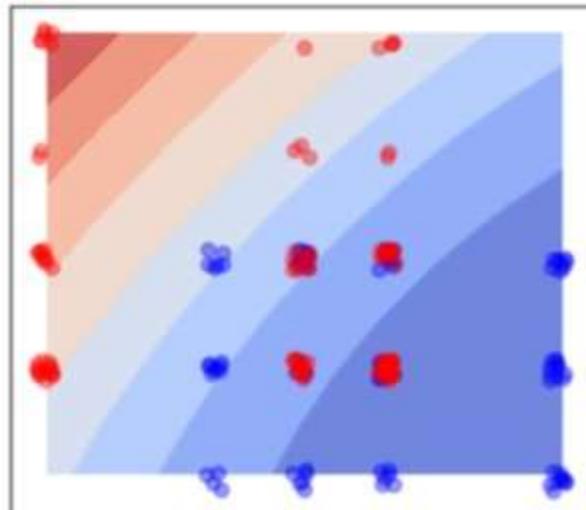
LM 0.71



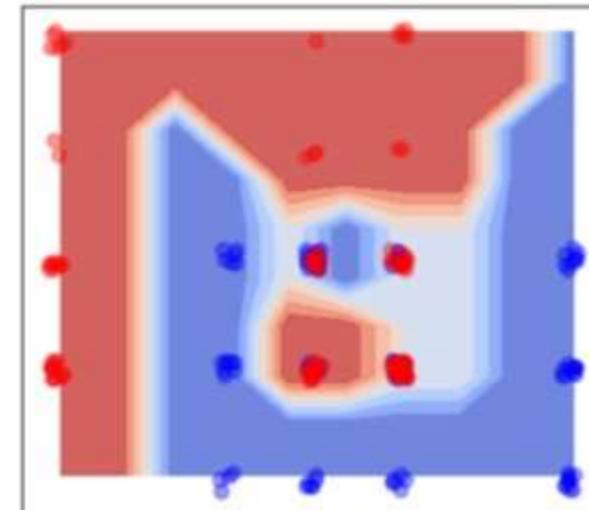
Bayes 0.90



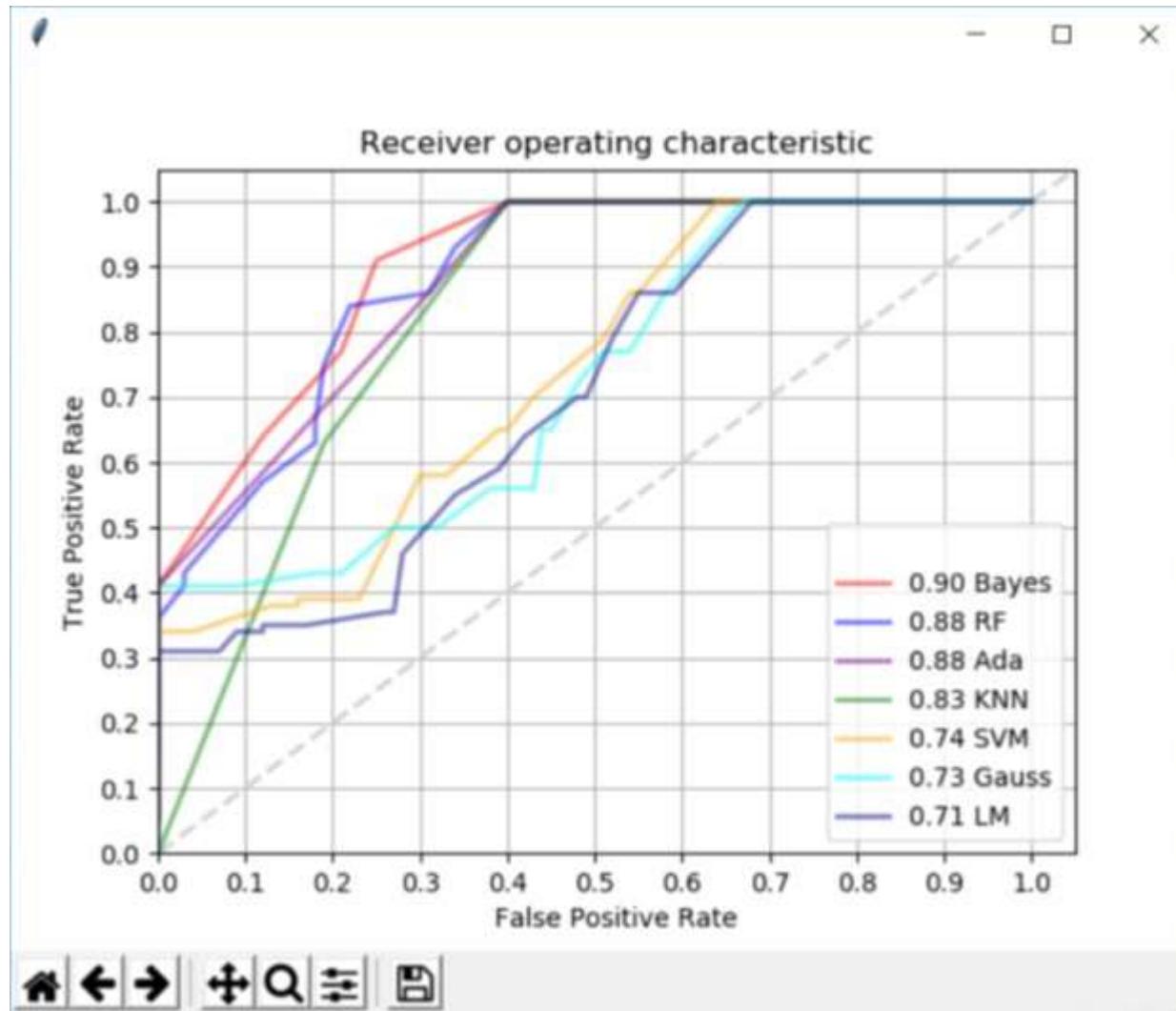
Gauss 0.73



KNN 0.83



Benchmarking the Classifiers



Benchmarking the Classifiers

```
def E2E_on_features_files_2_classes(Classifier, folder_out, filename_data_pos, filename_data_neg, filename_scs_pos=None, filename_scs_neg=None, fig=None):
    filename_data_grid = folder_out+'data_grid.txt'
    filename_scores_grid = folder_out+'scores_grid.txt'

    Pos = (IO.load_mat(filename_data_pos, numpy.chararray, '\t')).shape[0]
    Neg = (IO.load_mat(filename_data_neg, numpy.chararray, '\t')).shape[0]

    numpy.random.seed(125)
    idx_pos_train = numpy.random.choice(Pos, int(Pos/2), replace=False)
    idx_neg_train = numpy.random.choice(Neg, int(Neg/2), replace=False)
    idx_pos_test = [x for x in range(0, Pos) if x not in idx_pos_train]
    idx_neg_test = [x for x in range(0, Neg) if x not in idx_neg_train]

    generate_data_grid(filename_data_pos, filename_data_neg, filename_data_grid)

    model = learn_on_pos_neg_files(Classifier, filename_data_pos, filename_data_neg, '\t', idx_pos_train, idx_neg_train)
    score_feature_file(Classifier, filename_data_pos, filename_scs_pos=filename_scs_pos, delimiter='\t', append=0, rand_sel=idx_pos_test)
    score_feature_file(Classifier, filename_data_neg, filename_scs_neg=filename_scs_neg, delimiter='\t', append=0, rand_sel=idx_neg_test)
    score_feature_file(Classifier, filename_data_grid, filename_scs=filename_scores_grid)

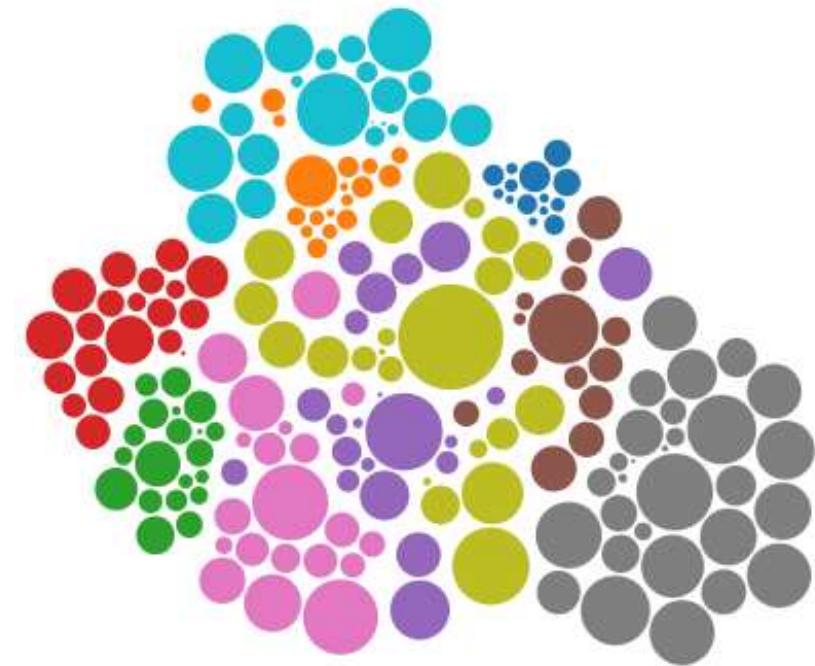
    model = learn_on_pos_neg_files(Classifier, filename_data_pos, filename_data_neg, '\t', idx_pos_test, idx_neg_test)
    score_feature_file(Classifier, filename_data_pos, filename_scs=filename_scs_pos, delimiter='\t', append=1, rand_sel=idx_pos_train)
    score_feature_file(Classifier, filename_data_neg, filename_scs=filename_scs_neg, delimiter='\t', append=1, rand_sel=idx_neg_train)

    tpr, fpr, auc = IO.get_roc_data_from_scores_file_v2(filename_scs_pos, filename_scs_neg)

    if(fig!=None):
        th = get_th_v2(filename_scs_pos, filename_scs_neg)
        IO.display_roc_curve_from_descriptions(plt.subplot(1,3,3),fig,filename_scs_pos,filename_scs_neg,delim='\t')
        IO.display_distributions(plt.subplot(1,3,2),fig, filename_scs_pos, filename_scs_neg,delim='\t')
        IO.plot_2D_scores(plt.subplot(1,3,1),fig,filename_data_pos,filename_data_neg,filename_data_grid, filename_scores_grid, th, noice_needed=0)
        plt.tight_layout()
        plt.show()

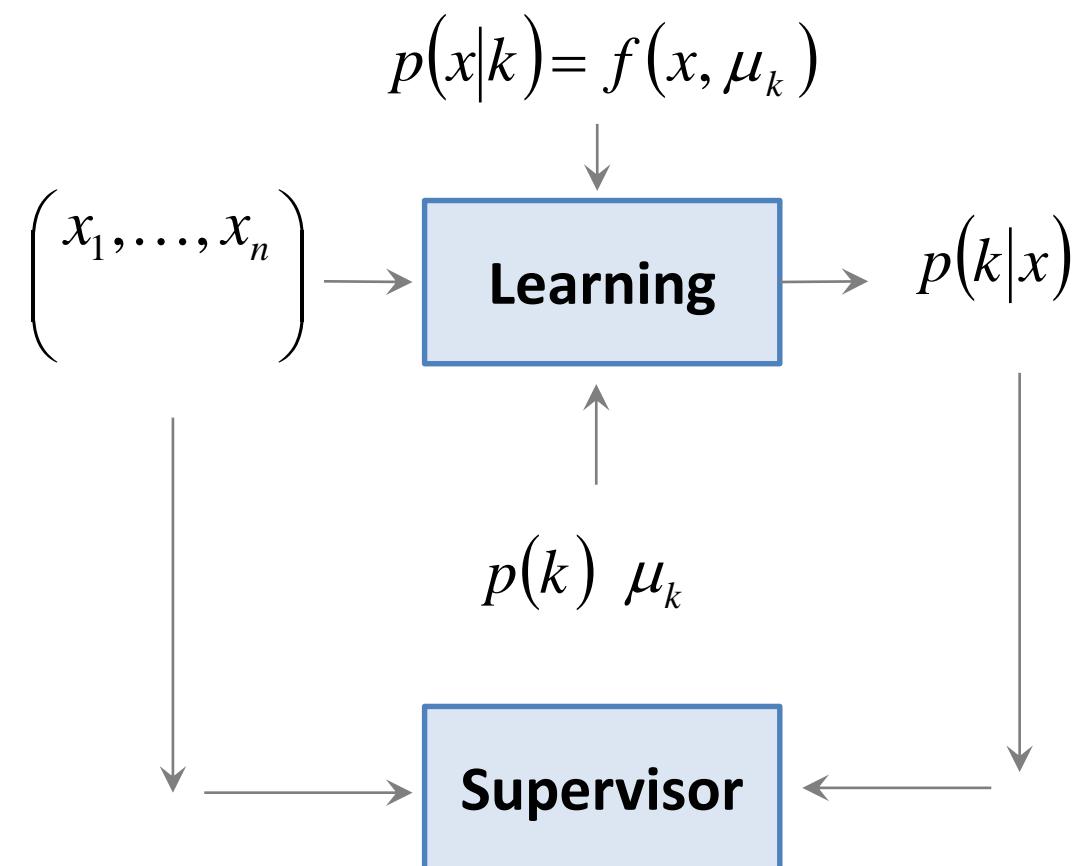
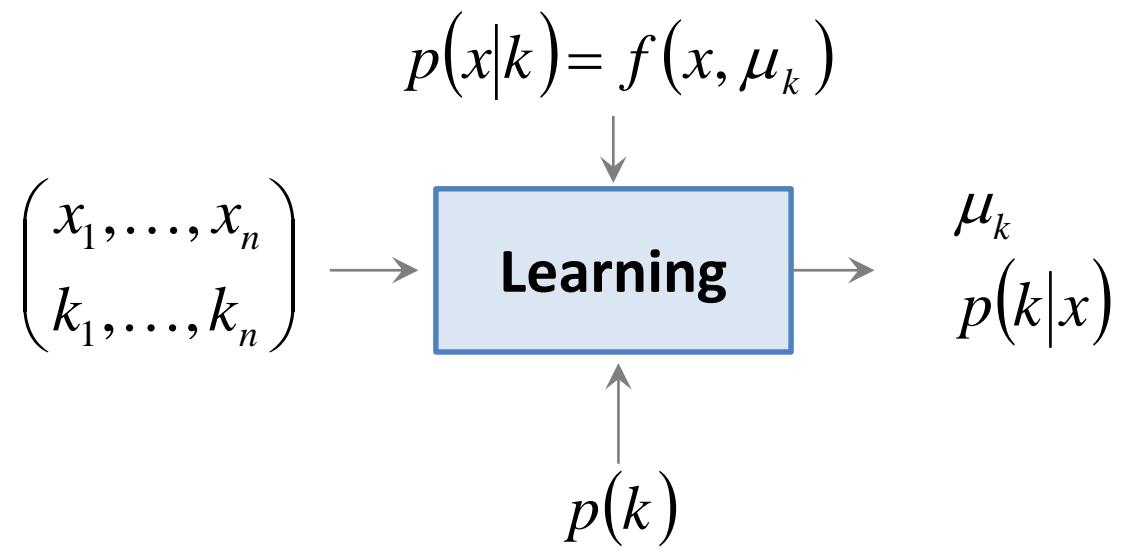
    return tpr, fpr, auc
```

1.10. Unsupervised learning



Unsupervised learning

EM algorithm



Unsupervised learning

EM algorithm



Unsupervised learning

EM algorithm



Unsupervised learning

EM algorithm



$$\mu_{\bullet} = \frac{1}{4}(0 + 0 + 3 + 5) = 2$$

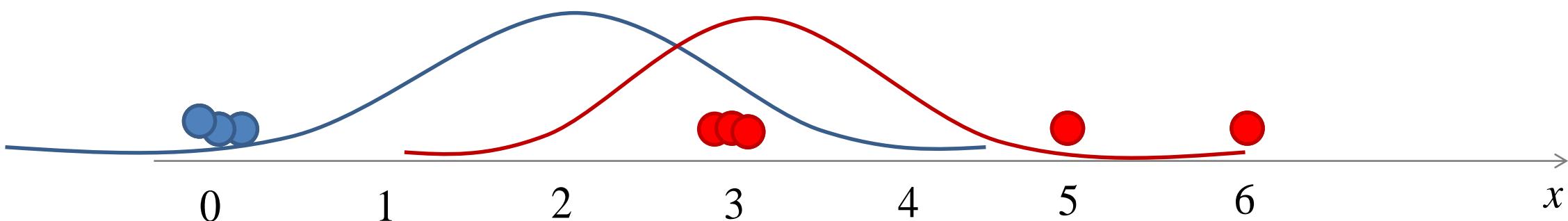
$$\sigma_{\bullet} = \frac{1}{4} \left(2^2 + 2^2 + 1^2 + 3^2 \right) = 4.5$$

$$\mu_{\bullet} = \frac{1}{4}(0 + 3 + 3 + 6) = 3$$

$$\sigma_{\bullet} = \frac{1}{4} \left(3^2 + 0^2 + 0^2 + 3^2 \right) = 4.5$$

Unsupervised learning

EM algorithm



$$\mu_{\bullet} = \frac{1}{4}(0 + 0 + 3 + 5) = 2$$

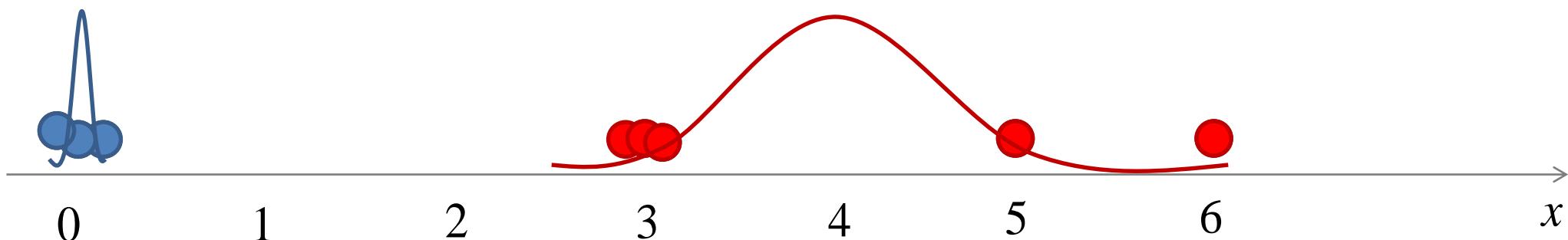
$$\sigma_{\bullet} = \frac{1}{4} \left(2^2 + 2^2 + 1^2 + 3^2 \right) = 4.5$$

$$\mu_{\bullet} = \frac{1}{4}(0 + 3 + 3 + 6) = 3$$

$$\sigma_{\bullet} = \frac{1}{4} \left(3^2 + 0^2 + 0^2 + 3^2 \right) = 4.5$$

Unsupervised learning

EM algorithm



$$\mu_{\bullet} = \frac{1}{3}(0 + 0 + 0) = 0$$

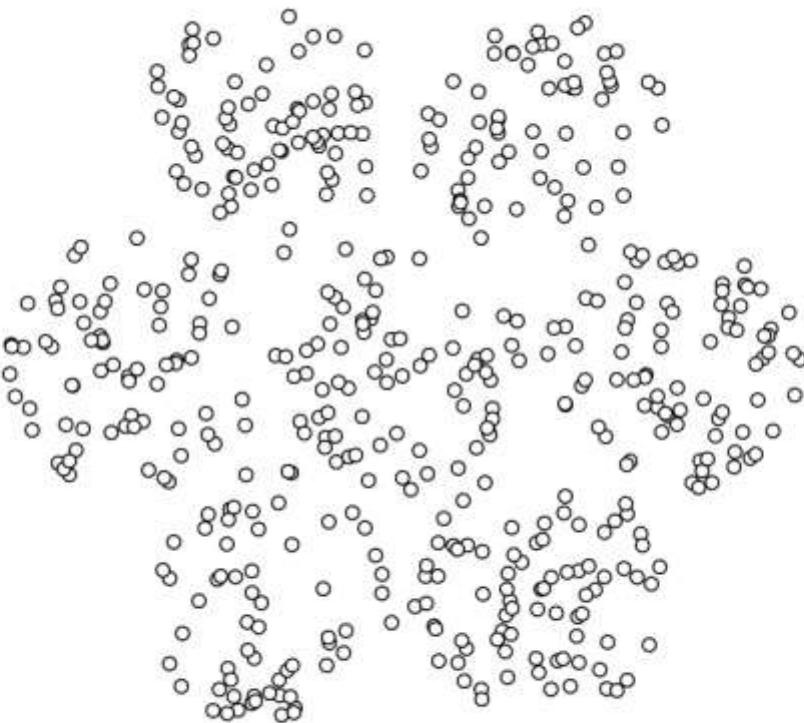
$$\sigma_{\bullet} = \frac{1}{3}(0^2 + 0^2 + 0^2) = 0$$

$$\mu_{\bullet} = \frac{1}{5}(3 + 3 + 3 + 5 + 6) = 4$$

$$\sigma_{\bullet} = \frac{1}{5}(1^2 + 1^2 + 1^2 + 1^2 + 2^2) = 1.6$$

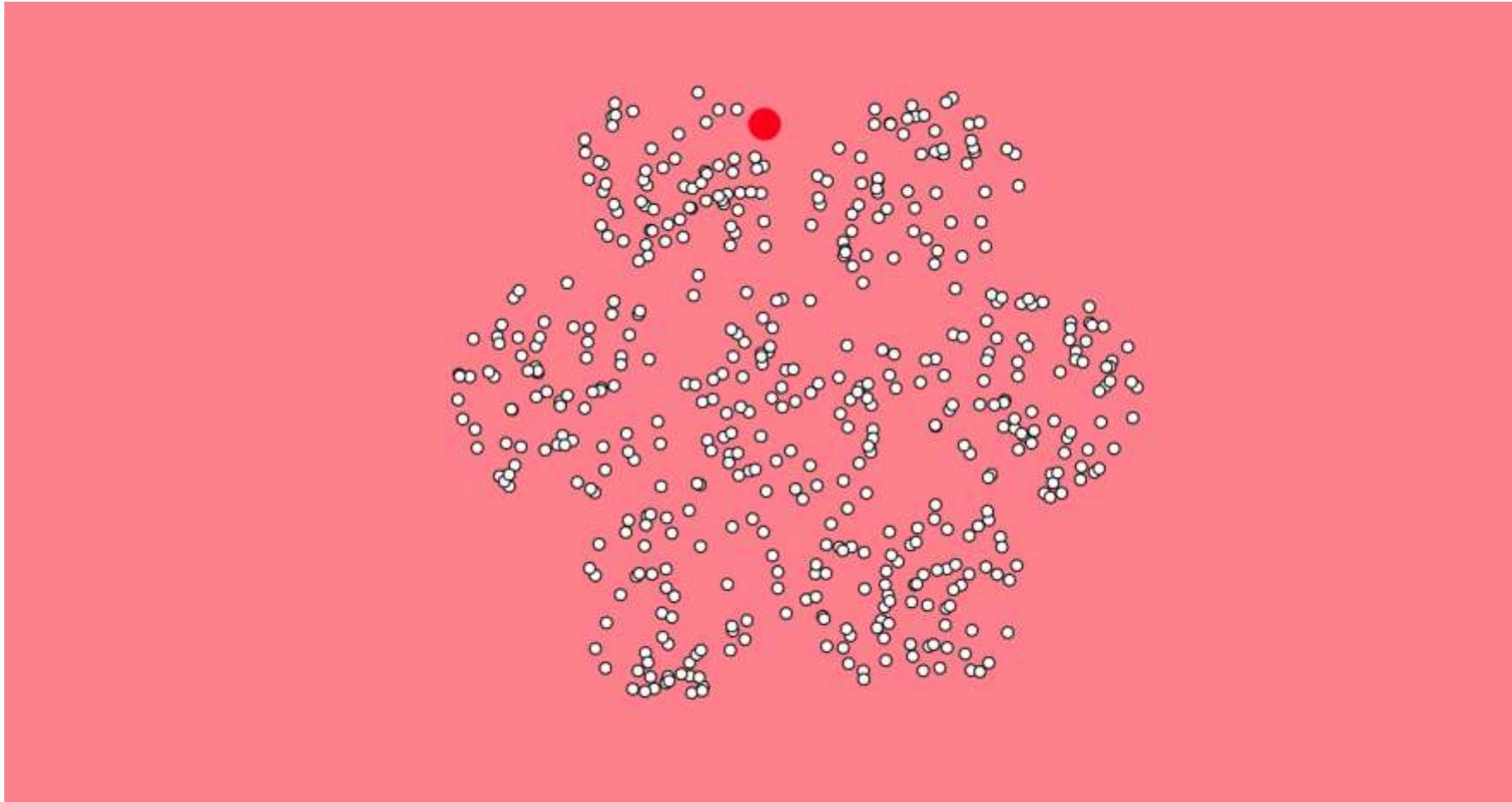
Unsupervised learning

K-means



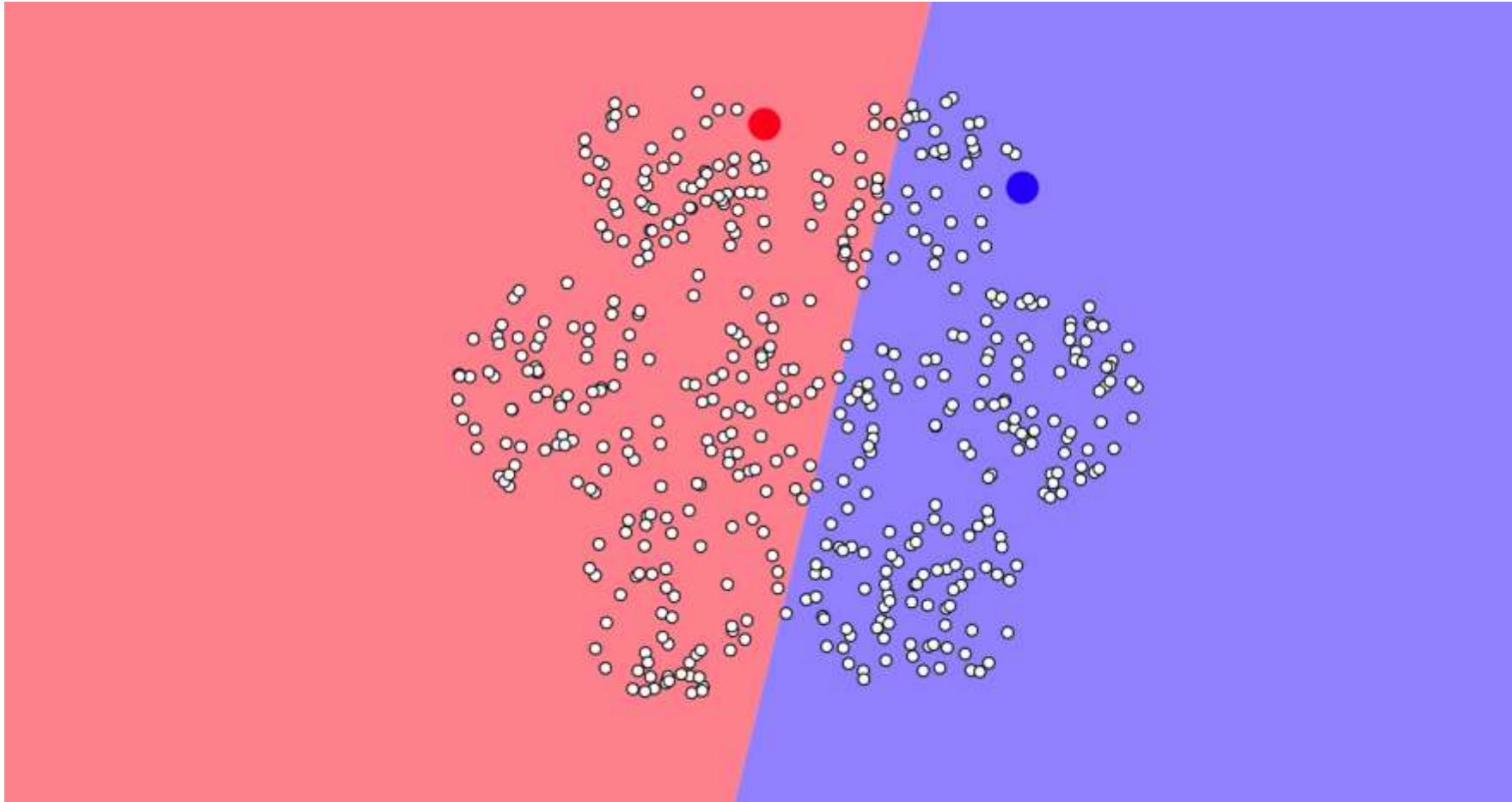
Unsupervised learning

K-means



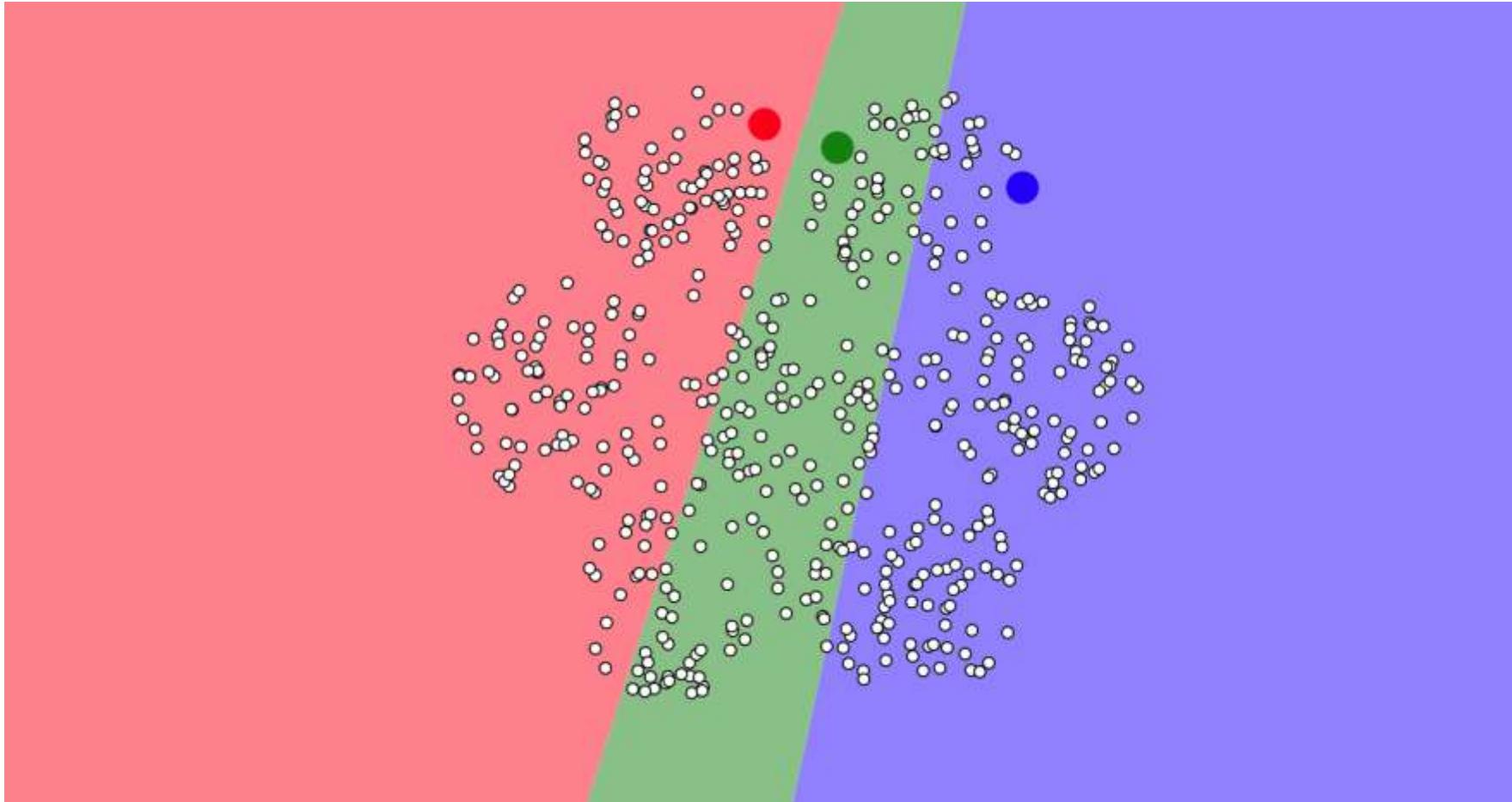
Unsupervised learning

K-means



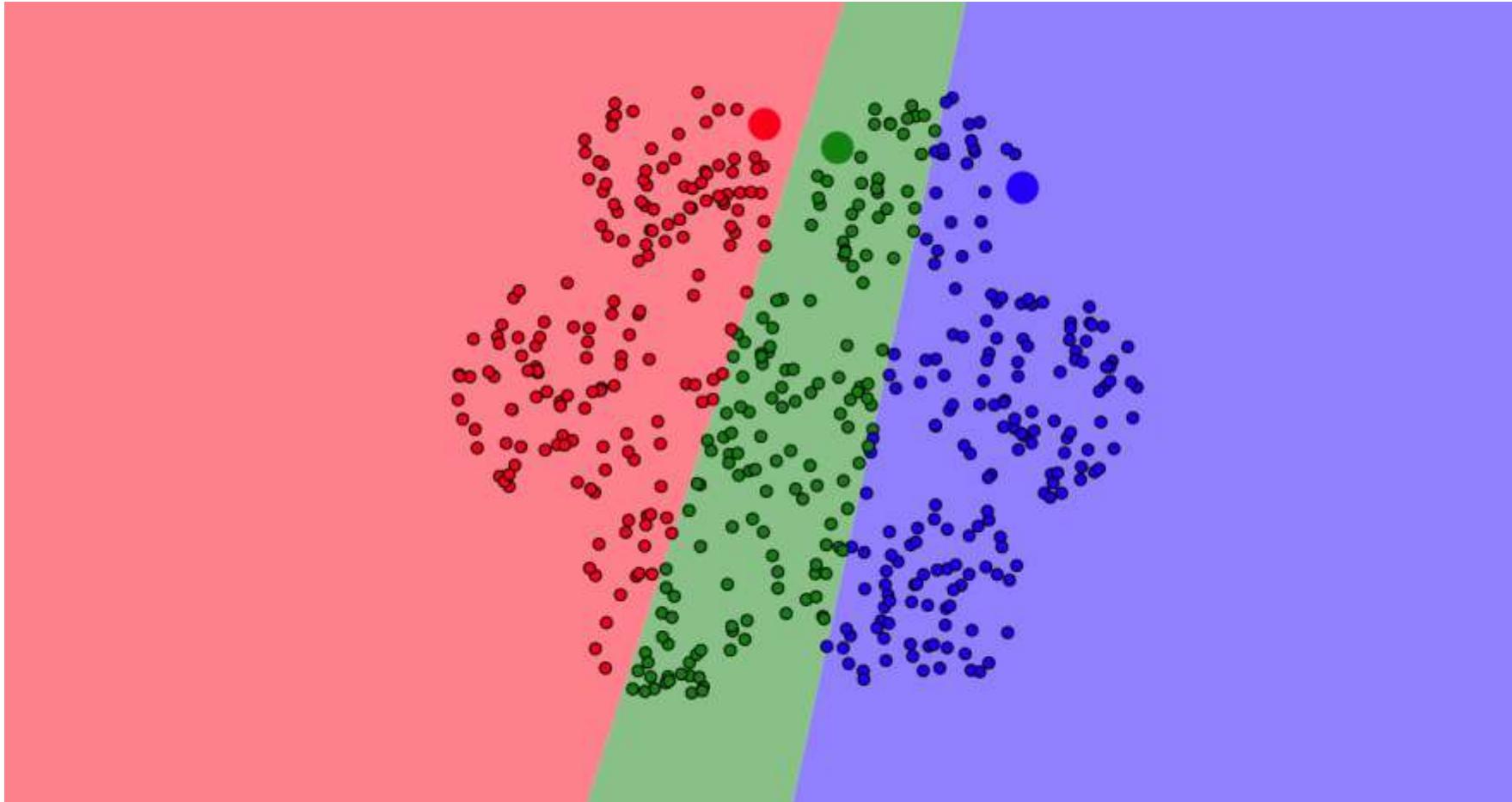
Unsupervised learning

K-means



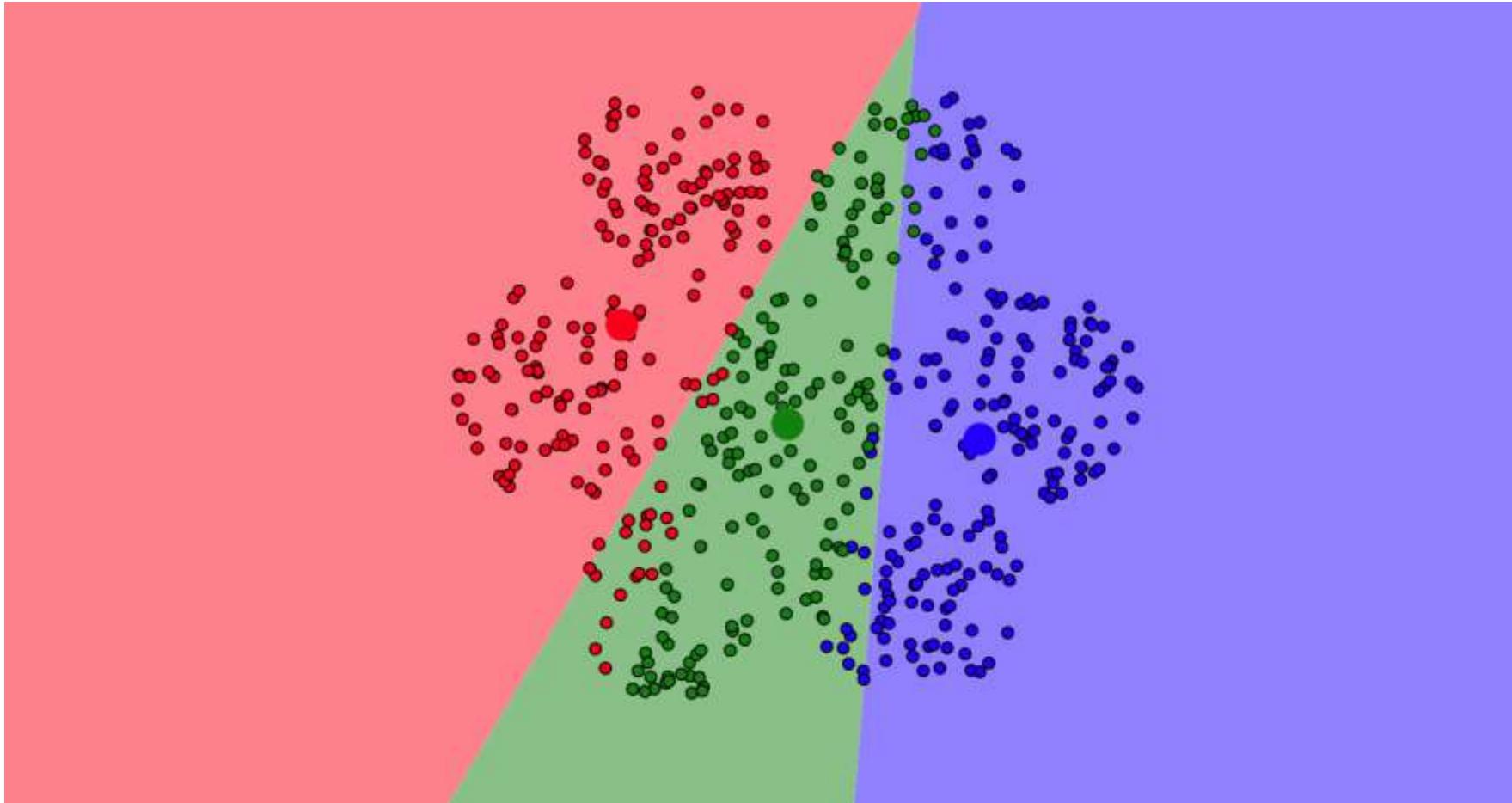
Unsupervised learning

K-means



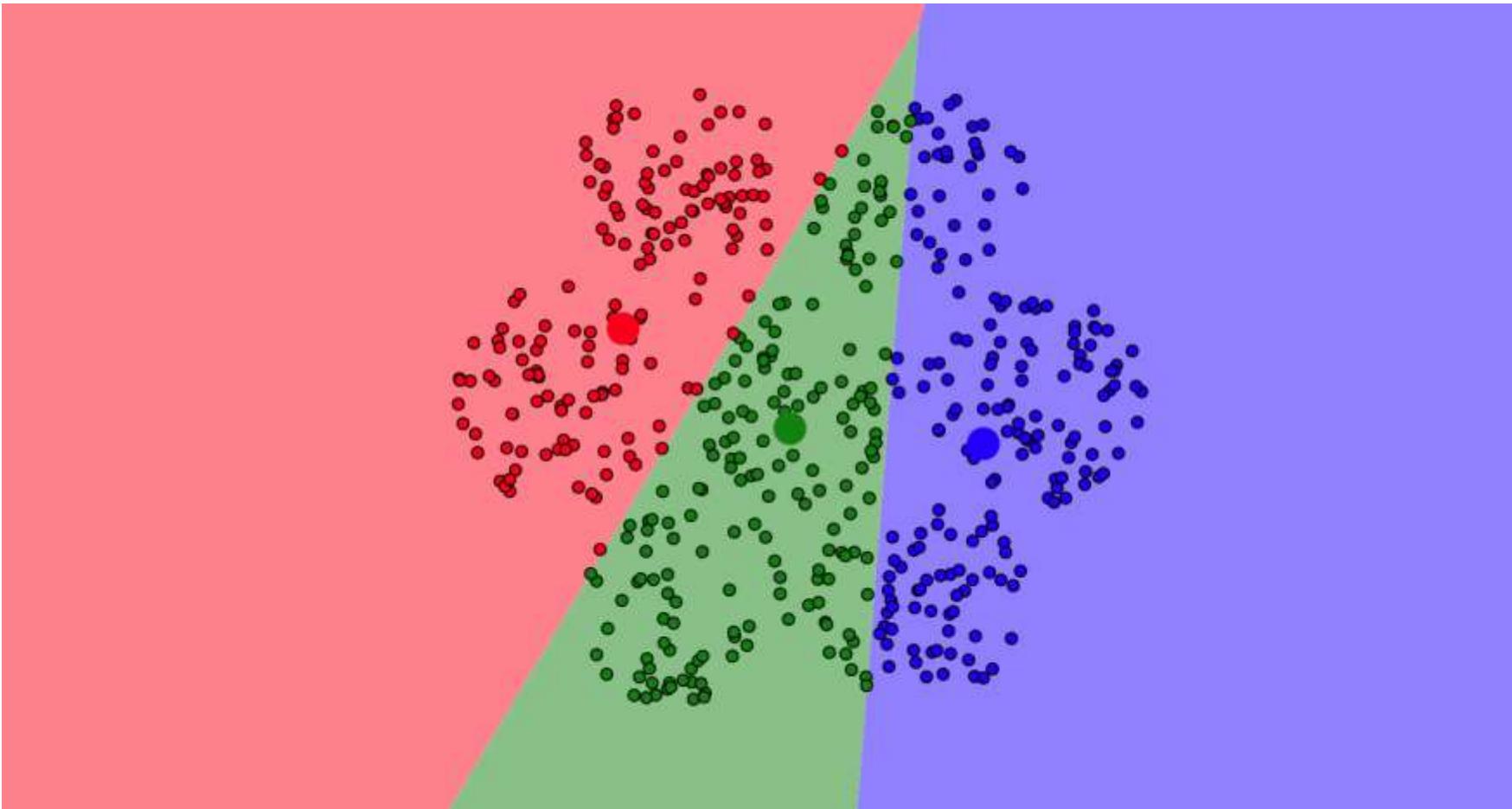
Unsupervised learning

K-means



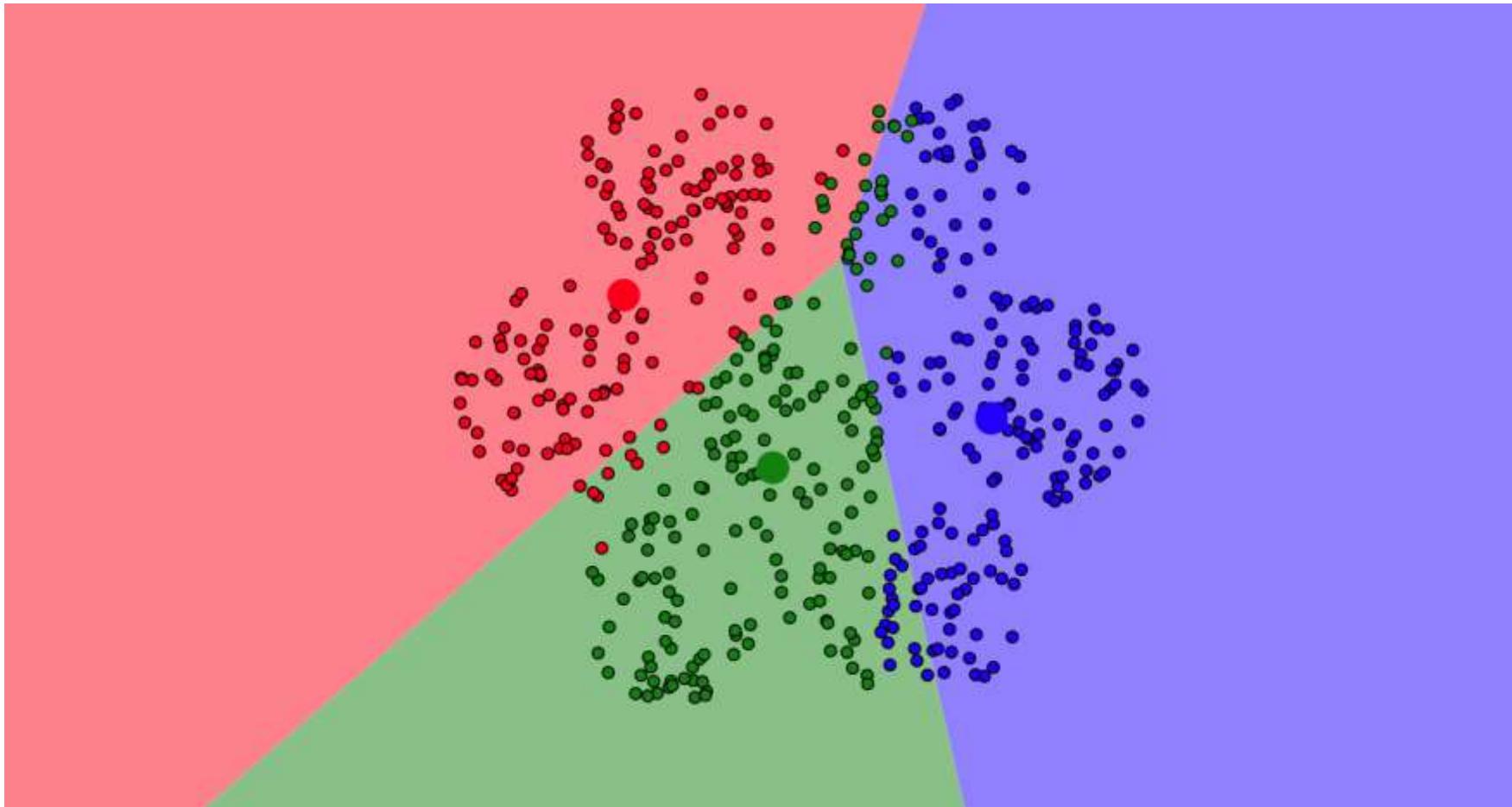
Unsupervised learning

K-means



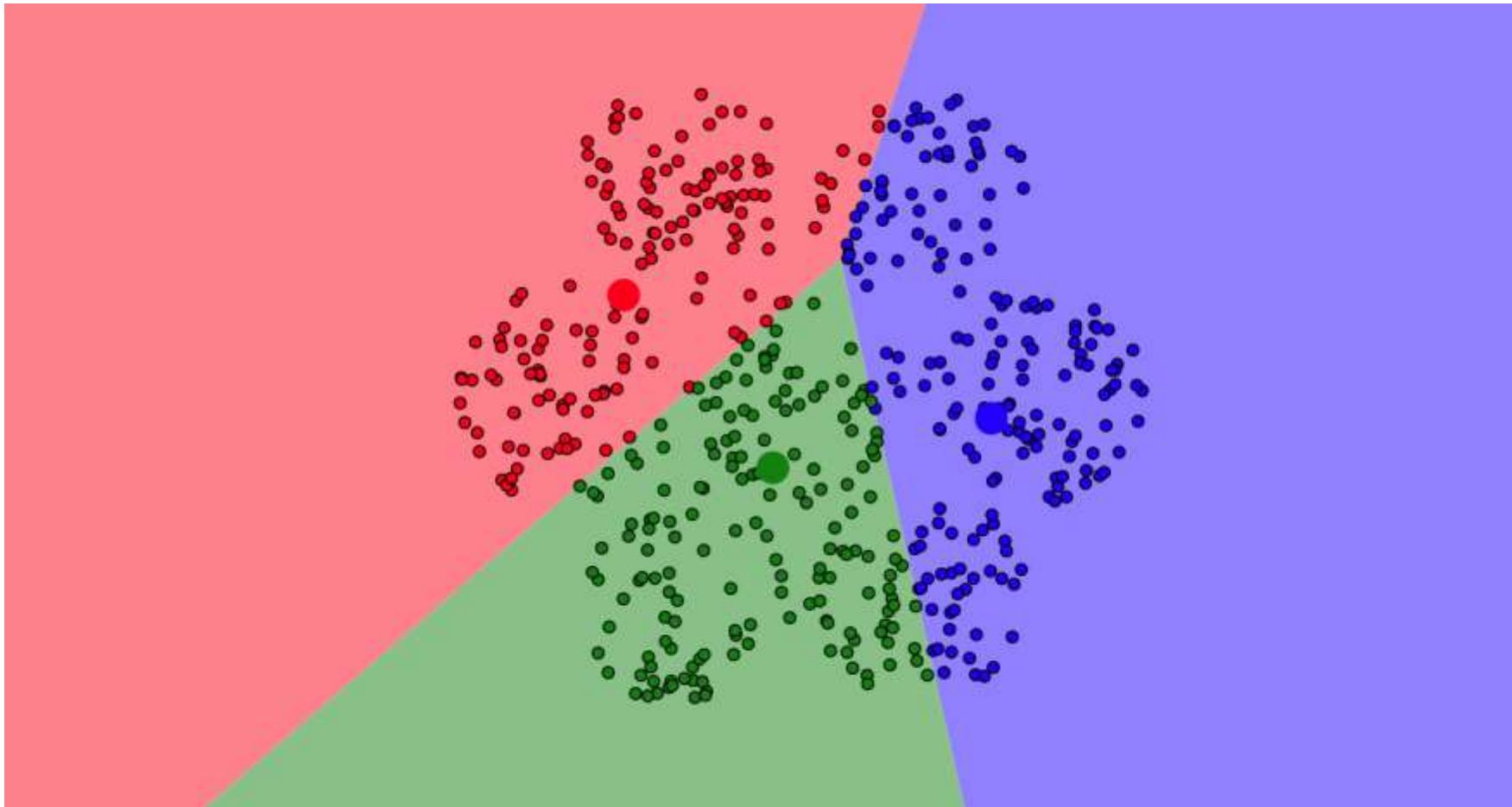
Unsupervised learning

K-means



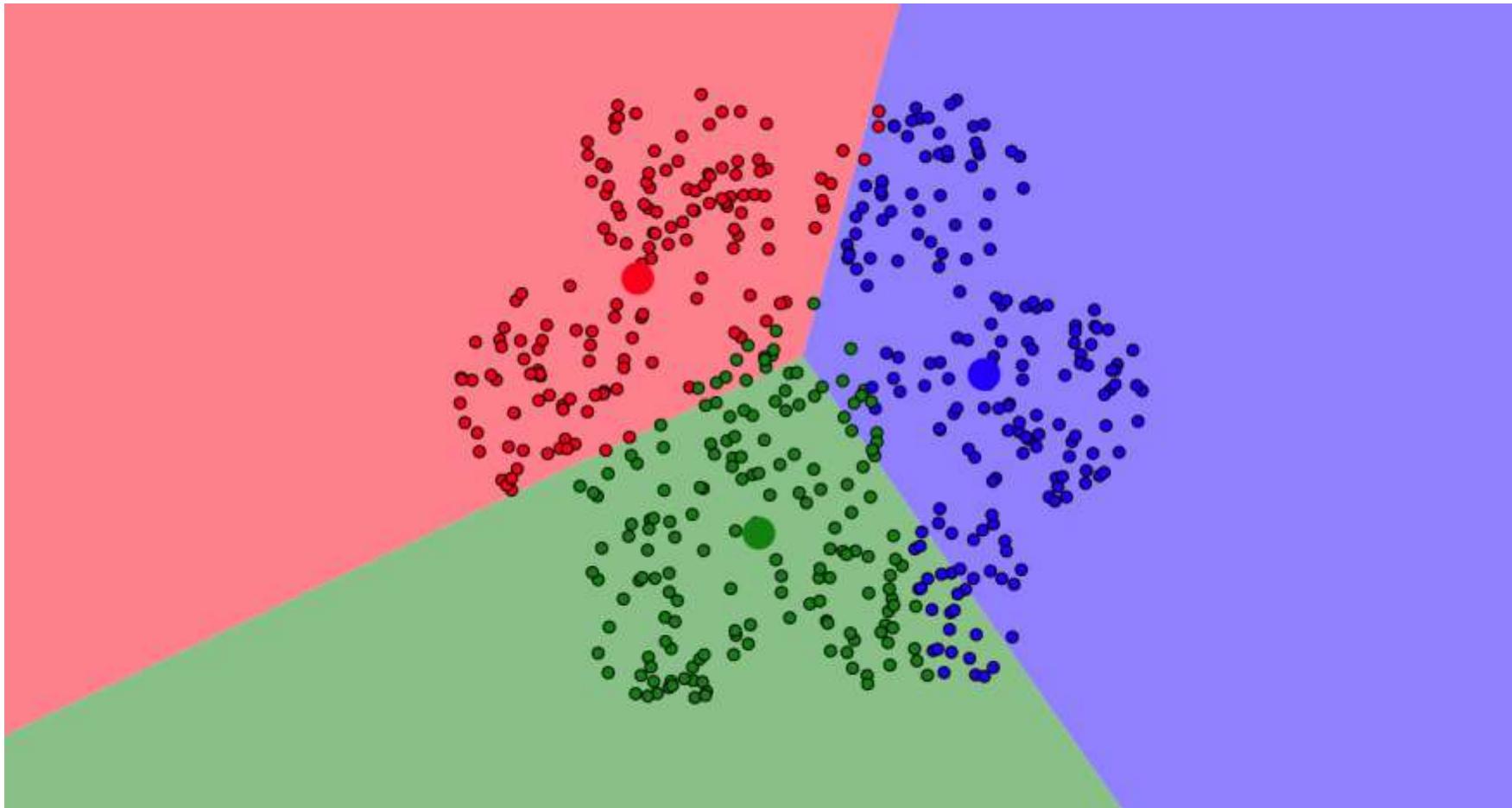
Unsupervised learning

K-means

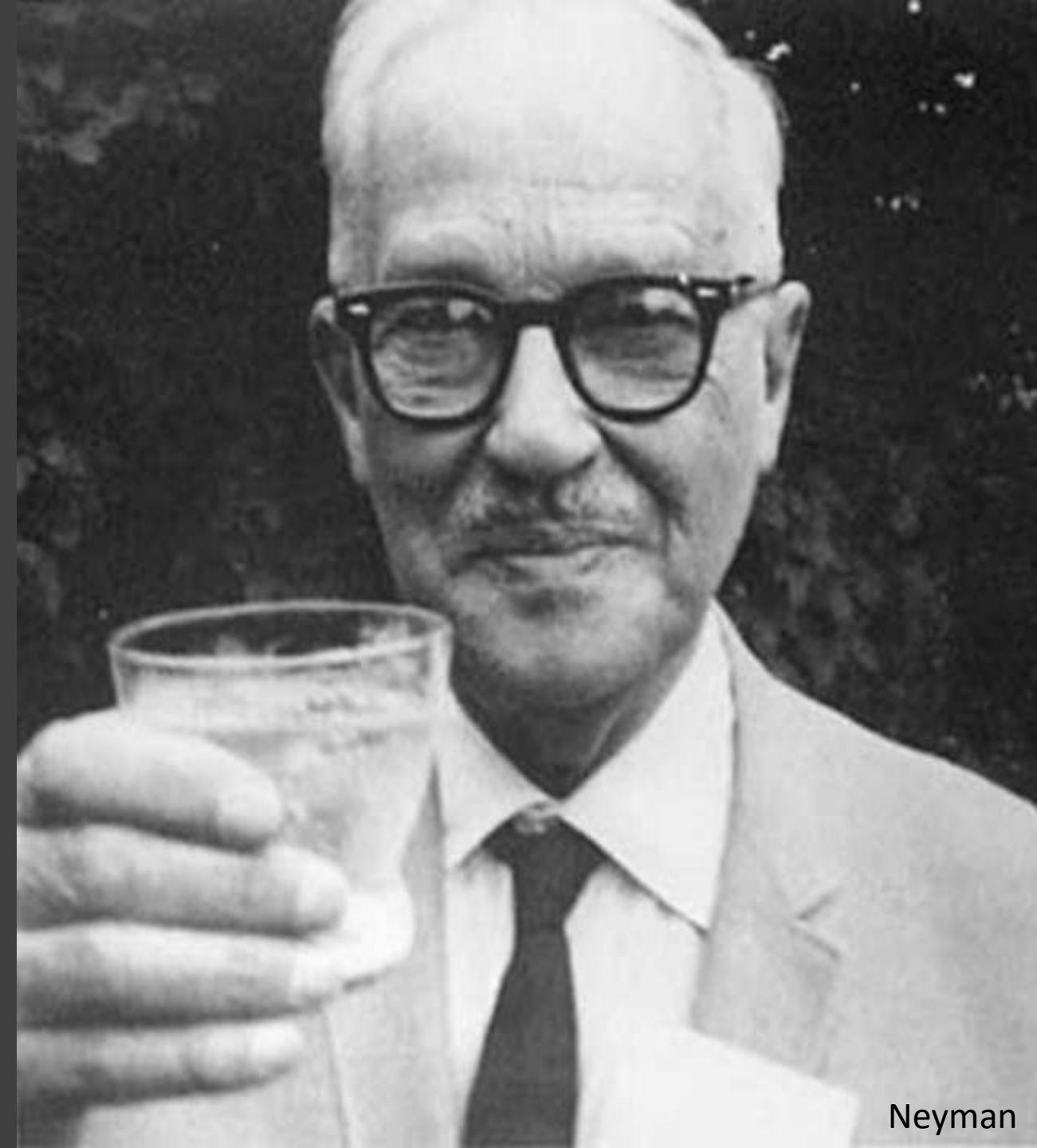


Unsupervised learning

K-means



1.11 Non-Bayesian approach



Neyman

Neyman-Pearson approach

Limitations of the Bayesian approach

$x \in X$ feature

$k \in K$ hidden state

$p(x|k)$ conditional probability to expose x by state k

$p(k)$ priory probability of the state

$W : K \times K \rightarrow R$ penalty function

Neyman-Pearson approach

Limitations of the Bayesian approach

$x \in X$ feature

$k \in K$ hidden state

$p(x|k)$ conditional probability to expose x by state k

$p(k)$ ~~prior probability of the state~~

$-W:K \times K \rightarrow R$ ~~penalty function~~

Neyman-Pearson approach

Limitations of the Bayesian approach: example

$x \in X$ heartbeat

$k \in K$ state: normal, sick

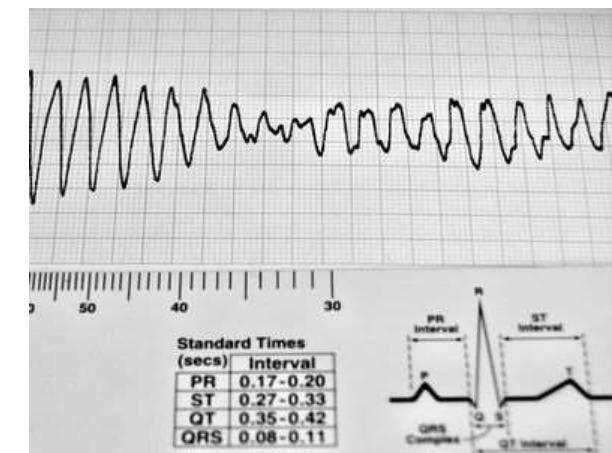
$p(x|k)$ distribution of the heartbeat for
normal and sick

$p(k)$???

W(normal, sick)

W(sick, normal)

???



Neyman-Pearson approach

The problem

$$x \in X \quad k \in K = \{normal, risky\}$$

$$p(x|k = norm) \quad X_{norm} \cup X_{risk} = X$$

$$p(x|k = risk) \quad X_{norm} \cap X_{risk} = 0$$

$$\text{error type 1} = P(\text{false alarm}) = \sum_{x \in X_{risk}} p(x|norm) \rightarrow \min$$

$$\text{error type 2} = P(\text{miss the risk}) = \sum_{x \in X_{norm}} p(x|risk) \leq \varepsilon$$

$$X_{norm}, X_{risk}=?$$

input

output

Neyman-Pearson approach

The problem

$$x \in X \quad k \in K = \{normal, risky\}$$

$$\alpha_{norm}(x) = \begin{cases} 1 & \text{when } x \in X_{norm} \\ 0 & \text{when } x \notin X_{norm} \end{cases}$$

$$\alpha_{risk}(x) = \begin{cases} 1 & \text{when } x \in X_{risk} \\ 0 & \text{when } x \notin X_{risk} \end{cases}$$

Neyman-Pearson approach

The problem

$$\sum_{x \in X_{risk}} p(x|norm) \rightarrow min$$

$$\sum_{x \in X_{norm}} p(x|risk) \leq \varepsilon$$

$$\left\{ \begin{array}{l} \sum_{x \in X} \alpha_{risk}(x) \cdot p(x|norm) \rightarrow min \\ - \sum_{x \in X} \alpha_{norm}(x) \cdot p(x|norm) \geq -\varepsilon \\ \alpha_{norm}(x) + \alpha_{risk}(x) = 1 \quad \forall x \in X \\ \alpha_{norm}(x) \geq 0 \quad \forall x \in X \\ \alpha_{risk}(x) \geq 0 \quad \forall x \in X \end{array} \right.$$

Neyman-Pearson approach

$p_1 \cdot x_1 + \dots + p_n \cdot x_n \rightarrow \max$	$b_1 \cdot y_1 + \dots + b_m \cdot y_m \rightarrow \min$
$a_{11} \cdot x_1 + \dots + a_{1n} \cdot x_n \leq b_1$	$y_1 \geq 0$
$a_{k1} \cdot x_1 + \dots + a_{kn} \cdot x_n \leq b_k$	$y_k \geq 0$
$a_{k+1,1} \cdot x_1 + \dots + a_{k+1,n} \cdot x_n = b_{k+1}$	$y_{k+1} - c\vartheta. \text{ neperem.}$
$a_{m1} \cdot x_1 + \dots + a_{mn} \cdot x_n = b_m$	$y_m - c\vartheta. \text{ neperem.}$
$x_1 \geq 0$	$a_{11} \cdot y_1 + \dots + a_{m1} \cdot y_m \geq p_1$
$x_s \geq 0$	$a_{1s} \cdot y_1 + \dots + a_{ms} \cdot y_m \geq p_s$
$x_{s+1} - c\vartheta. \text{ neperem.}$	$a_{1,s+1} \cdot y_1 + \dots + a_{m,s+1} \cdot y_m = p_{s+1}$
$x_n - c\vartheta. \text{ neperem.}$	$a_{1n} \cdot y_1 + \dots + a_{mn} \cdot y_m = p_n$

Neyman-Pearson approach

$$\left\{ \begin{array}{l}
 p_1 \cdot x_1 + \dots + p_n \cdot x_n \rightarrow \max \\
 a_{11} \cdot x_1 + \dots + a_{1n} \cdot x_n \leq b_1 \\
 a_{k1} \cdot x_1 + \dots + a_{kn} \cdot x_n \leq b_k \\
 a_{k+1,1} \cdot x_1 + \dots + a_{k+1,n} \cdot x_n = b_{k+1} \\
 a_{m1} \cdot x_1 + \dots + a_{mn} \cdot x_n = b_m \\
 x_1 \geq 0 \\
 x_s \geq 0 \\
 x_{s+1} - c\sigma. \text{ нерем.} \\
 x_n - c\sigma. \text{ нерем.}
 \end{array} \right. \quad \left\{ \begin{array}{l}
 b_1 \cdot y_1 + \dots + b_m \cdot y_m \rightarrow \min \\
 y_1 \geq 0 \\
 y_k \geq 0 \\
 y_{k+1} - c\sigma. \text{ нерем.} \\
 y_m - c\sigma. \text{ нерем.} \\
 a_{11} \cdot y_1 + \dots + a_{m1} \cdot y_m \geq p_1 \\
 a_{1s} \cdot y_1 + \dots + a_{ms} \cdot y_m \geq p_s \\
 a_{1,s+1} \cdot y_1 + \dots + a_{m,s+1} \cdot y_m = p_{s+1} \\
 a_{1n} \cdot y_1 + \dots + a_{mn} \cdot y_m = p_n
 \end{array} \right.$$

$$(a_{11} \cdot x_1 + \dots + a_{1n} \cdot x_n - b_1) \cdot y_1 = 0$$

$$(a_{11} \cdot y_1 + \dots + a_{m1} \cdot y_m - p_1) \cdot x_1 = 0$$

Neyman-Pearson approach

Solution

$$\left\{ \begin{array}{l} \sum_{x \in X} \alpha_{risk}(x) \cdot p(x|norm) \rightarrow min \\ - \sum_{x \in X} \alpha_{norm}(x) \cdot p(x|norm) \geq -\varepsilon \\ \alpha_{norm}(x) + \alpha_{risk}(x) = 1 \quad \forall x \in X \\ \alpha_{norm}(x) \geq 0 \quad \forall x \in X \\ \alpha_{risk}(x) \geq 0 \quad \forall x \in X \end{array} \right.$$

$$\left\{ \begin{array}{l} -\varepsilon \cdot \tau + 1 \cdot t(x_1) + \dots + 1 \cdot t(x_i) \rightarrow max \\ \tau \geq 0 \\ t(x) \text{ free var} \\ -p(x|risk) \cdot \tau + 1 \cdot t(x) \leq 0 \\ 1 \cdot t(x) \leq p(x|norm) \end{array} \right.$$

Neyman-Pearson approach

Solution

$$x \in X_{norm} \Rightarrow \alpha_{norm}(x) = 1, \alpha_{risk}(x) = 0 \Rightarrow \begin{cases} -p(x|risk) \cdot \tau + 1 \cdot t(x) = 0 \\ 1 \cdot t(x) < p(x|norm) \end{cases} \Rightarrow$$
$$\Rightarrow p(x|risk) \cdot \tau \leq p(x|norm)$$

$$x \in X_{risk} \Rightarrow \alpha_{norm}(x) = 0, \alpha_{risk}(x) = 1 \Rightarrow \begin{cases} -p(x|risk) \cdot \tau + 1 \cdot t(x) \leq 0 \\ 1 \cdot t(x) = p(x|norm) \end{cases} \Rightarrow$$
$$\Rightarrow p(x|risk) \cdot \tau \geq p(x|norm)$$

Neyman-Pearson approach

Solution

$$\frac{p(x|norm)}{p(x|risk)} \stackrel{\text{norm}}{\gtrless} \stackrel{\text{risk}}{\tau}$$

Neyman-Pearson approach

Some Variations: two **risky** states

$$\sum_{x \in X_{risk}} p(x|norm) \rightarrow \min$$

$$\sum_{x \in X_{norm}} p(x|risk1) \leq \varepsilon$$

$$\sum_{x \in X_{norm}} p(x|risk2) \leq \varepsilon$$

Neyman-Pearson approach

Some Variations: two **normal** states

$$\sum_{x \in X_{risk}} p(x|norm1) + \sum_{x \in X_{risk}} p(x|norm2) \rightarrow min$$

$$\sum_{x \in X_{norm}} p(x|risk) \leq \varepsilon$$

Neyman-Pearson approach

Minimax problem

$$\max \left\{ \sum_{x \notin X_1} p(x|1), \sum_{x \notin X_2} p(x|2) \dots, \sum_{x \notin X_K} p(x|K) \right\} \rightarrow \min$$

1.12. Labeling



M Schlesinger



D Schlesinger



Flach



Hlavác

Labeling

Definitions

T set of objects

$\in T$ t object

K set of labels

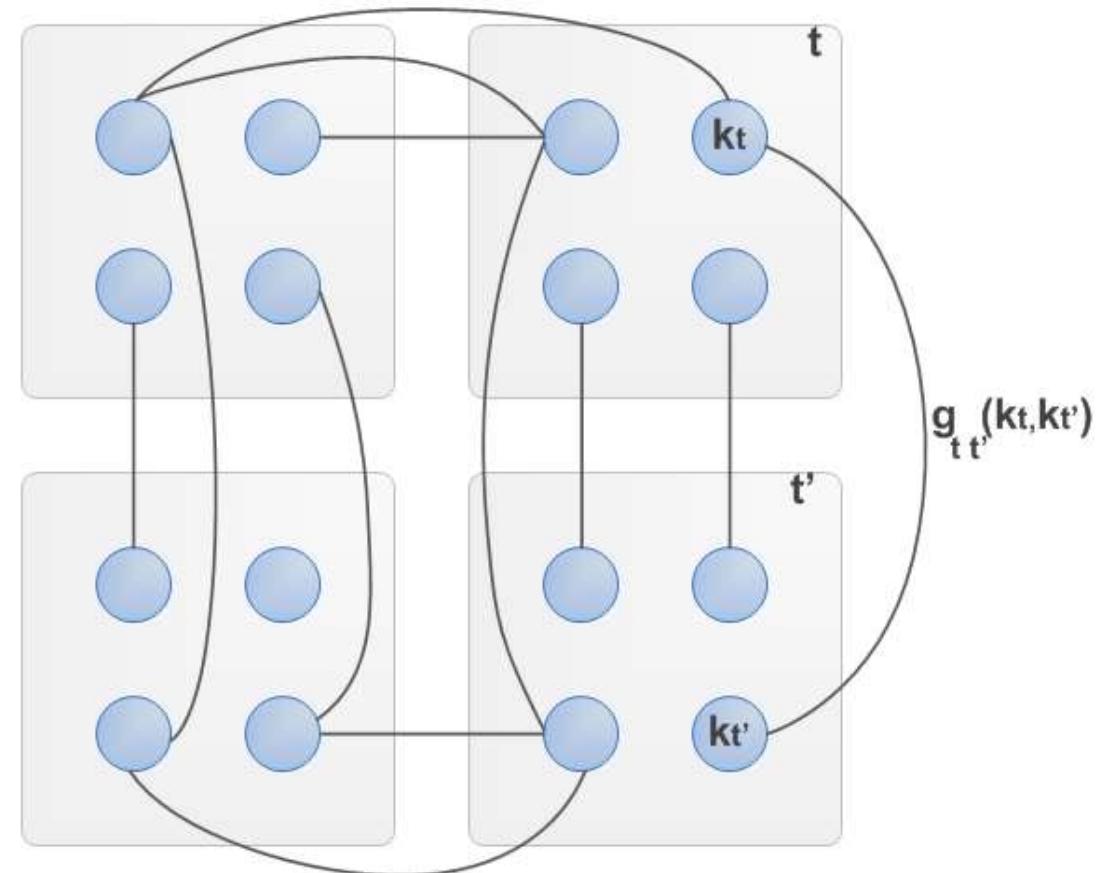
$k_t \in K$ label for object t

$\bar{k} : T \rightarrow K$ labeling

K^T all possible labelings

$t' \in \tau(t) \in T$ neighbor of t

$g_{tt'}(k_t, k_{t'})$ weight of an edge



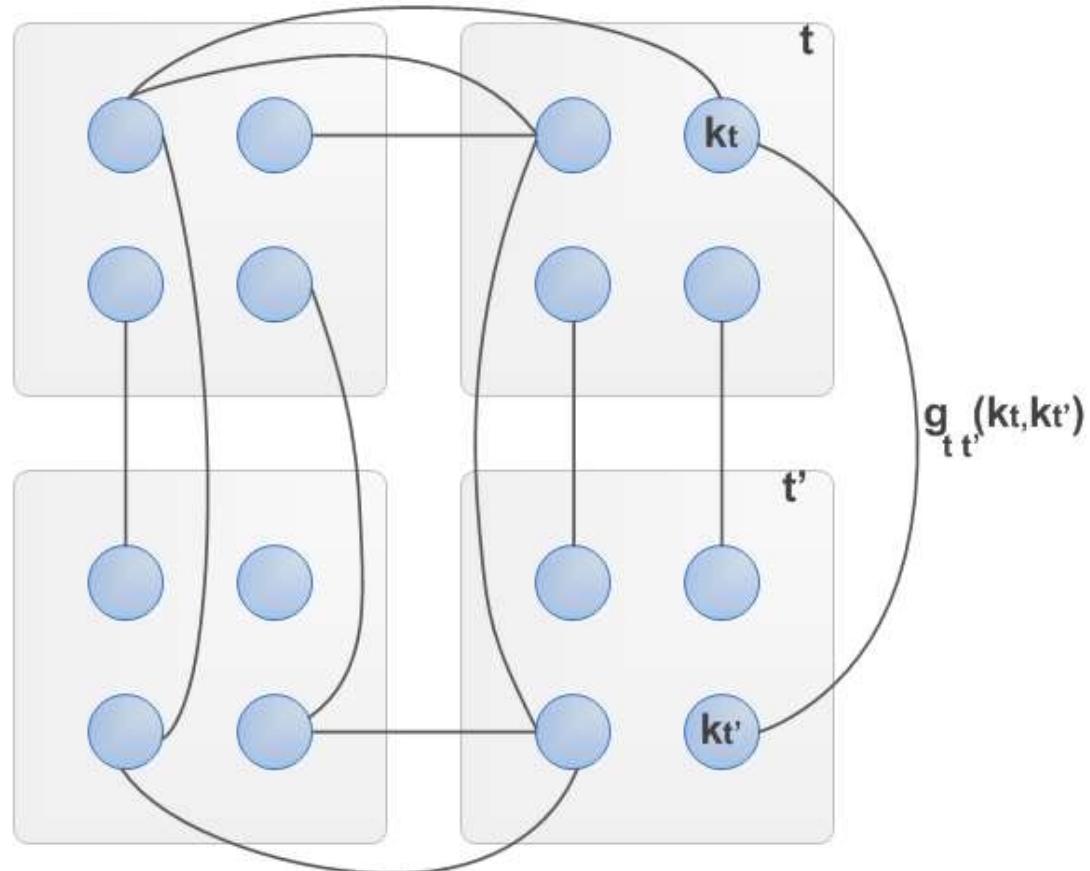
Labeling

The (OR, AND) problem

$$g_{tt'}(k_t, k_{t'}) \in \{0,1\}$$

Find the valid labeling

$$k^* = \underset{\bar{k} \in K^T}{\vee} \quad \underset{\substack{t \in T \\ t' \in \tau(t)}}{\&} \quad g_{tt'}(k_t, k_{t'})$$



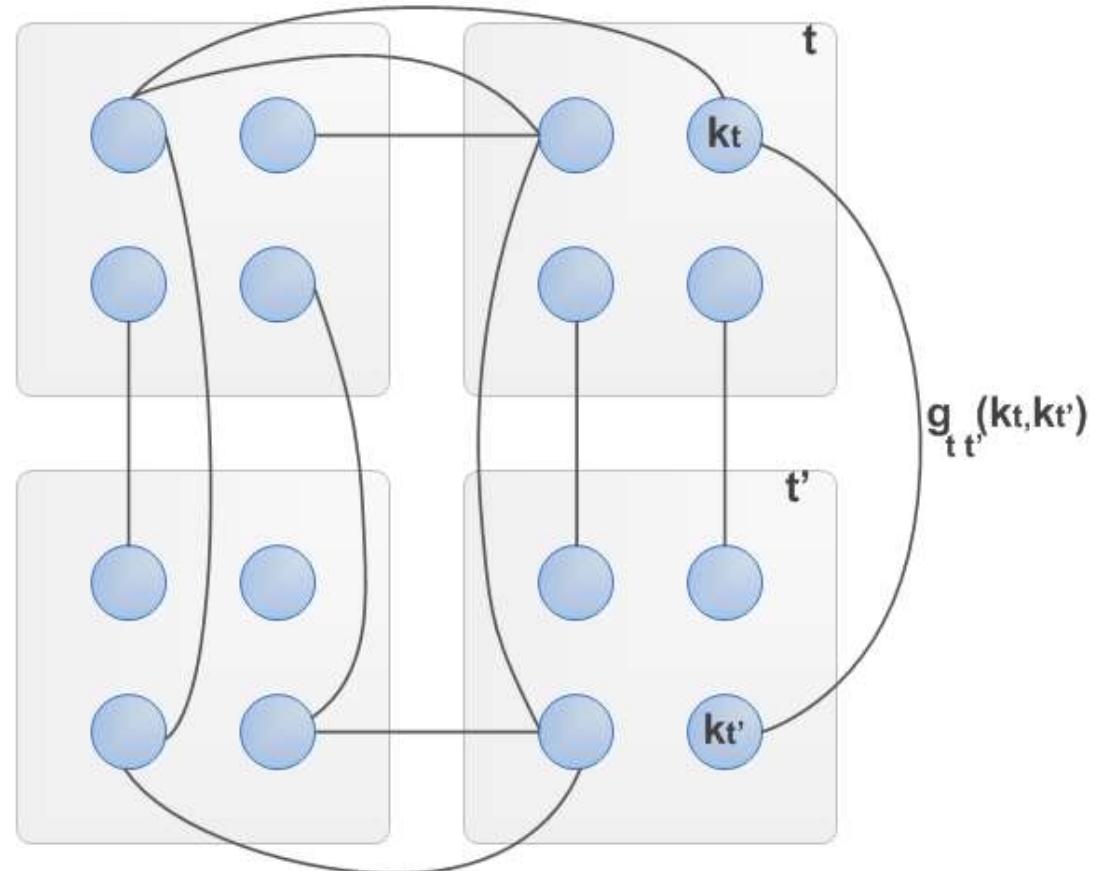
Labeling

The (MAX, +) problem

$$g_{tt'}(k_t, k_{t'}) \in R$$

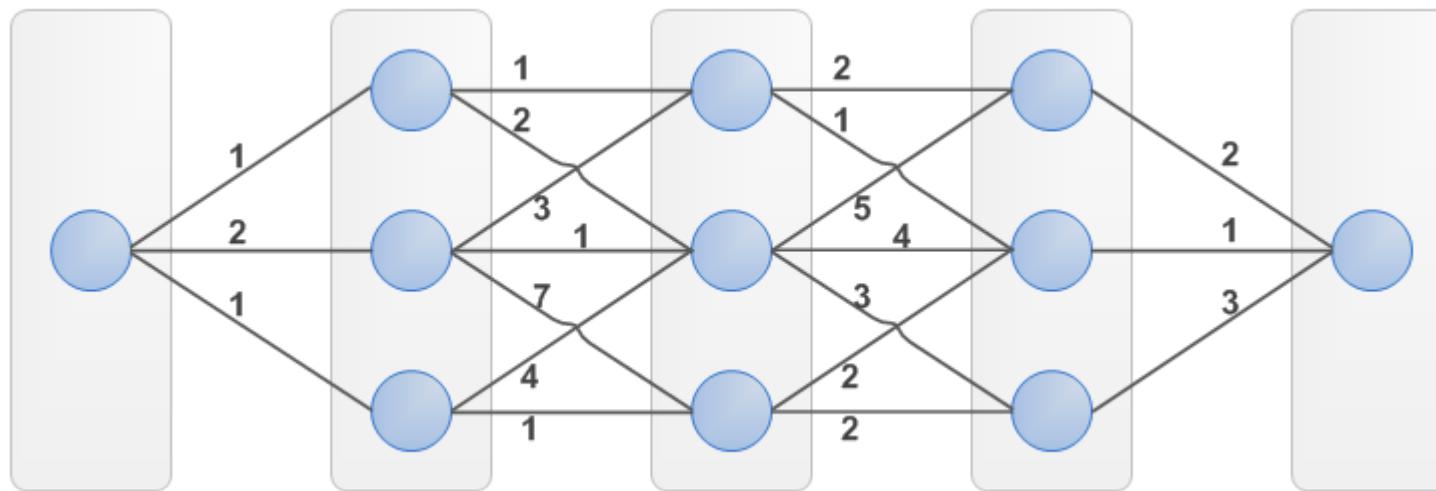
Find a labeling with maximal sum

$$k^* = \arg \max_{\bar{k} \in K^T} \sum_{\substack{t \in T \\ t' \in \tau(t)}} g_{tt'}(k_t, k_{t'})$$



Labeling

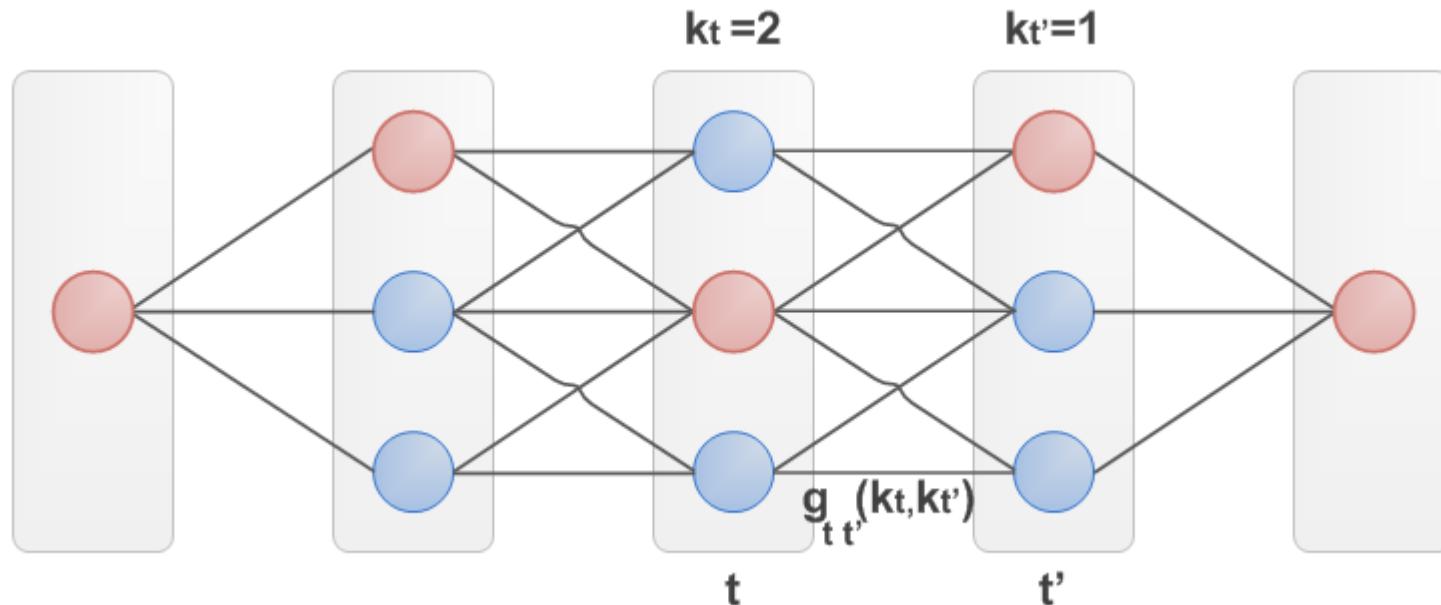
The (MAX, +) problem: the best path on graph



Labeling

The **(MAX, +)** problem: the best path on graph

$$k^* = \arg \max_{\bar{k} \in K^T} \sum_{\substack{t \in T \\ t' \in \tau(t)}} g_{tt'}(k_t, k_{t'})$$



Labeling

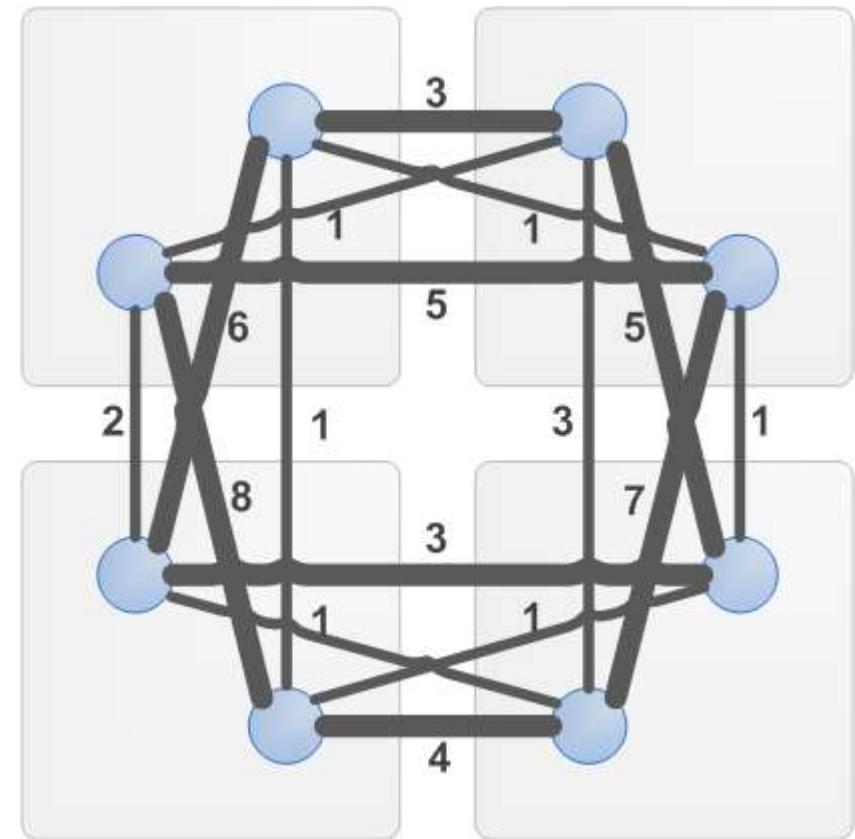
The (MIN, +) problem: travelling salesman problem

$T = \text{города}$

$K = \text{дороги}$

$g_{tt'}(k_t, k_{t'}) = \text{расстояние между городами}$

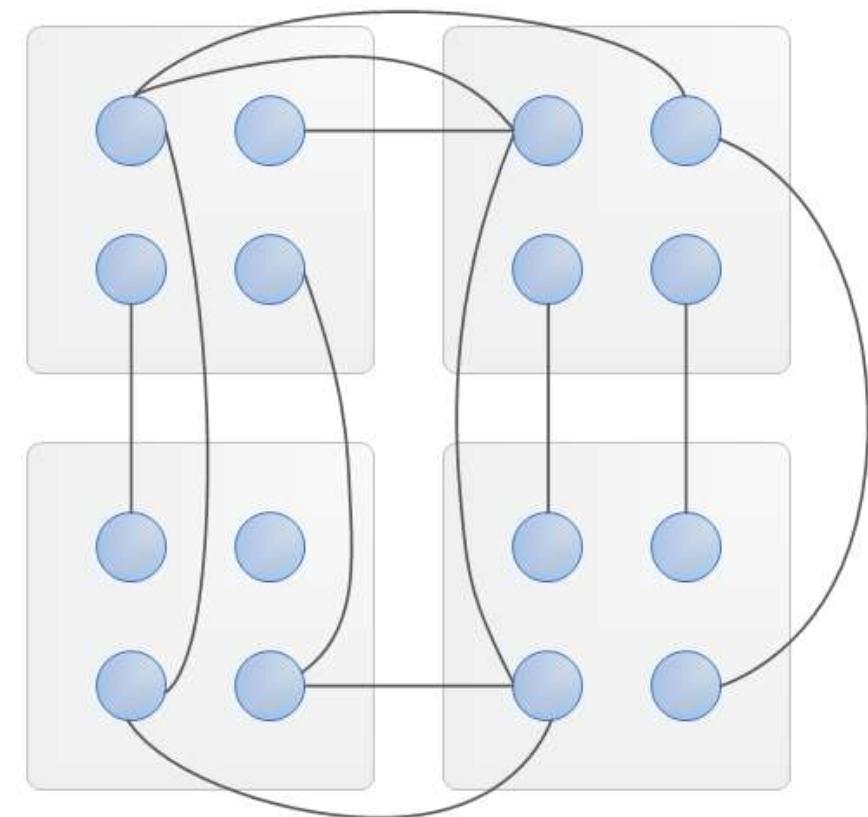
$$k^* = \min_{\bar{k} \in K^T} \sum_{\substack{t \in T \\ t' \in \tau(t)}} g_{tt'}(k_t, k_{t'})$$



Labeling

The (OR, AND) problem: example

$$k^* = \underset{\bar{k} \in K^T}{\vee} \quad \underset{t \in T}{\&} \quad g_{tt'}(k_t, k_{t'})$$

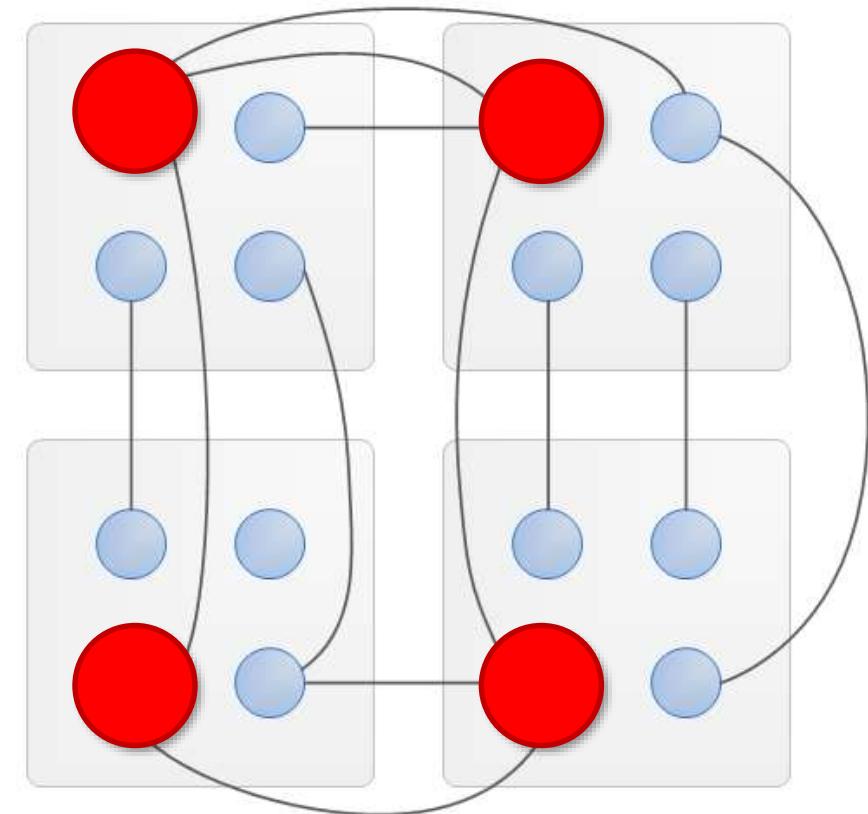


Labeling

The (OR, AND) problem: example

$$k^* = \underset{\bar{k} \in K^T}{\vee} \quad \underset{t \in T}{\&} \quad g_{tt'}(k_t, k_{t'})$$

Labels in all the neighbor objects should be connected by an edge



Labeling

The (MIN, MAX) problem: clustering

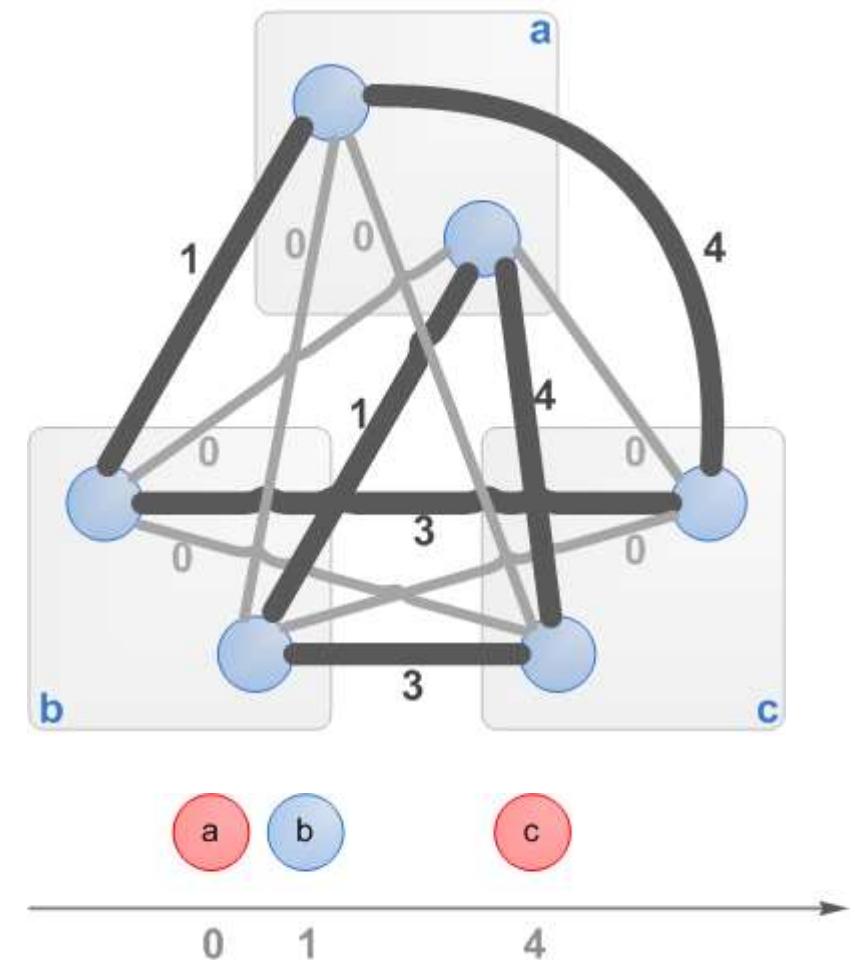
$T = \text{objects}$

$K = \text{clusters}$

$g_{tt'}(k_t, k_{t'}) = \text{distance between the objects}$

The distance between most different objects of the same cluster should be minimal

$$k^* = \min_{\bar{k} \in K^T} \max_{\substack{t \in T \\ t' \in \tau(t)}} g_{tt'}(k_t, k_{t'})$$



Labeling

The general labeling problem

(\oplus, \otimes)

$(+, \times)$ Count of valid labelings

(or, and) Valid labeling

$(\max, +)$ Travelling salesman problem

$(\min, +)$ Travelling salesman problem

(\max, \times) Optimization on Gibbs field

(\min, \times) Optimization on Gibbs field

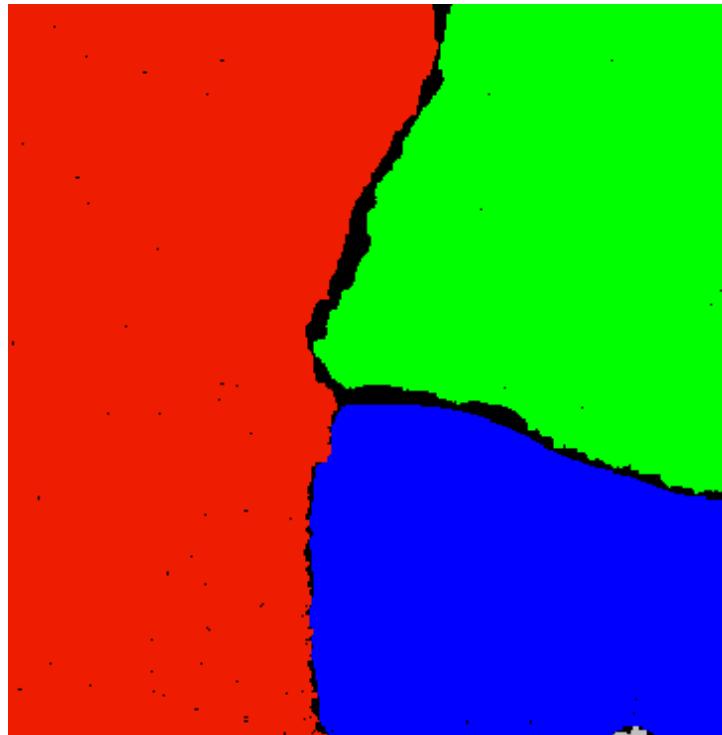
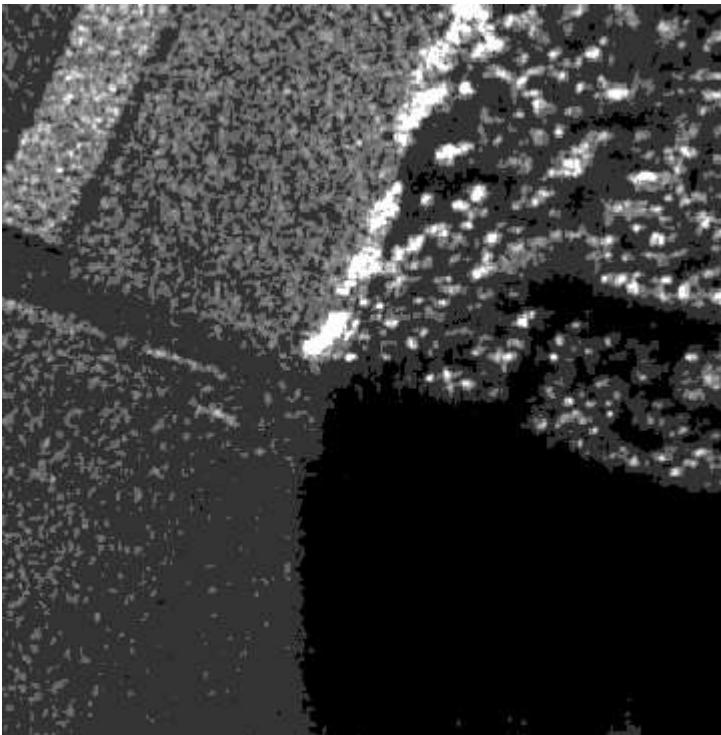
(\max, \min) Hamiltonian path

(\min, \max) Clustering

$$k^* = \bigoplus_{\bar{k} \in K^T} \bigotimes_{\substack{t \in T \\ t' \in \tau(t)}} g_{tt'}(k_t, k_{t'})$$

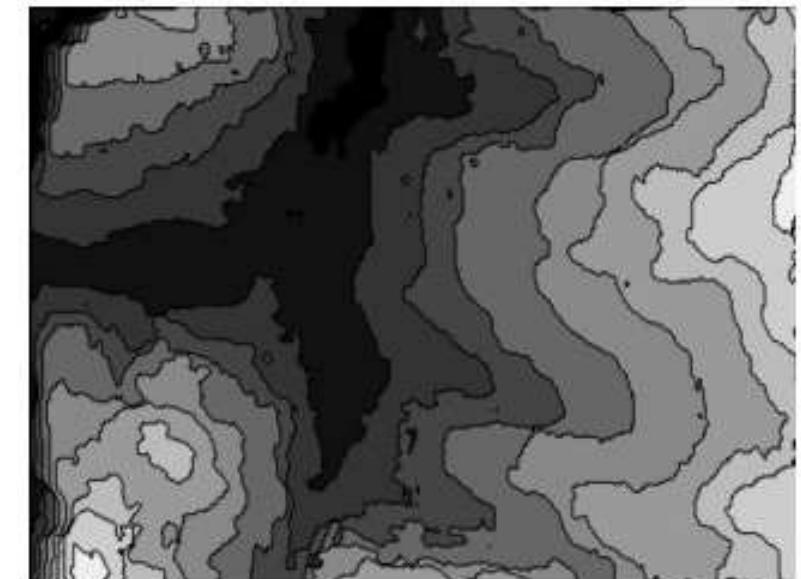
Labeling

Some applications: texture segmentation



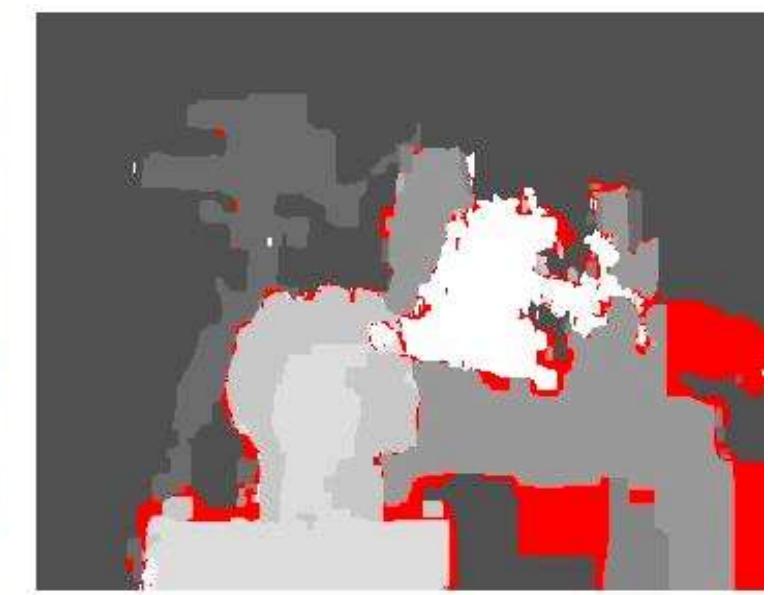
Labeling

Some applications: stereo vision



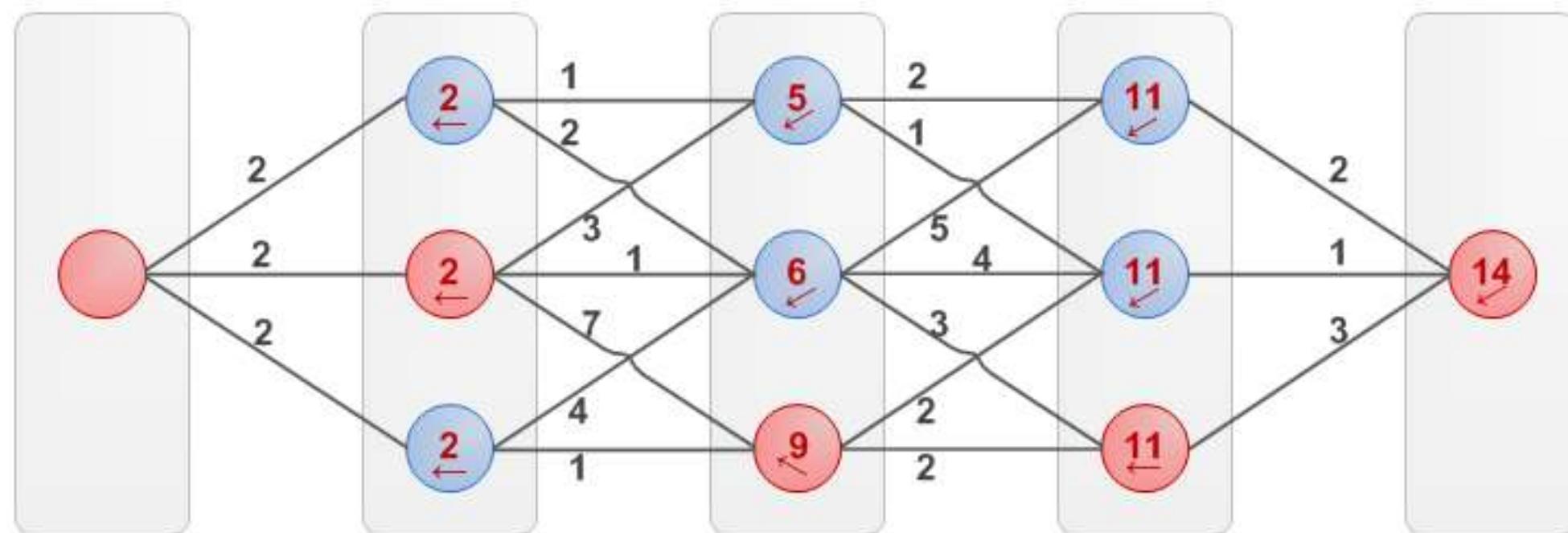
Labeling

Some applications: stereo vision



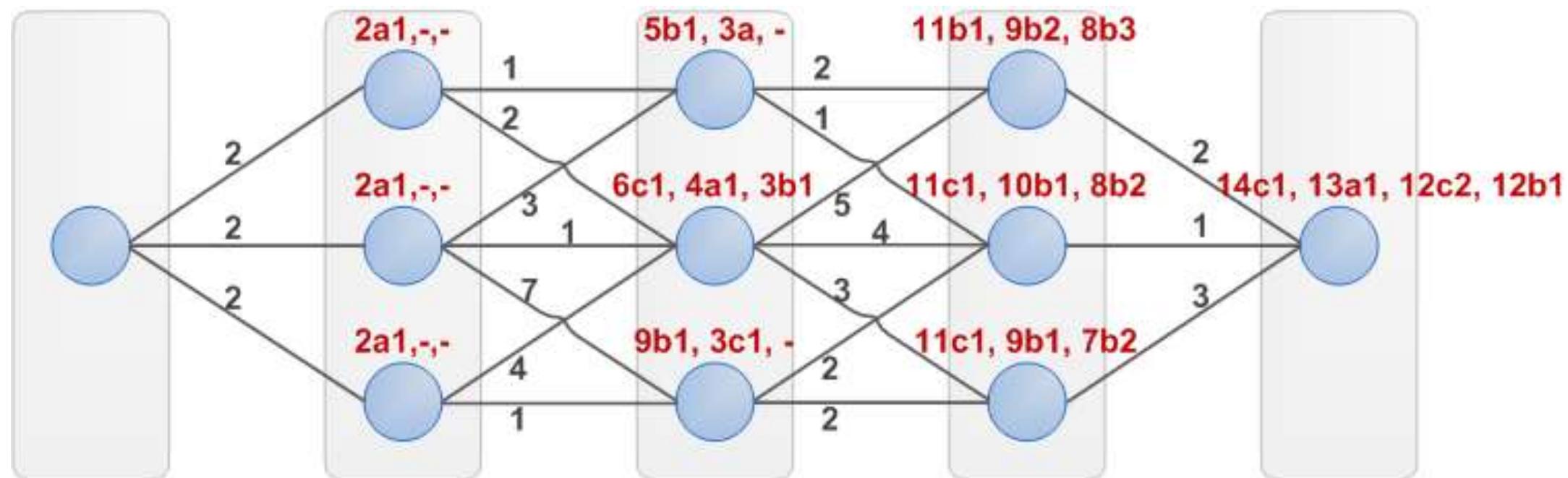
Labeling

(MAX, +) labeling problem on 1D chain structure



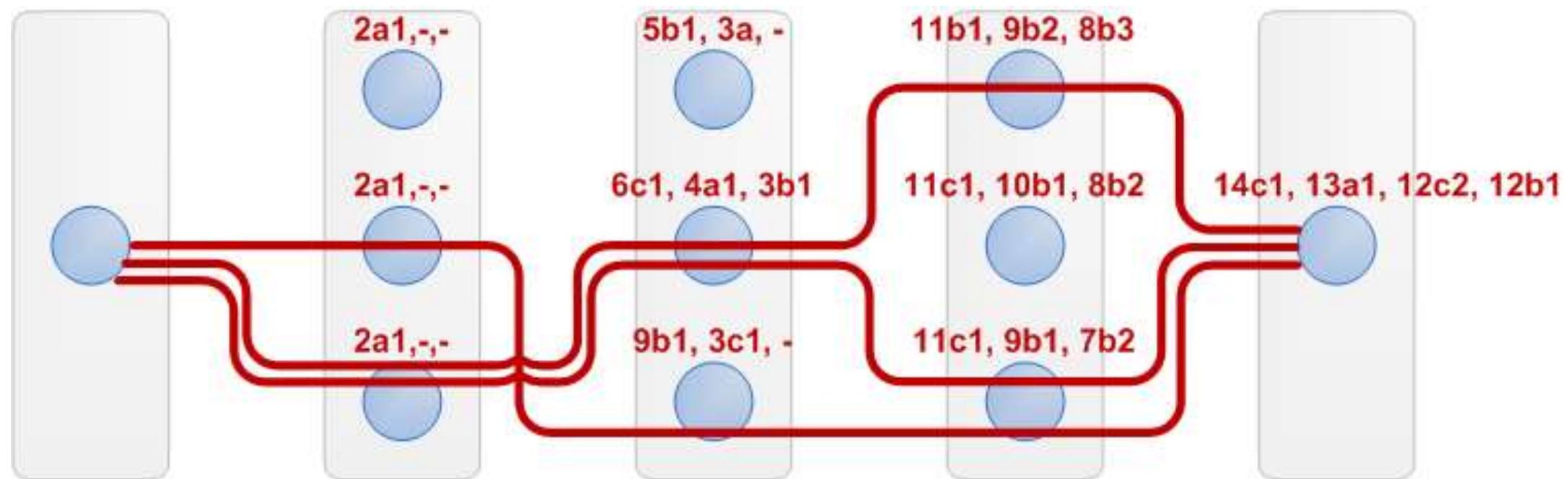
Labeling

(MAX, +) labeling problem on 1D chain structure



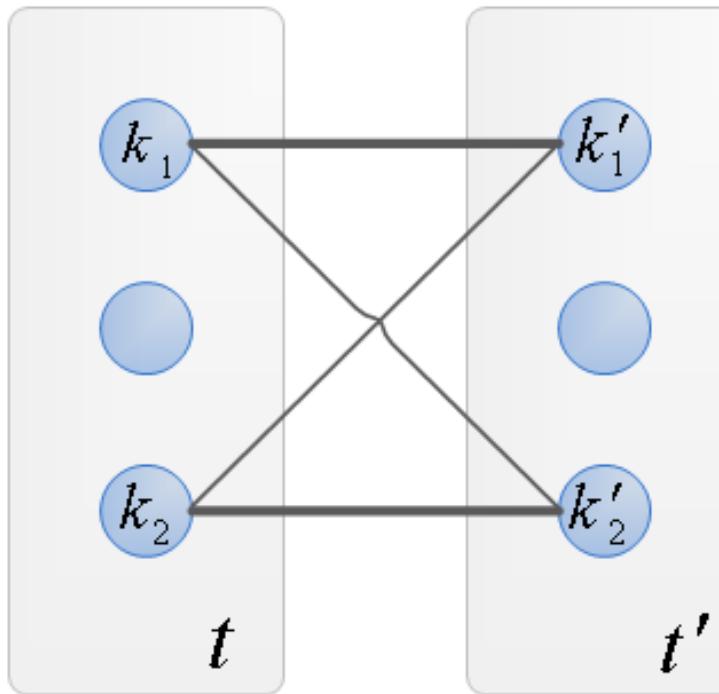
Labeling

(MAX, +) labeling problem on 1D chain structure: N best paths



Labeling

(OR, AND) labeling problem with supermodular weights

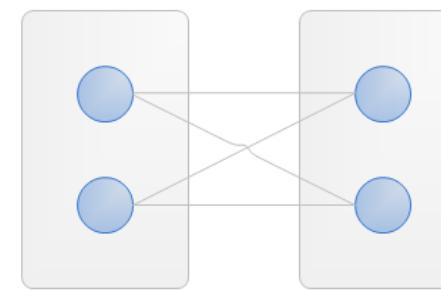
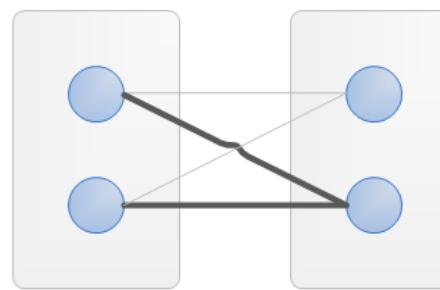
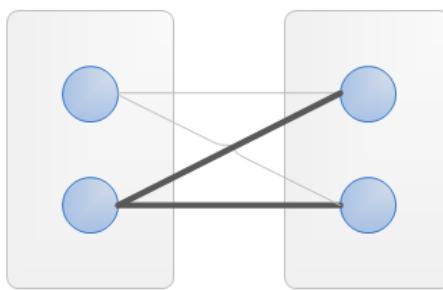
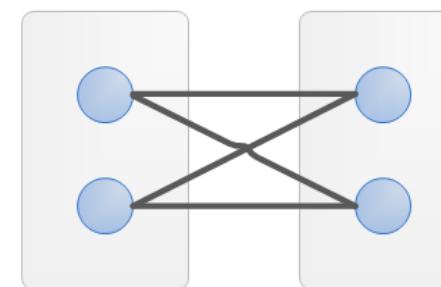
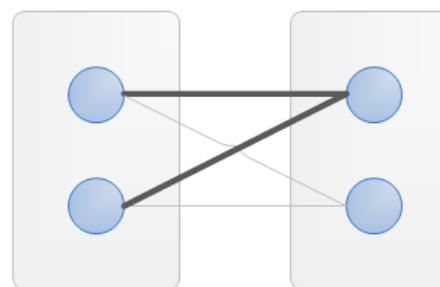
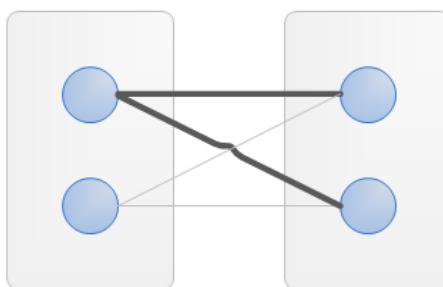
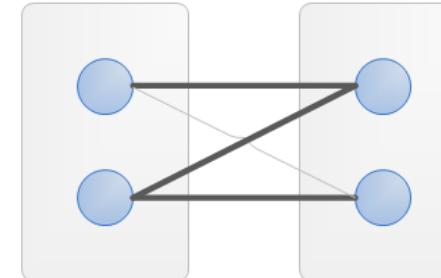
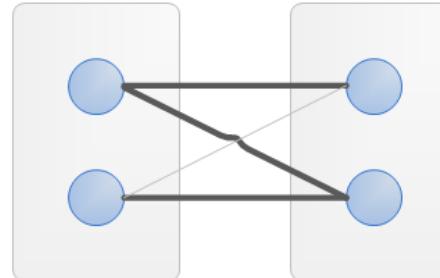
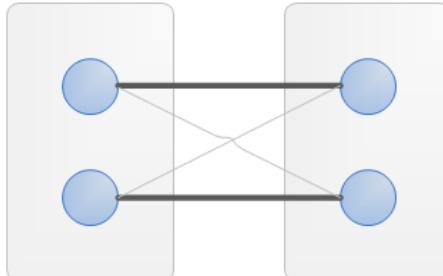


$$k_1 > k_2, \quad k'_1 > k'_2$$

$$g(k_1, k'_1) \& g(k_2, k'_2) \geq g(k_1, k'_2) \& g(k_2, k'_1)$$

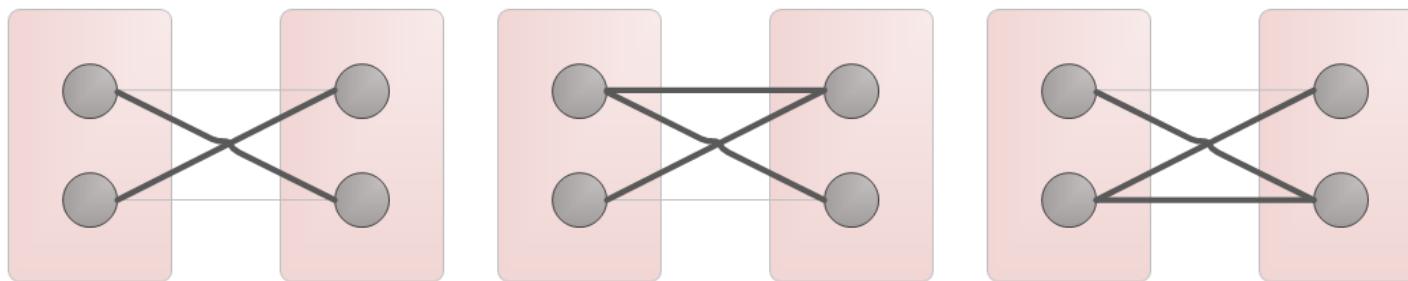
Labeling

Supermodular weights



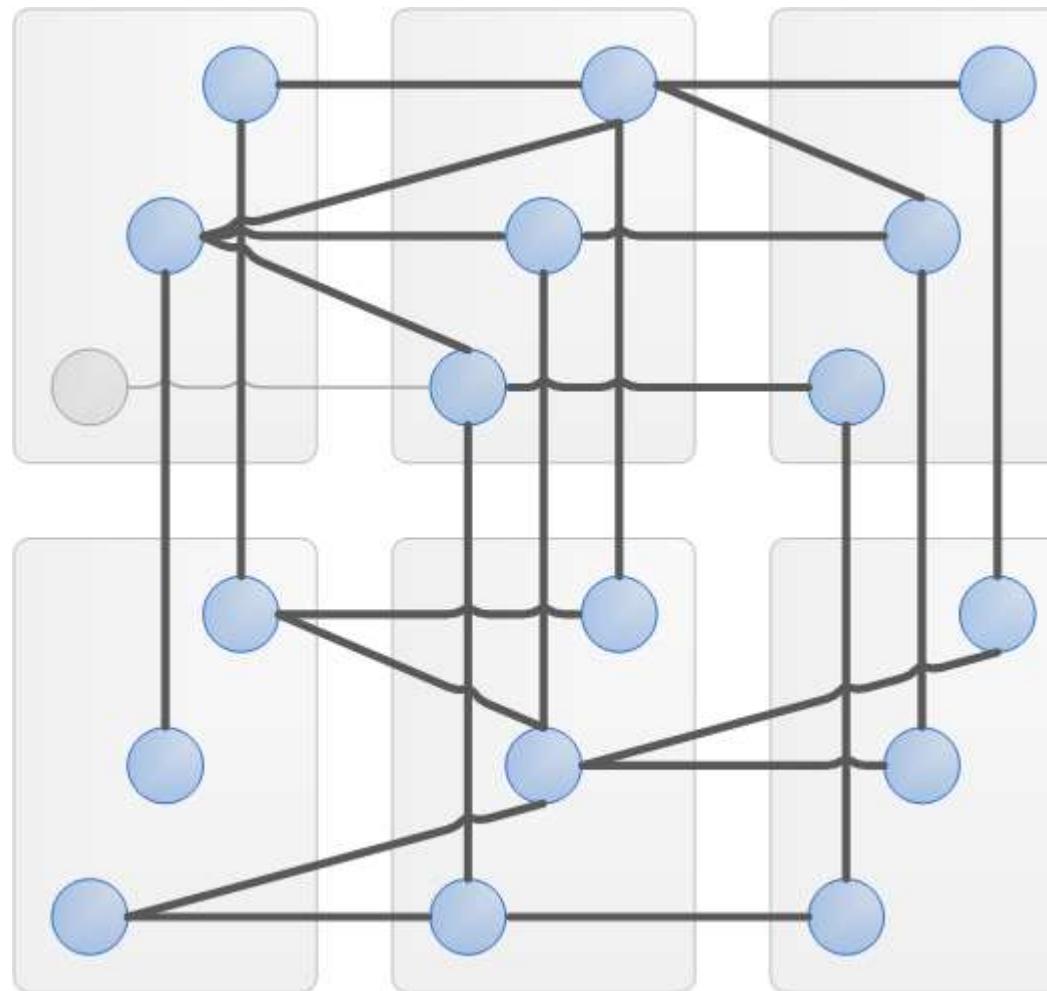
Labeling

NOT supermodular weights



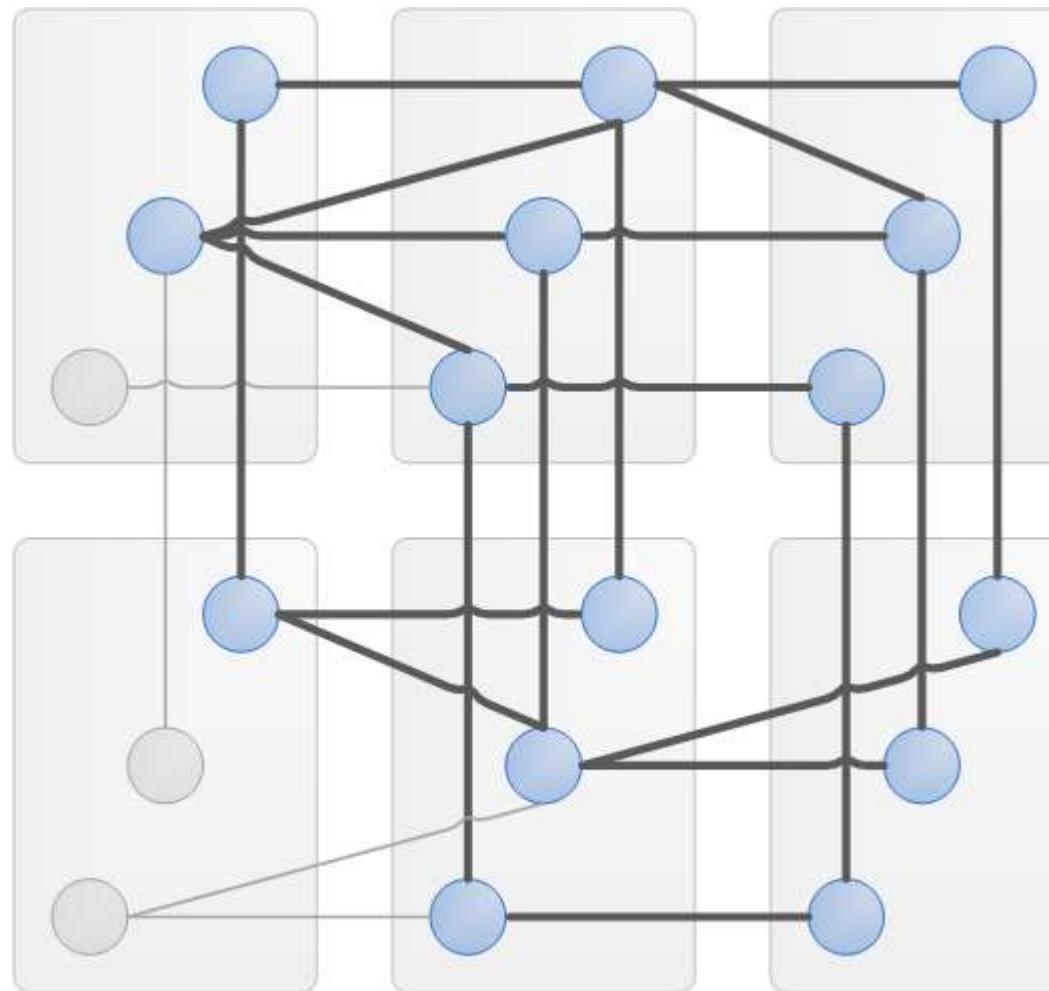
Labeling

(OR, AND) labeling problem with supermodular weights: example



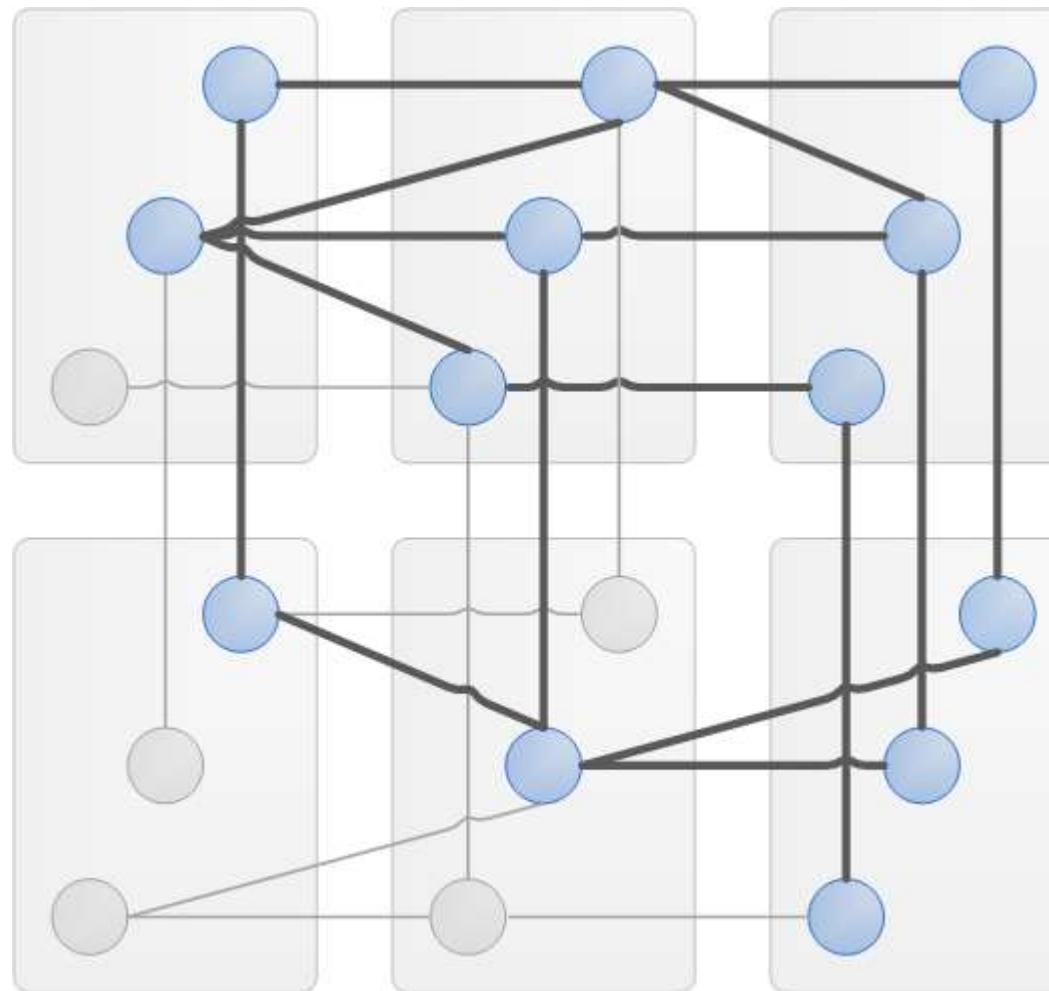
Labeling

(OR, AND) labeling problem with supermodular weights: example



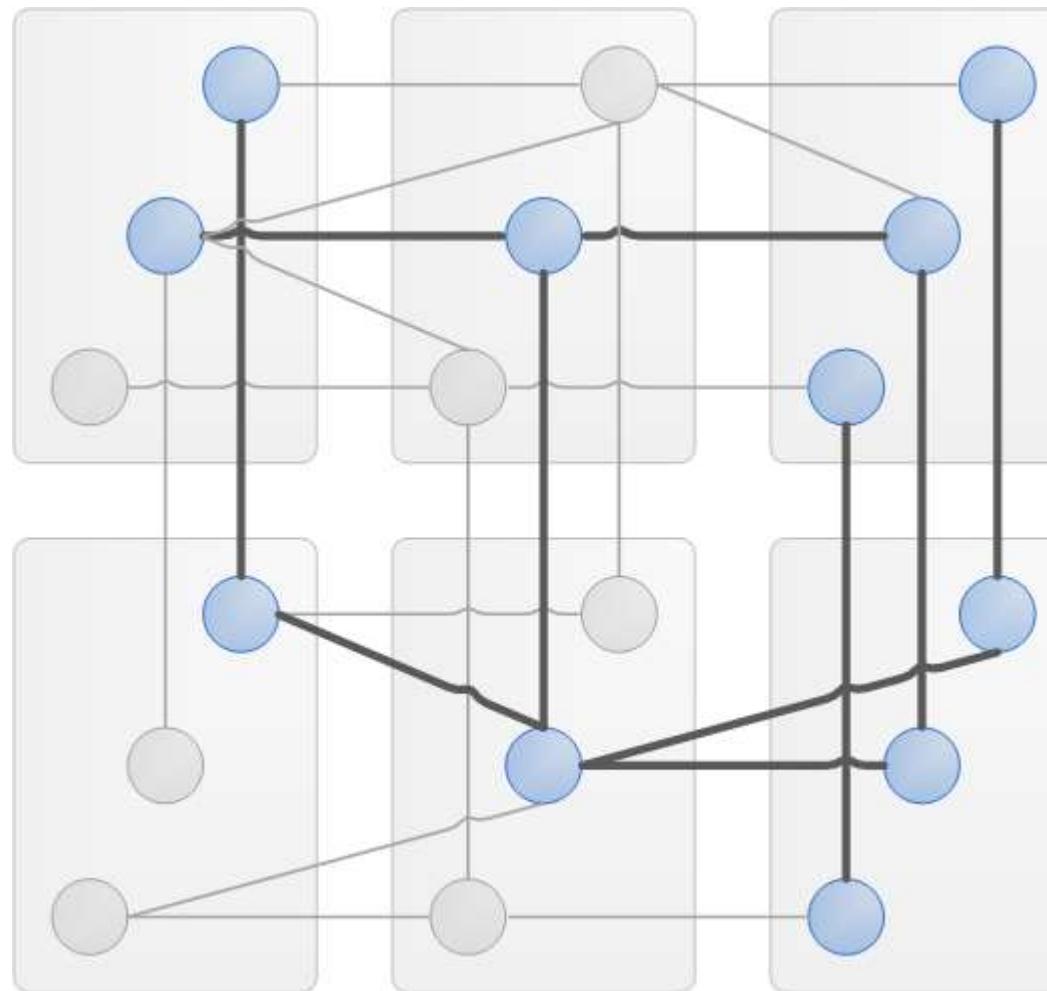
Labeling

(OR, AND) labeling problem with supermodular weights: example



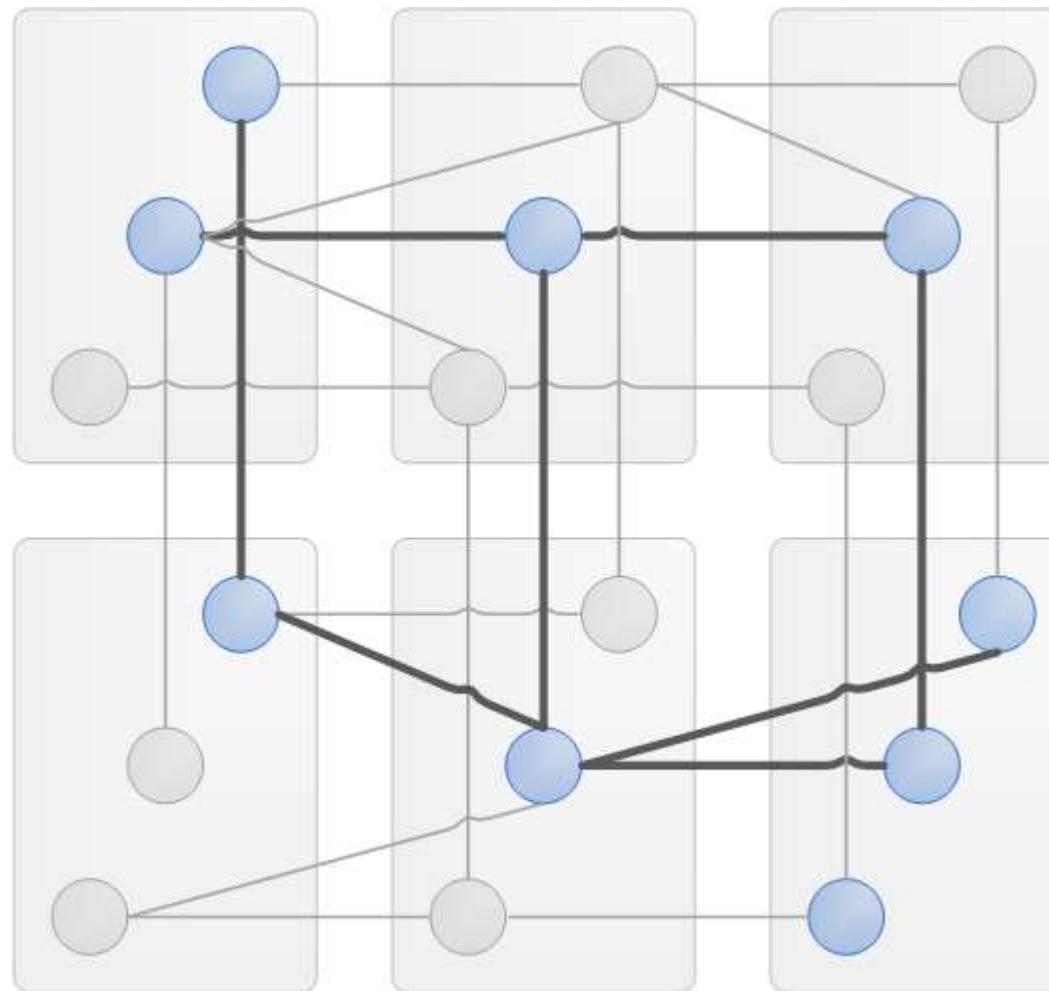
Labeling

(OR, AND) labeling problem with supermodular weights: example



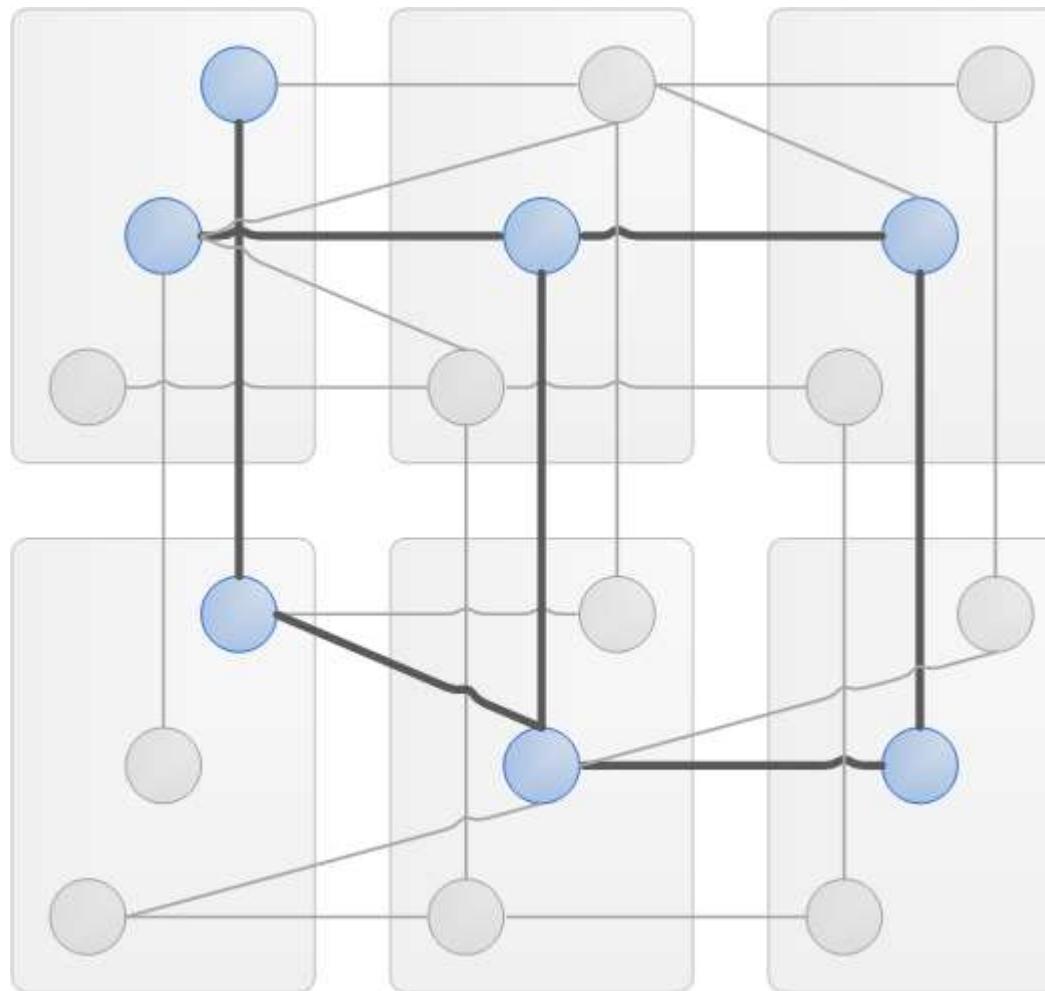
Labeling

(OR, AND) labeling problem with supermodular weights: example



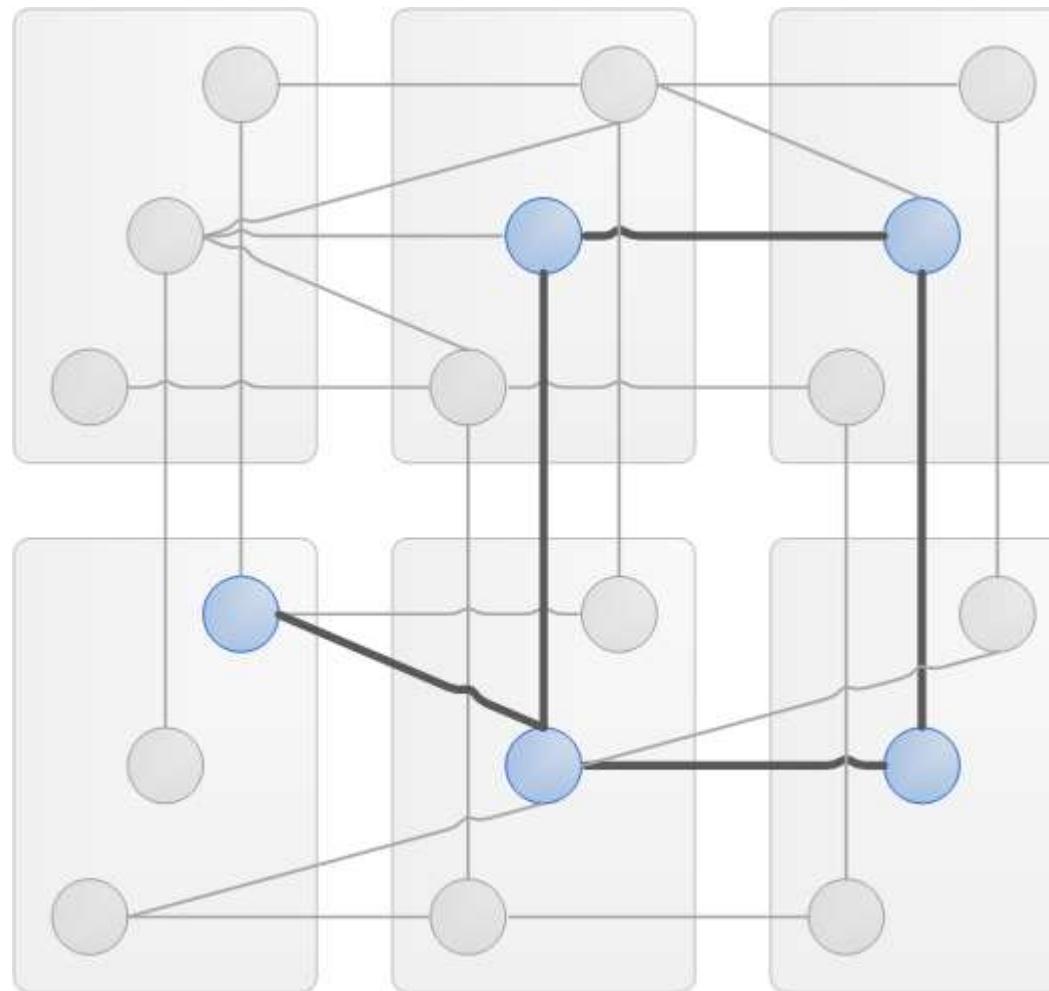
Labeling

(OR, AND) labeling problem with supermodular weights



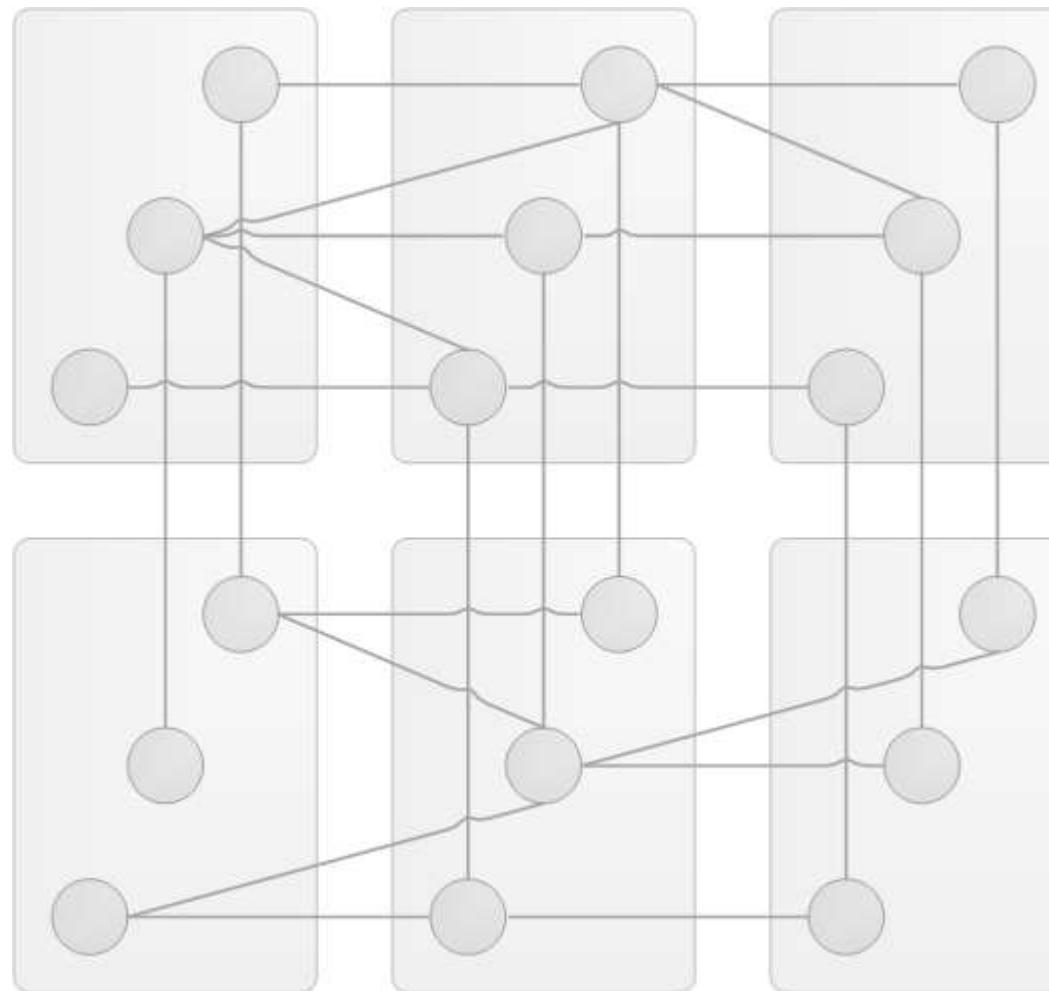
Labeling

(OR, AND) labeling problem with supermodular weights: example



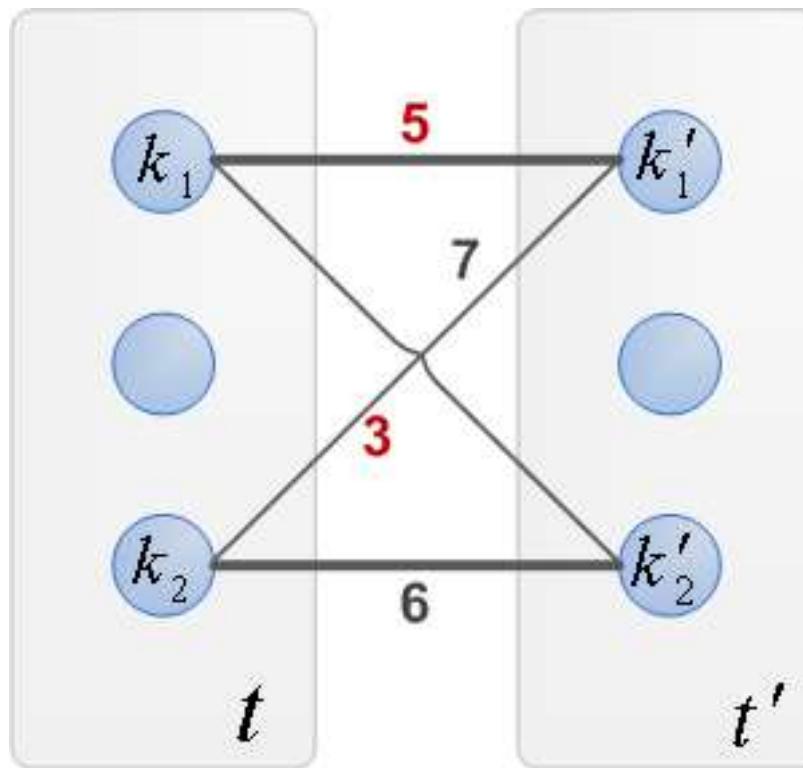
Labeling

(OR, AND) labeling problem with supermodular weights: example



Labeling

(MAX, MIN) labeling problem with supermodular weights

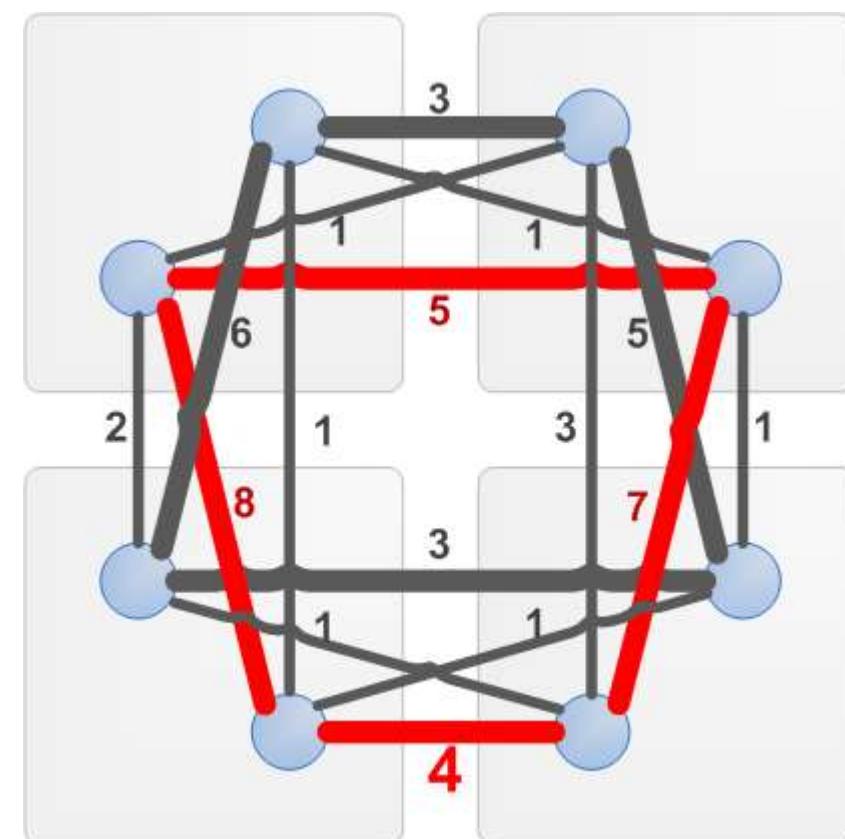
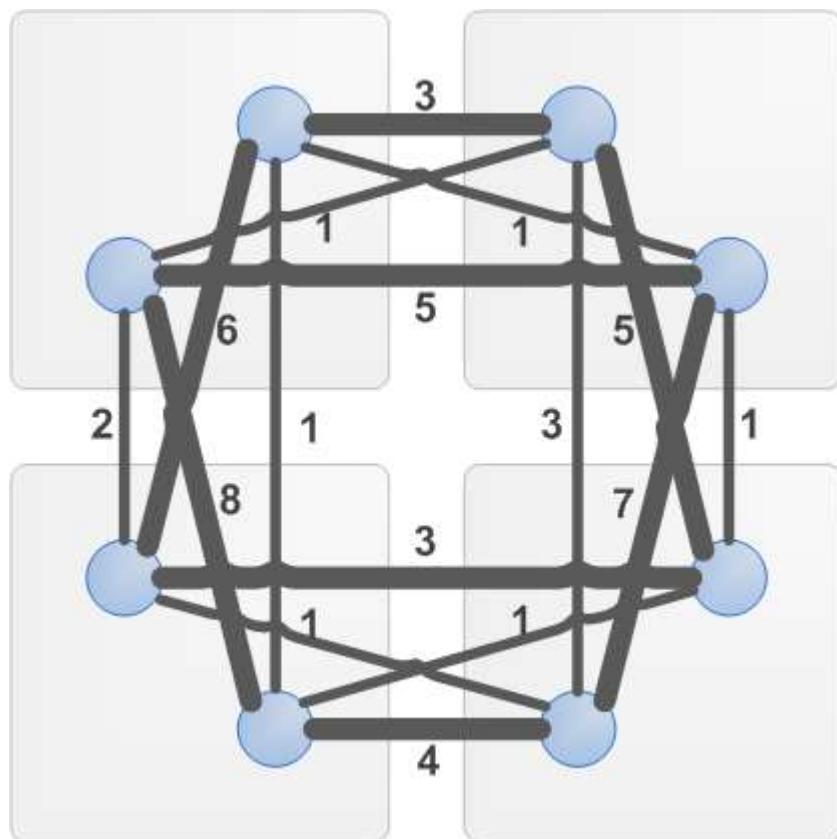


$$k_1 > k_2, k'_1 > k'_2$$

$$\min[g(k_1, k'_2), g(k_2, k'_1)] \leq \min[g(k_1, k'_1), g(k_2, k'_2)]$$

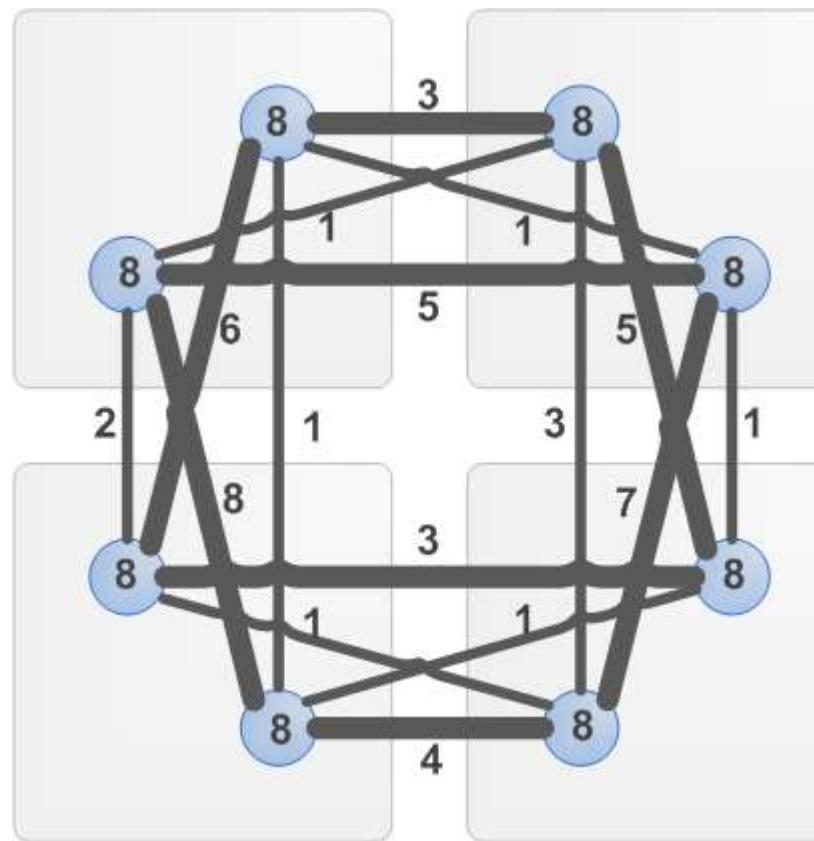
Labeling

(MAX, MIN) labeling problem with supermodular weights



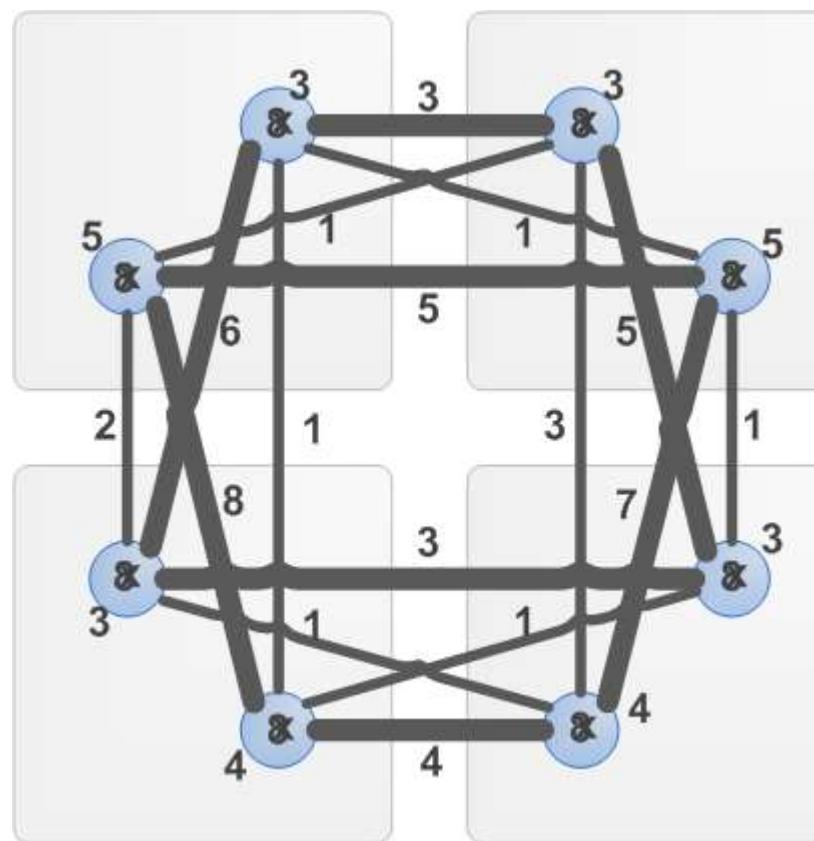
Labeling

(MAX, MIN) labeling problem with supermodular weights



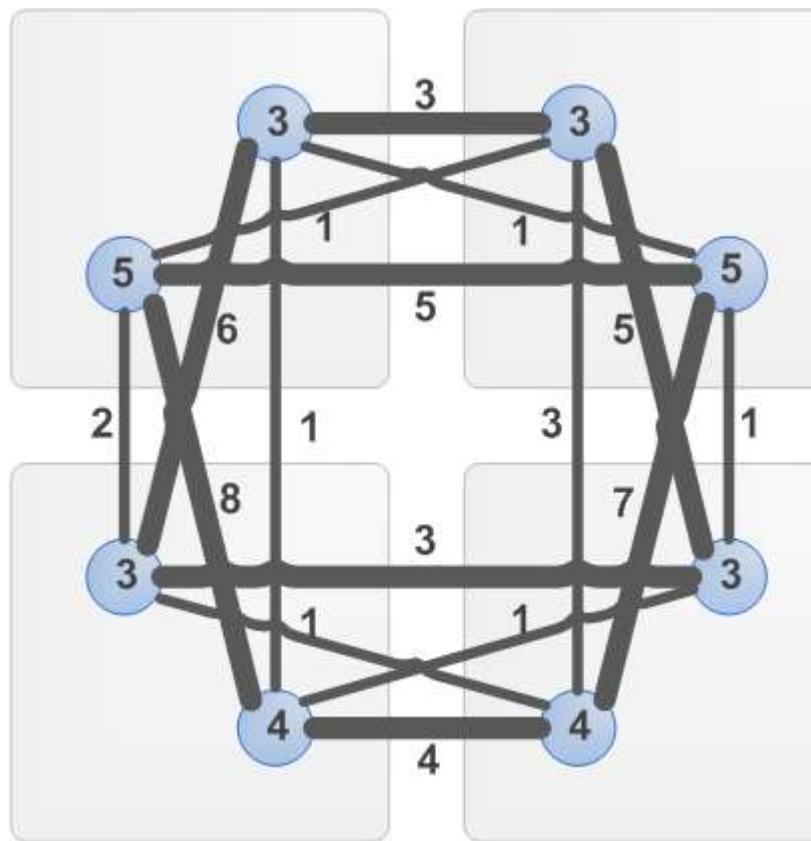
Labeling

(MAX, MIN) labeling problem with supermodular weights



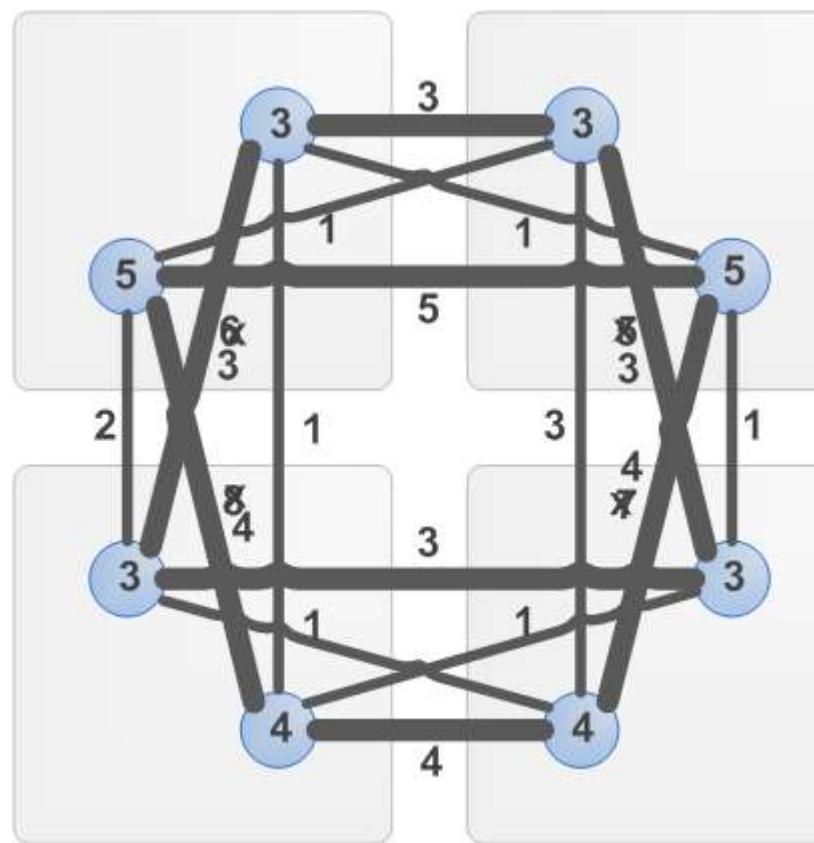
Labeling

(MAX, MIN) labeling problem with supermodular weights



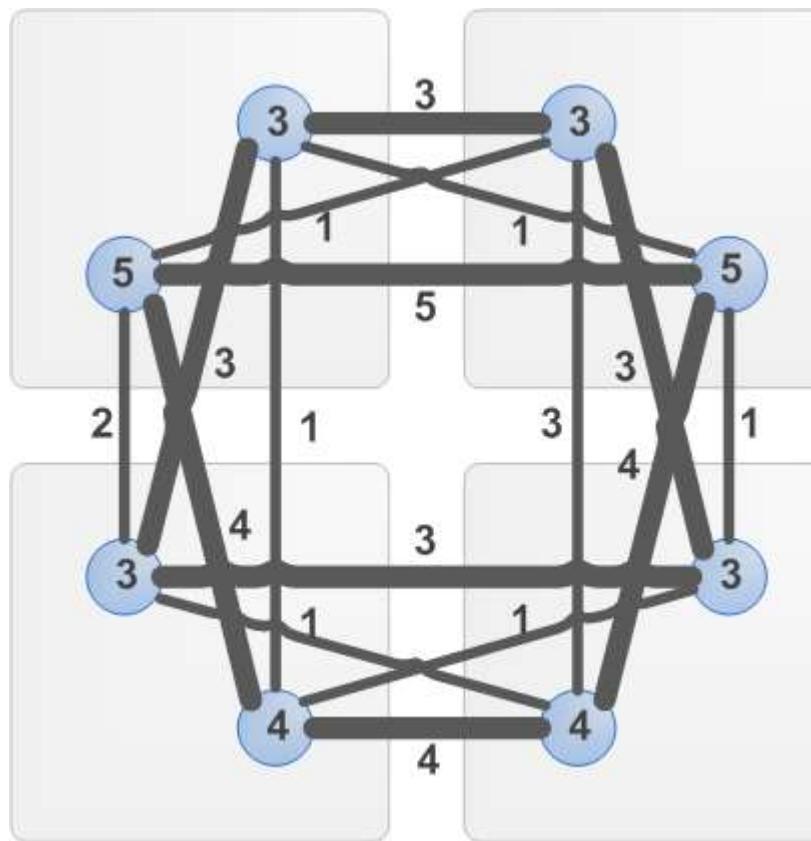
Labeling

(MAX, MIN) labeling problem with supermodular weights



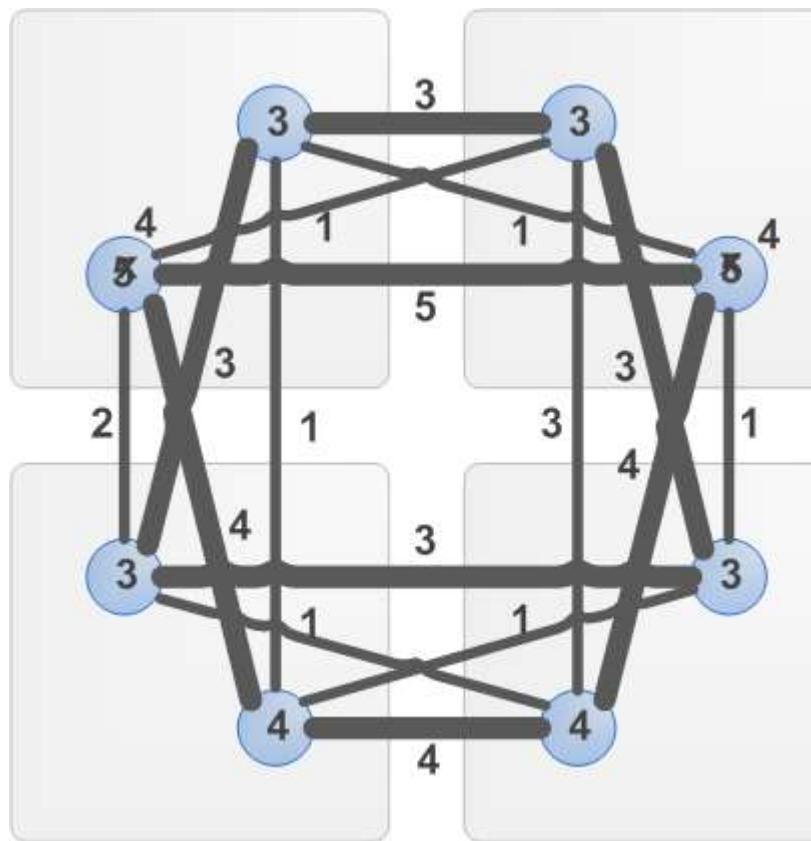
Labeling

(MAX, MIN) labeling problem with supermodular weights



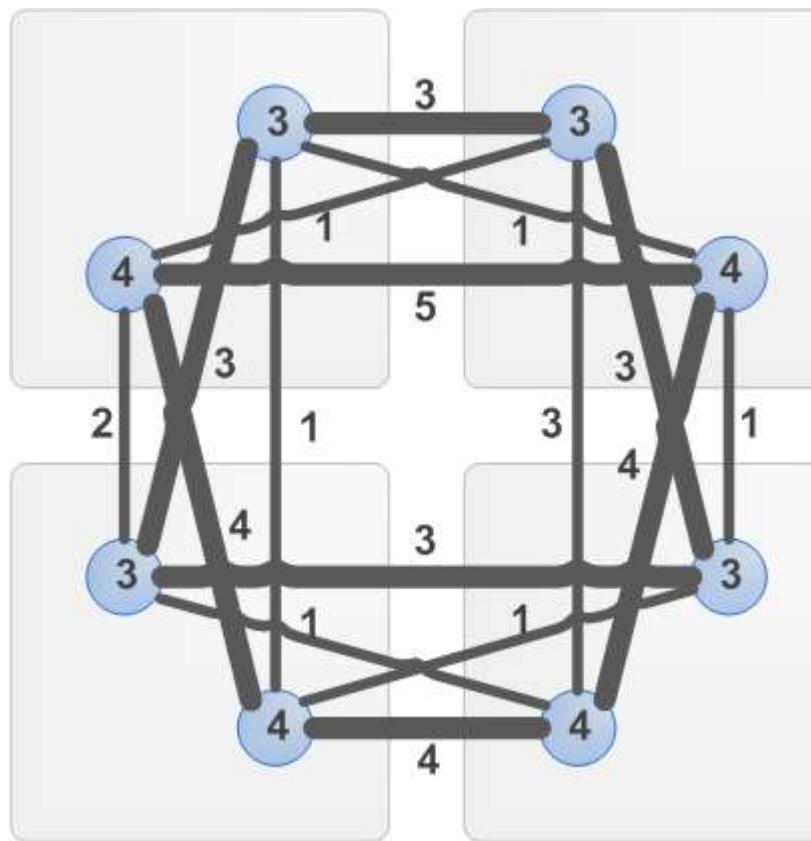
Labeling

(MAX, MIN) labeling problem with supermodular weights



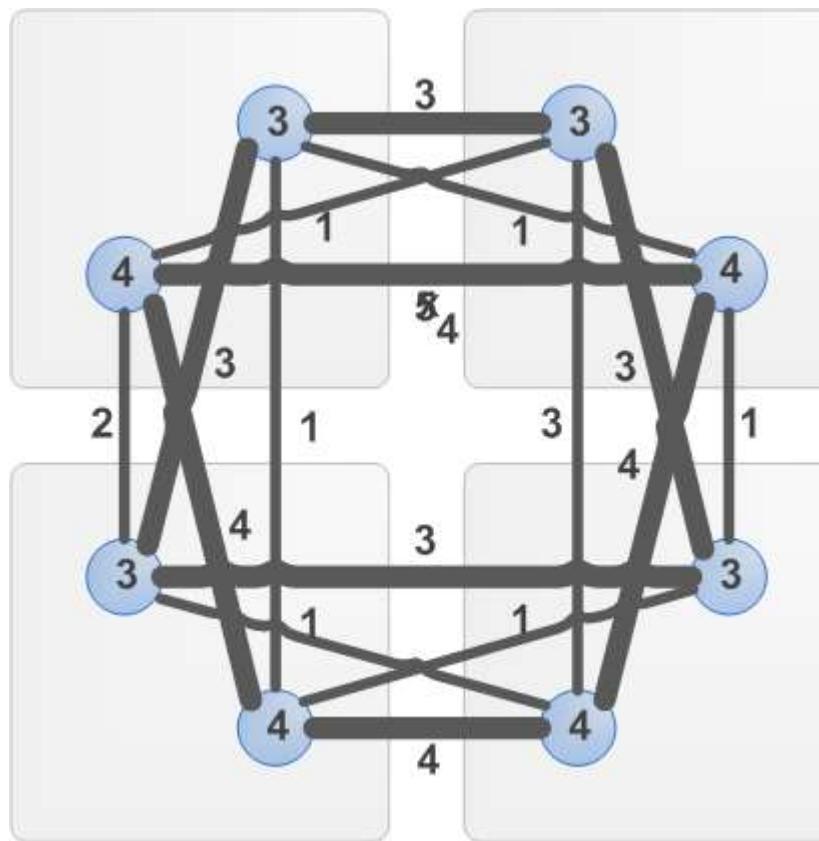
Labeling

(MAX, MIN) labeling problem with supermodular weights



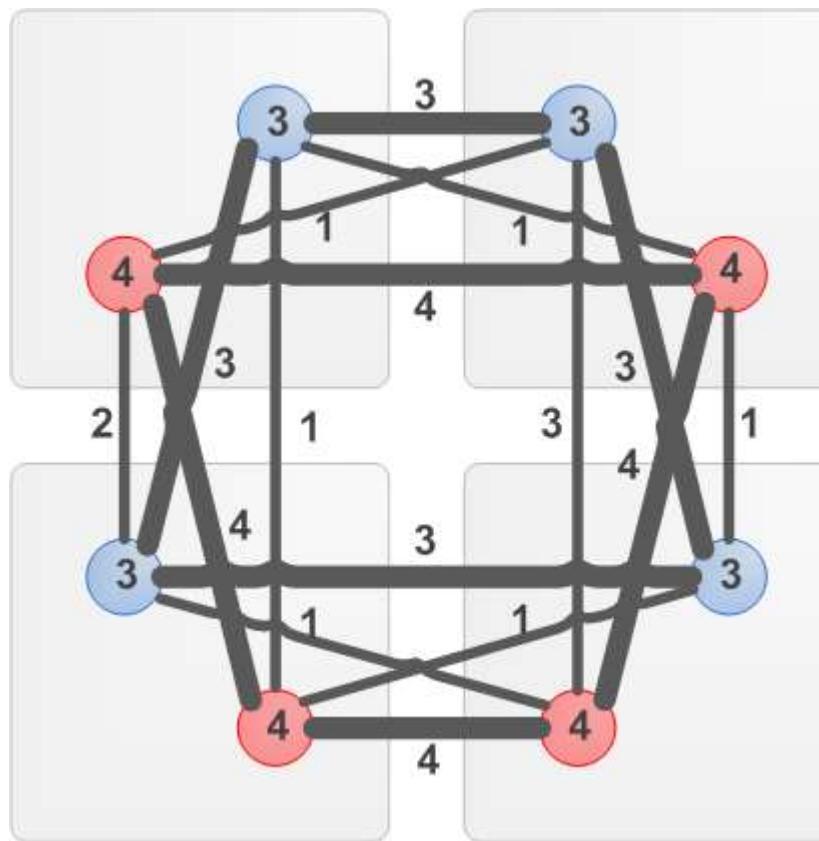
Labeling

(MAX, MIN) labeling problem with supermodular weights



Labeling

(MAX, MIN) labeling problem with supermodular weights



Labeling

(MIN, MAX) labeling problem: clustering

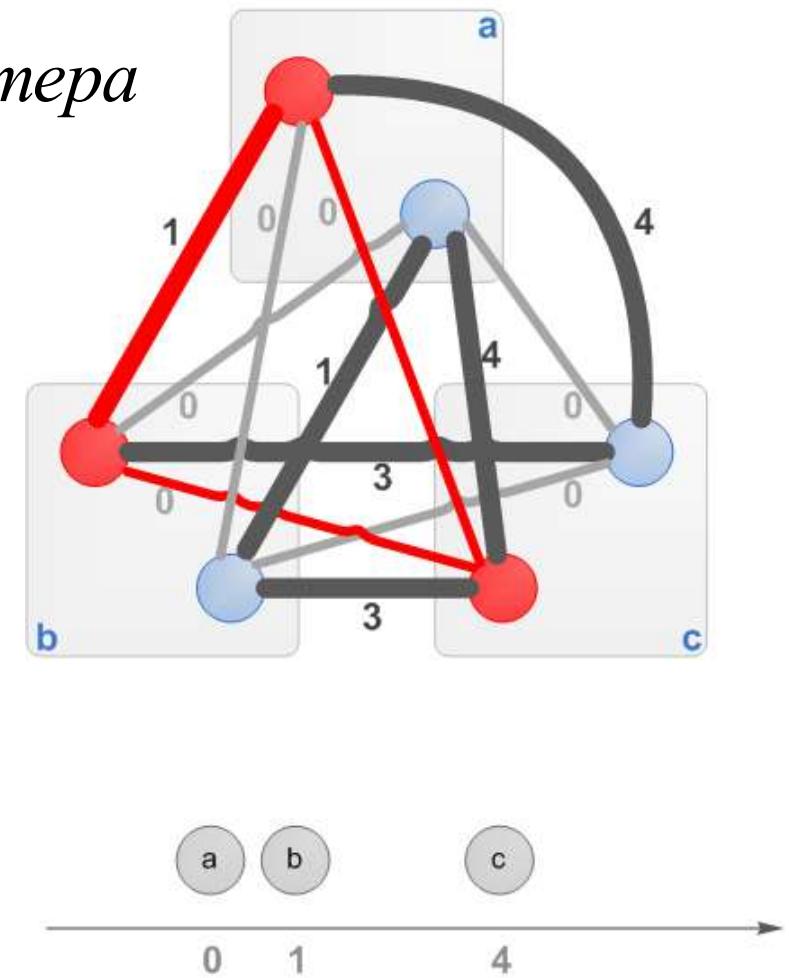
$g_{tt'}(k_t, k_{t'})$ = отличие между объектами одного кластера

T = объекты

K = кластеры

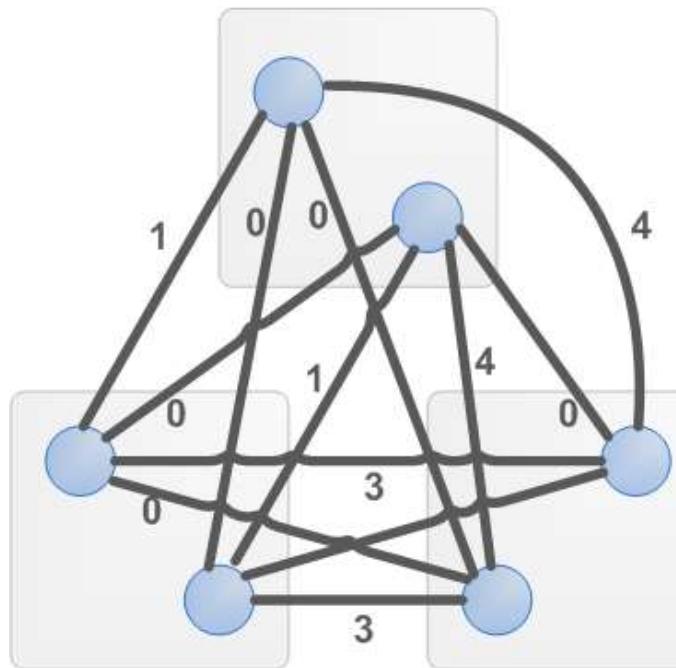
The distance between most different objects of the same cluster should be minimal

$$k^* = \min_{\bar{k} \in K^T} \max_{\substack{t \in T \\ t' \in \tau(t)}} g_{tt'}(k_t, k_{t'})$$



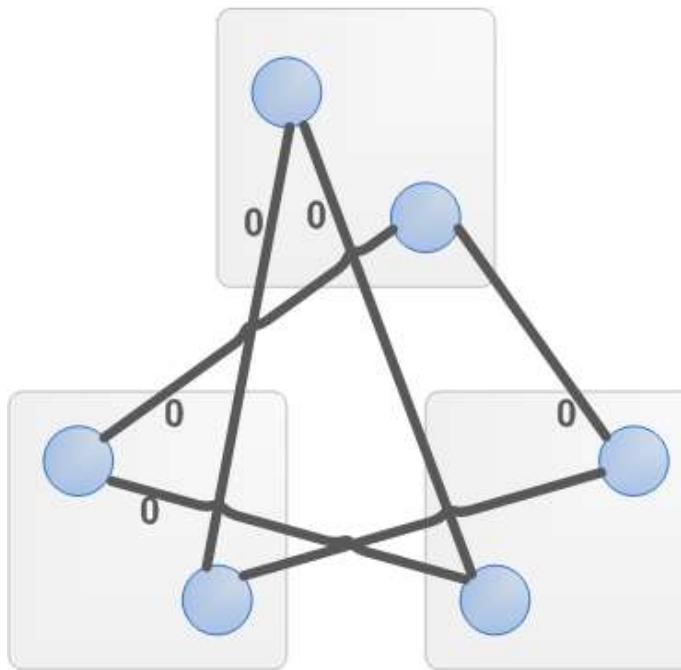
Labeling

(MIN, MAX) labeling problem: clustering



Labeling

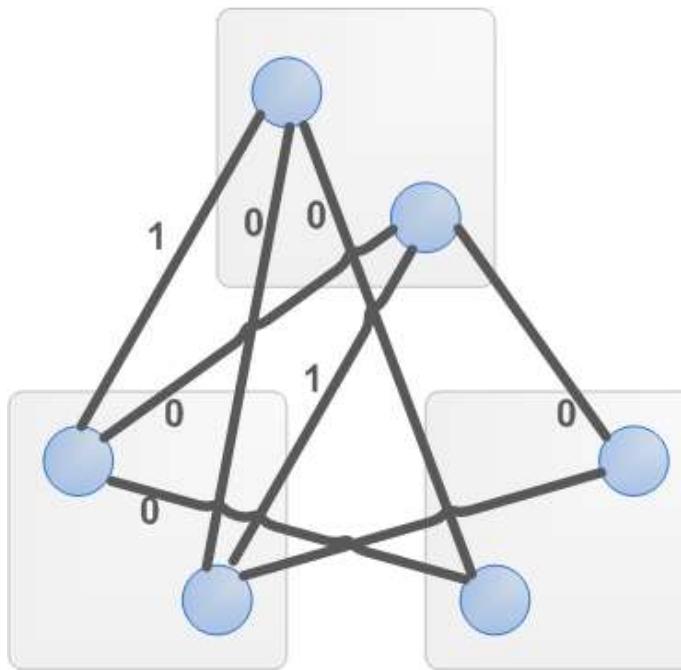
(MIN, MAX) labeling problem: clustering



Remove all edges having weight > 0
Solve (OR,AND) problem

Labeling

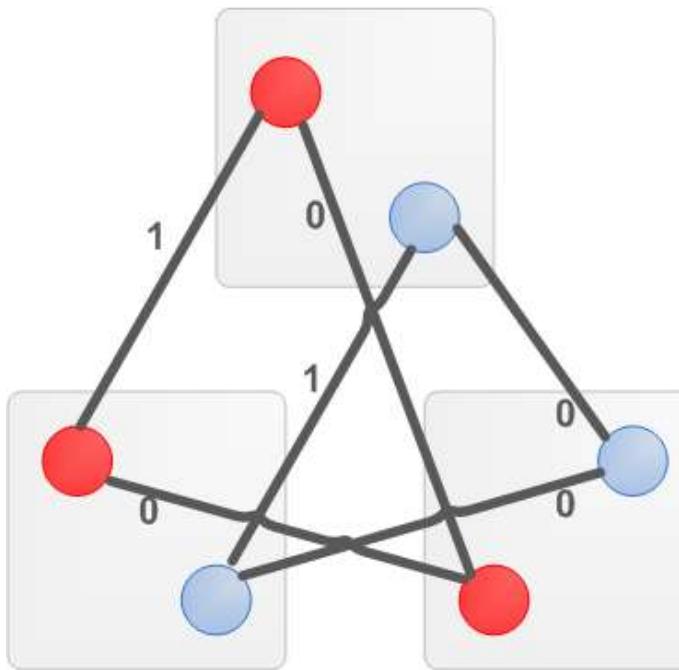
(MIN, MAX) labeling problem: clustering



Remove all edges having weight ≥ 1
Solve (OR,AND) problem

Labeling

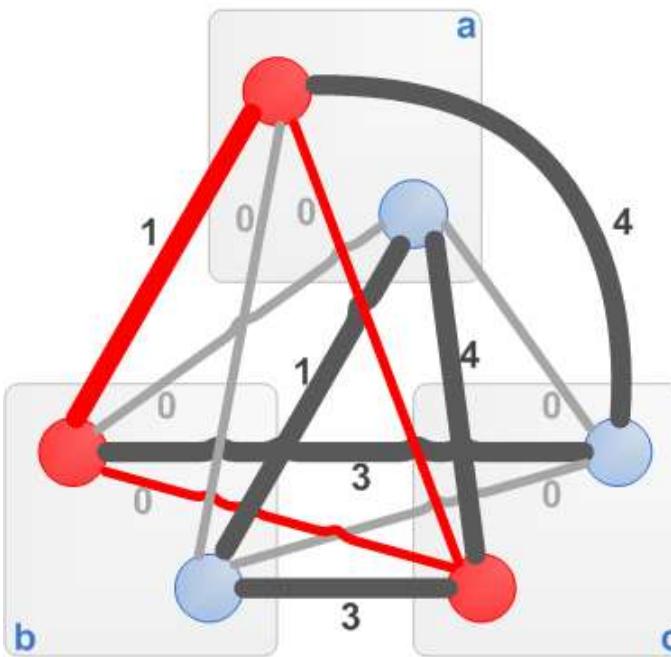
(MIN, MAX) labeling problem: clustering



Remove all edges having weight ≥ 1
Solve (OR,AND) problem

Labeling

(MIN, MAX) labeling problem: clustering



distance between most different objects of the same cluster = 1



1.13. Object detection



Object detection

```
def demo_cascade():

    if USE_CAMERA:
        cap = cv2.VideoCapture(0)
    else:
        cap = []
        frame = cv2.imread(filename_in)

    while (True):
        if USE_CAMERA:
            ret, frame = cap.read()

        gray_rgb = tools_image.desaturate(frame)
        faces = faceCascade.detectMultiScale(frame, scaleFactor=1.1,minNeighbors=5,minSize=(30, 30))

        for (x, y, w, h) in faces:
            cv2.rectangle(gray_rgb, (x, y), (x + w, y + h), (0, 255, 0), 2)

        cv2.imshow('frame', gray_rgb)
        key = cv2.waitKey(1)
        if key & 0xFF == 27:
            break

        if (key & 0xFF == 13) or (key & 0xFF == 32):
            cv2.imwrite(filename_out, gray_rgb)

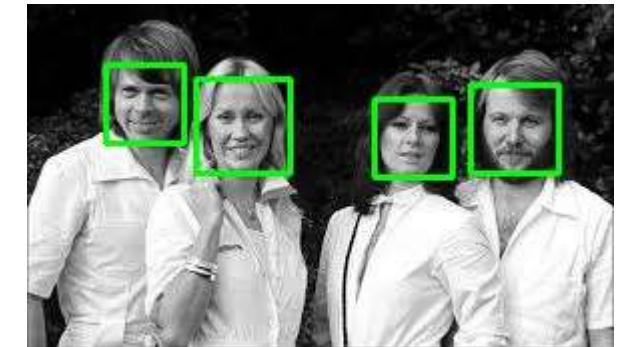
    if USE_CAMERA:
        cap.release()

    cv2.destroyAllWindows()
    if not USE_CAMERA:
        cv2.imwrite(filename_out, gray_rgb)

return
```

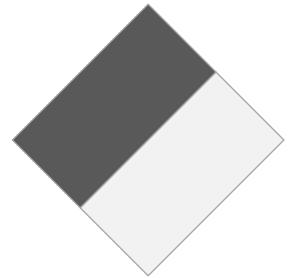
Object detection

haarcascade_eye.xml
haarcascade_eye_tree_eyeglasses.xml
haarcascade_frontalcatface.xml
haarcascade_frontalcatface_extended.xml
haarcascade_frontalface_alt.xml
haarcascade_frontalface_alt2.xml
haarcascade_frontalface_alt_tree.xml
haarcascade_frontalface_default.xml
haarcascade_fullbody.xml
haarcascade_lefteye_2splits.xml
haarcascade_licence_plate_rus_16stages.xml
haarcascade_lowerbody.xml
haarcascade_profileface.xml
haarcascade_righteye_2splits.xml
haarcascade_russian_plate_number.xml
haarcascade_smile.xml
haarcascade_upperbody.xml



Object detection: Haar cascades

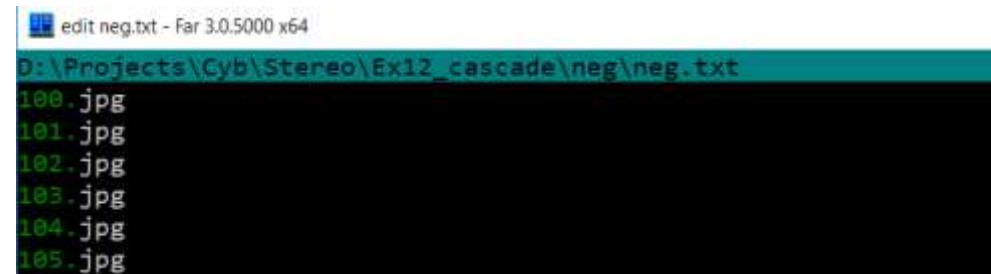
1. Create samples
2. Verify samples
3. Train the cascade



Object detection: create samples from single positive

```
opencv_createsamples -vec pos.vec  
                      -img pos_single\object.jpg  
                      -bg neg\neg.txt  
                      -num 100  
                      -bgcolor 0  
                      -bgthresh 0  
                      -w 28 -h 28  
                      -maxxangle 1.1  
                      -maxyangle 1.1  
                      -maxzangle 0.5  
                      -maxidev 40
```

result vec file
input positive sample file
list of negative samples



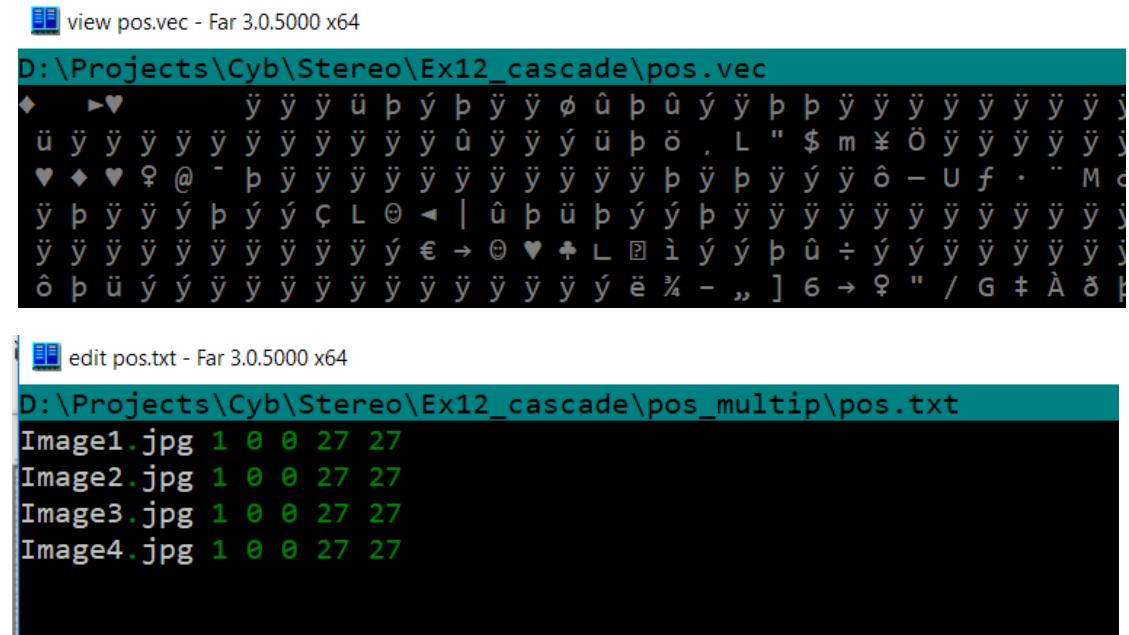
A screenshot of a text editor window titled "edit neg.txt - Far 3.0.5000 x64". The window shows a list of file names in green text, each ending in ".jpg". The files are numbered from 100 to 105. The path "D:\Projects\Cyb\Stereo\Ex12_cascade\neg\neg.txt" is visible at the top of the editor.

```
D:\Projects\Cyb\Stereo\Ex12_cascade\neg\neg.txt  
100.jpg  
101.jpg  
102.jpg  
103.jpg  
104.jpg  
105.jpg
```

Object detection: create samples from many positive

```
opencv_createsamples -vec pos.vec  
                      -info pos_multip\pos.txt  
                      -num 4  
                      -w 28 -h 28
```

result file
list of pos samples



The screenshot shows two terminal windows side-by-side.

The left terminal window, titled "view pos.vec - Far 3.0.5000 x64", displays the command:

```
D:\Projects\Cyb\Stereo\Ex12_cascade\pos.vec
```

The right terminal window, titled "edit pos.txt - Far 3.0.5000 x64", displays the command:

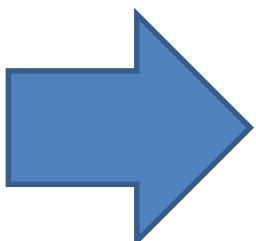
```
D:\Projects\Cyb\Stereo\Ex12_cascade\pos_multip\pos.txt
```

Both windows show the following output:

```
Image1.jpg 1 0 0 27 27  
Image2.jpg 1 0 0 27 27  
Image3.jpg 1 0 0 27 27  
Image4.jpg 1 0 0 27 27
```

Object detection: verify vec file

```
opencv_createsamples -vec pos.vec  
-show
```



Object detection: training

```
opencv_traincascade -data output_folder
                    -vec ../positive/positives.vex
                    -bg ../negative/bg_train.txt
                    -numPos 800
                    -numNeg 400
                    -numStages 10
                    -w 20
                    -h 20
```

```
C:\Users\dryabokon\source\Digits\ML\data\x12_single\output_
<?xml version="1.0"?>
<opencv_storage>
<cascade>
  <stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>40</height>
  <width>40</width>
  <stageParams>
    <boostType>GAB</boostType>
    <minHitRate>9.9500000476837158e-01</minHitRate>
    <maxFalseAlarm>5.000000000000000e-01</maxFalseAlarm>
    <weightTrimRate>9.499999999999996e-01</weightTrimRate>
    <maxDepth>1</maxDepth>
    <maxWeakCount>100</maxWeakCount>
  </stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount>
    <featSize>1</featSize>
    <mode>BASIC</mode>
  </featureParams>
```

*existing folder
made by opencv_createsamples
list of negative samples*

```
edit_bg_train.txt - Fe 3.0.5000 x64  
D:\Projects\Cyb\Stereo\ExIII_cascade\negative\bg_train.txt  
../negative/bg/_02.jpg  
../negative/bg/_22.jpg  
../negative/bg/_32.jpg  
../negative/bg/_712.jpg  
../negative/bg/_156.jpg  
../negative/bg/_416.jpg  
../negative/bg/_275.jpg  
../negative/bg/_311.jpg  
../negative/bg/_152.jpg  
../negative/bg/_451.jpg  
../negative/bg/_696.jpg  
../negative/bg/_136.jpg  
../negative/bg/_286.jpg  
../negative/bg/_116.jpg  
../negative/bg/_398.jpg  
../negative/bg/_787.jpg
```

Train the Cascades

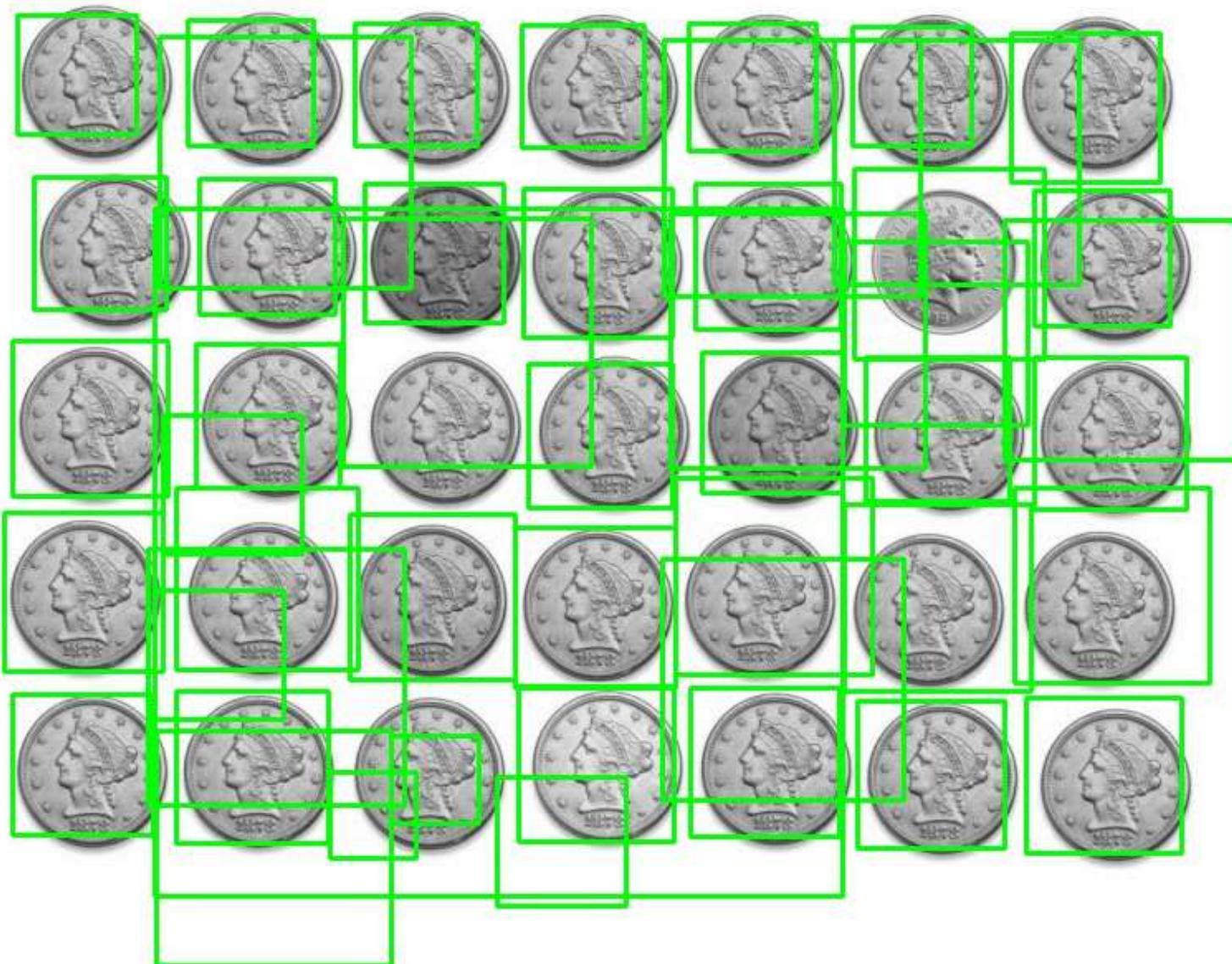
```
import cv2
import os
import tools_image
# -----
path = 'data/ex12_single/'
filename_in = path + 'pos_single/object_to_detect.jpg'
filename_out = 'data/output/detect_result.jpg'
object_detector= cv2.CascadeClassifier(path + 'output_single/cascade.xml')

# -----
def train_cascade():
    os.chdir(path)
    os.system('1-create_vec_from_single.bat')#os.system('2-verify_vec_single.bat')
    os.system('3-train_from_single.bat')
    return
# -----


if __name__ == '__main__':
    #train_cascade()

    image = tools_image.desaturate(cv2.imread(filename_in))
    objects, rejectLevels, levelWeights = \
        object_detector.detectMultiScale3(image, scaleFactor=1.05, minSize=(20, 20), outputRejectLevels=True)
    for (x, y, w, h) in objects:cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.imwrite(filename_out,image)
```

Object detection: example of training



1.14. Detection of patterns and anomalies



Detection of patterns and anomalies

Input data:

- Still image of multiple similar objects sampled across the grid-like structure.

Output:

- Pattern – an image that best fits (represents) the sampled instances of the similar objects
- Markup (labeling) of an image to indicate the positions of the patterns and outliers as well as their confidence level

Limitations

- Samples on the input image have limited pose variance so their view angle is similar (e.g. front view only, no side or back view)

Detection of patterns and anomalies

Synthetic image: a template is sampled thru the grid



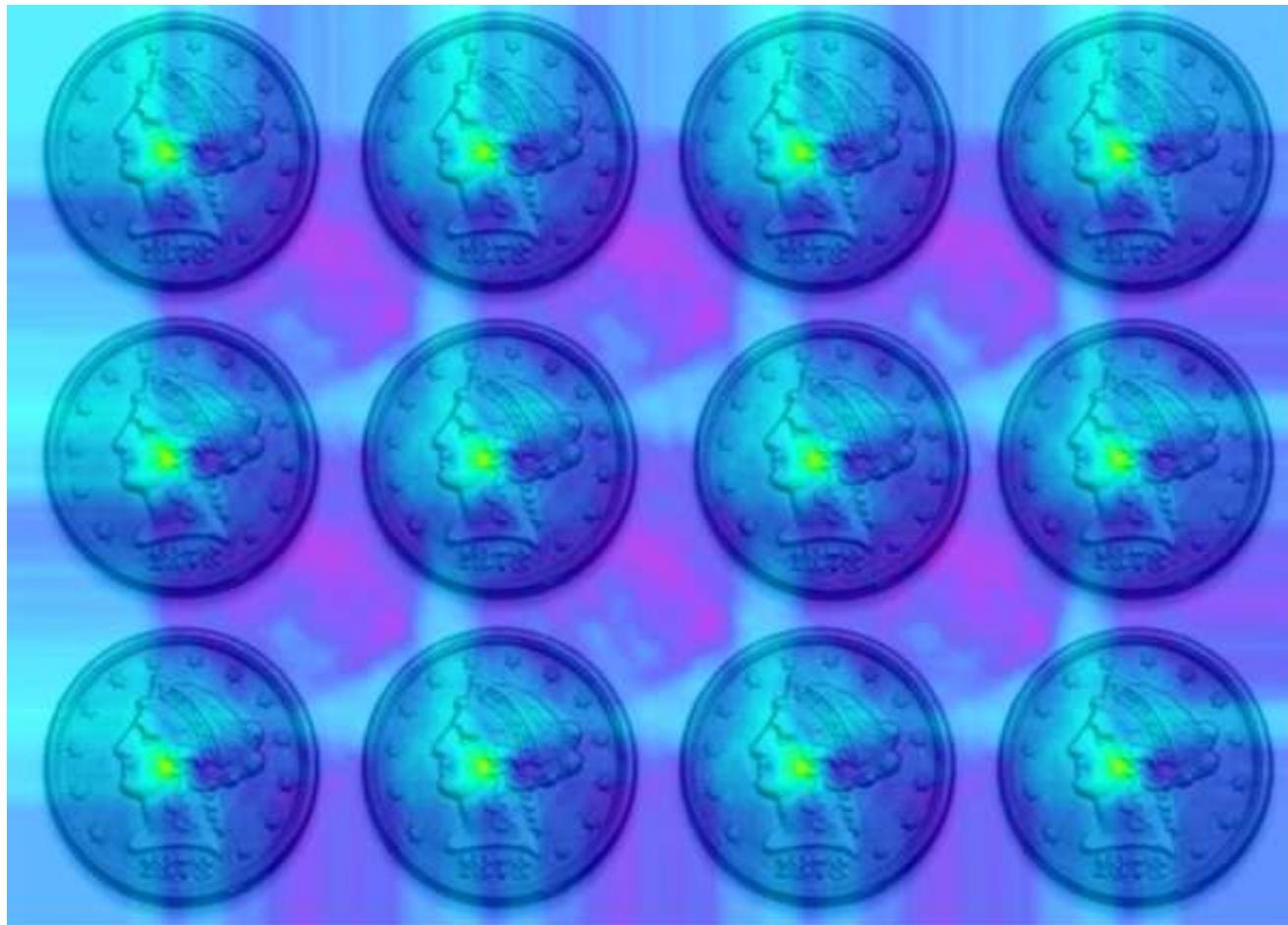
Detection of patterns and anomalies

Input image: added some variance : Euclidian translation + gamma correction



Detection of patterns and anomalies

Result heatmap: confidence level to match the pattern



pattern
(auto detected)

Detection of patterns and anomalies

Result mark-up: positions of the patterns



Detection of patterns and anomalies

Real image



Detection of patterns and anomalies

Real image



Detection of patterns and anomalies

Real image



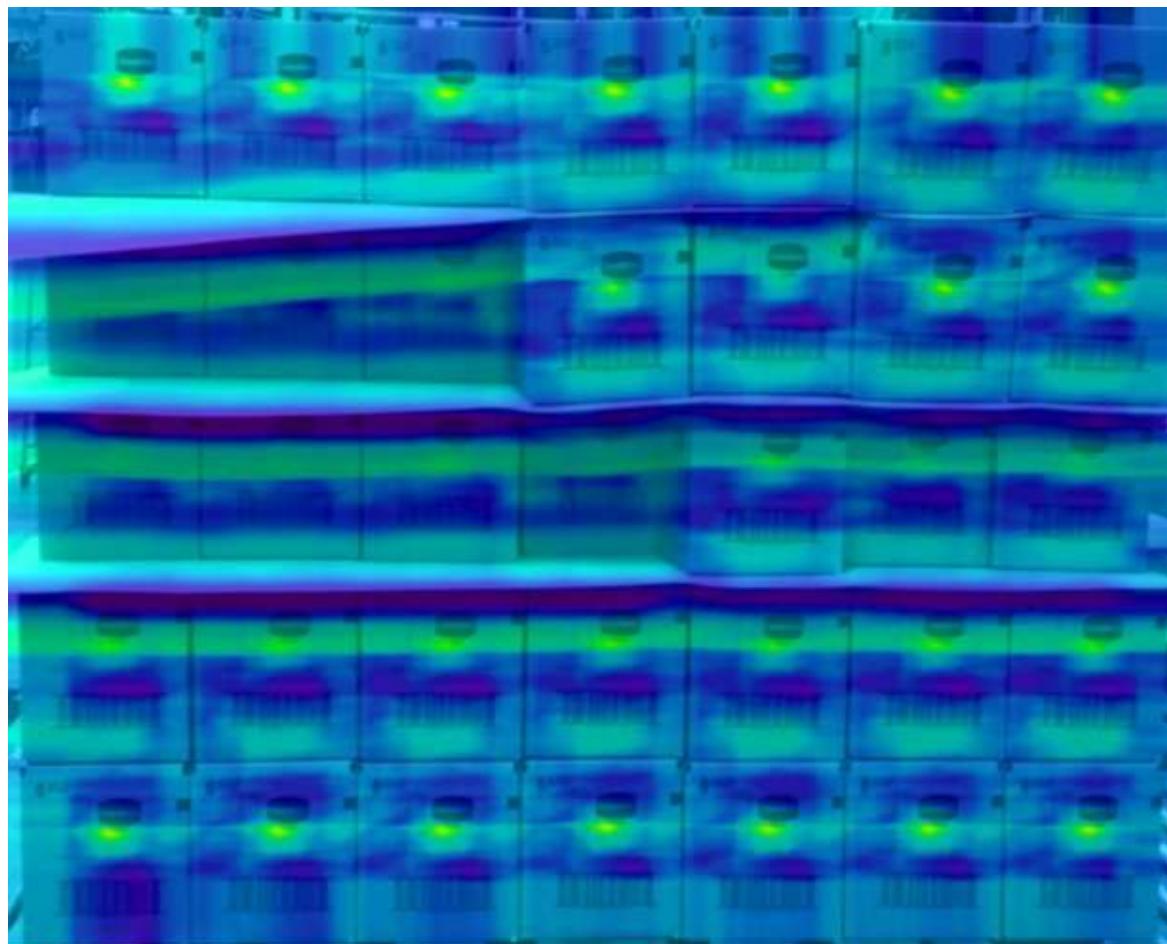
Detection of patterns and anomalies

Real image



Detection of patterns and anomalies

Result heatmap: confidence level to match the pattern



pattern
(auto detected)

Detection of patterns and anomalies

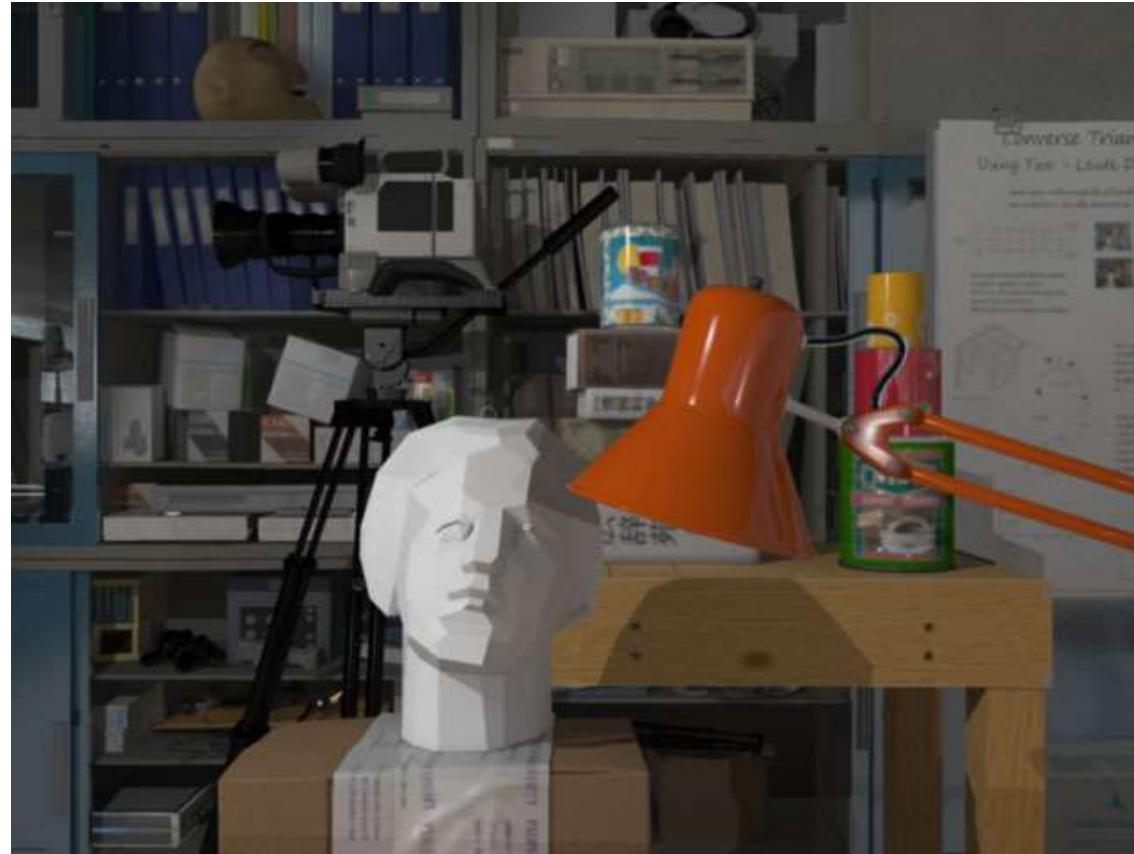
Result mark-up: positions of the patterns + outlier area



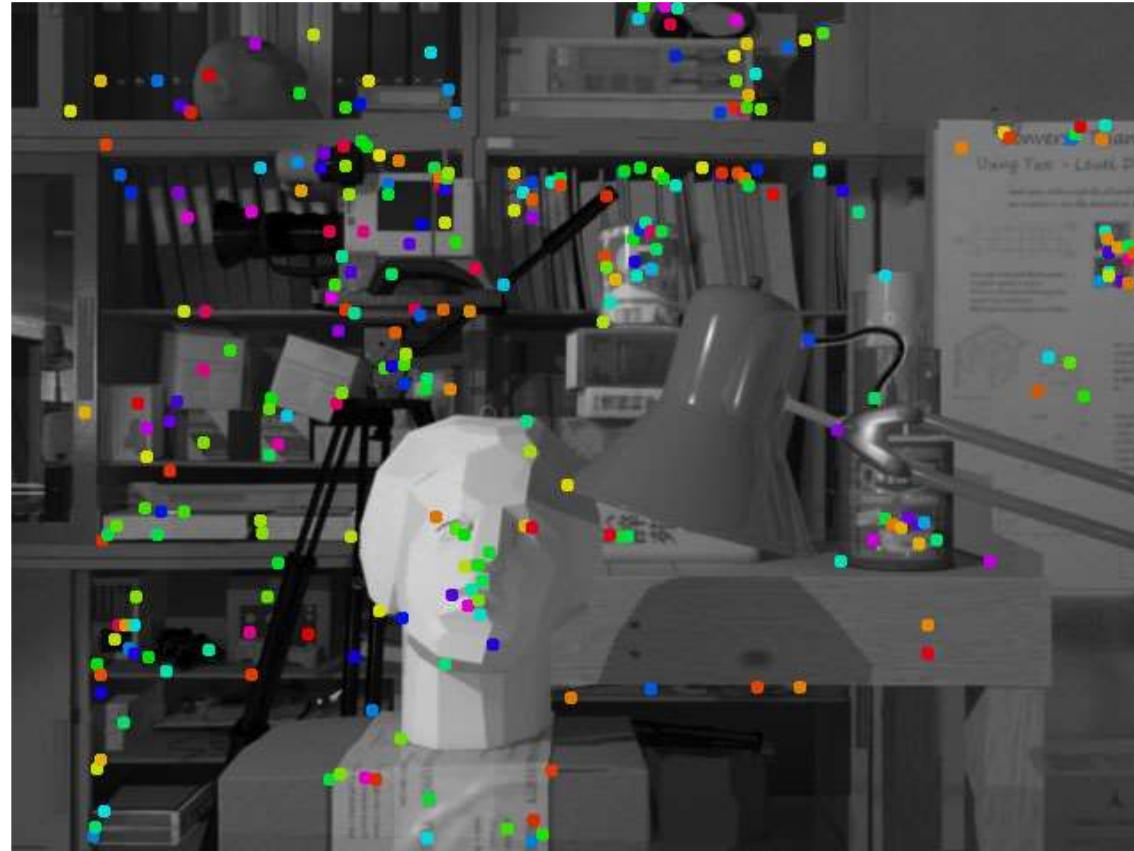
15. Stereo Vision



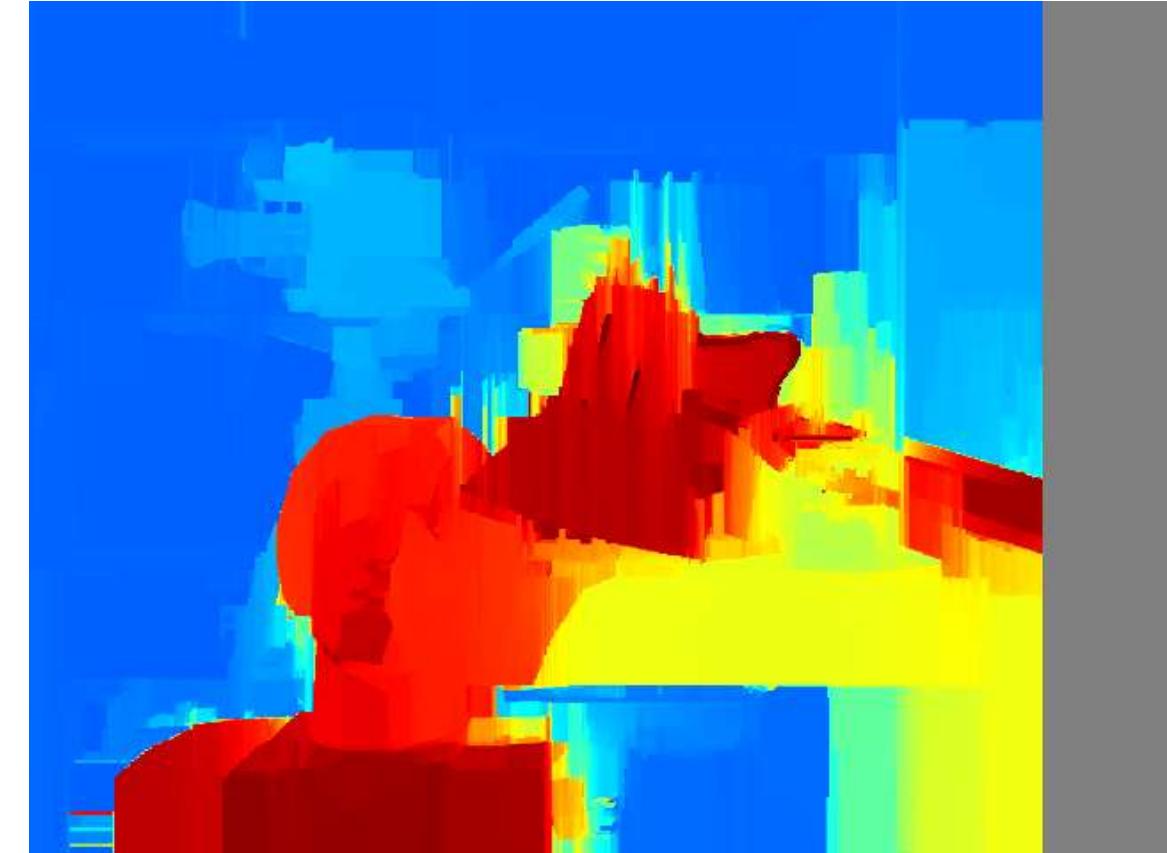
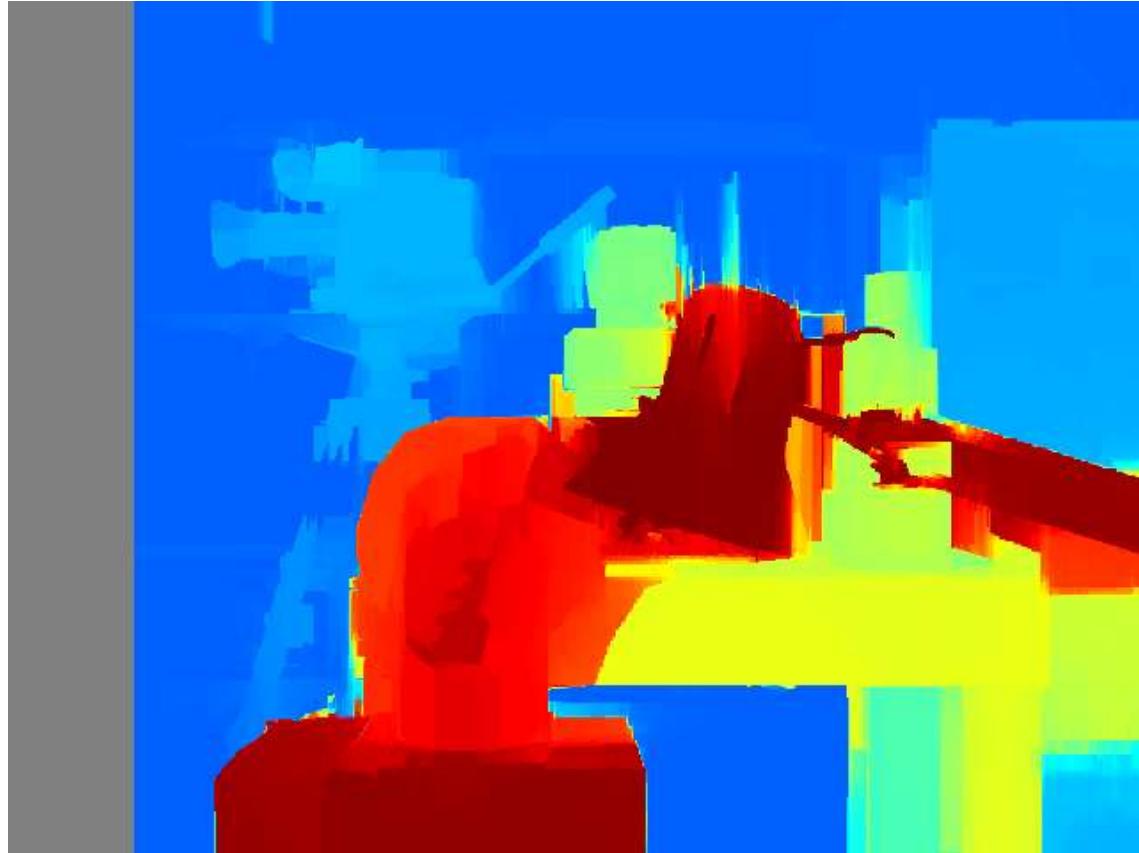
Stereo vision



Stereo vision



Stereo vision



Stereo vision

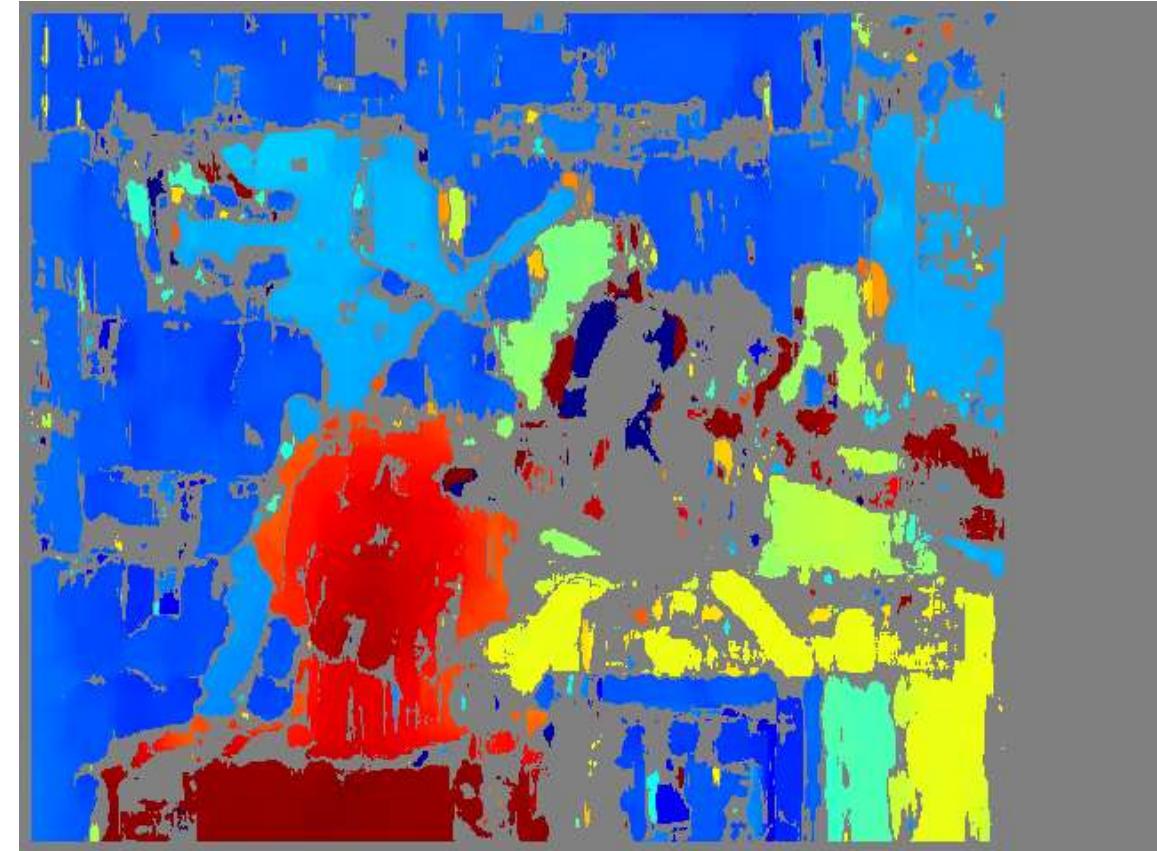
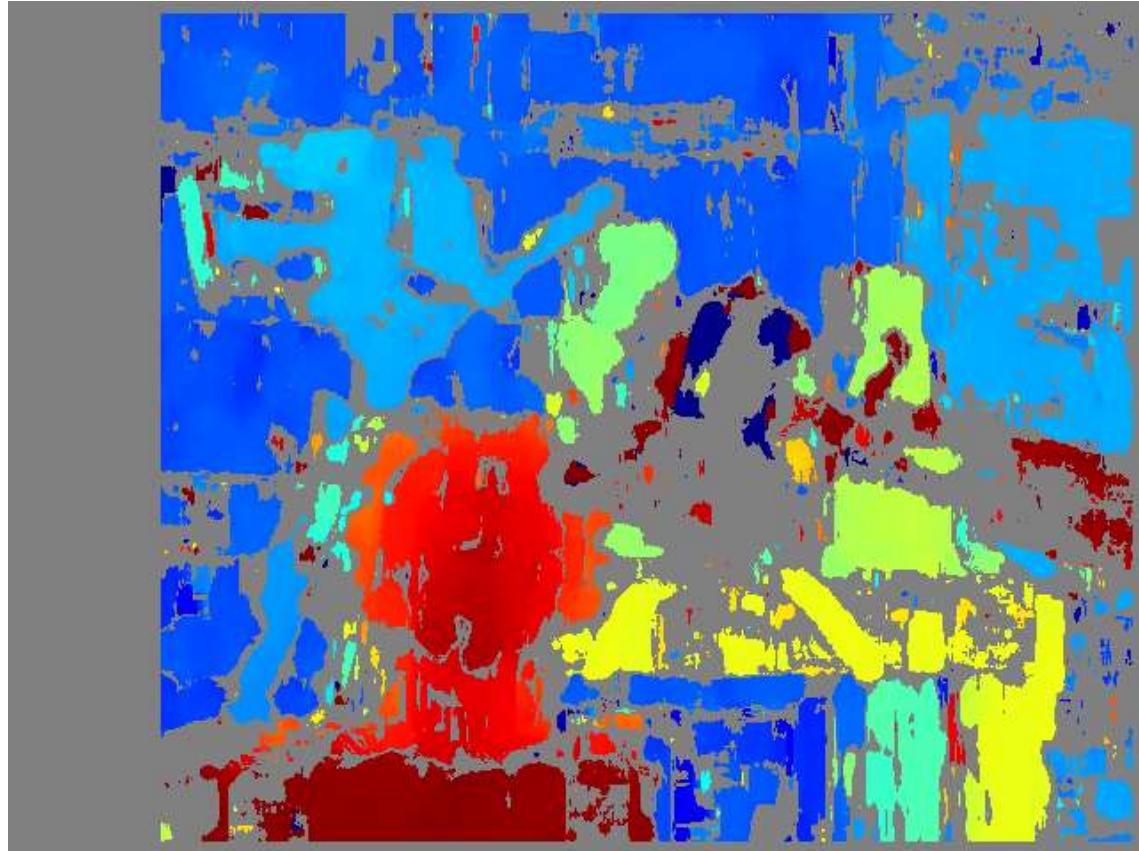
```
def get_disparity_v_01(imgL, imgR, disp_v1, disp_v2, disp_h1, disp_h2):
    window_size = 7
    left_matcher = cv2.StereoSGBM_create(
        minDisparity=disp_h1,
        numDisparities=int(0 + (disp_h2 - disp_h1) / 16) * 16,
        blockSize=5,
        P1=8 * 3 * window_size ** 2,
        P2=32 * 3 * window_size ** 2,
        disp12MaxDiff=disp_h2,
        uniquenessRatio=15,
        speckleWindowSize=0,
        speckleRange=2,
        preFilterCap=63,
        mode=cv2.STEREO_SGBM_MODE_SGBM_3WAY
    )

    right_matcher = cv2.ximgproc.createRightMatcher(left_matcher)
    dispr = right_matcher.compute(imgR, imgL)
    displ = left_matcher.compute(imgL, imgR)

    wls_filter = cv2.ximgproc.createDisparityWLSFilter(matcher_left=left_matcher)
    wls_filter.setLambda(80000)
    wls_filter.setSigmaColor(1.2)

    filteredImg_L = wls_filter.filter(displ, imgL, None, dispr)
    wls_filter = cv2.ximgproc.createDisparityWLSFilter(matcher_left=right_matcher)
    filteredImg_R = wls_filter.filter(dispr, imgR, None, displ)
```

Stereo vision

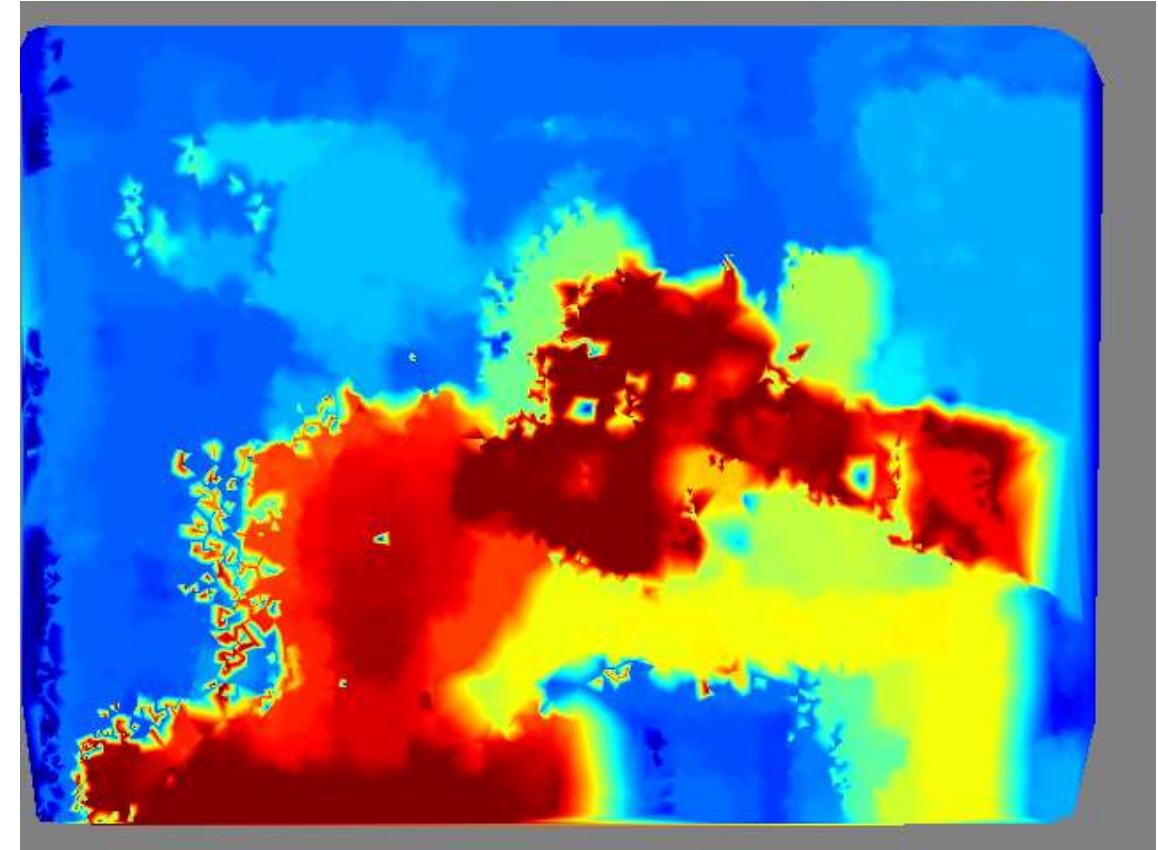
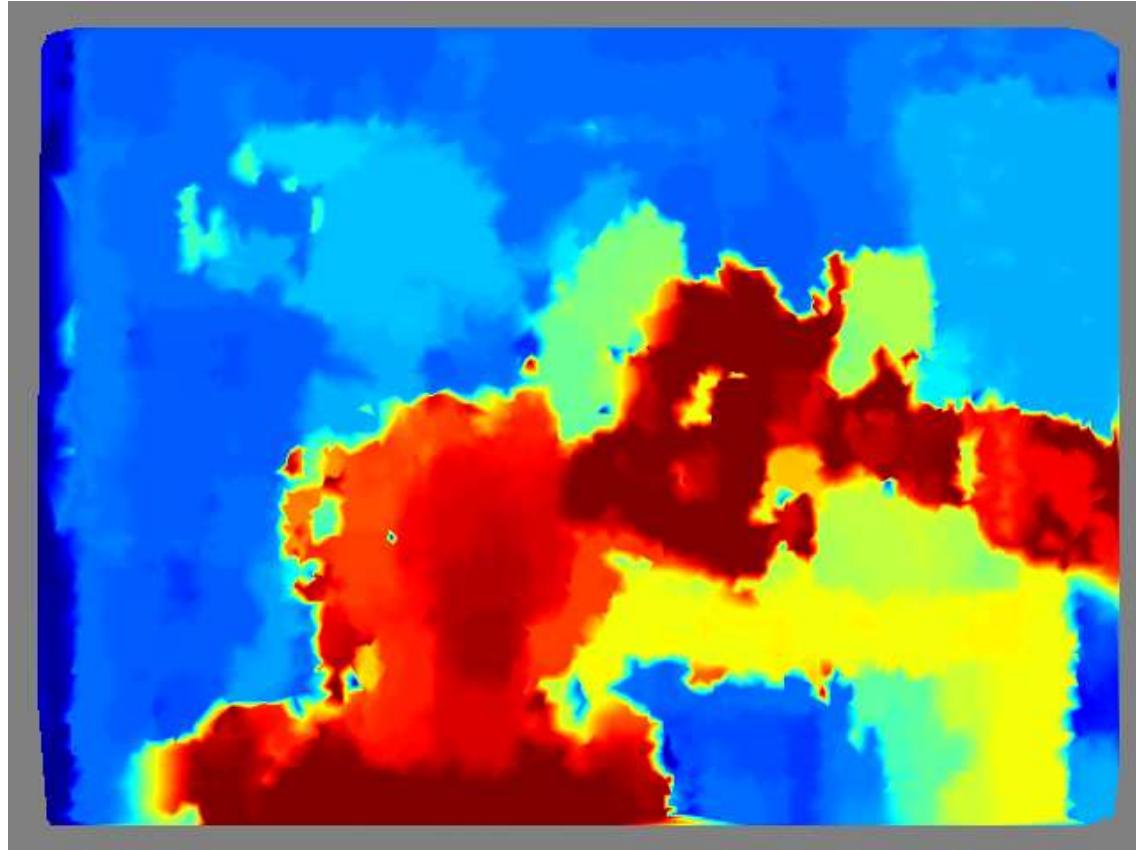


Stereo vision

```
def get_disparity_v_02(imgL, imgR, disp_v1, disp_v2, disp_h1, disp_h2):
    max = numpy.maximum(math.fabs(disp_h1), math.fabs(disp_h2))
    levels = int(1 + (max) / 16) * 16
    stereo = cv2.StereoBM_create(numDisparities=levels, blockSize=15)
    displ = stereo.compute(imgL, imgR)

    dispr = numpy.flip(stereo.compute(numpy.flip(imgR, axis=1), numpy.flip(imgL, axis=1)), axis=1)
    return -displ / 16, -dispr / 16
```

Stereo vision



Stereo vision

```
def get_best_matches(image1,image2,disp_v1, disp_v2, disp_h1, disp_h2, window_size=15,step=10):

    N = int(image1.shape[0] * image1.shape[1] / step / step)
    rand_points = numpy.random.rand(N,2)
    rand_points[:, 0] = window_size + (rand_points[:, 0]*(image1.shape[0]-2*window_size))
    rand_points[:, 1] = window_size + (rand_points[:, 1]*(image1.shape[1]-2*window_size))

    coord1,coord2,quality = [],[],[]

    for each in rand_points:
        row,col = int(each[0]),int(each[1])

        template = tools_image.crop_image(image1,row-window_size,col-window_size,row+window_size,col+window_size)
        top,left,bottom,right = row - window_size + disp_v1, col - window_size + disp_h1, row + window_size + disp_v2, col + window_size + disp_h2
        field = tools_image.crop_image(image2,top,left,bottom,right)

        q = cv2.matchTemplate(field, template, method=cv2.TM_CCOEFF_NORMED)
        q = q[1:, 1:]
        q = (q + 1) * 128

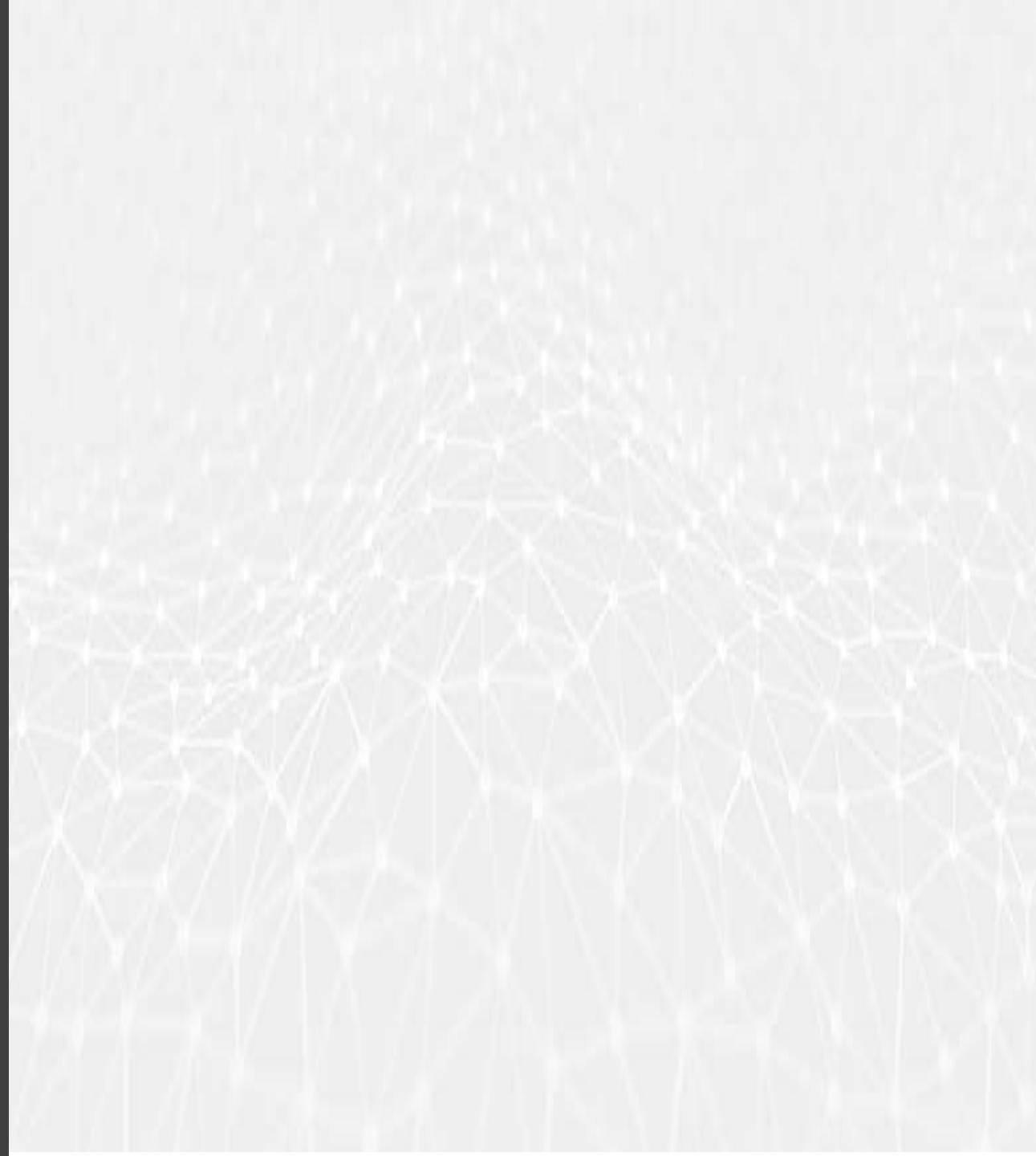
        idx = numpy.argmax(q.flatten())
        q_best = numpy.max(q.flatten())
        qq=q[int(idx/q.shape[1]) , idx % q.shape[1]]

        if q_best>0:
            dr = int(idx/q.shape[1])+disp_v1
            dc = idx % q.shape[1]+disp_h1

            if col + dc>=0 and col + dc<image1.shape[1] and row + dr>=0 and row + dr<image1.shape[0]:
                coord1.append([col      ,row      ])
                coord2.append([col+ dc, row+dr])
                quality.append(q_best)

    return numpy.array(coord1), numpy.array(coord2),numpy.array(quality)
```

2.1. CNNs out of box



CNNs out of box

The popular networks

Classification

- LeNet [Model](#)
- AlexNet [Model](#)
- VGG [Model](#)
- ResNet [Paper](#)
- YOLO9000 [Paper](#)
- DenseNet [Paper](#)

Segmentation

- FCN8 [Paper](#)
- SegNet [Paper](#)
- U-Net [Paper](#)
- E-Net [Paper](#)
- ResNetFCN [Paper](#)
- PSPNet [Paper](#)
- Mask RCNN [Paper](#)

Detection

- Faster RCNN [Paper](#)
- SSD [Paper](#)
- YOLOv2 [Paper](#)
- R-FCN [Paper](#)

CNNs out of box

Some datasets available for research

0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9

MNIST: 10 classes, ~7000 ex. per class



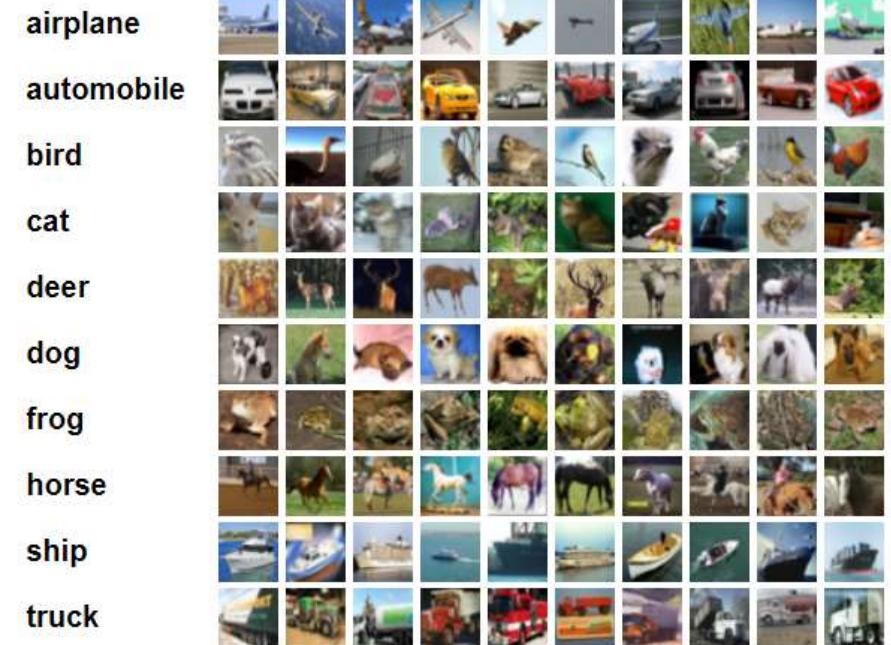
ImageNet: 1000 classes, ~100 ex per class

CNNs out of box

Some datasets available for research



The Street View House Numbers
10 classes, ~2000 ex. per class



CIFAR: 10 classes, 6000 ex. per class
100 classes, 600 ex per class

CNNs out of box

Some datasets available for research



Olivetti database: 40 classes

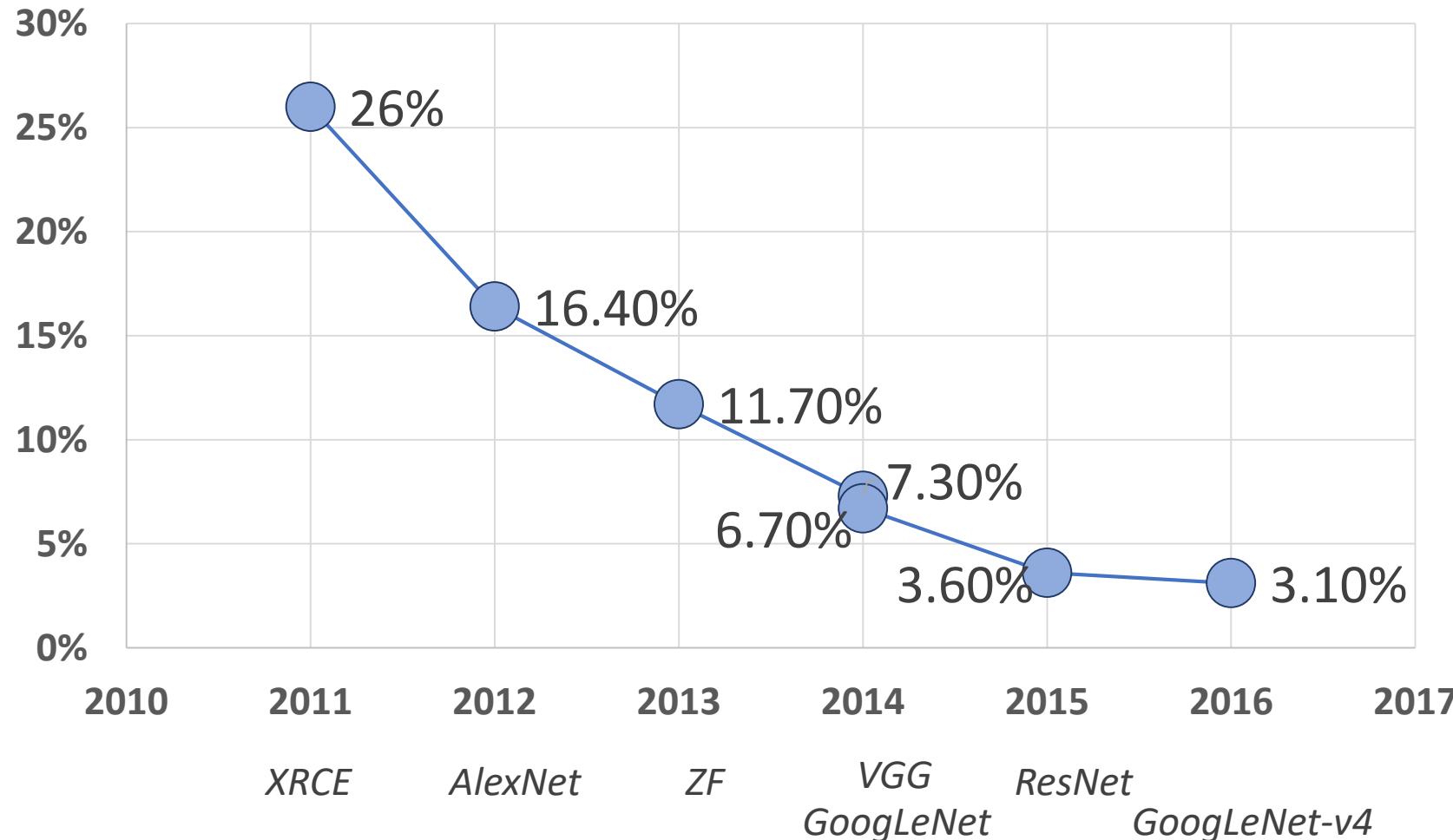
The screenshot shows the Kaggle datasets interface. At the top, there's a navigation bar with back, forward, and search icons, followed by the URL <https://www.kaggle.com/datasets>. Below the navigation is a blue header bar with the word "Datasets". Underneath the header, there are three tabs: "Public" (selected), "Your Datasets", and "Favorites". Below the tabs are filters for "Sizes", "File types", "Licenses", and "Tags". The main content area displays four dataset cards:

- Heart Disease UCI**: 101 datasets. Description: "https://archive.ics.uci.edu/ml/datasets/Heart+Disease". Last updated: 7 months ago (Version 1). Tags: biology, health, classification, binary clas...
- Graduate Admissions**: 424 datasets. Description: "Predicting admission from important parameters". Last updated: a month ago (Version 2). Tags: regression, model com..., random for..., + 2 more...
- FIFA 19 complete player dataset**: 334 datasets. Description: "18k+ FIFA 19 players, ~90 attributes extracted from the late...". Last updated: a month ago. Tags: sports, data visuali..., regression, + 2 more...
- Suicide Rates Overview 1985 to 2016**: 54 datasets. Description: "Compares socio-economic info with suicide rates by year &...". Last updated: a month ago. Tags: world, demograph..., economics

.. and much more @ kaggle

CNNs out of box

ImageNet Classification error



CNNs out of box

Usage of the neural networks

identify the name of a street (in France) from an image
compressing and decompressing images
semantic image segmentation
real-world image text extraction.
image classification
image-to-text
unsupervised learning
3D object reconstruction
image matching and retrieval
automatic speech recognition
localizing and identifying multiple objects in a single image
computer vision
discovery of latent 3D keypoints
predicting future video frames

CNNs out of box

The screenshot shows a browser window with the URL https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector/1. The page has a header "TensorFlow Hub" with a search icon. Below the header, there's a breadcrumb navigation: "← imagenet/resnet_v2_50/feature_vector".

Usage

This module implements the common signature for computing [image feature vectors](#). It can be used like

```
module = hub.Module("https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector/1")
height, width = hub.get_expected_image_size(module)
images = ... # A batch of images with shape [batch_size, height, width, 3].
features = module(images) # Features with shape [batch_size, num_features].
```

...or using the signature name `image_feature_vector`. The output for each image in the batch is a feature vector of size `num_features = 2048`.

For this module, the size of the input image is fixed to `height x width = 224 x 224` pixels. The input `images` are expected to have color values in the range `[0,1]`, following the [common image input](#) conventions.

CNNs out of box

```
import tensorflow_hub as hub
import urllib.request
import cv2
import numpy
import json
import tensorflow as tf
# -----
URL = 'https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json'
data = json.loads(urllib.request.urlopen(URL).read().decode())
class_names = [data['%d'%i][1] for i in range(0,999)]
# -----
def example_predict():

    module = hub.Module("https://tfhub.dev/google/imagenet/resnet_v2_50/classification/1")
    img = cv2.imread('data/ex-natural/dog/dog_0000.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)).astype(numpy.float32)
    img = numpy.array([img]).astype(numpy.float32) / 255.0
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

    outputs = module(dict(images=img), signature="image_classification", as_dict=True)
    prob = outputs["default"].eval(session=sess)[0]
    idx = numpy.argsort(-prob)[0]

    print(class_names[idx], prob[idx])
    sess.close()

    return
# -----
if __name__ == '__main__':
    example_predict()
```

CNNs out of box

← → ⌂ https://keras.io/getting-started/faq/#how-can-i-use-pre-trained-models-in-keras



Keras Documentation

Search docs

Home

Why use Keras

GETTING STARTED

Guide to the Sequential model

Guide to the Functional API

FAQ

How should I cite Keras?

How can I run Keras on GPU?

How can I run a Keras model on multiple GPUs?

What does "sample", "batch", "epoch" mean?

How can I save a Keras model?

Why is the training loss much higher than the testing loss?

How can I obtain the output of an intermediate layer?

How can I use Keras with datasets that don't fit in memory?

How can I interrupt training when the validation loss isn't decreasing

How can I use pre-trained models in Keras?

Code and pre-trained weights are available for the following image classification models:

- Xception
- VGG16
- VGG19
- ResNet50
- Inception v3
- Inception-ResNet v2
- MobileNet v1
- DenseNet
- NASNet
- MobileNet v2

They can be imported from the module `keras.applications`:

```
from keras.applications.xception import Xception
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.resnet50 import ResNet50
from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_resnet_v2 import InceptionResNetV2
from keras.applications.mobilenet import MobileNet
from keras.applications.densenet import DenseNet121
from keras.applications.densenet import DenseNet169
from keras.applications.densenet import DenseNet201
from keras.applications.nasnet import NASNetLarge
from keras.applications.nasnet import NASNetMobile
from keras.applications.mobilenet_v2 import MobileNetV2

model = VGG16(weights='imagenet', include_top=True)
```

CNNs out of box

```
from keras.applications import MobileNet
from keras.applications.xception import Xception
from keras.applications.mobilenet import preprocess_input
import urllib.request
import cv2
import numpy
import json
from keras import backend as K
K.set_image_dim_ordering('tf')
# -----
URL = 'https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json'
data = json.loads(urllib.request.urlopen(URL).read().decode())
class_names = [data['%d'%i][1] for i in range(0,999)]
# -----
def example_predict():

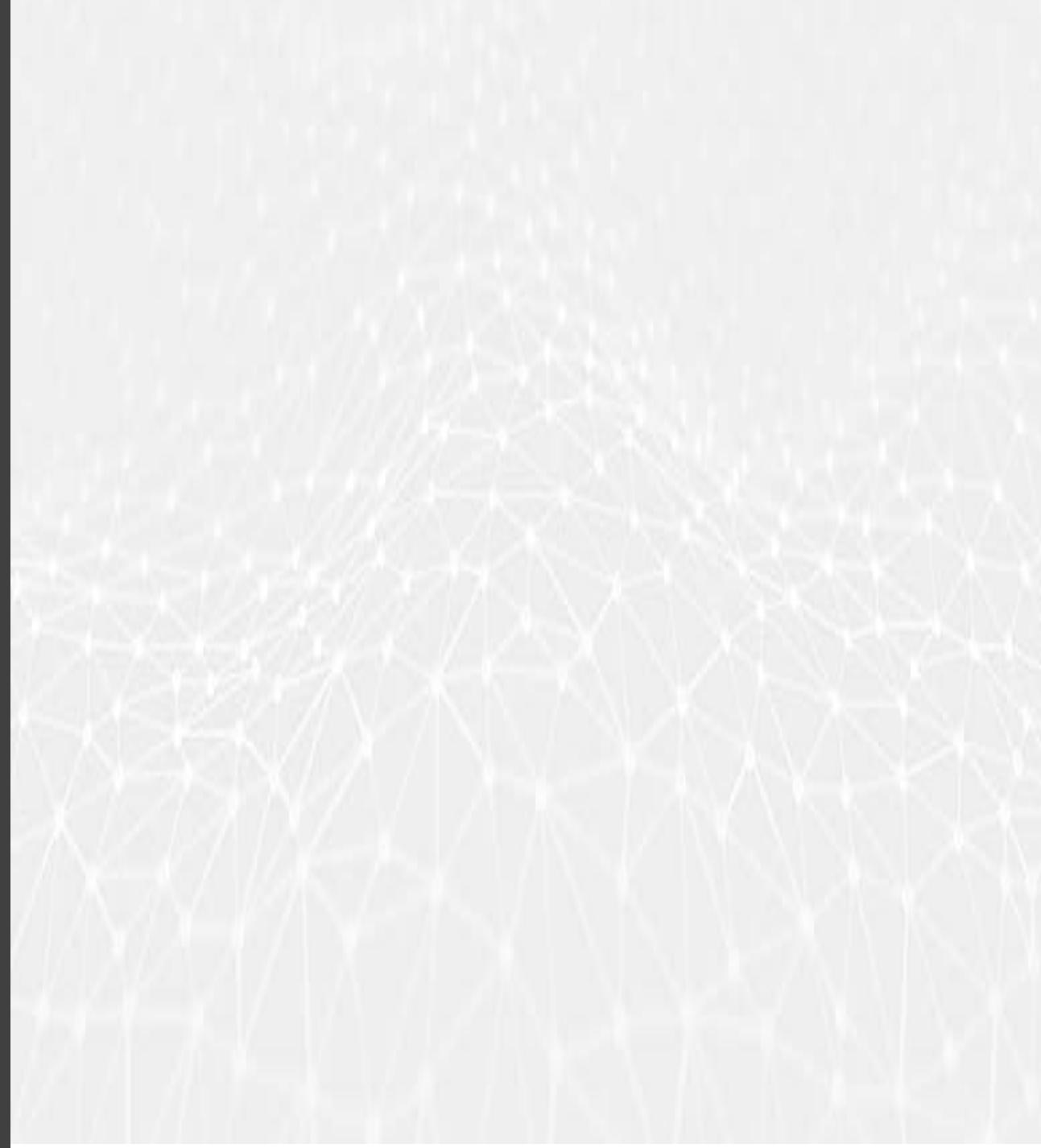
    CNN = MobileNet()
    #CNN = Xception()

    img = cv2.imread('data/ex-natural/dog/dog_0000.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)).astype(numpy.float32)

    prob = CNN.predict(preprocess_input(numpy.array([img])))
    idx = numpy.argsort(-prob[0])[0]
    print(class_names[idx], prob[0, idx])

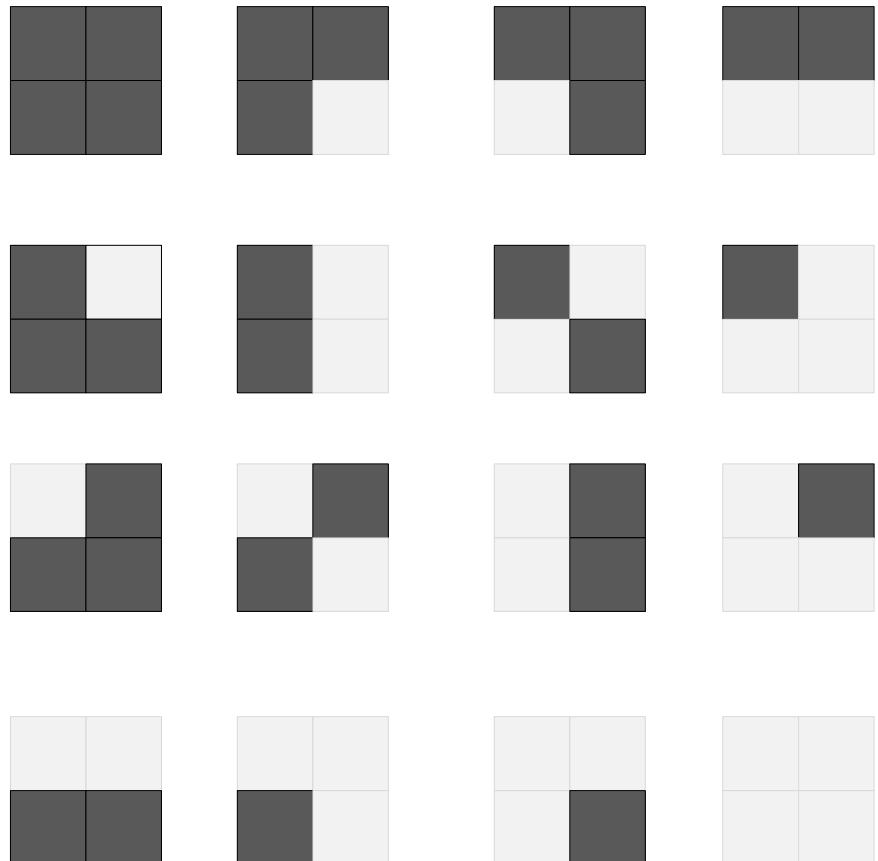
    return
# -----
if __name__ == '__main__':
    example_predict()
```

2.2. Basic operations



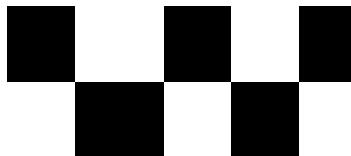
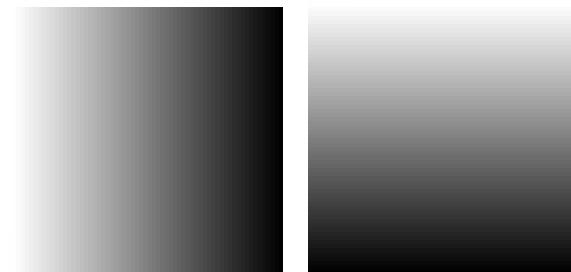
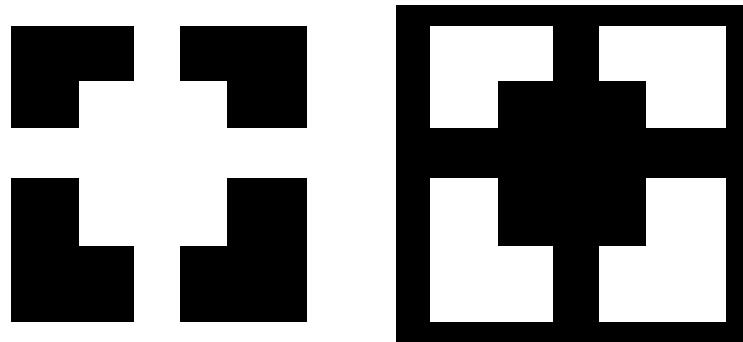
Basic operations

Convolution



Basic operations

Convolution

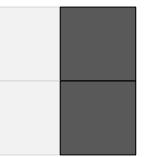
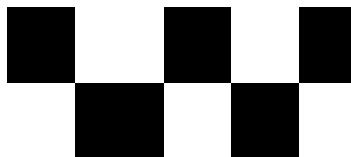
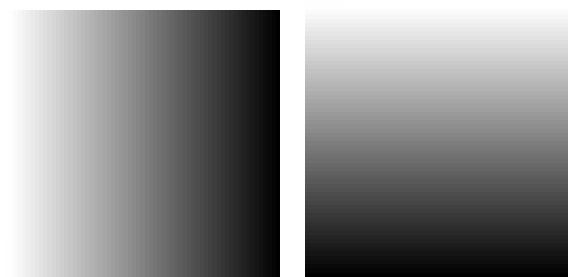
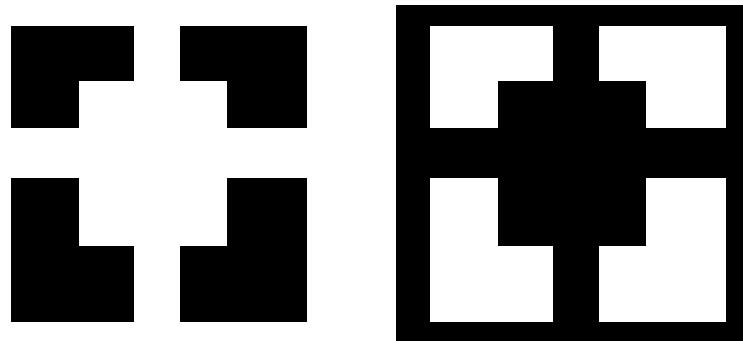


$$\otimes \begin{array}{c} \text{dark gray} \\ \hline \text{white} \end{array} =$$



Basic operations

Convolution

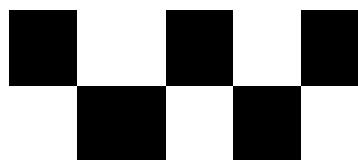
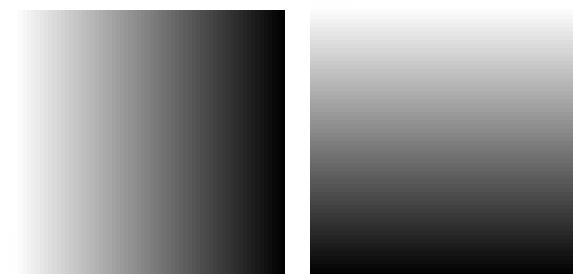
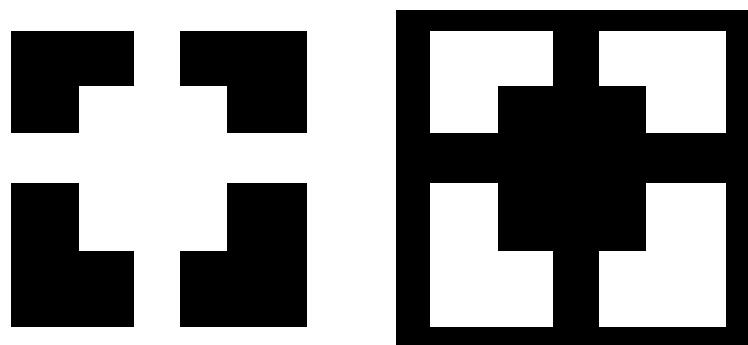


=



Basic operations

Convolution

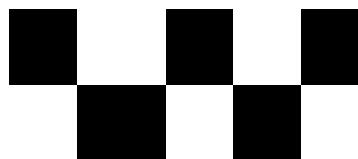
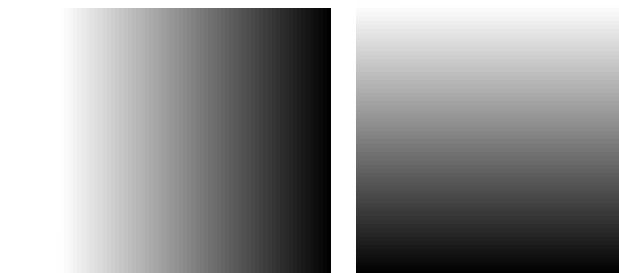
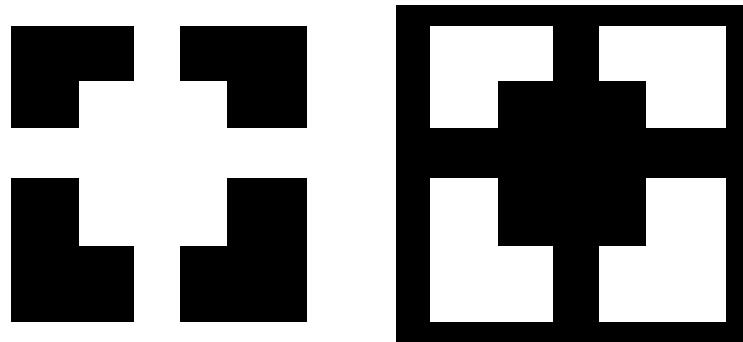


=

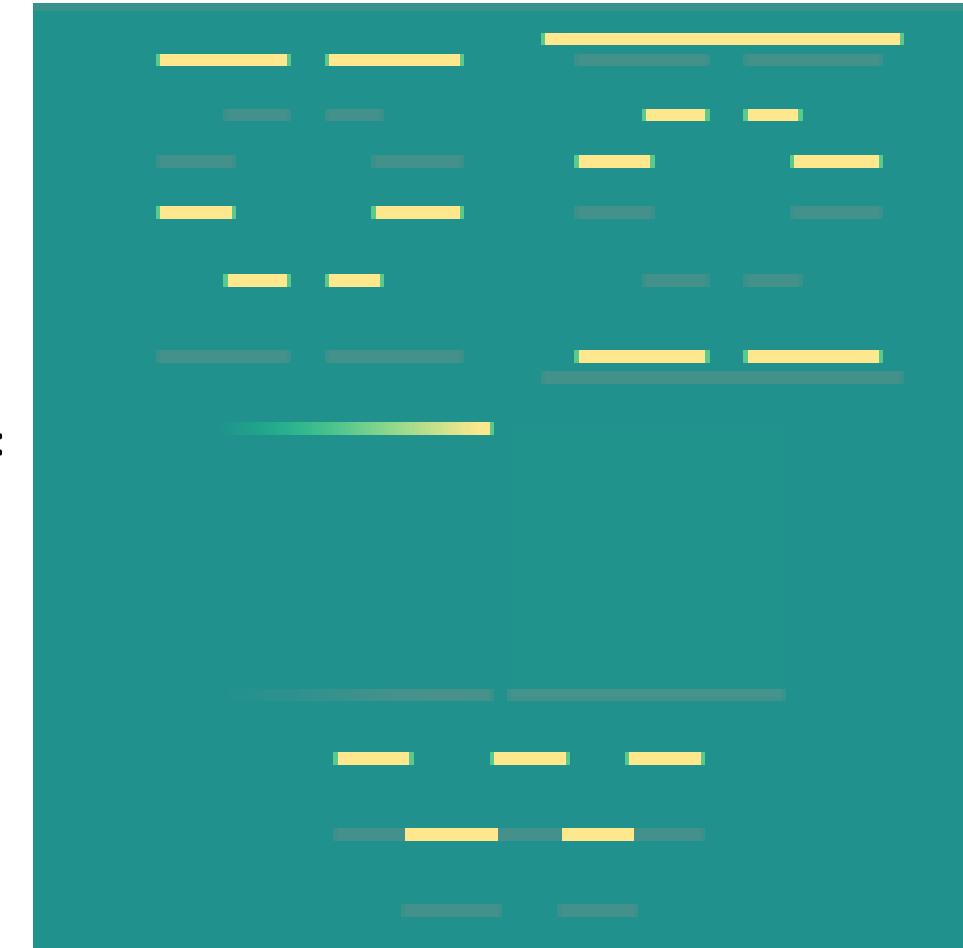


Basic operations

Convolution

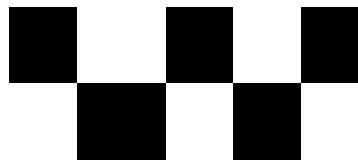
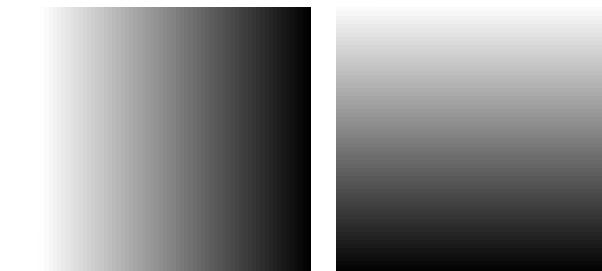
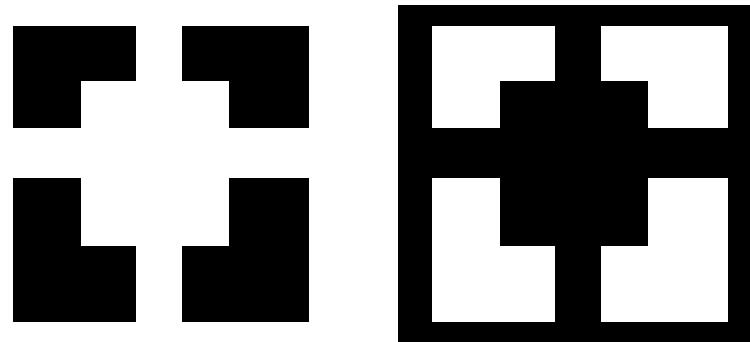


$$\otimes \quad \begin{matrix} & & \\ & & \\ \text{---} & \text{---} & \text{---} \\ & & \end{matrix} =$$

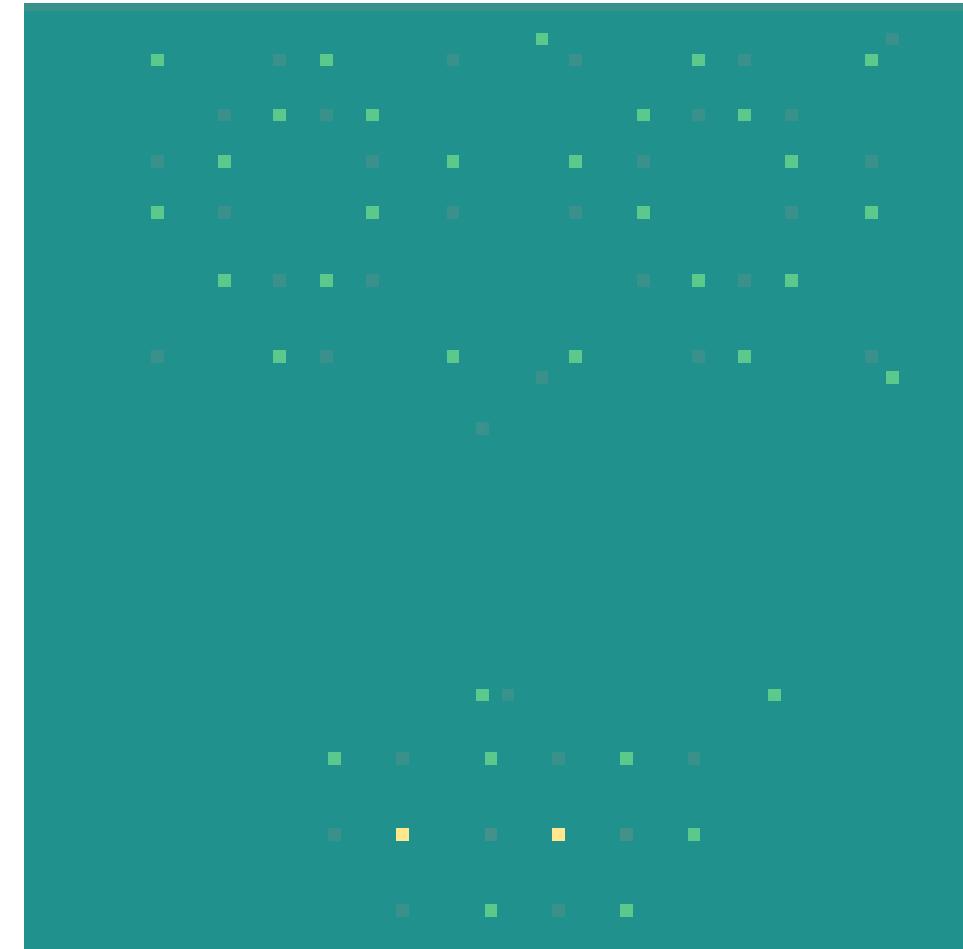


Basic operations

Convolution

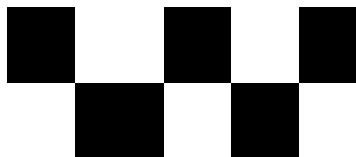
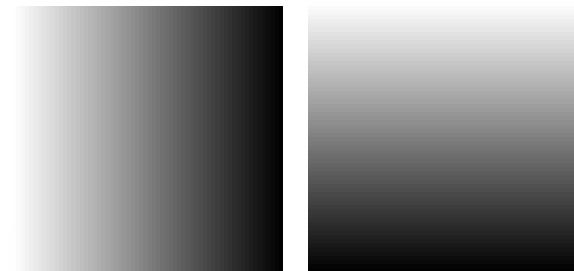
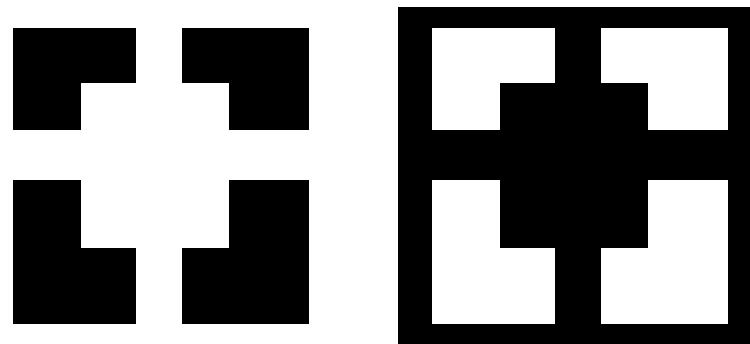


$$\otimes \begin{matrix} \text{dark gray} & \text{white} \\ \text{white} & \text{dark gray} \end{matrix} =$$

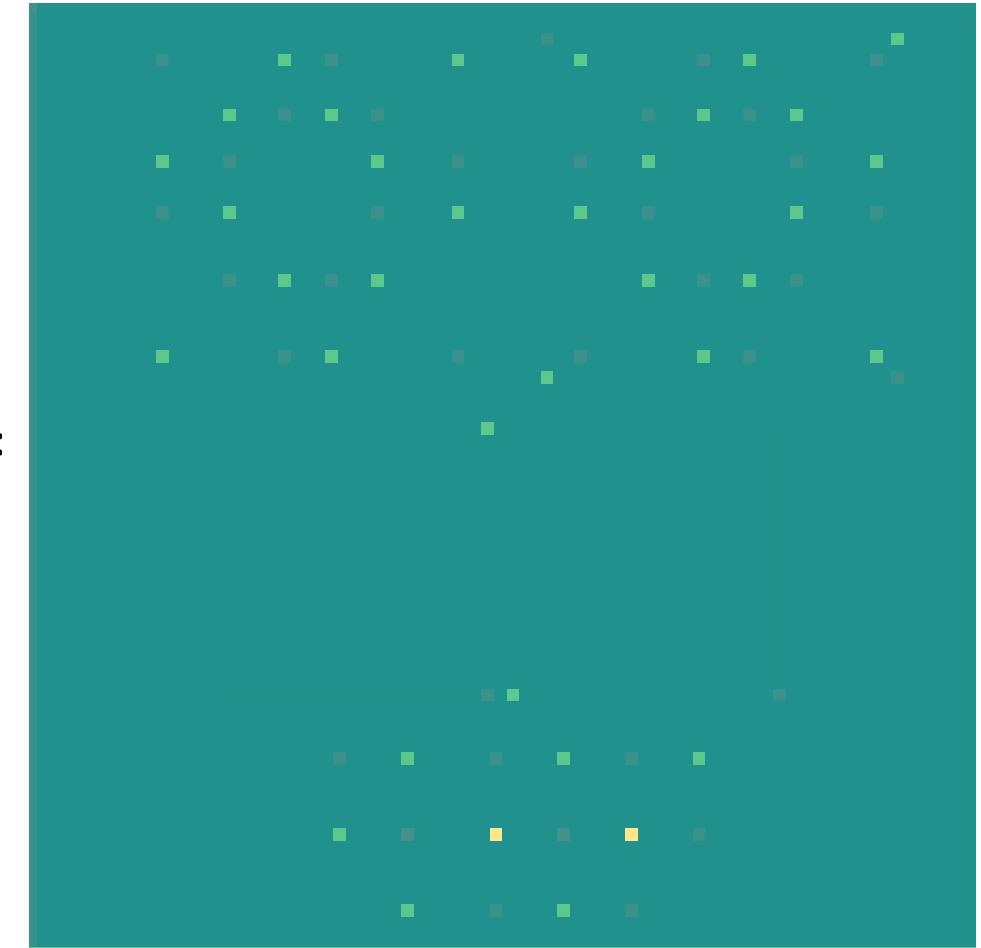


Basic operations

Convolution

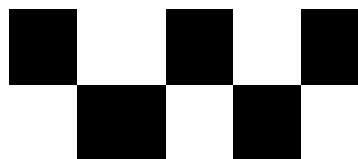
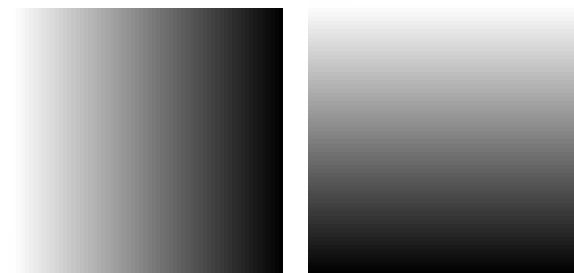
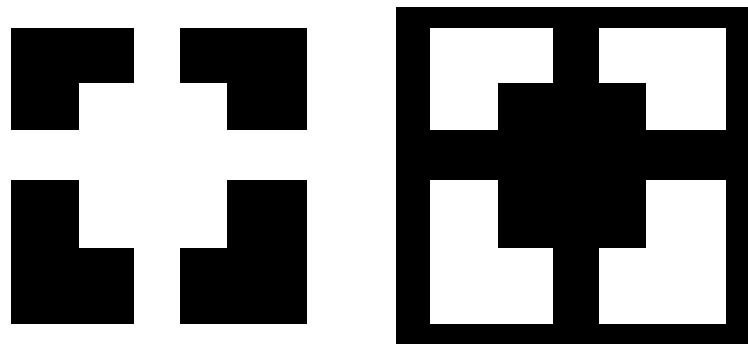


=

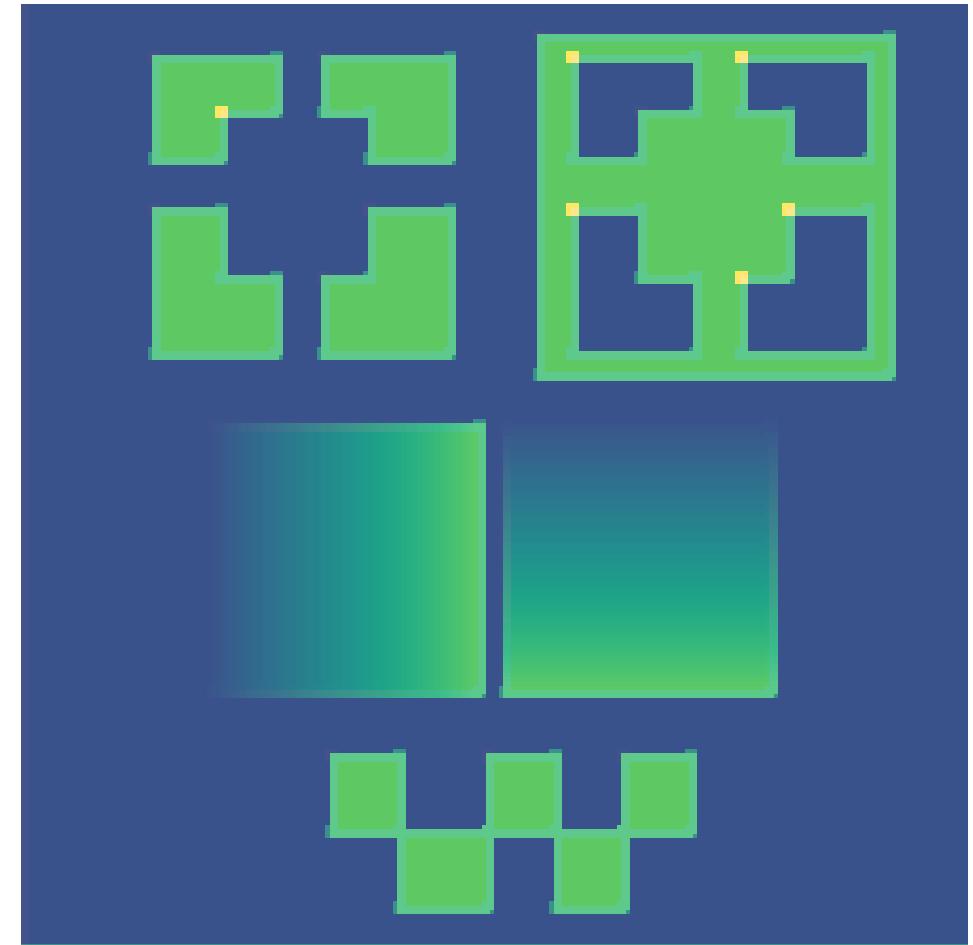


Basic operations

Convolution

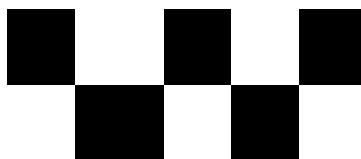
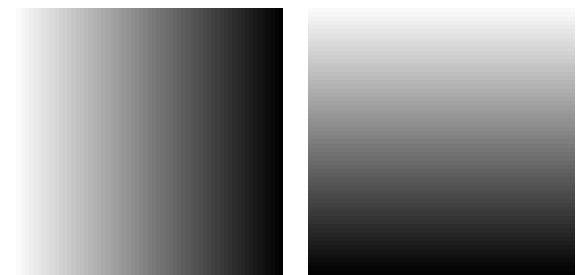
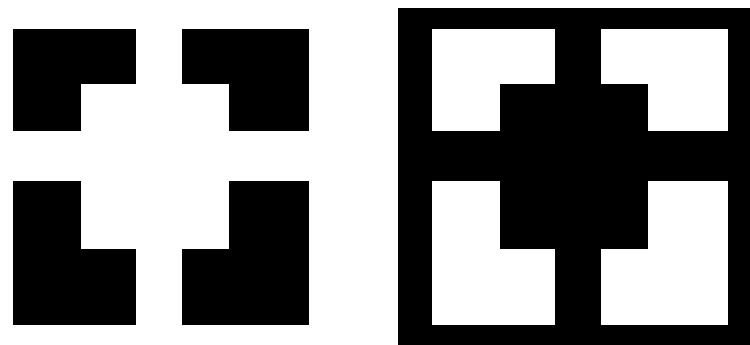


$$\otimes \begin{matrix} & & \\ & \text{---} & \\ & & \end{matrix} =$$

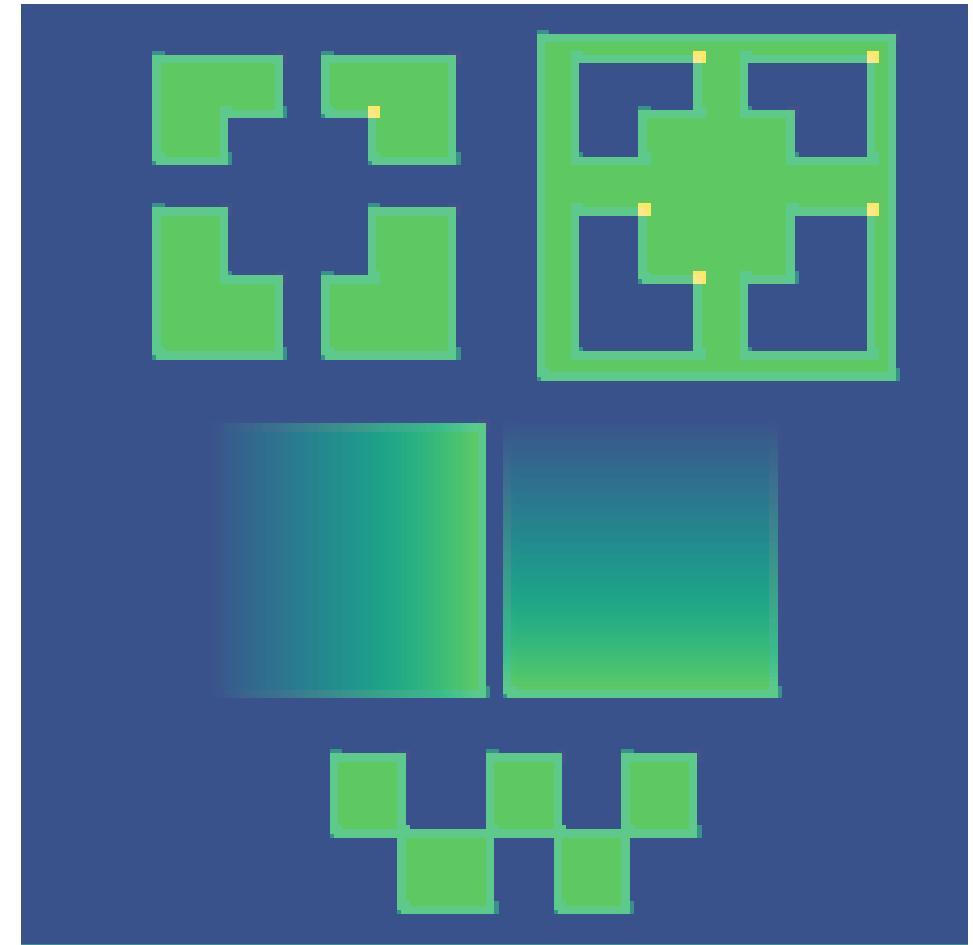


Basic operations

Convolution

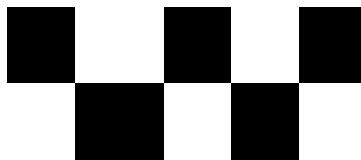
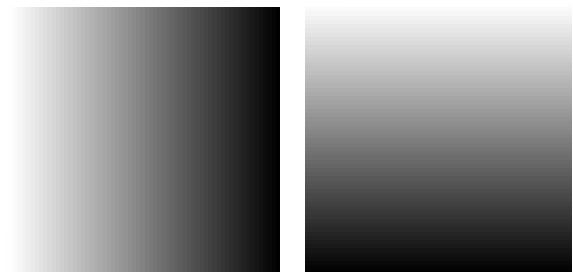


$$\otimes \begin{matrix} & & \\ & \text{dark gray} & \\ & & \end{matrix} =$$

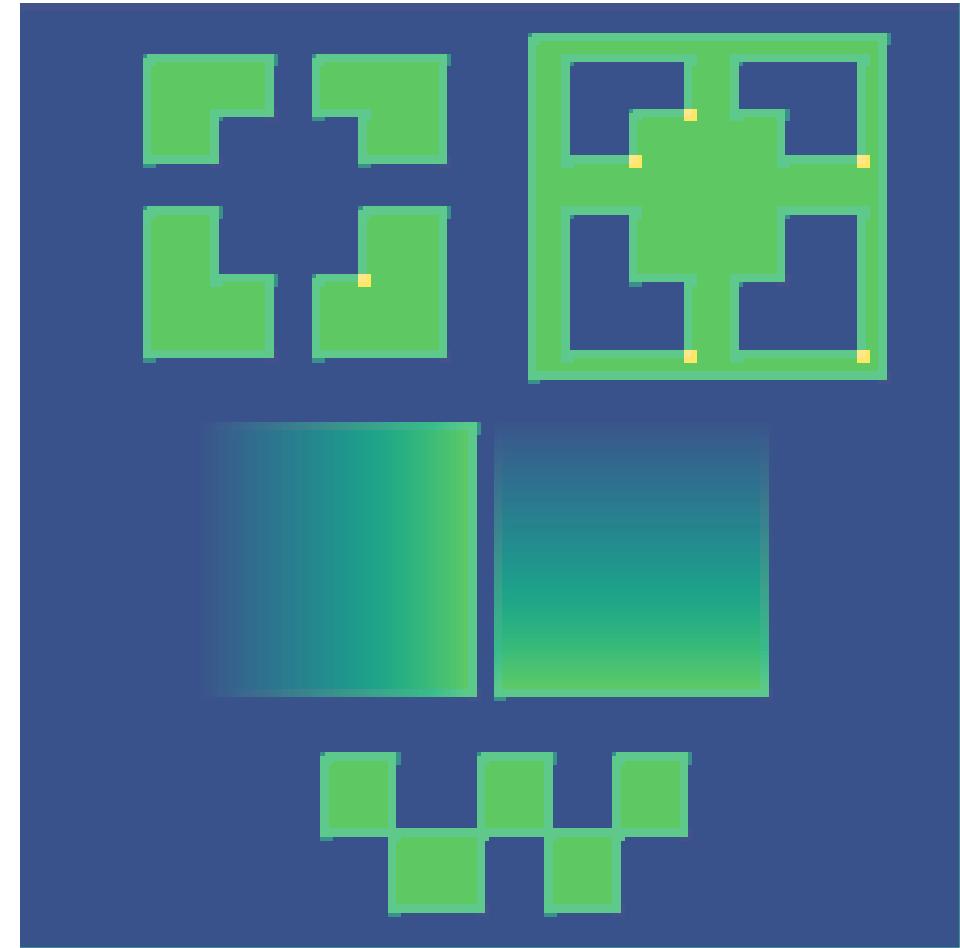


Basic operations

Convolution

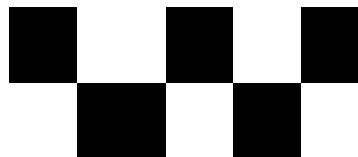
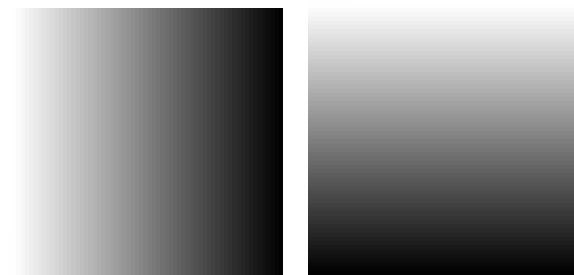
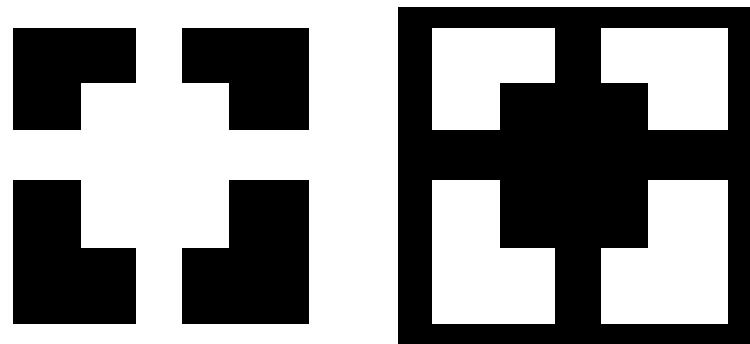


$$\otimes \begin{matrix} & & \\ & & \\ \textcolor{gray}{\otimes} & & \\ & & \end{matrix} =$$

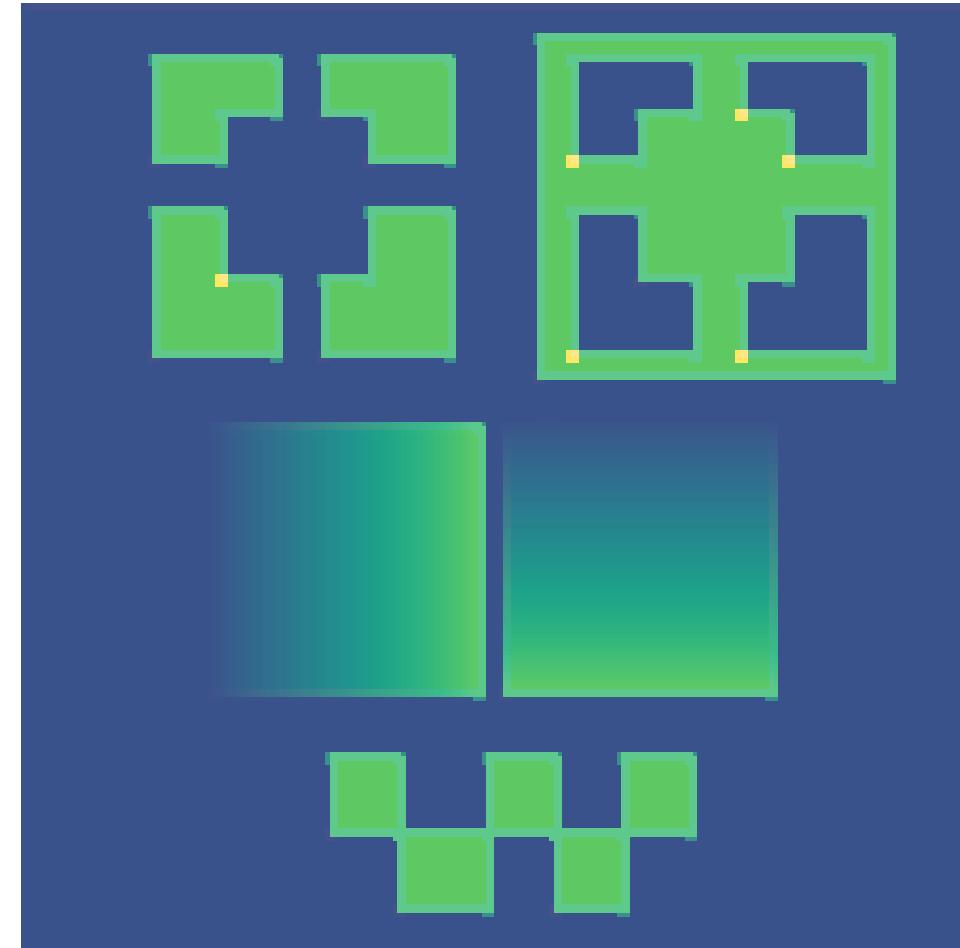


Basic operations

Convolution

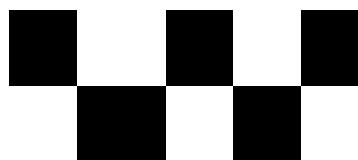
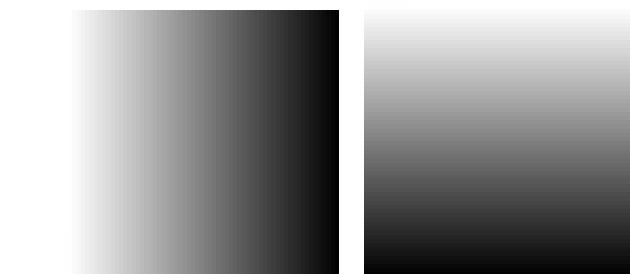
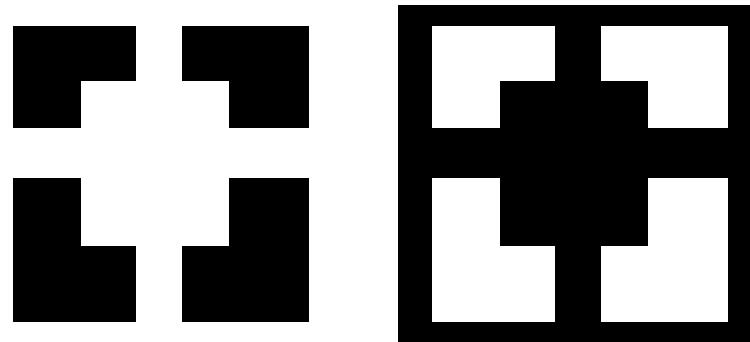


=



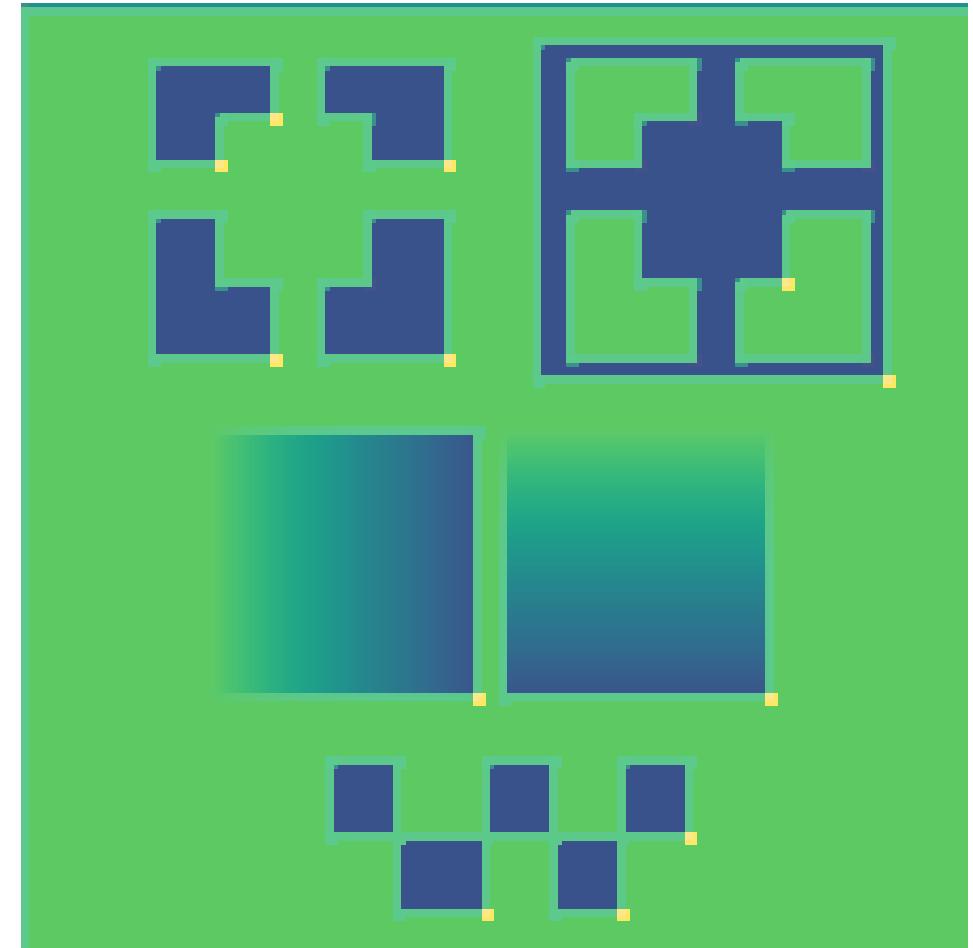
Basic operations

Convolution



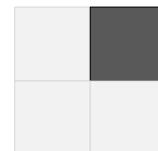
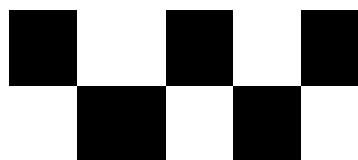
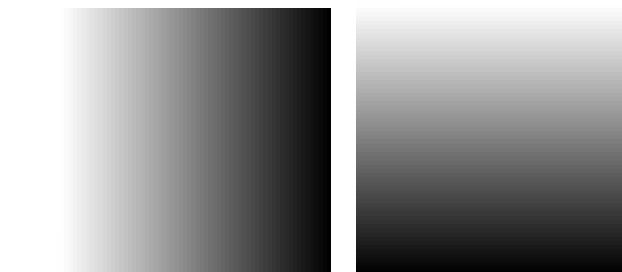
$$\otimes \begin{matrix} \text{dark gray} \\ \text{white} \end{matrix}$$

=

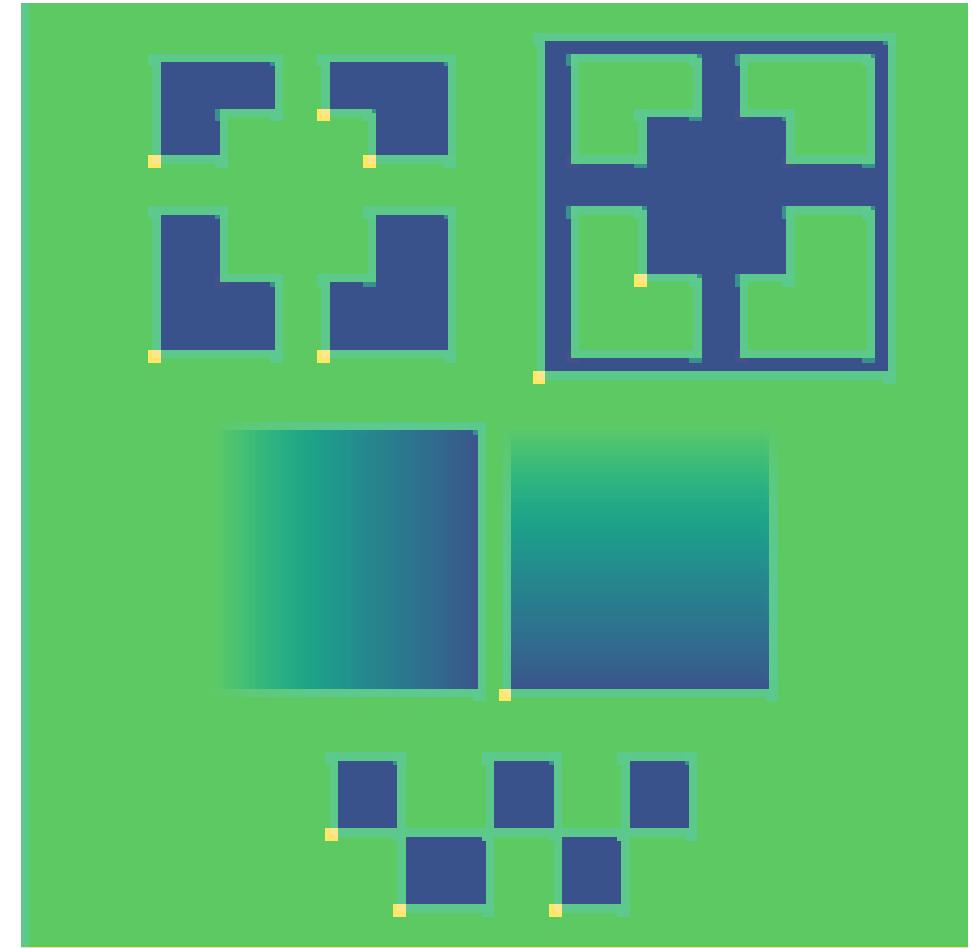


Basic operations

Convolution

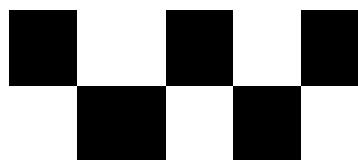
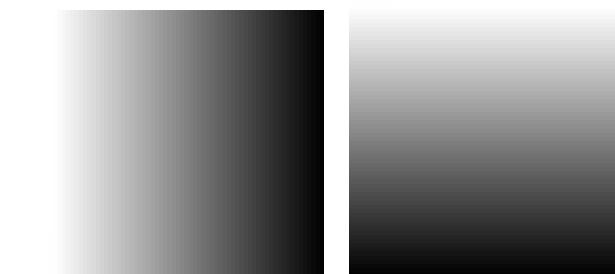


=

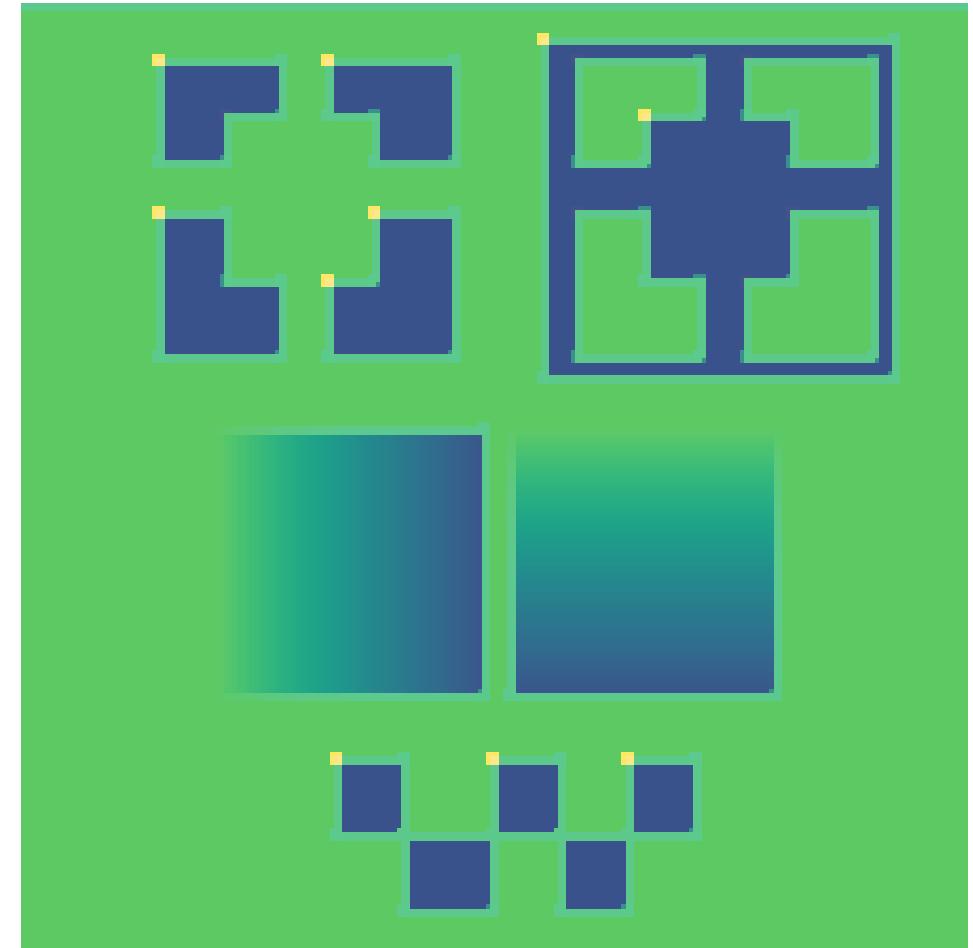


Basic operations

Convolution

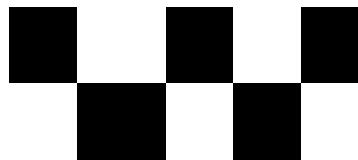
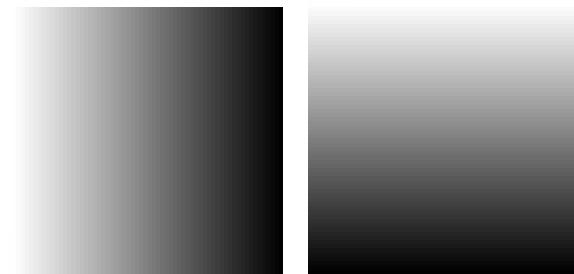


$$\otimes \quad \begin{matrix} & & \\ & & \\ & \text{---} & \\ & & \end{matrix} =$$

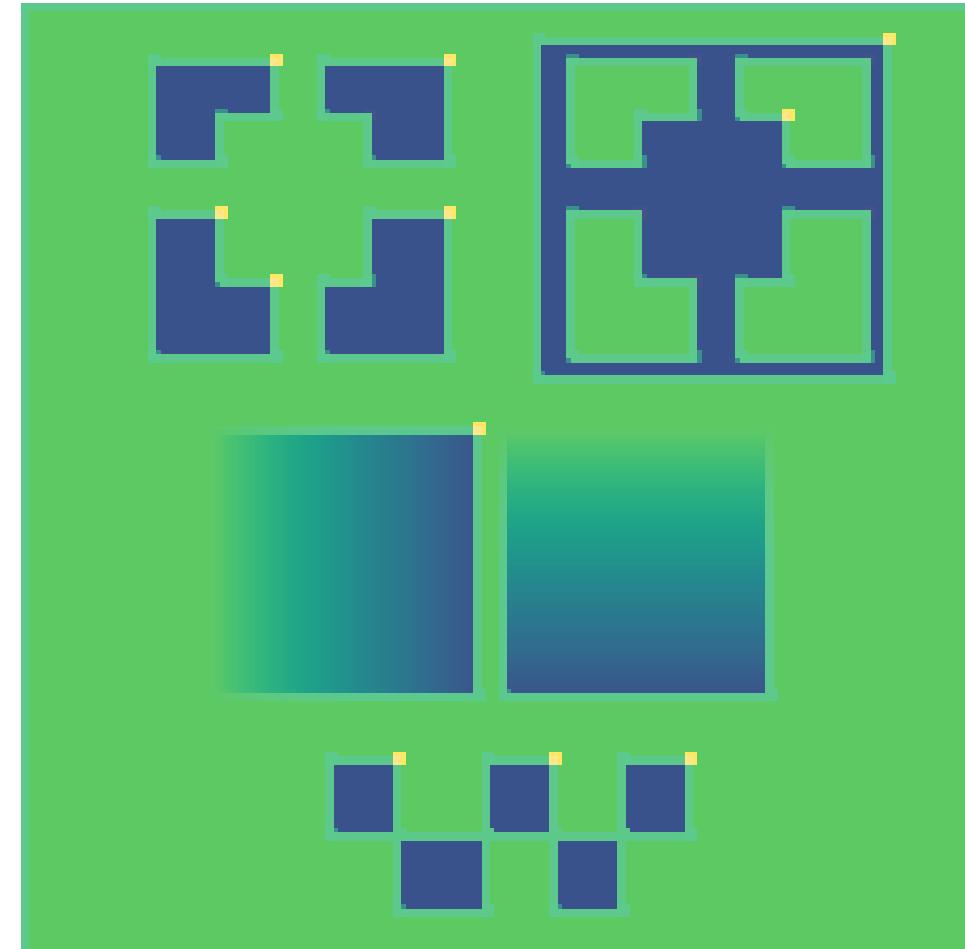


Basic operations

Convolution

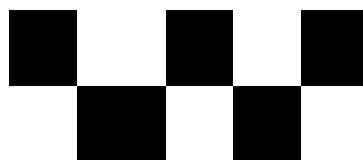
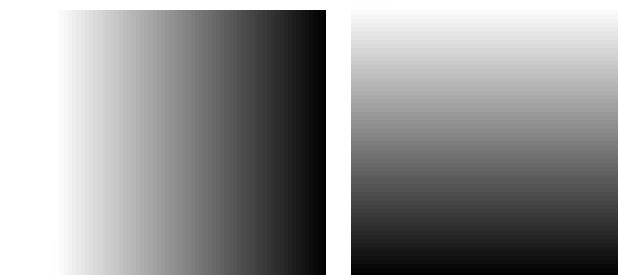
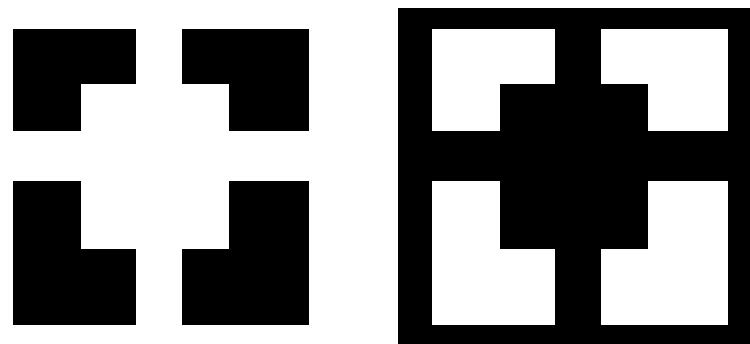


=

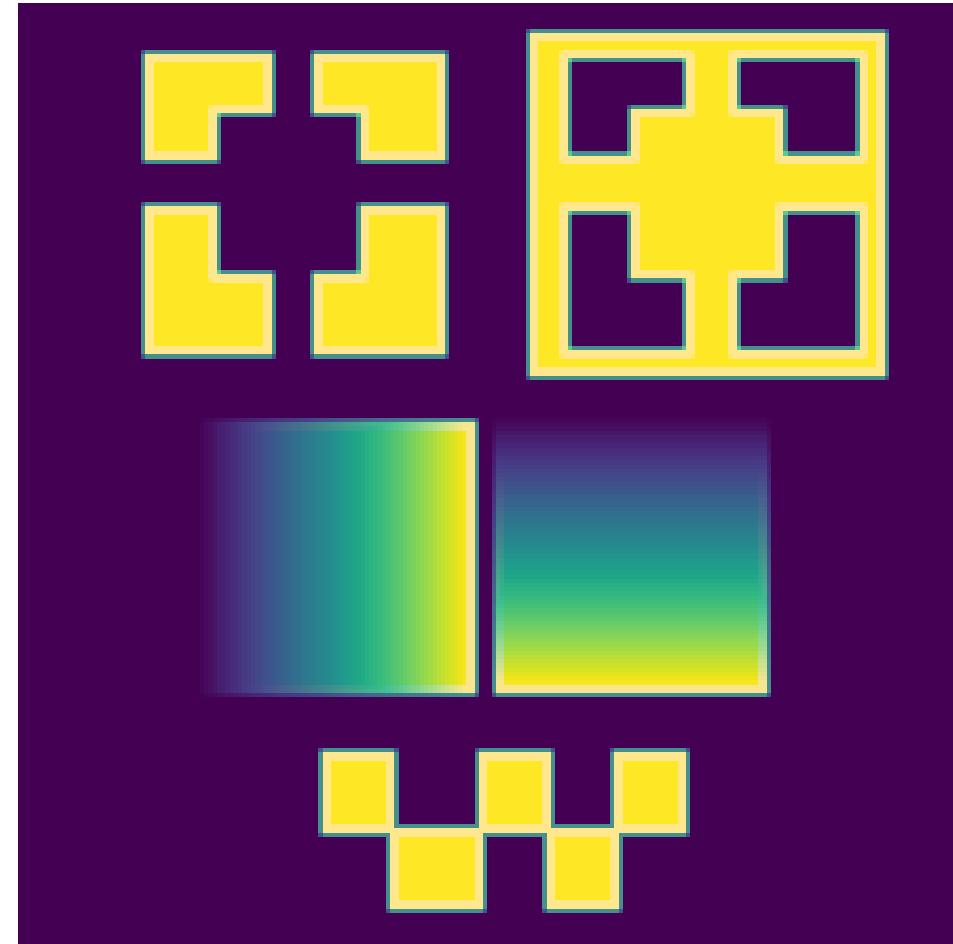


Basic operations

Convolution

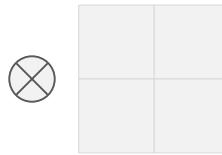
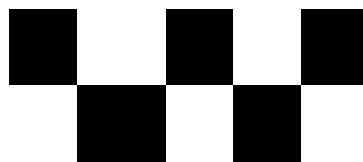
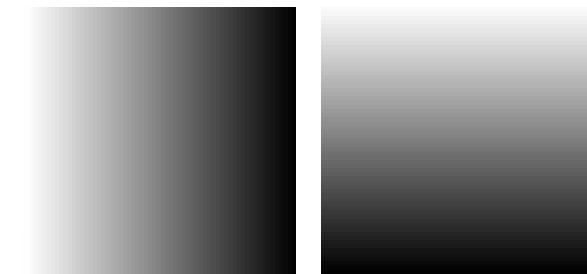
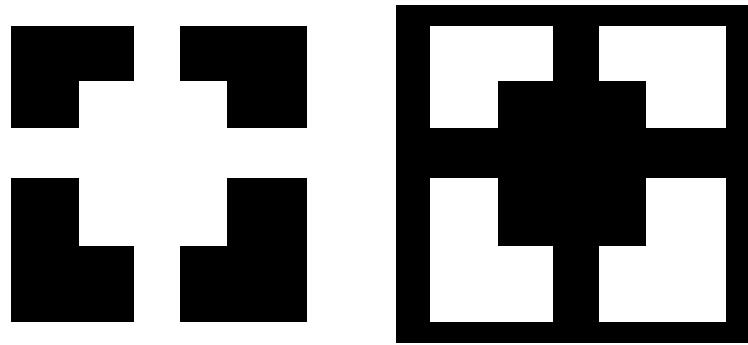


$$\otimes \begin{array}{|c|c|}\hline & \otimes \\ \hline \otimes & \end{array} =$$

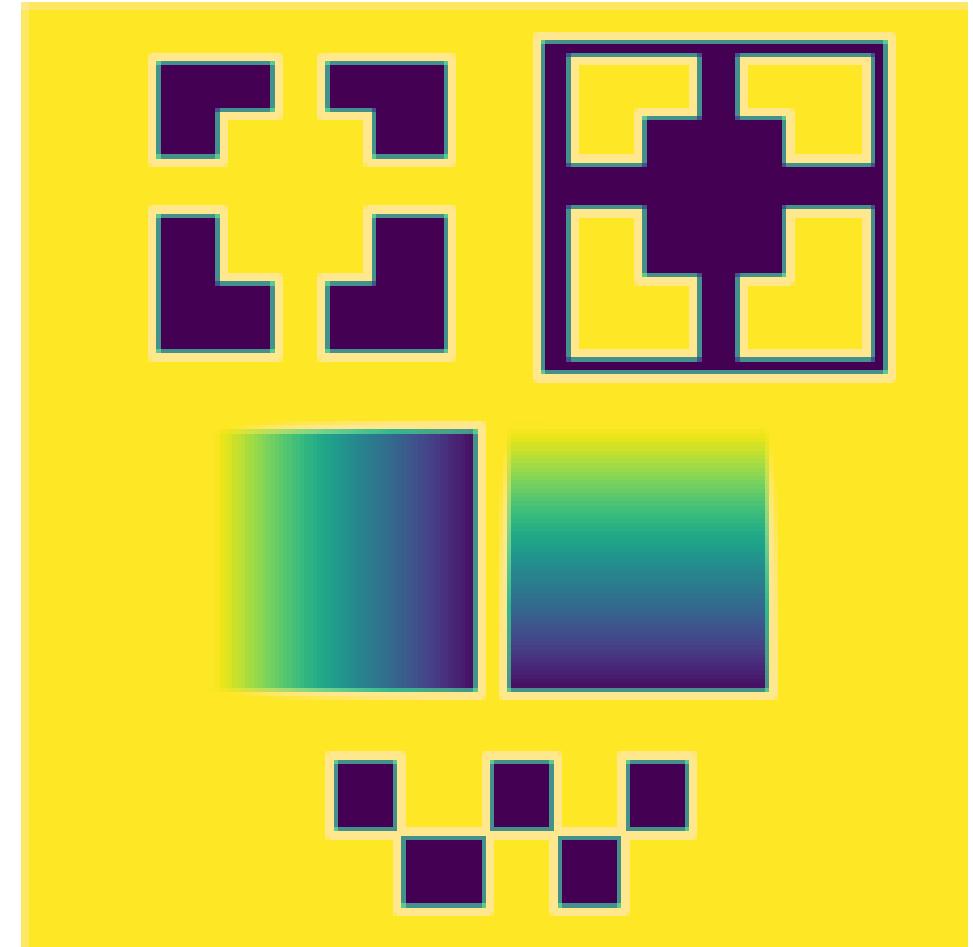


Basic operations

Convolution

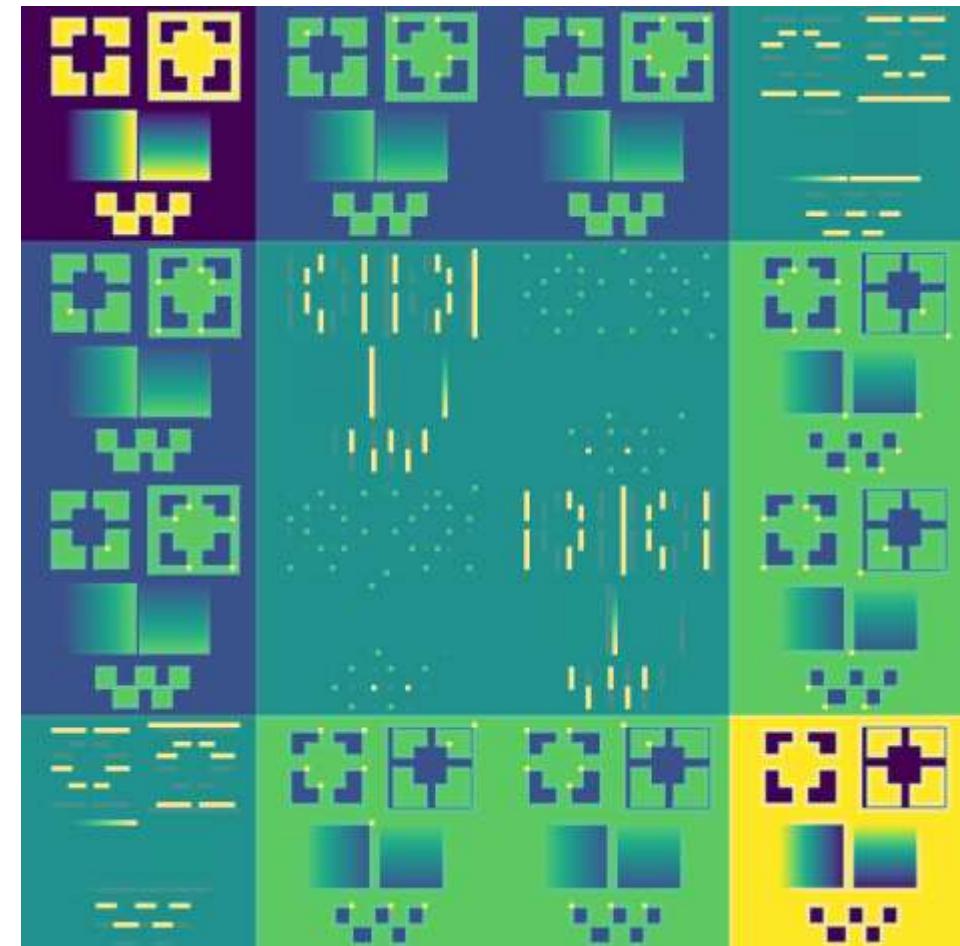
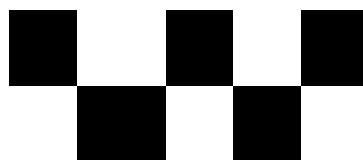
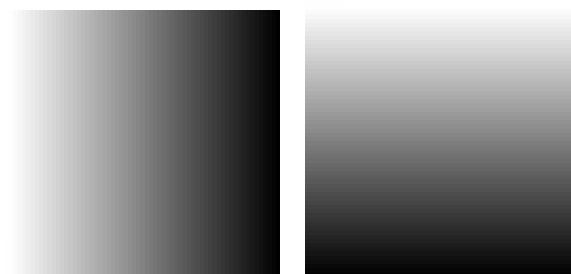


=



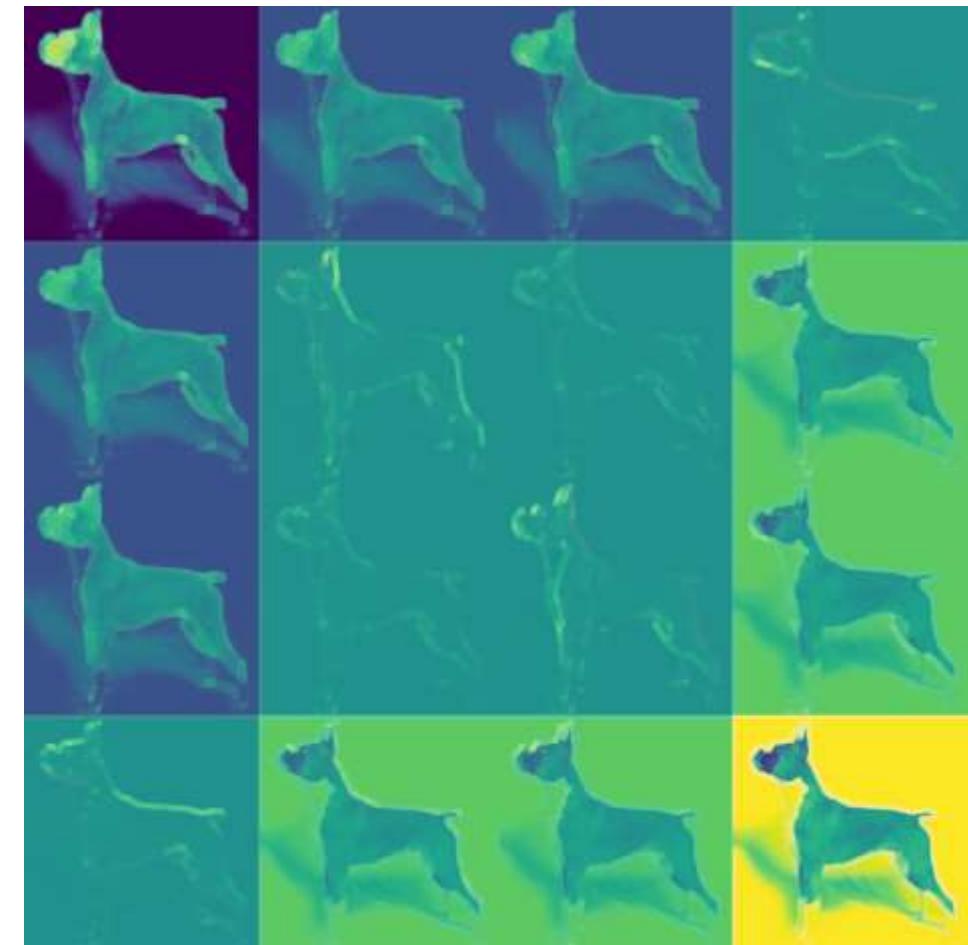
Basic operations

Convolution



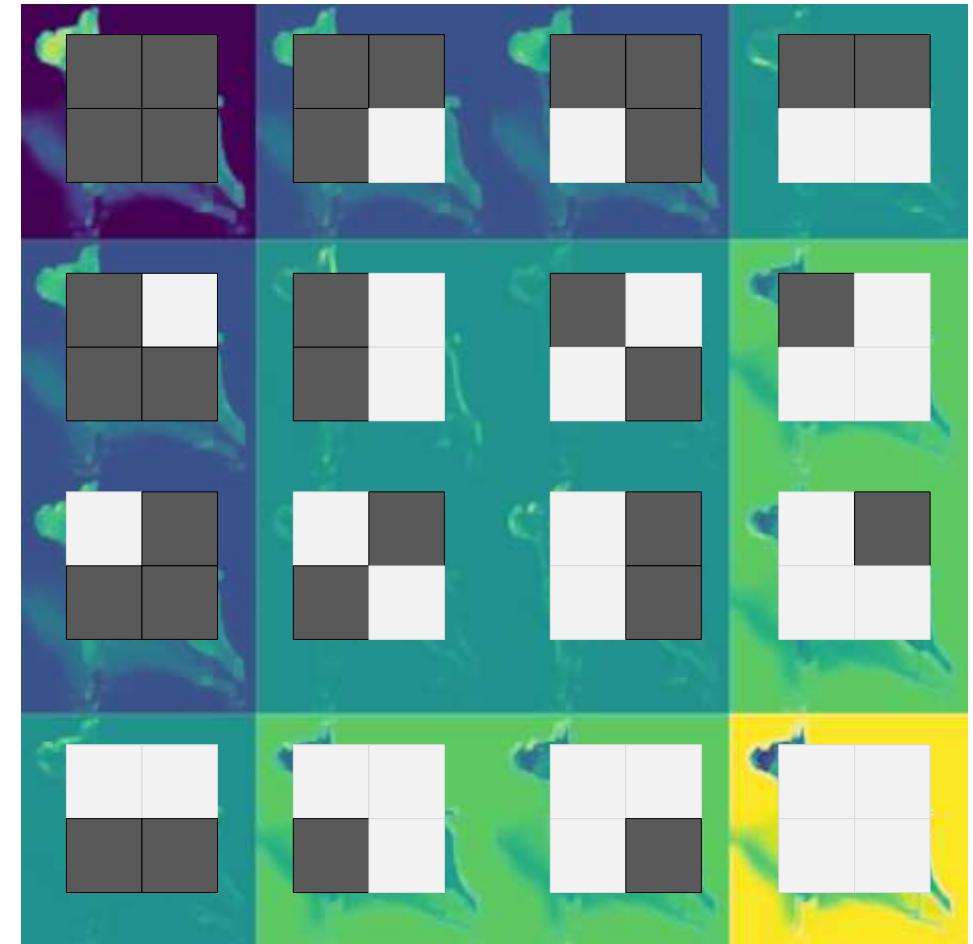
Basic operations

Convolution



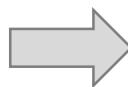
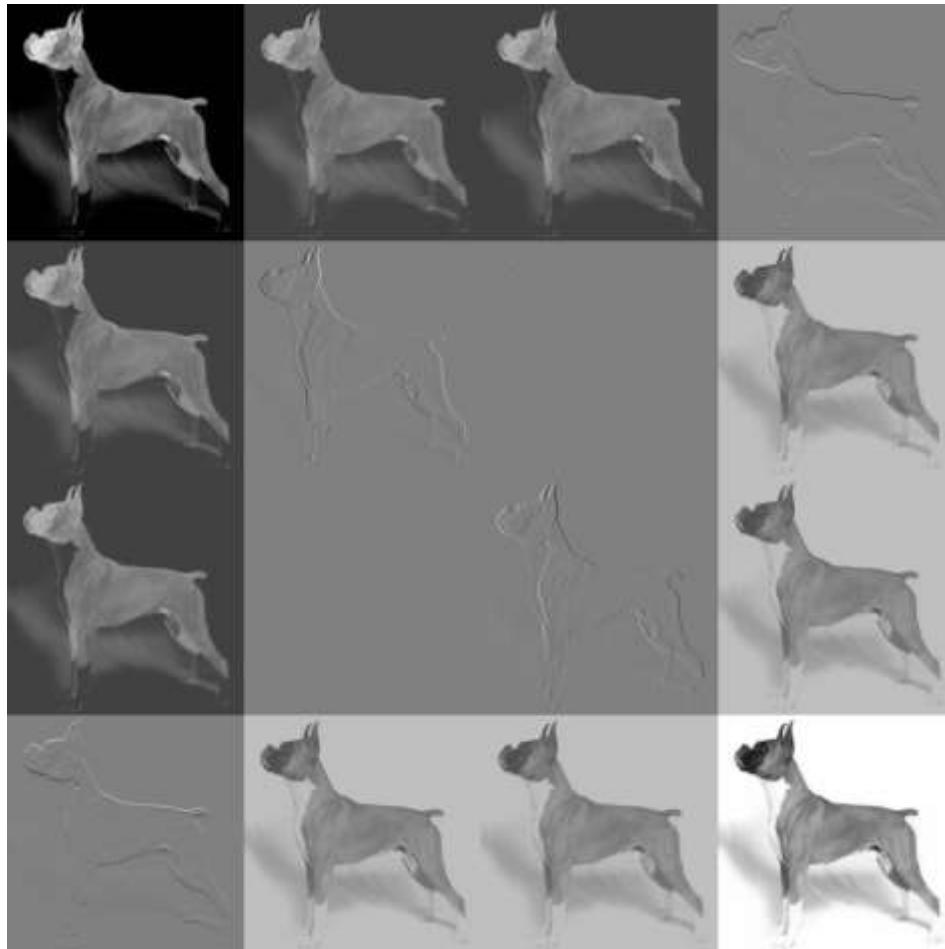
Basic operations

De-Convolution



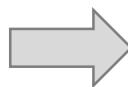
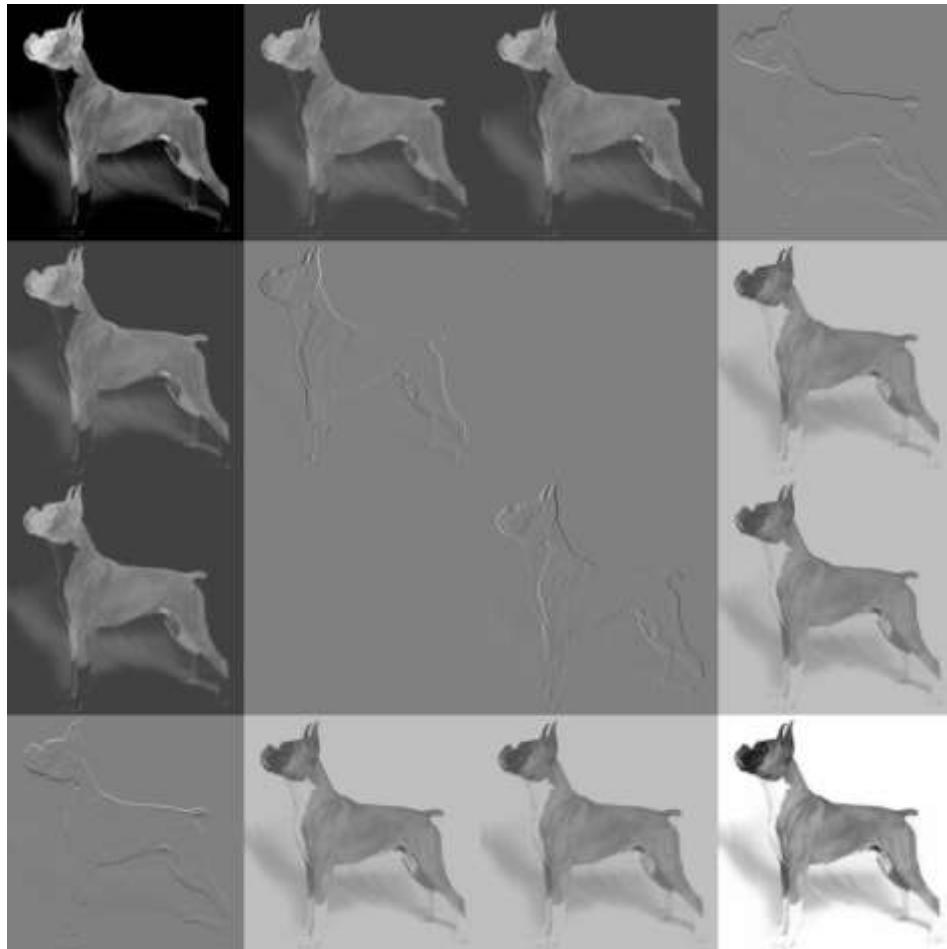
Basic operations

Pool: avg



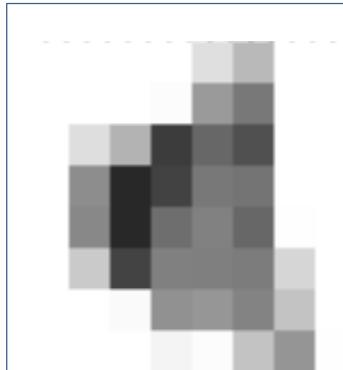
Basic operations

Pool: max



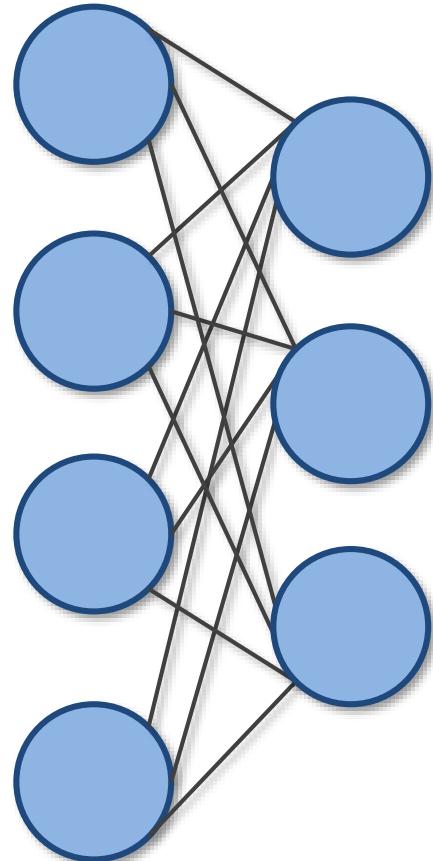
Basic operations

Flatten



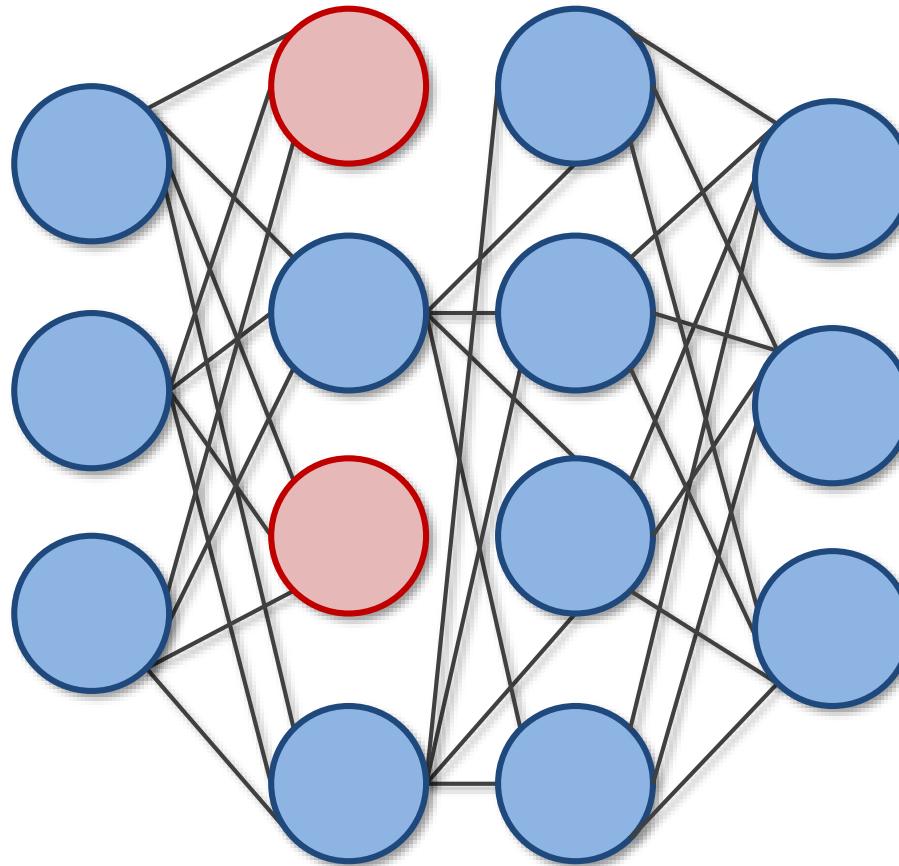
Basic operations

Fully Connected



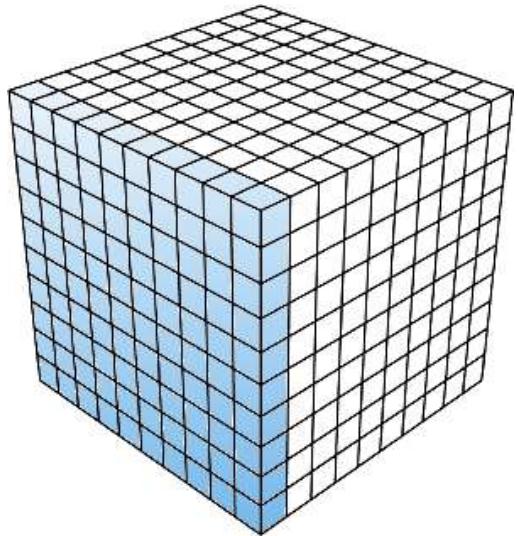
Basic operations

Dropout

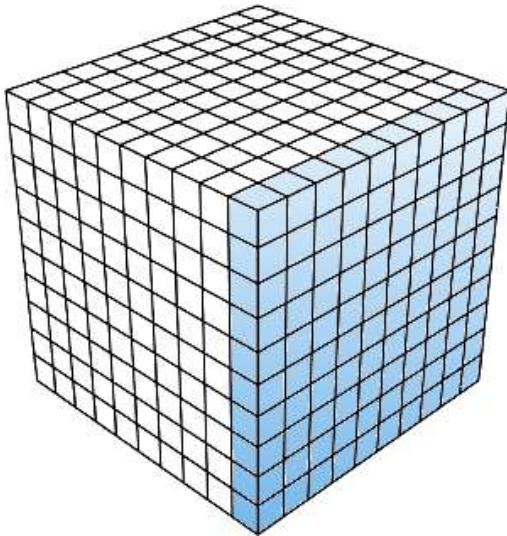


Basic operations

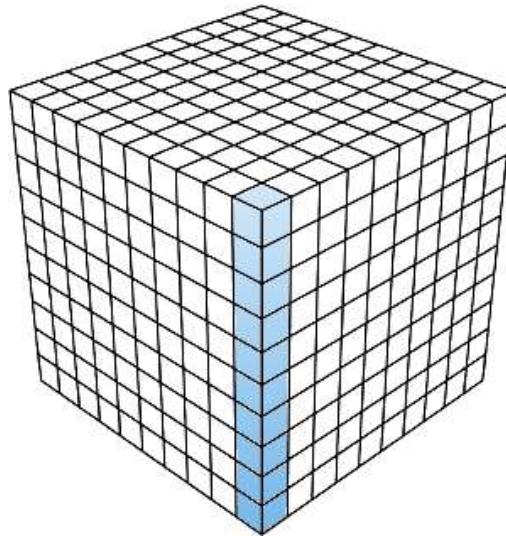
Normalization



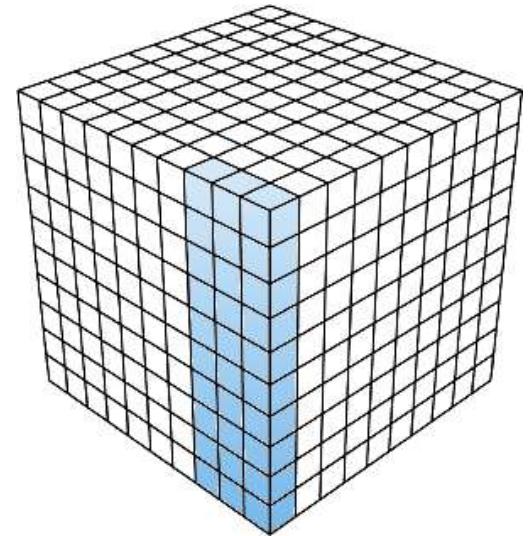
layer



batch



instance

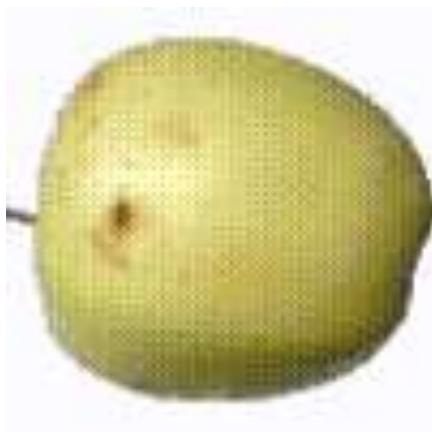


group

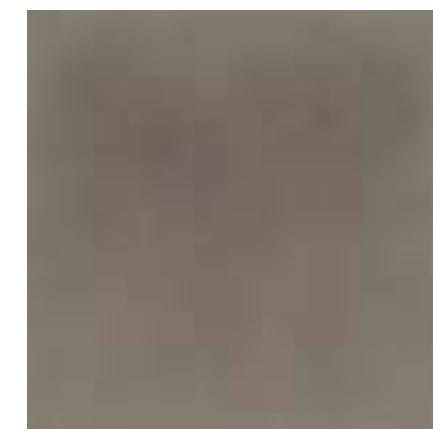
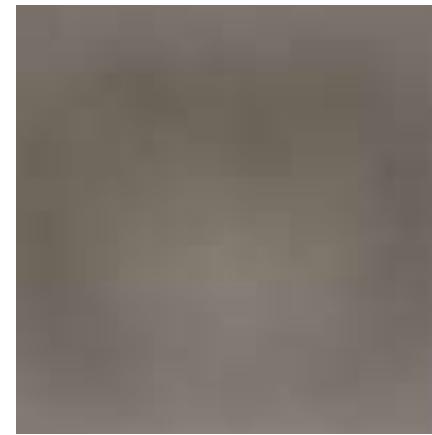
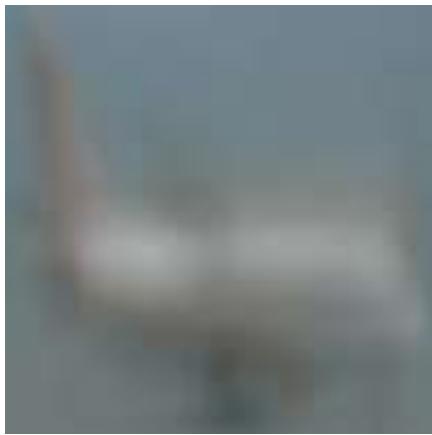
2.3. Feature extraction



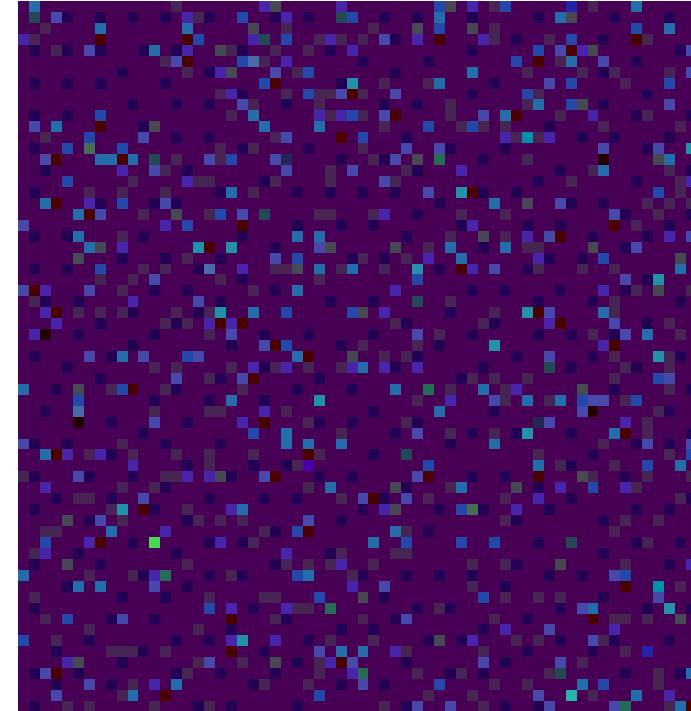
Feature extraction



Feature extraction



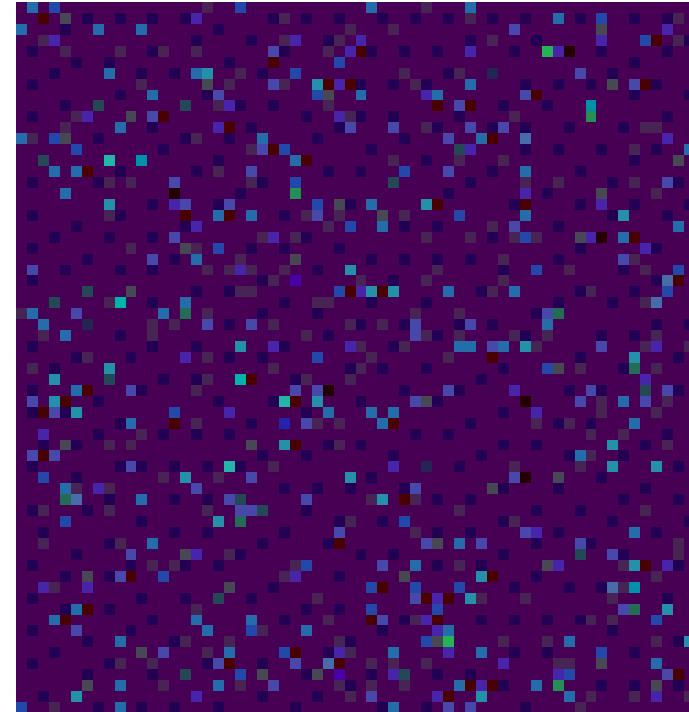
Extract the features



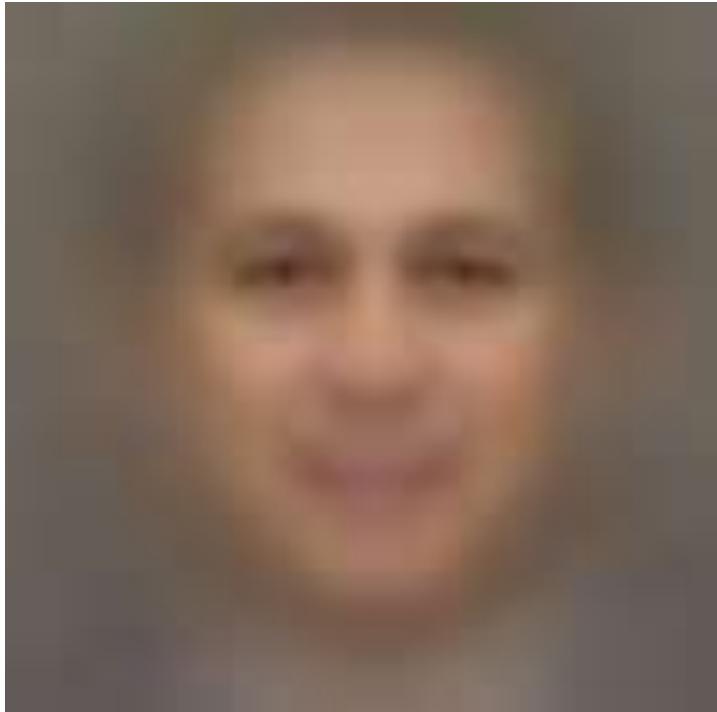
Feature extraction



Feature extraction



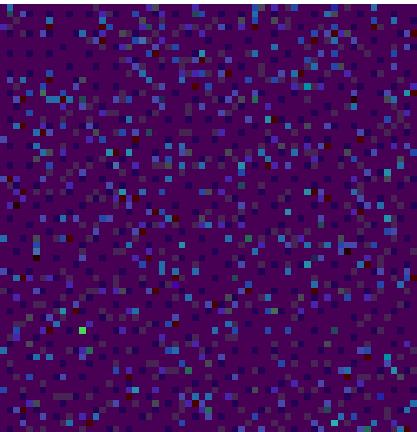
Feature extraction



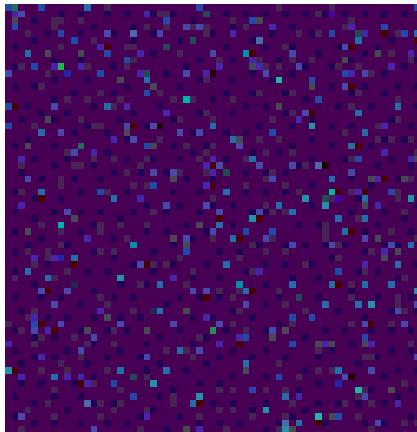
Feature extraction



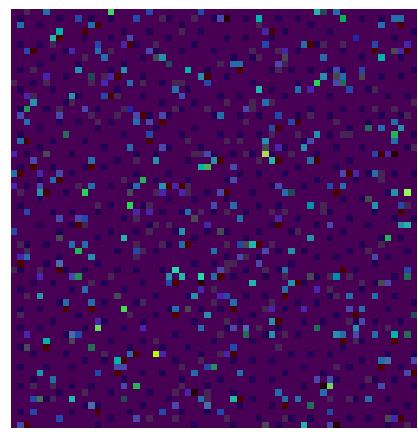
airplane



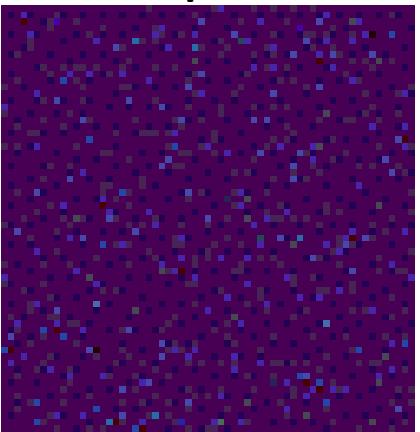
car



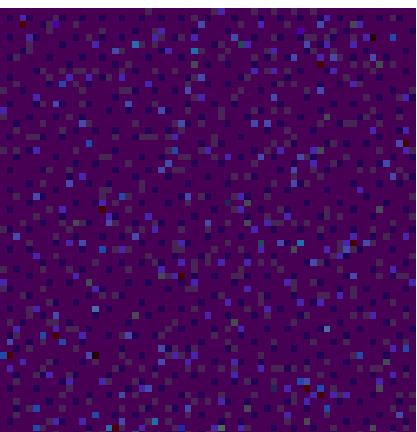
cat



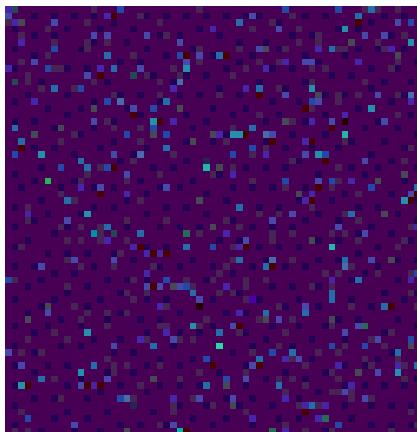
dog



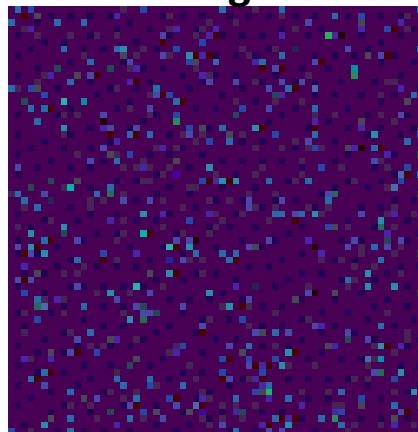
flower



fruit



motorbike



person

Feature extraction



airplane



car



cat



dog



flower



fruit



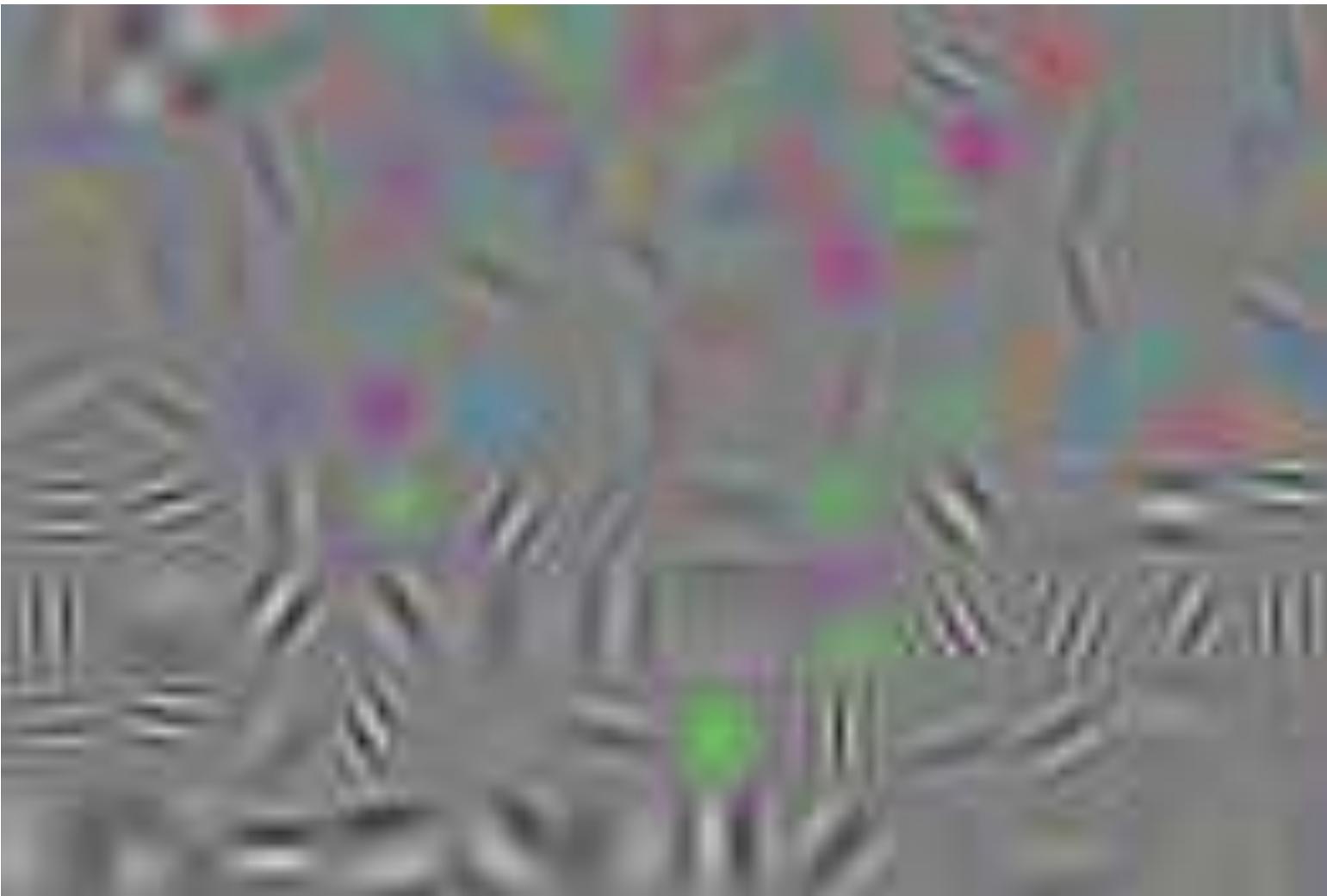
motorbike



person

AlexNet

Feature extraction



AlexNet

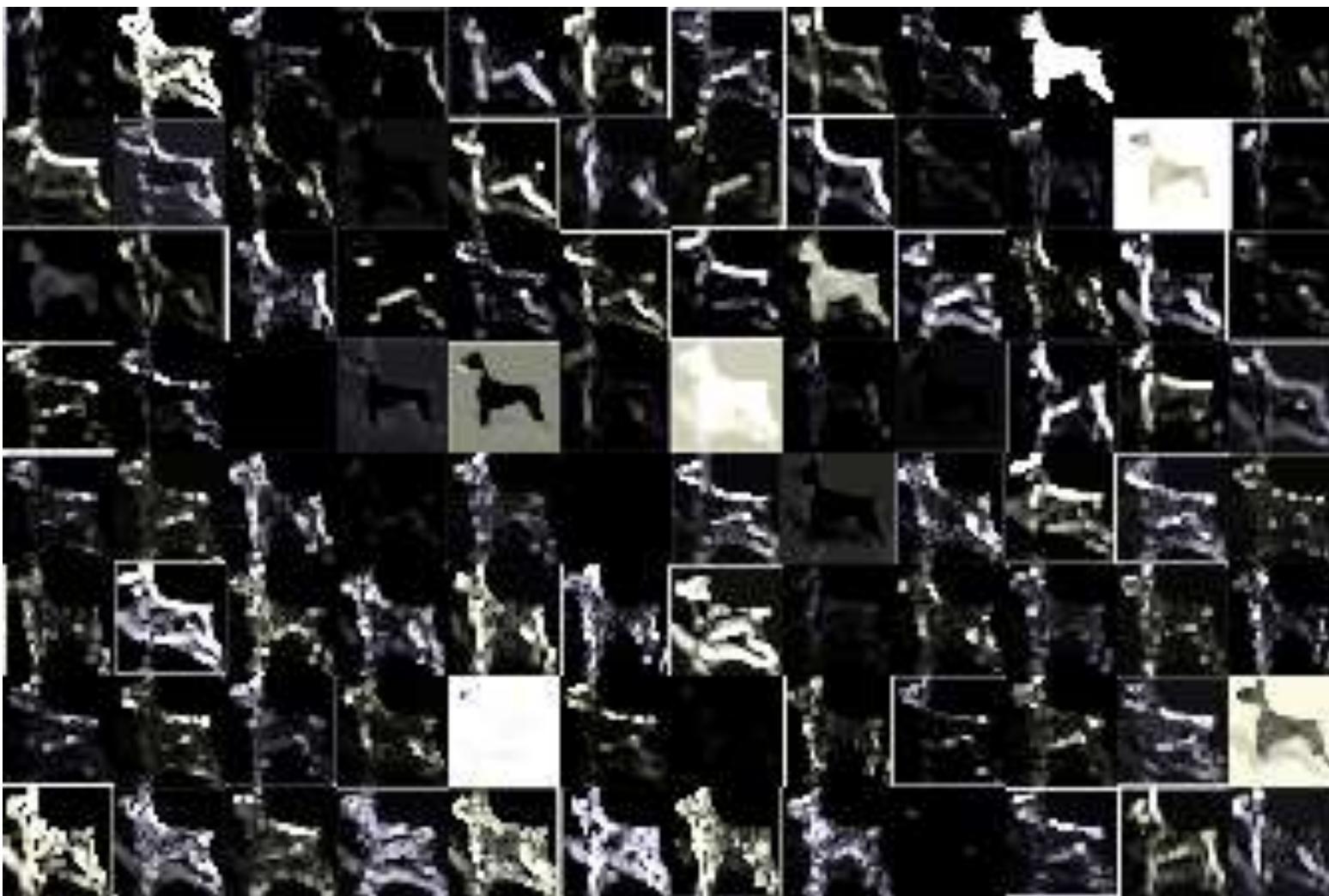


Feature extraction



How CNN constructs the features

AlexNet



AlexNet

Feature extraction



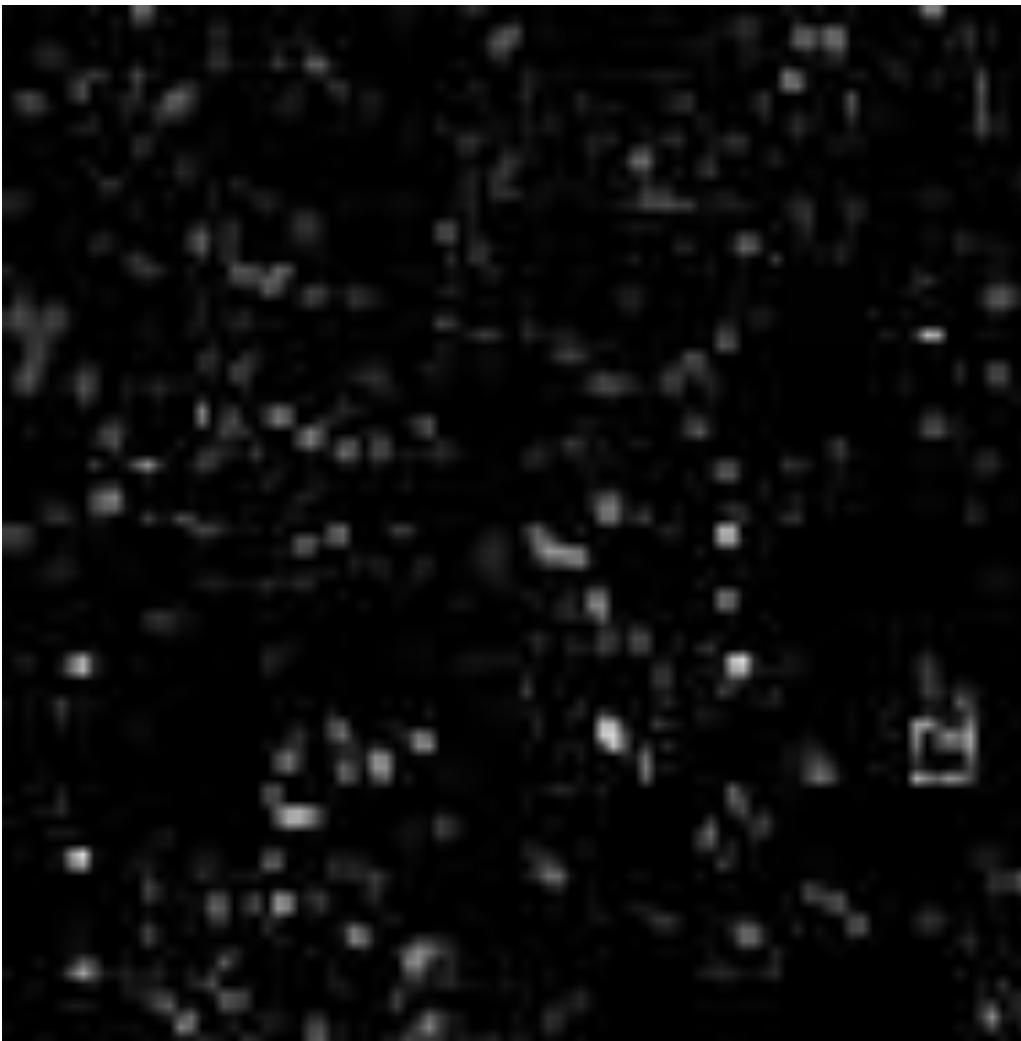
AlexNet

Feature extraction



AlexNet

Feature extraction



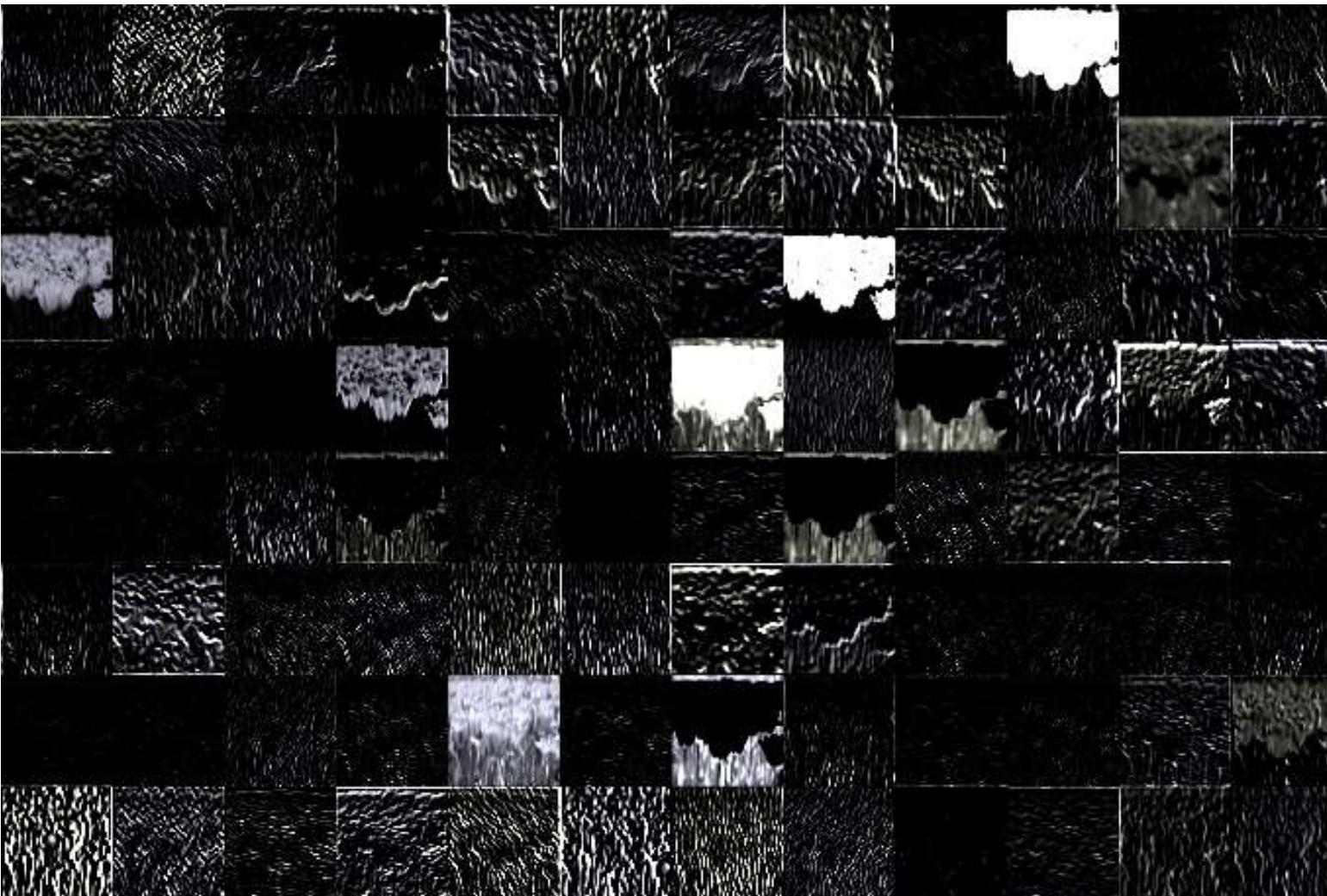
AlexNet

Feature extraction



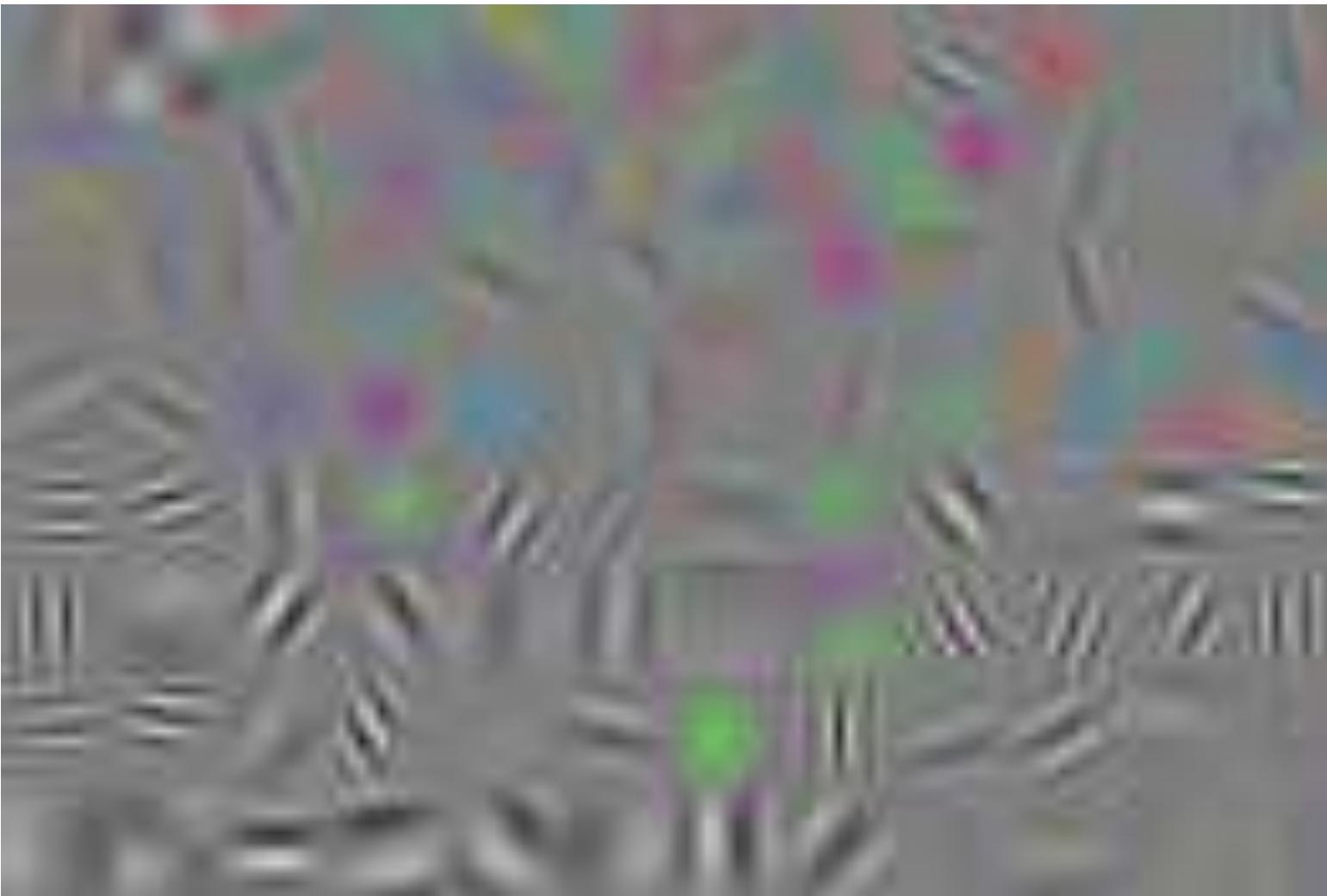
AlexNet

Feature extraction



AlexNet

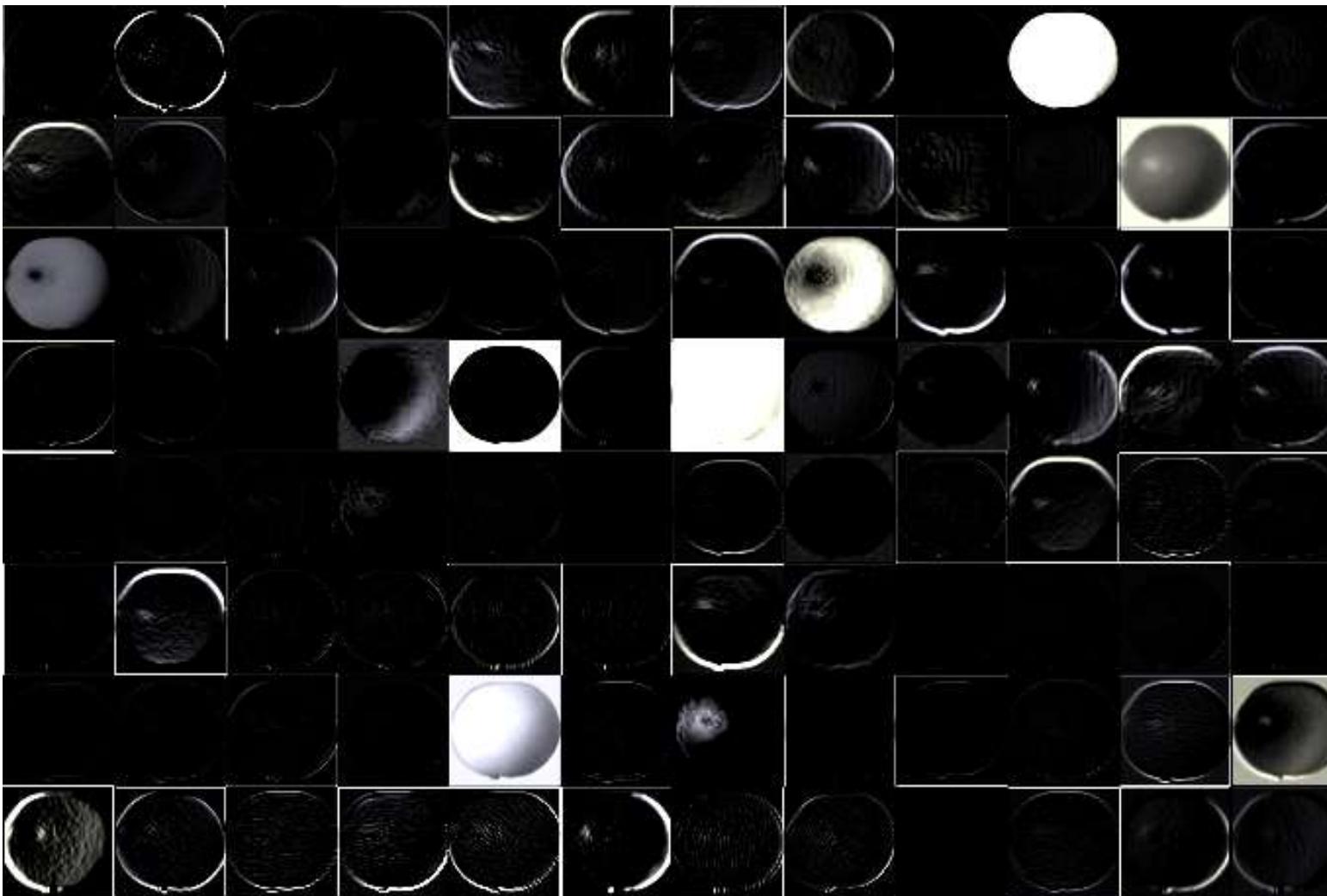
Feature extraction



AlexNet



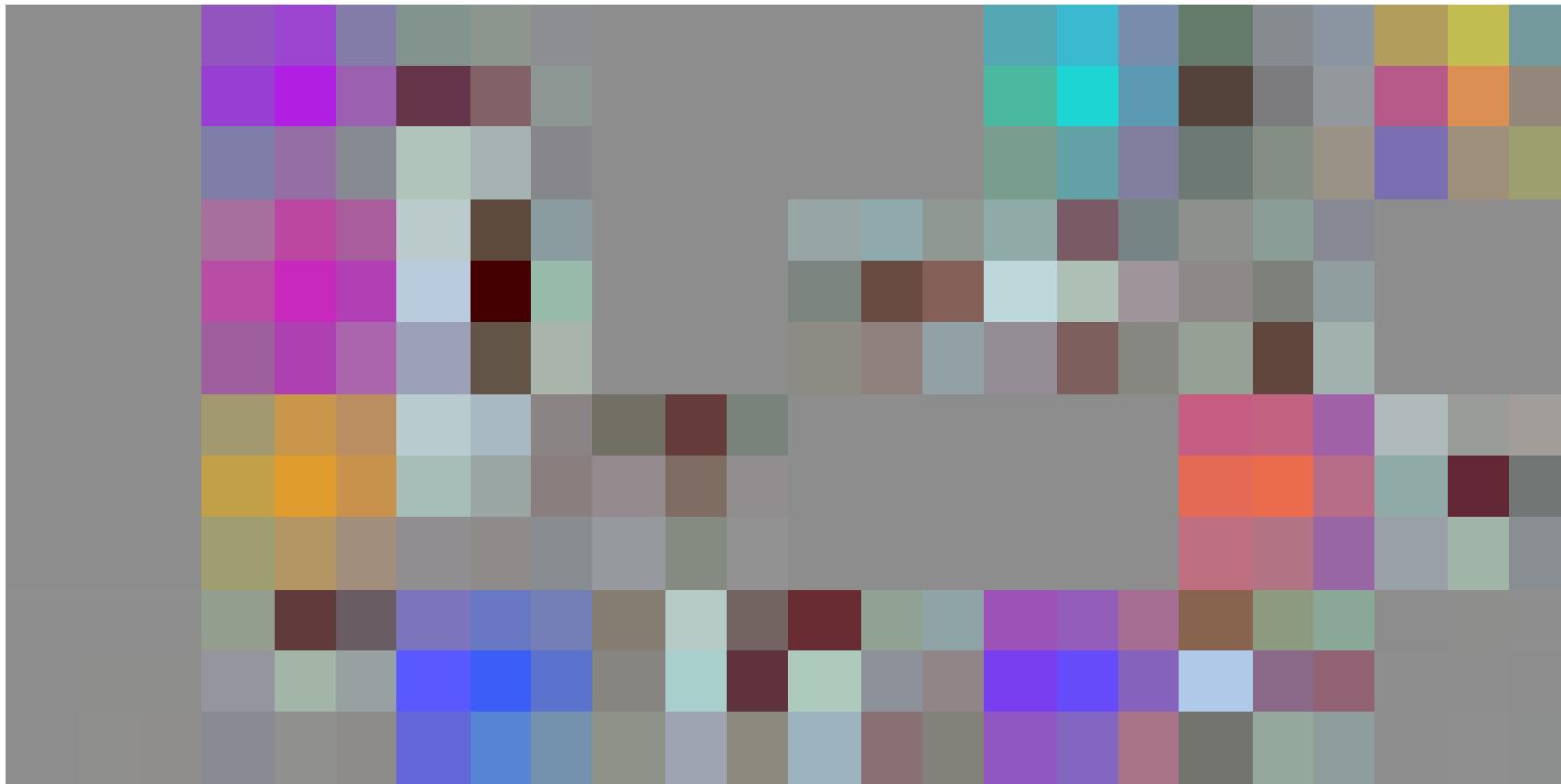
Feature extraction



MobileNet

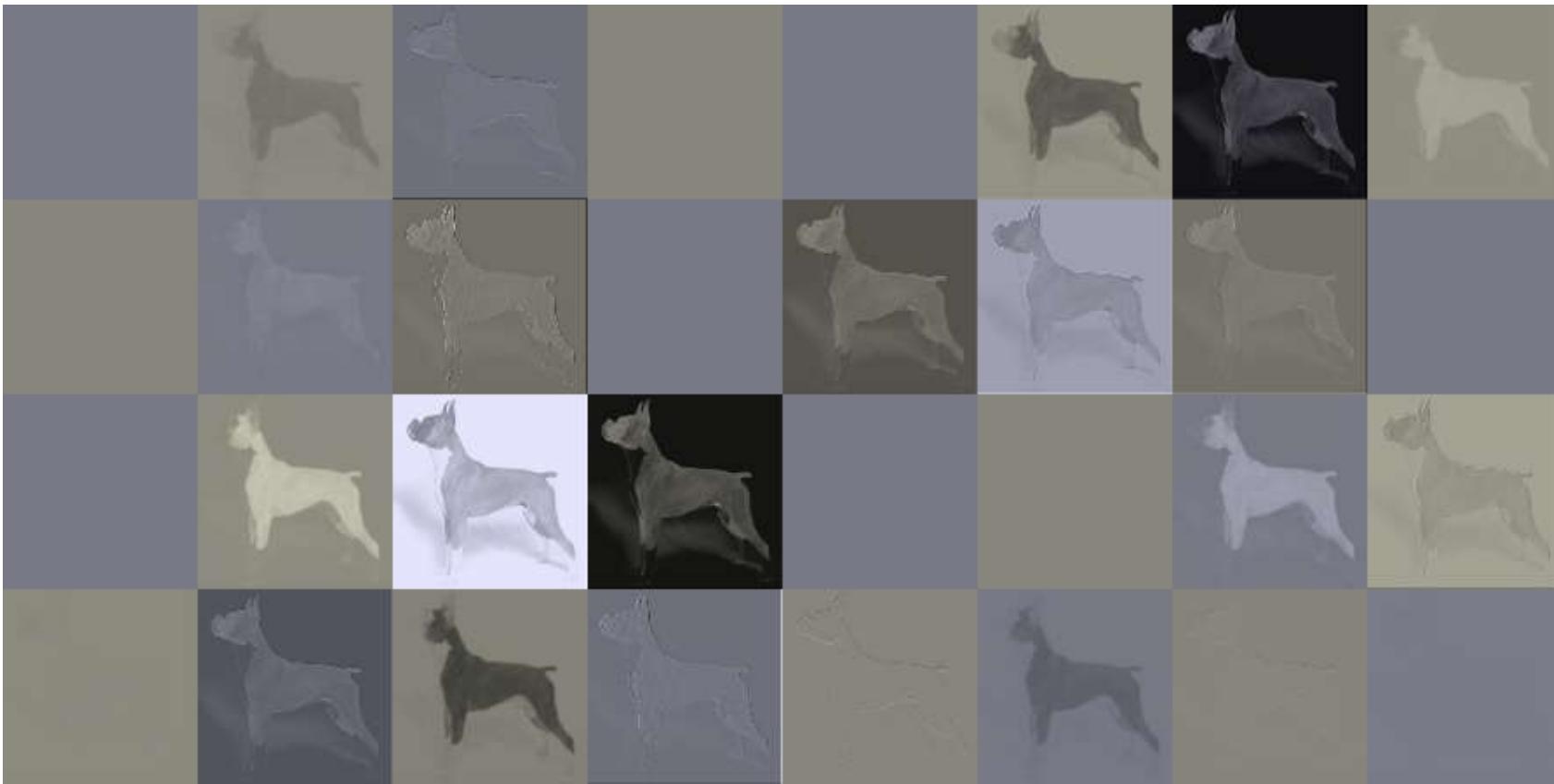


Feature extraction



MobileNet

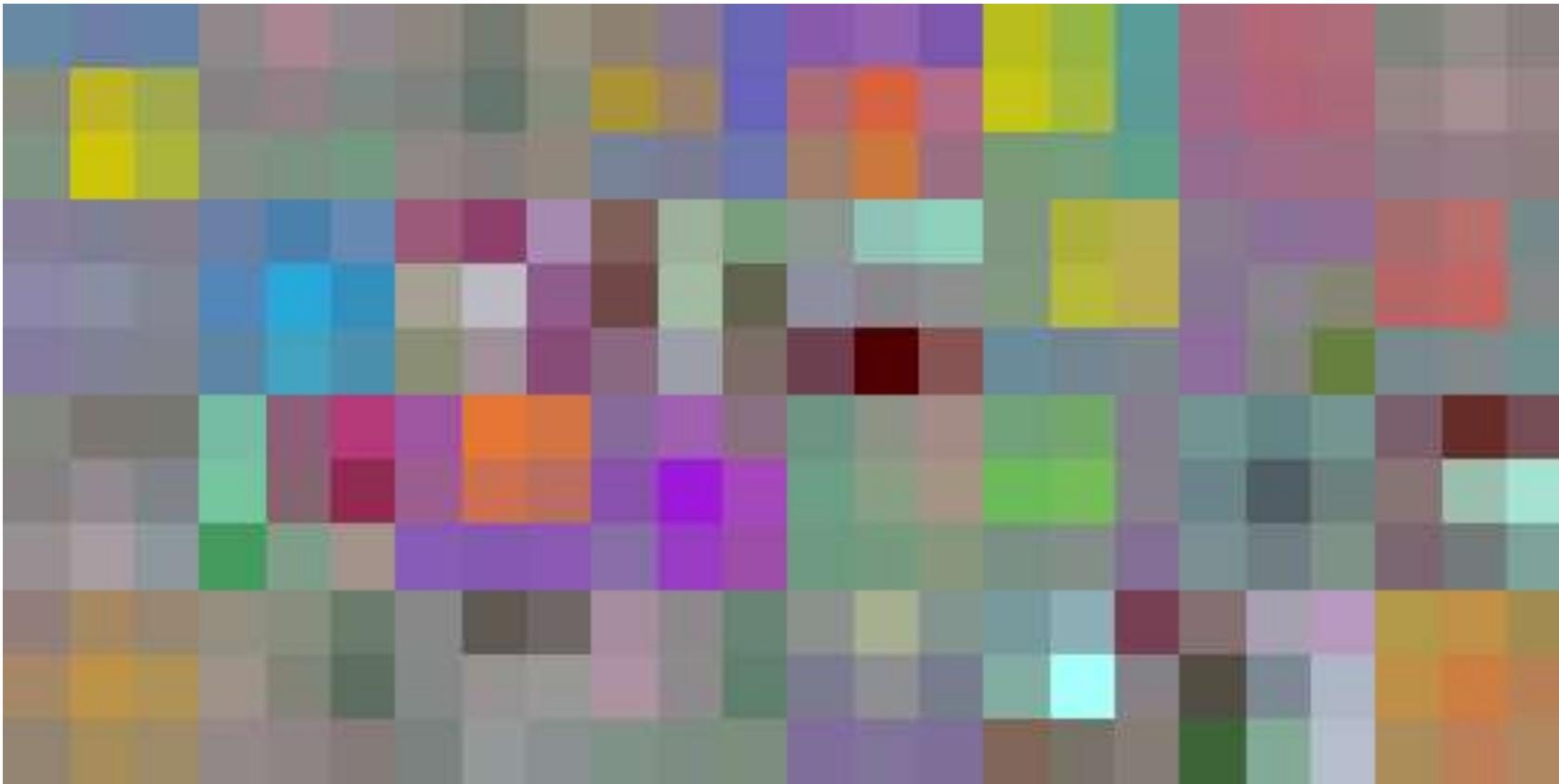
Feature extraction



Xception



Feature extraction



MobileNet

Feature extraction



2.4. Transfer learning

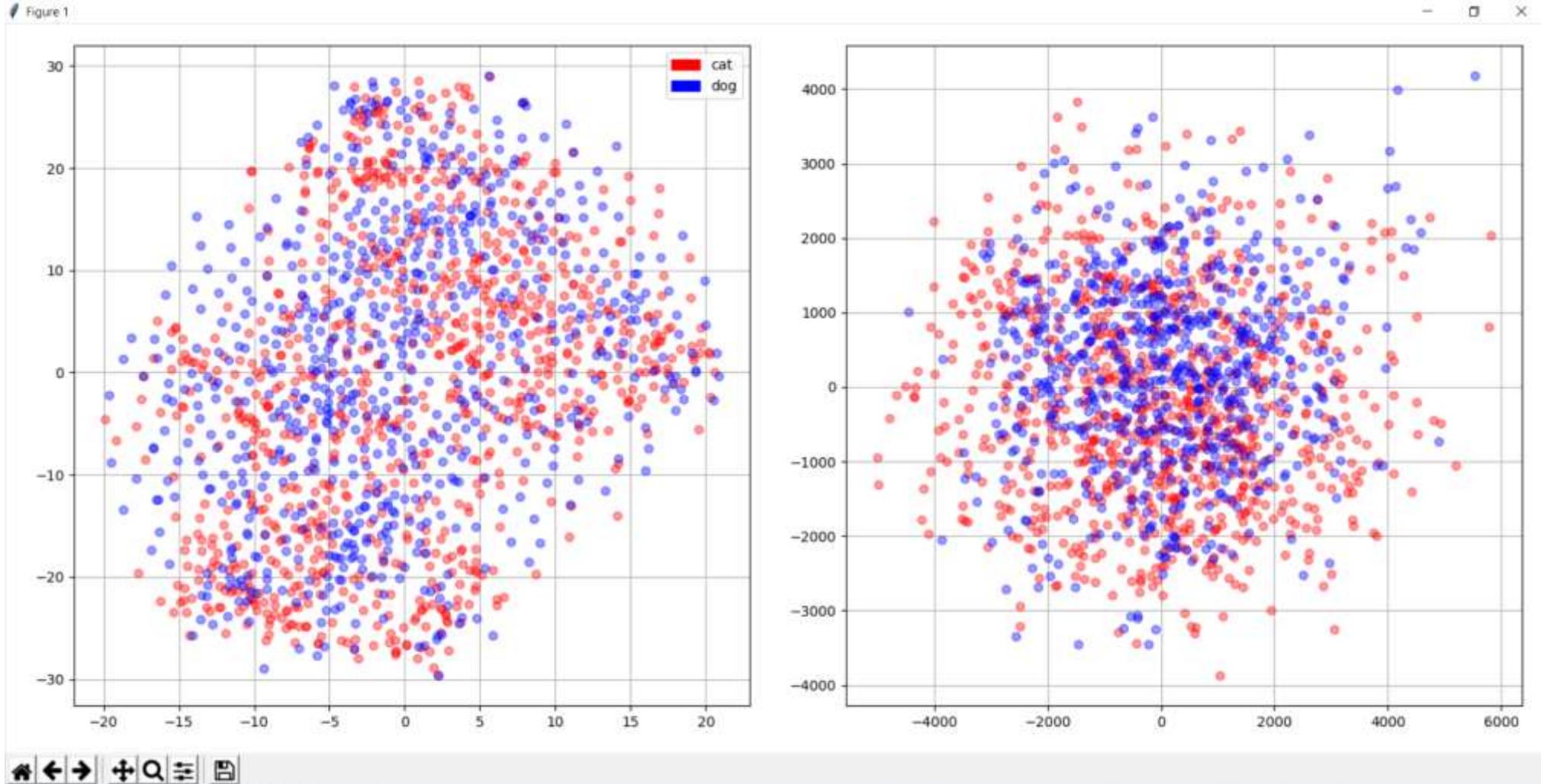


Transfer learning

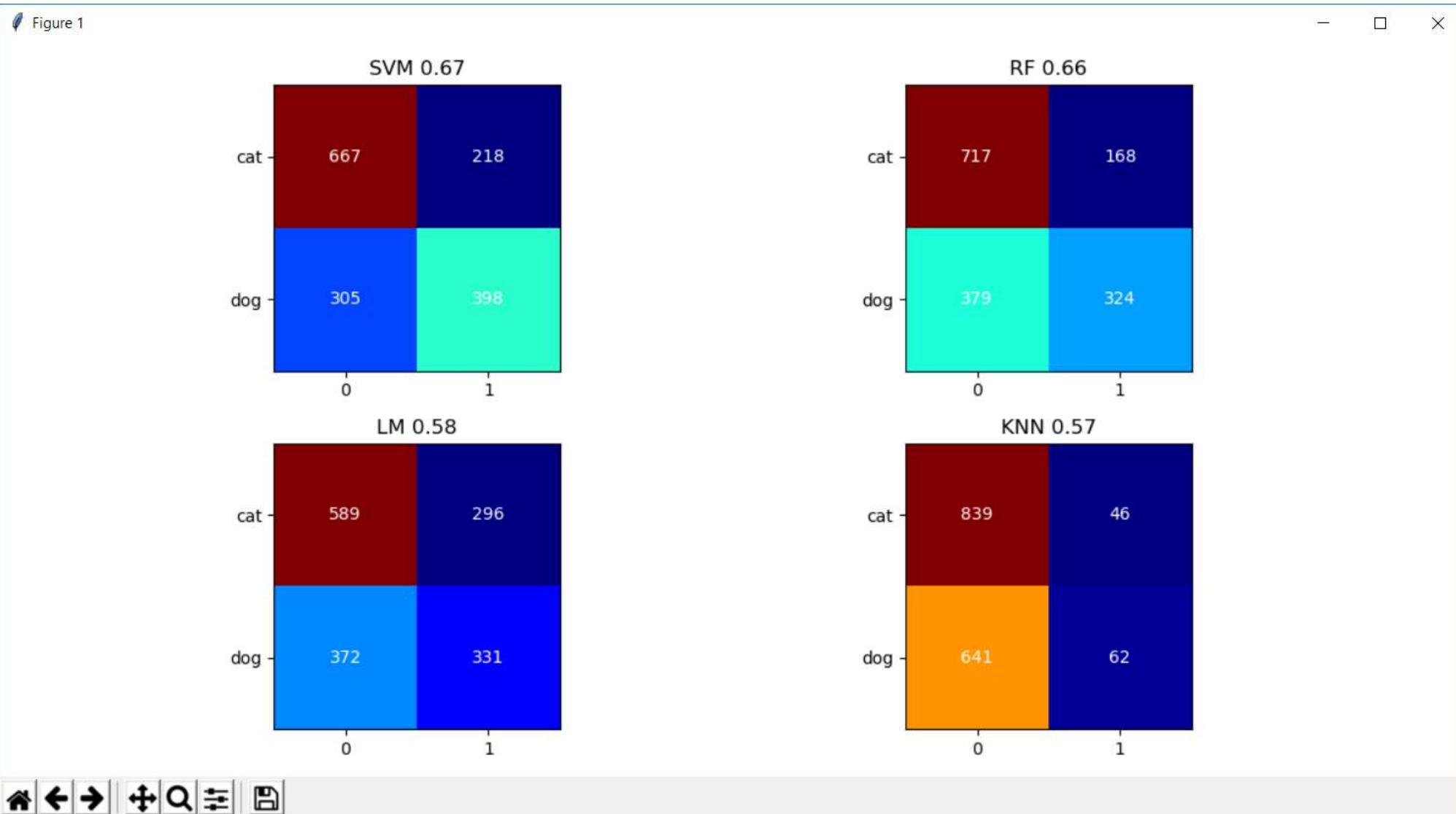
2.5. Fine tuning



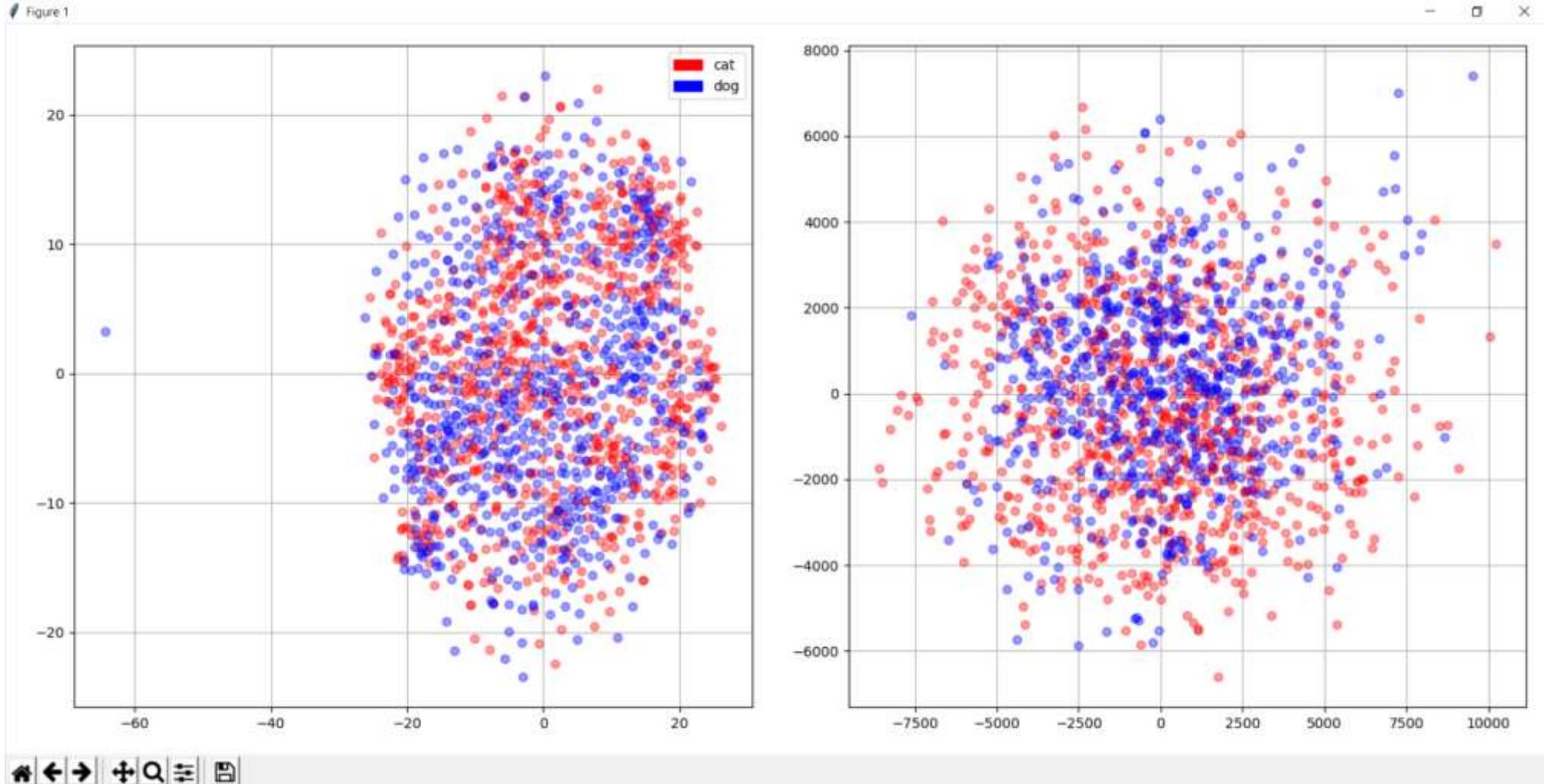
Finetuning



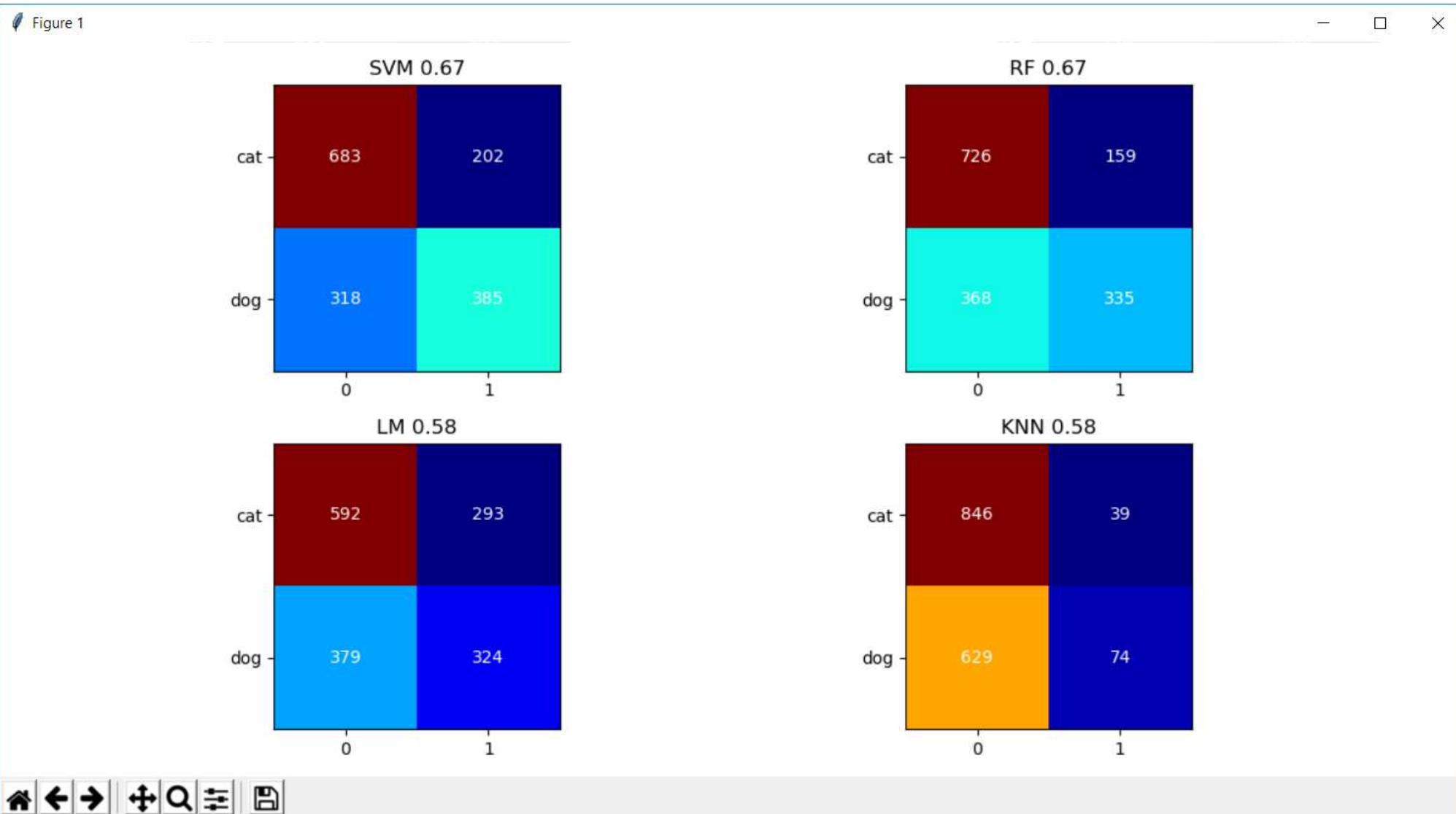
Finetuning



Finetuning



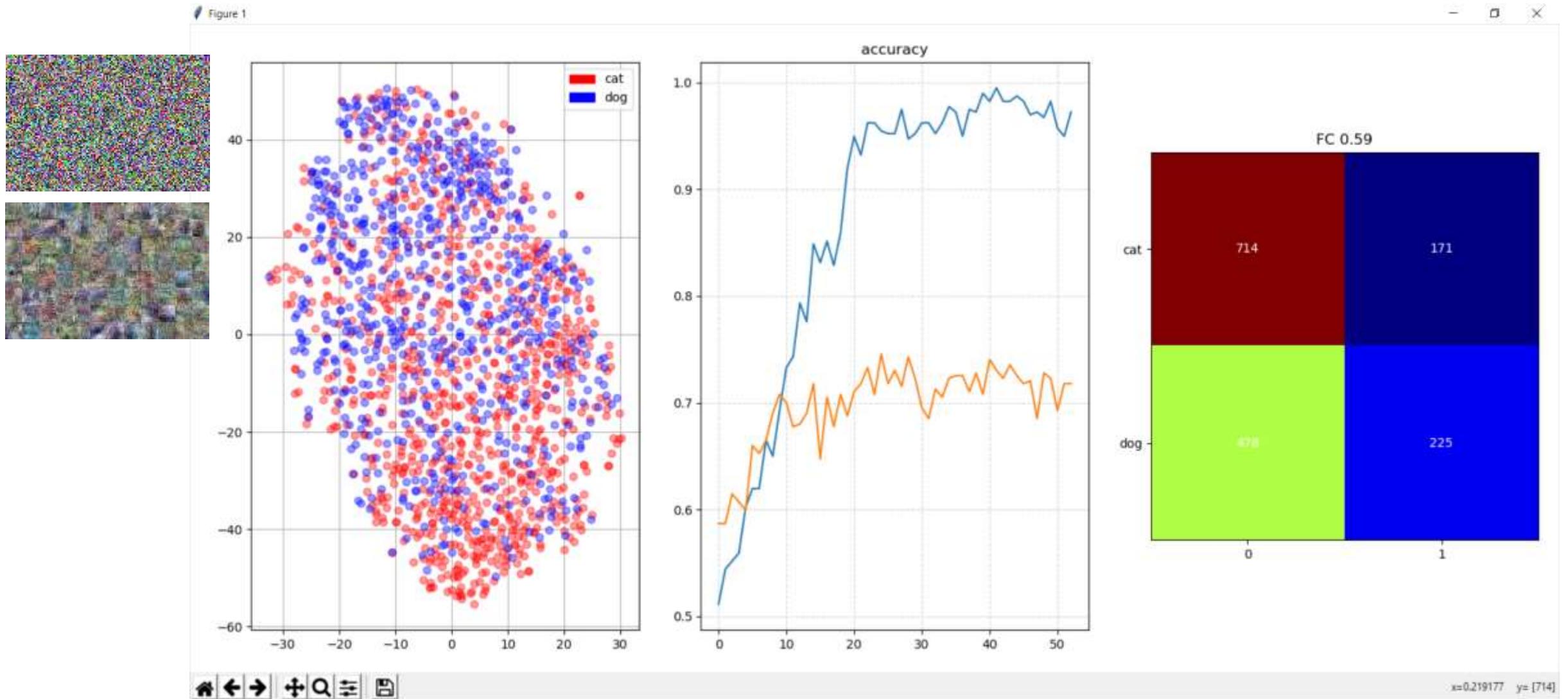
Finetuning



Finetuning



Finetuning



Finetuning

Random before learning

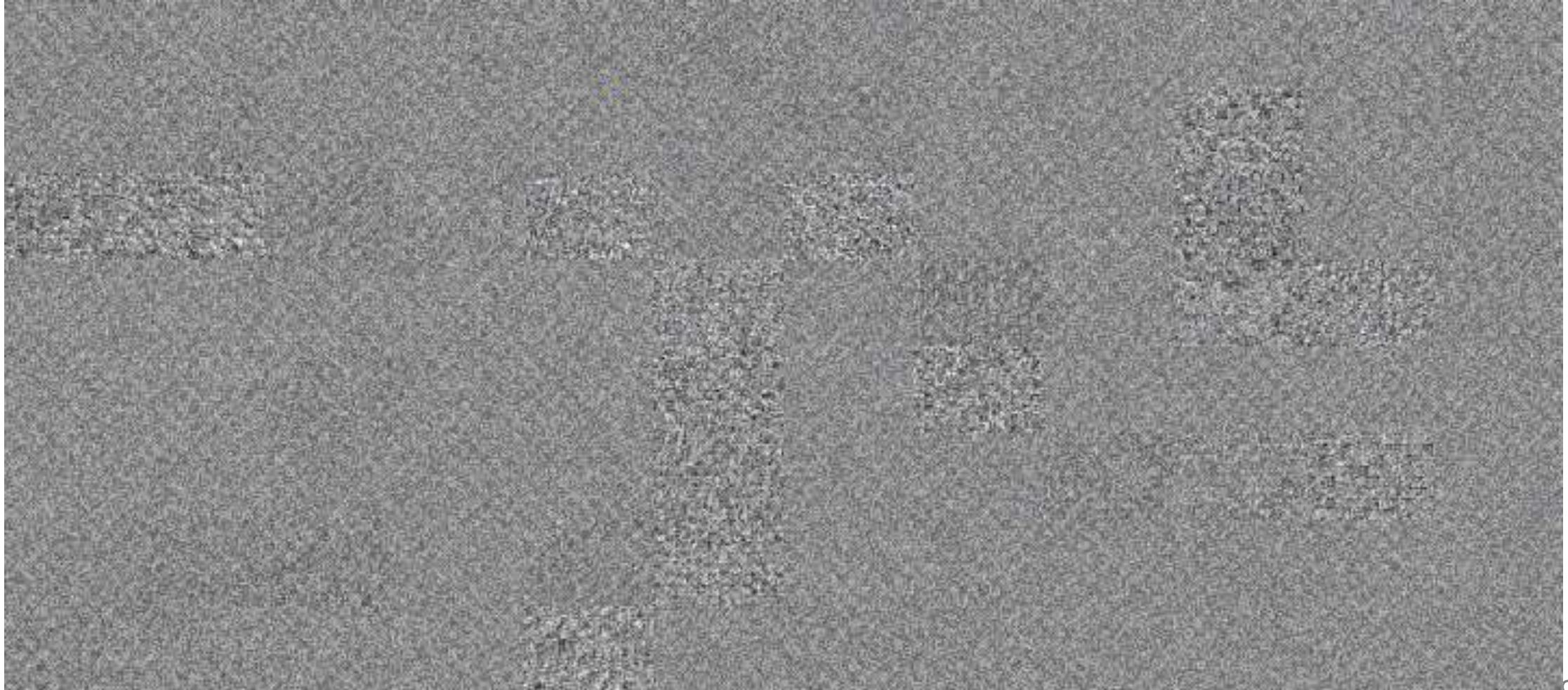


After learning

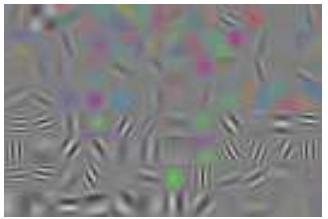


Finetuning

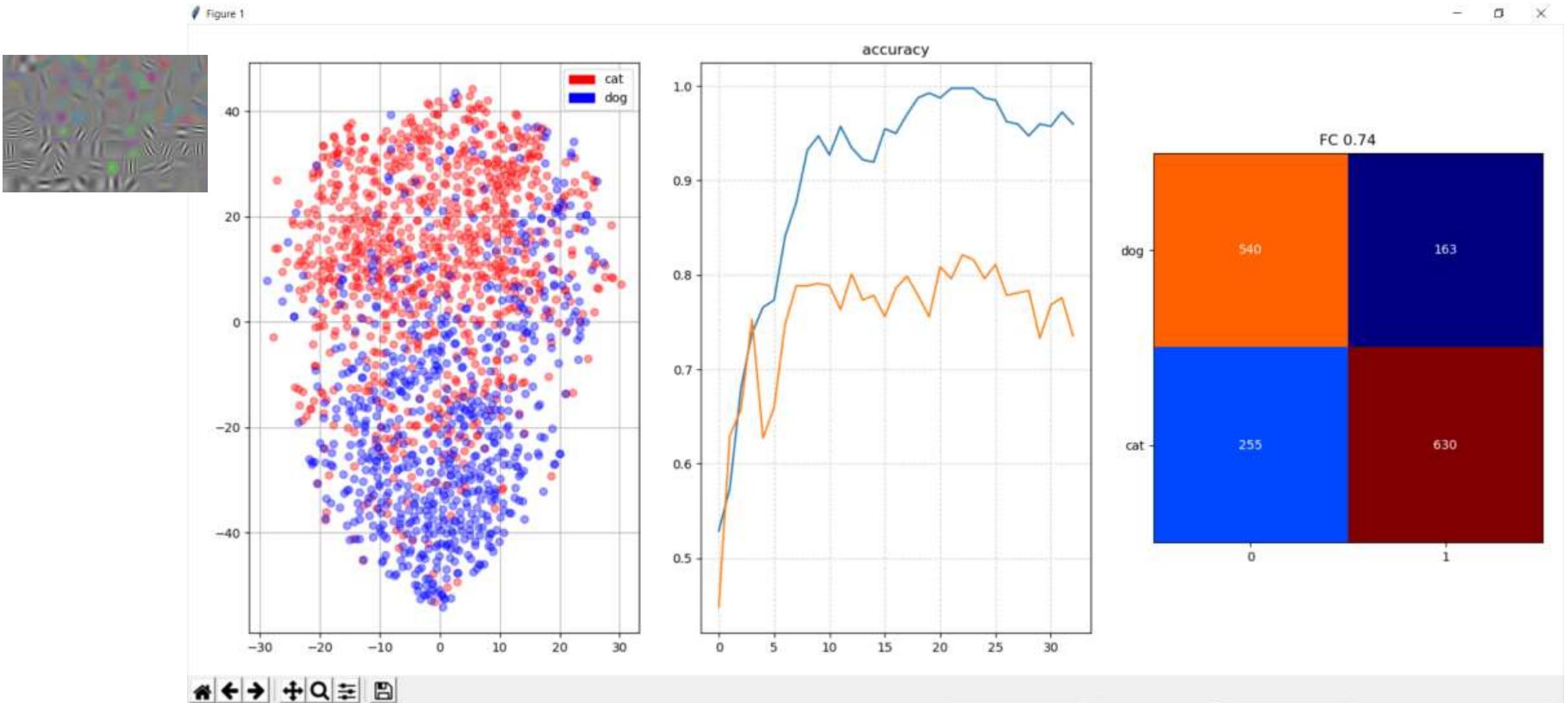
Filters after learning



Finetuning



Finetuning

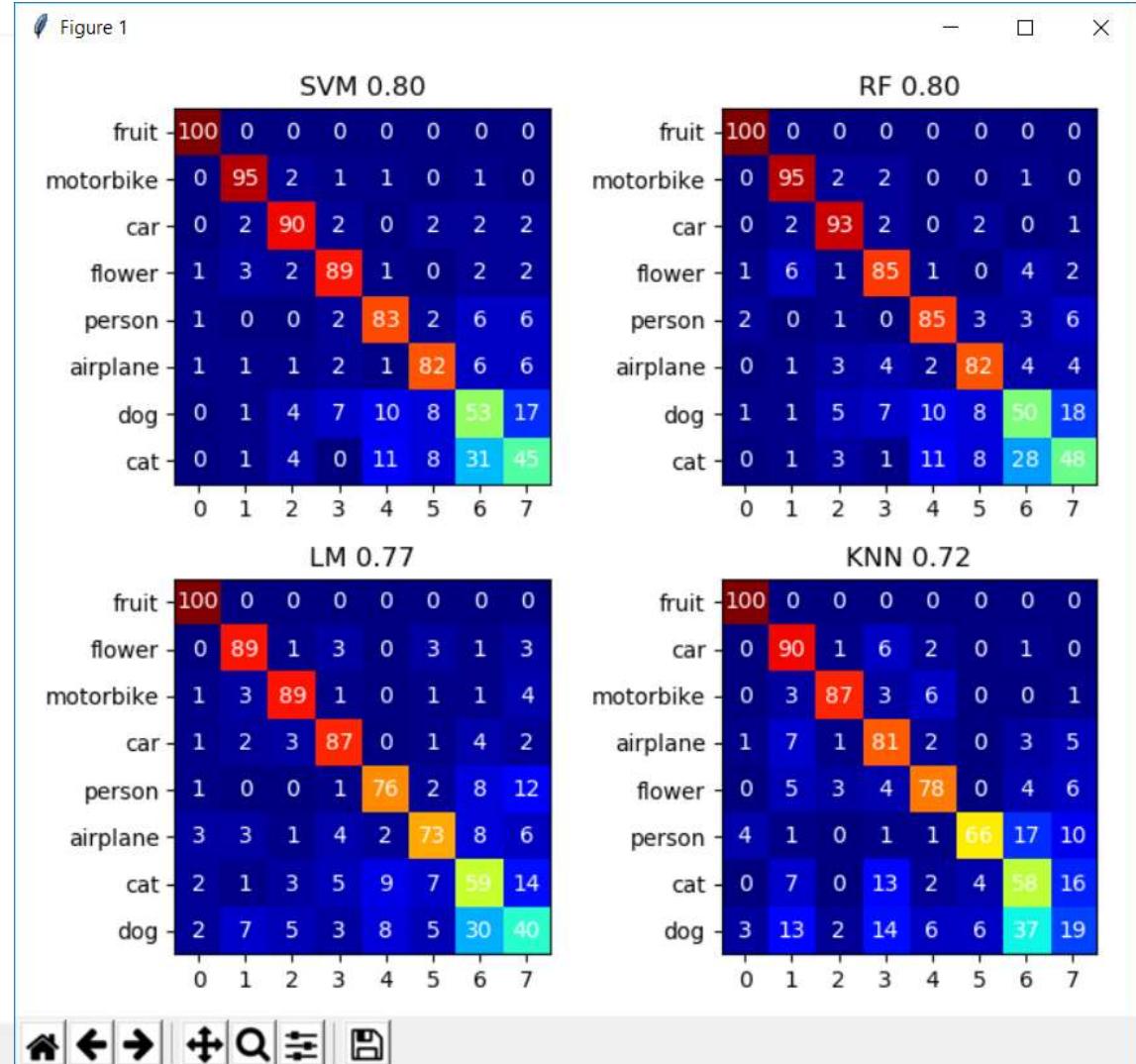
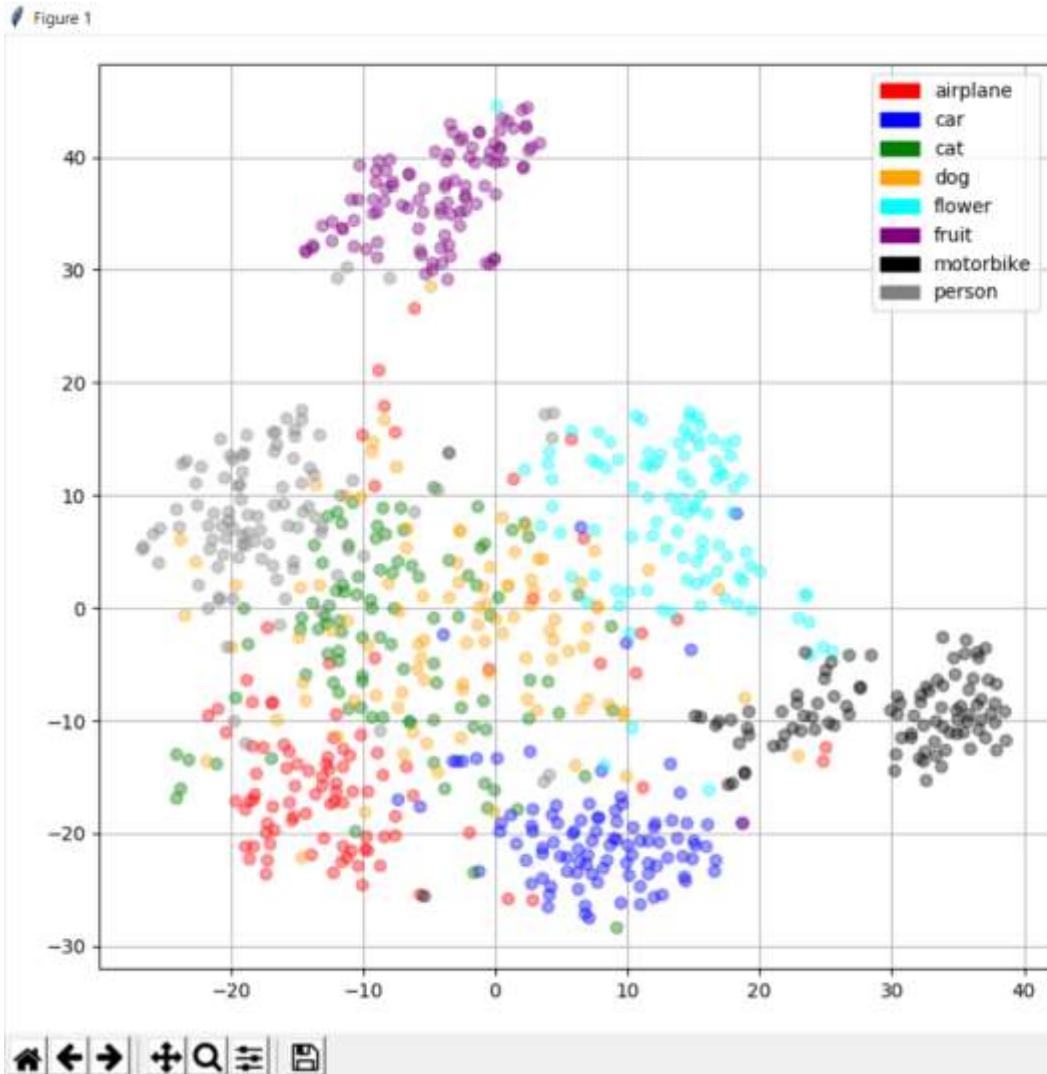


2.6. Benchmarking the CNNs



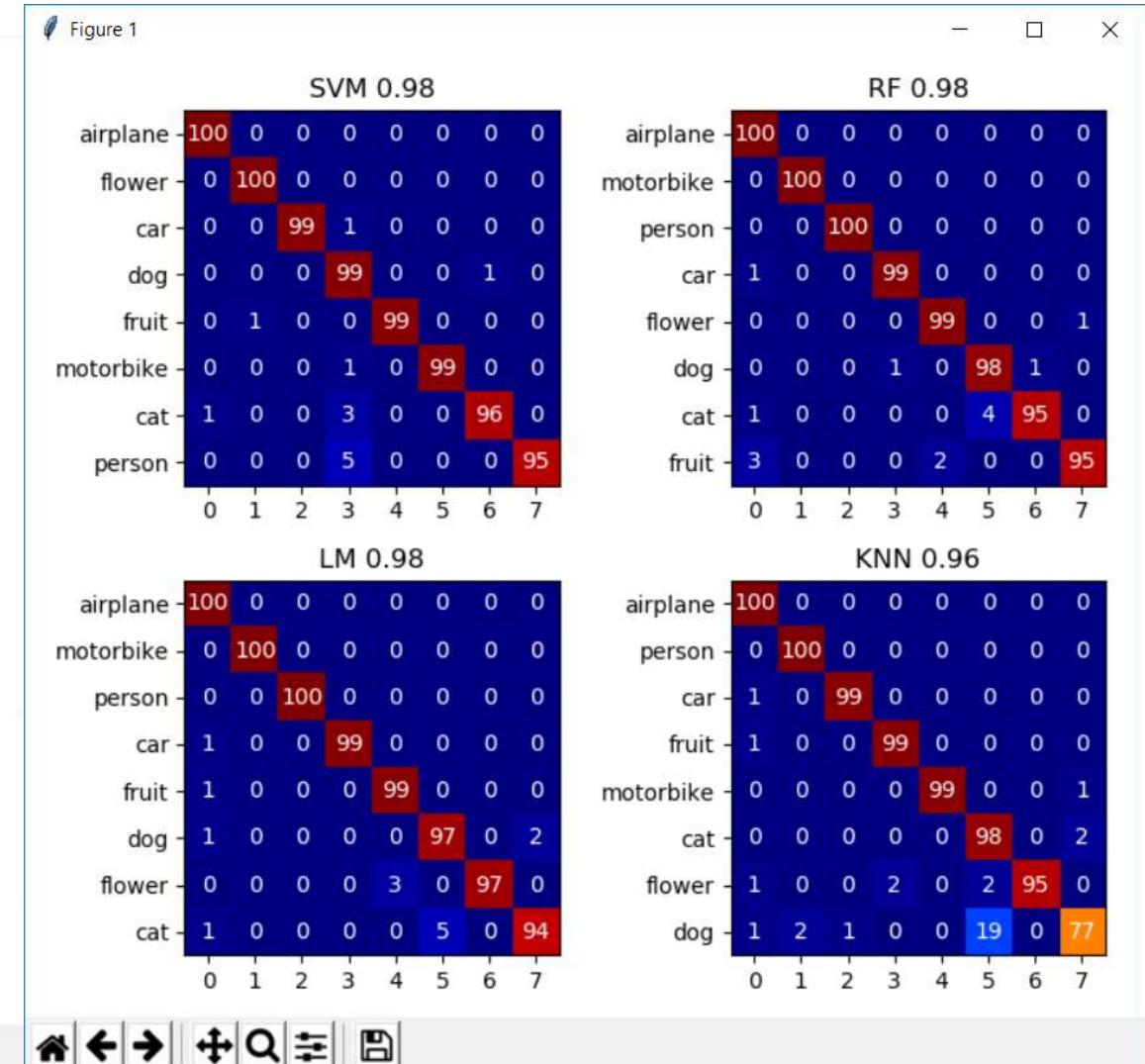
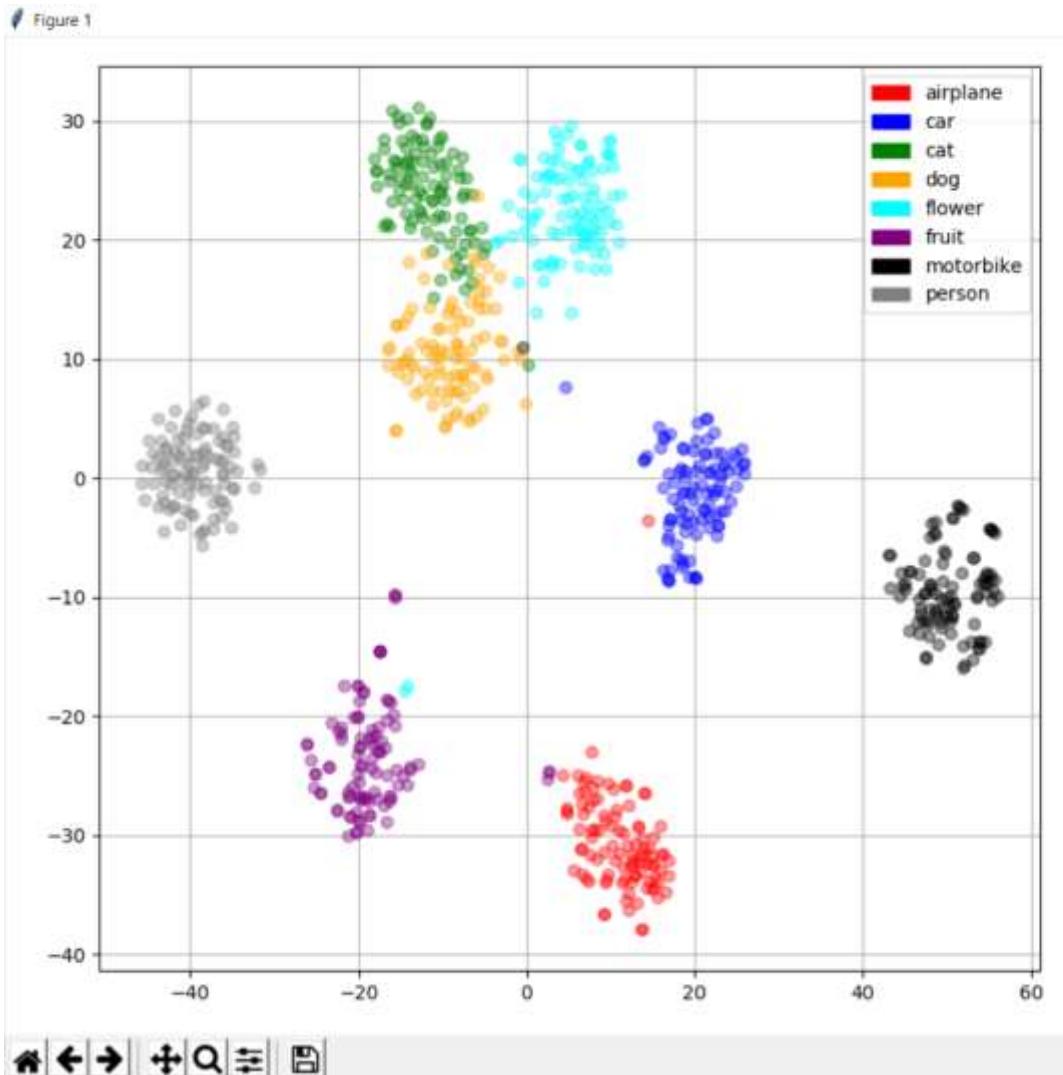
2 conv layers

Benchmarking the CNNs



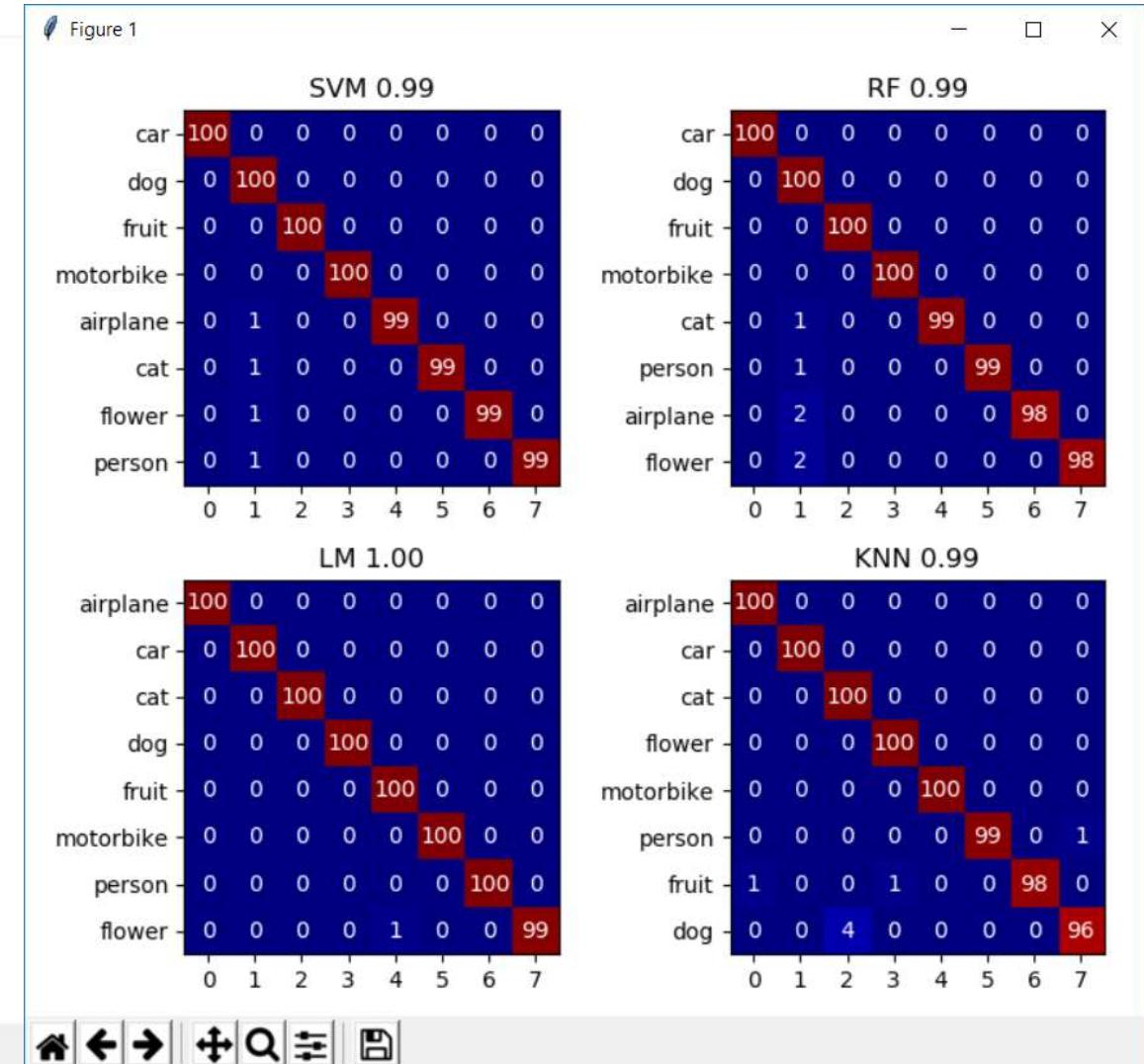
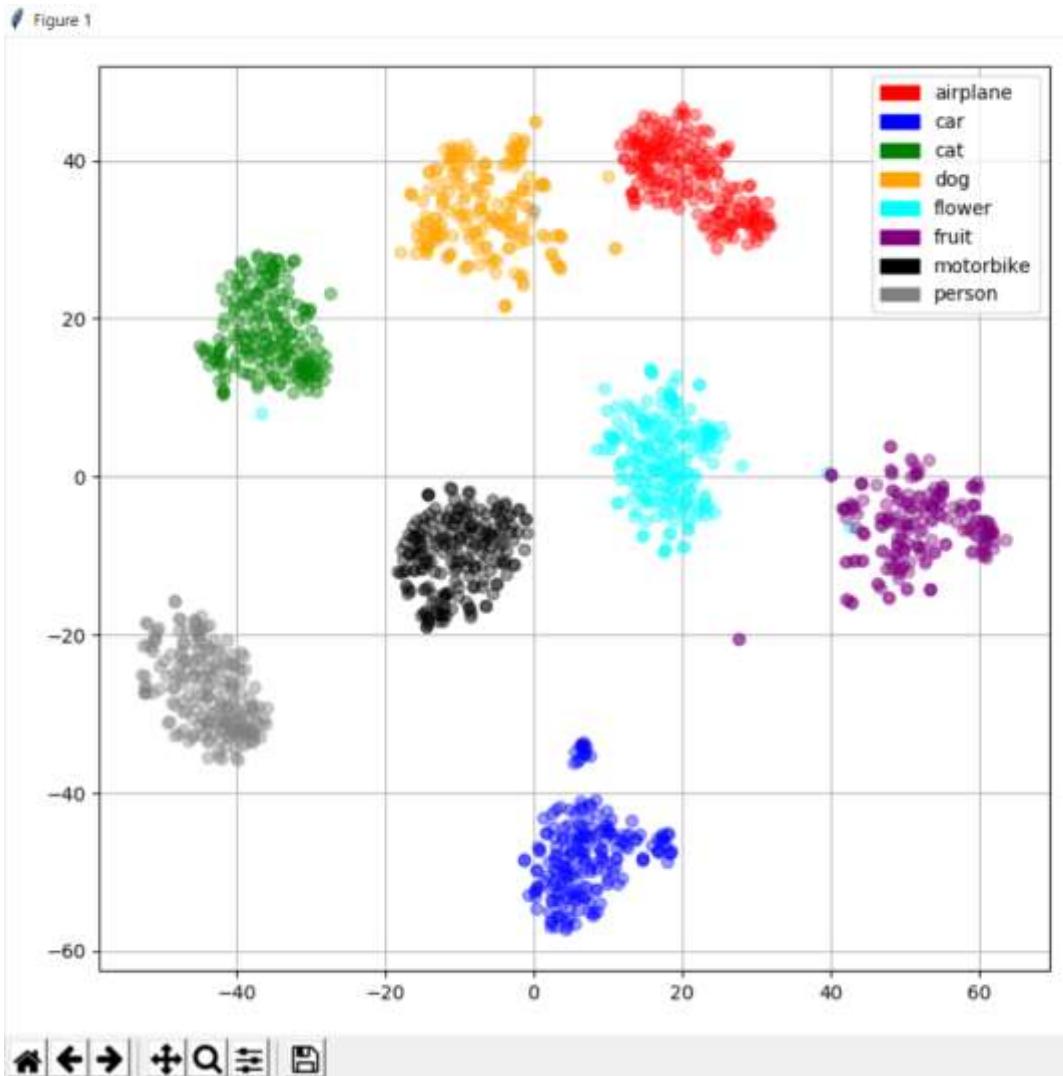
AlexNet

Benchmarking the CNNs



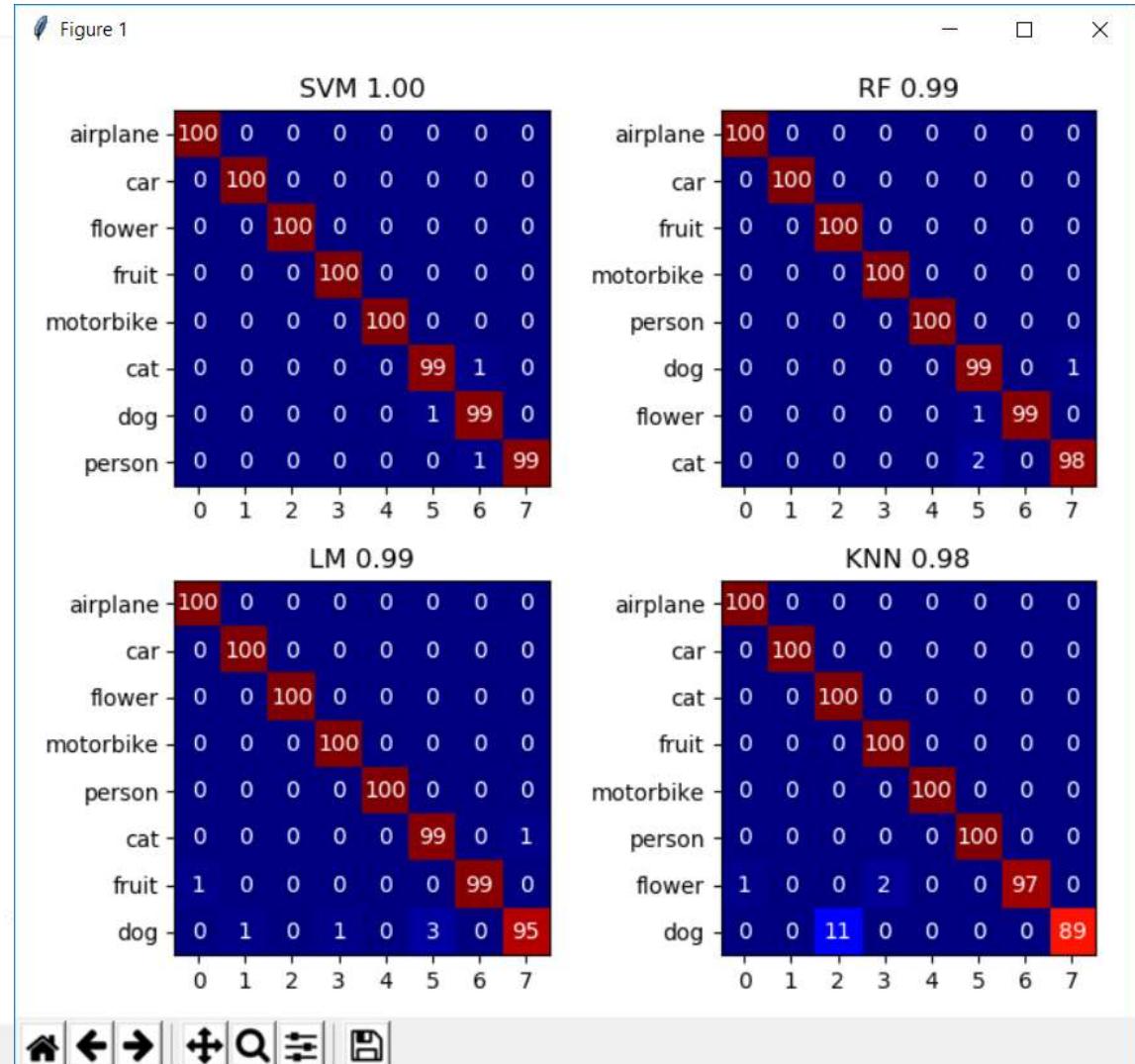
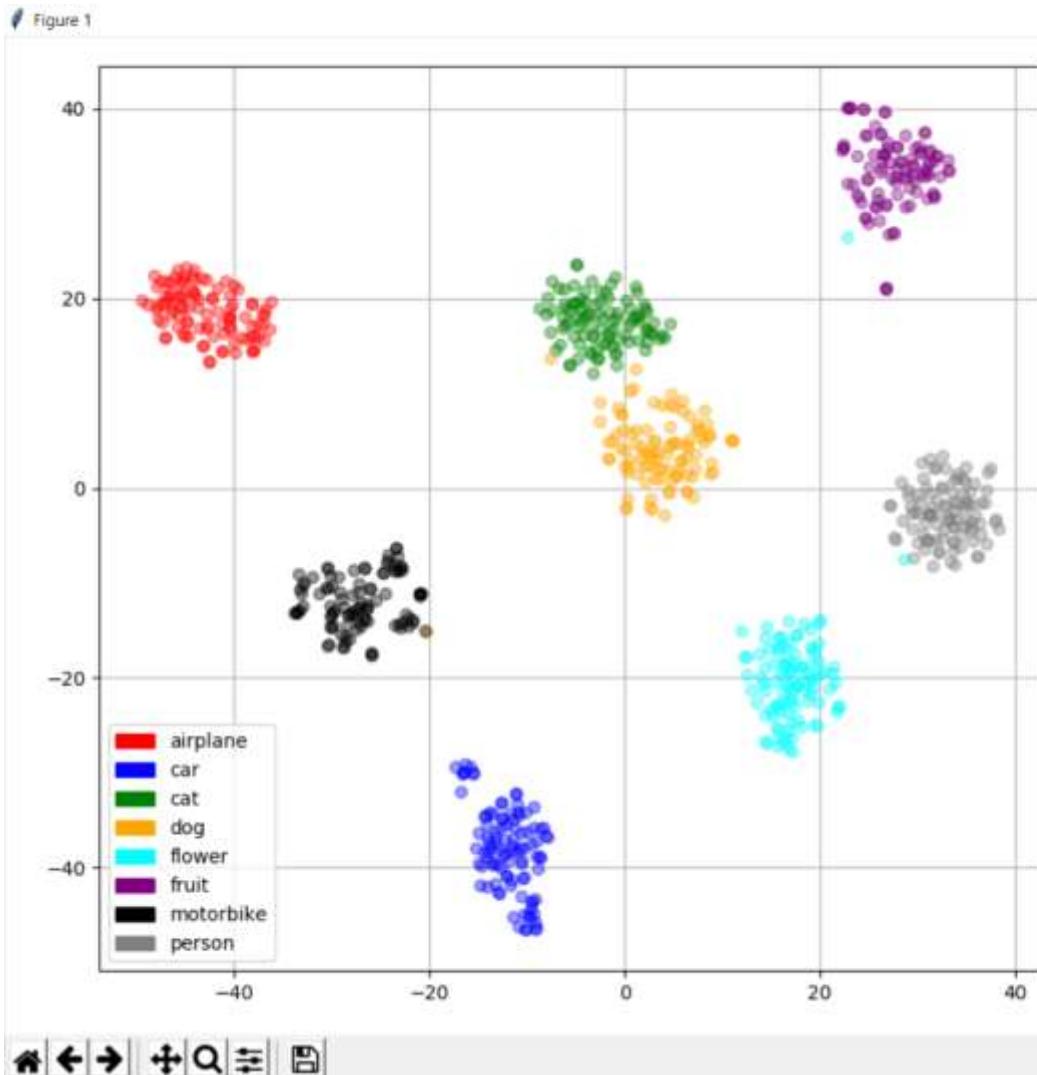
Inception

Benchmarking the CNNs



MobileNet

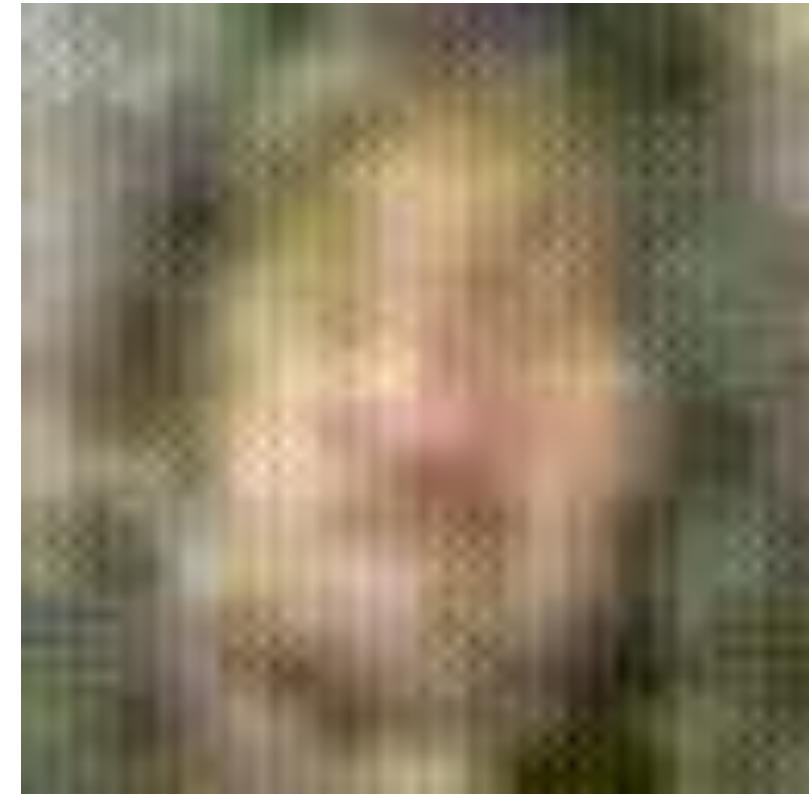
Benchmarking the CNNs



2.7. Reconstruct features back into images



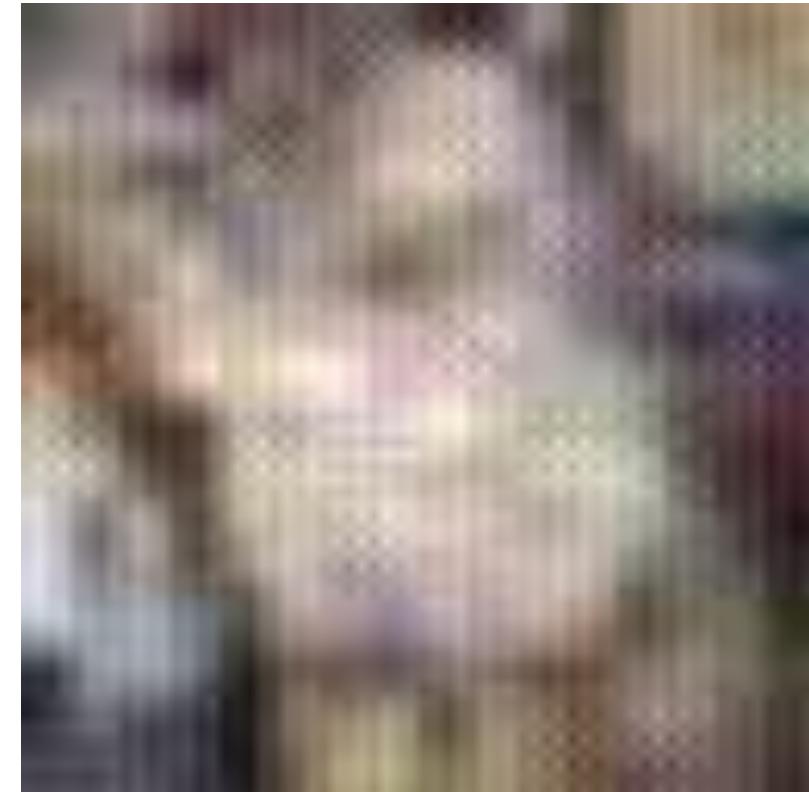
Reconstruct features back into images



Reconstruct features back into images



Reconstruct features back into images



Reconstruct features back into images

Features



airplane



car



cat



dog



flower



fruit



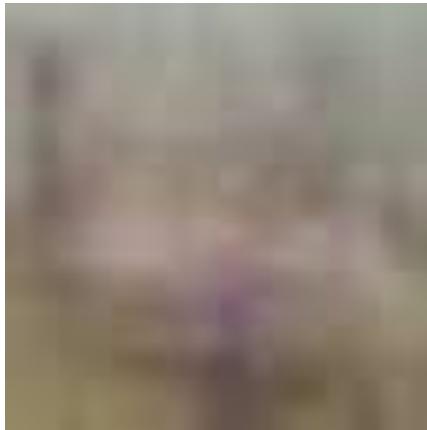
motorbike



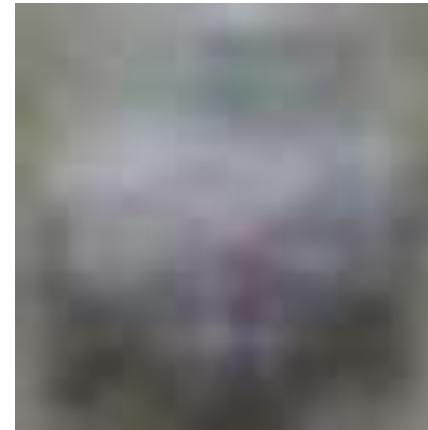
person

Reconstruct features back into images

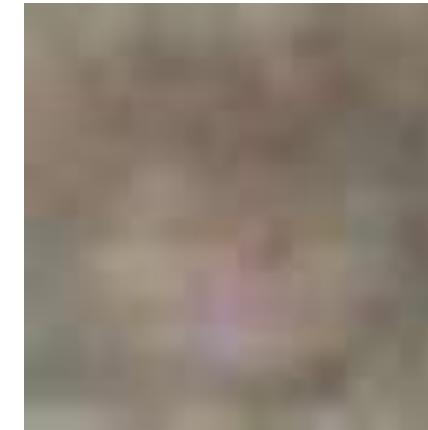
Reconstructed



airplane



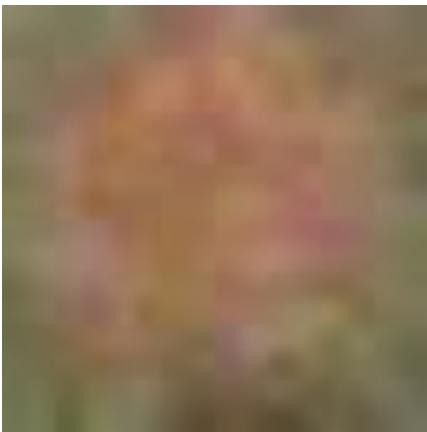
car



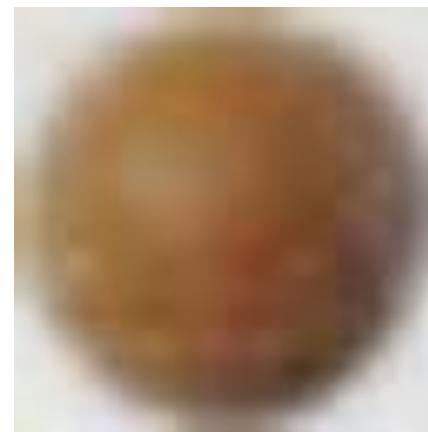
cat



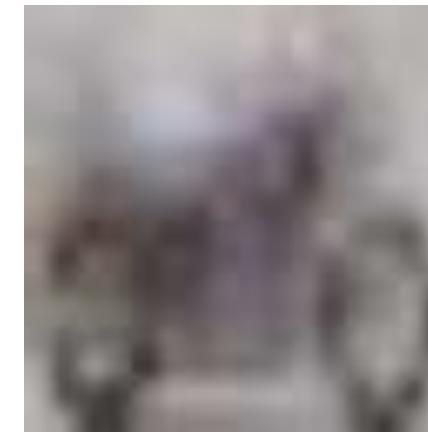
dog



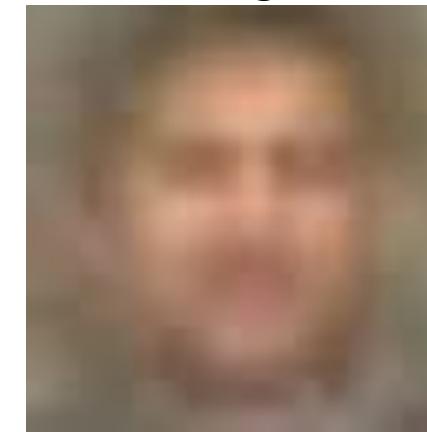
flower



fruit



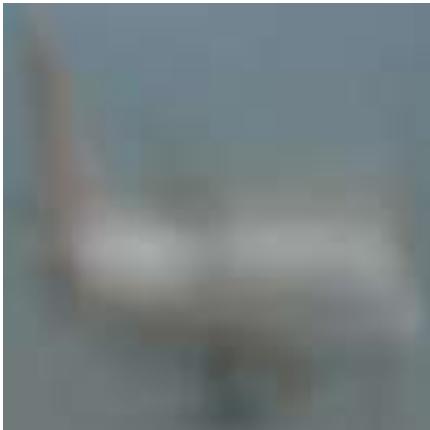
motorbike



person

Reconstruct features back into images

Original



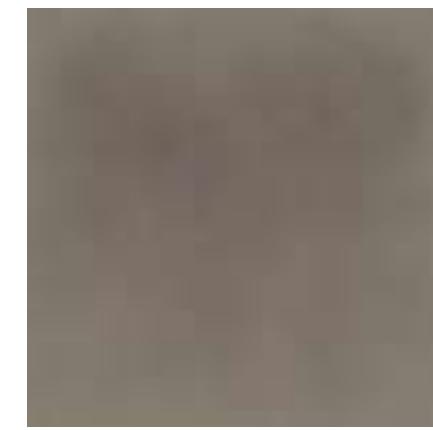
airplane



car



cat



dog



flower



fruit



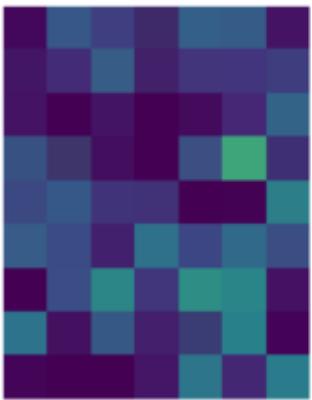
motorbike



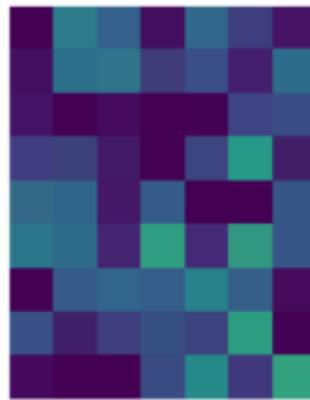
person

Reconstruct features back into images

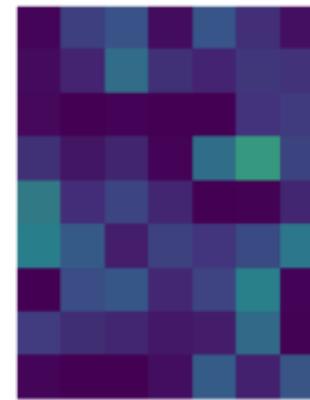
Features



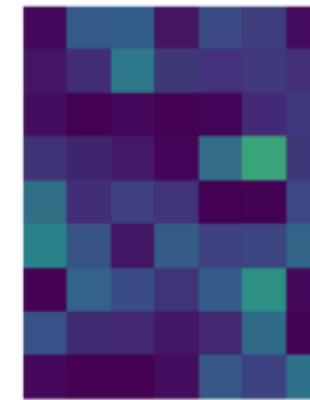
airplane



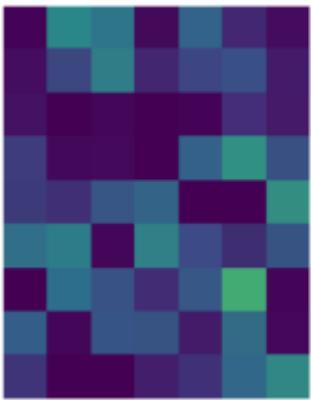
car



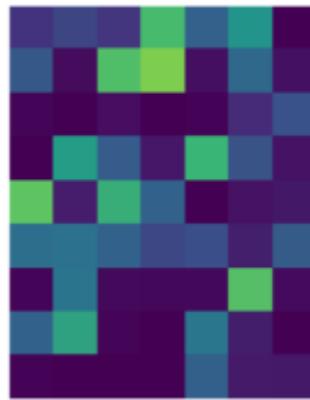
cat



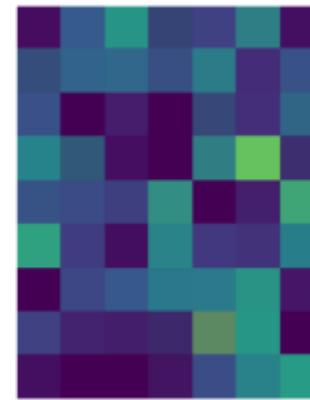
dog



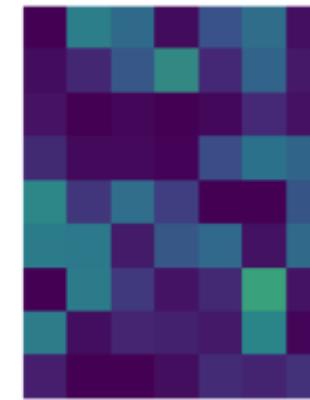
flower



fruit



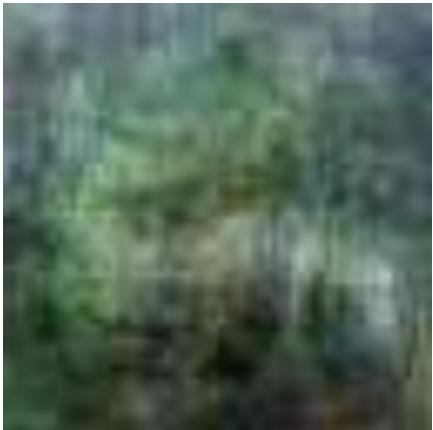
motorbike



person

Reconstruct features back into images

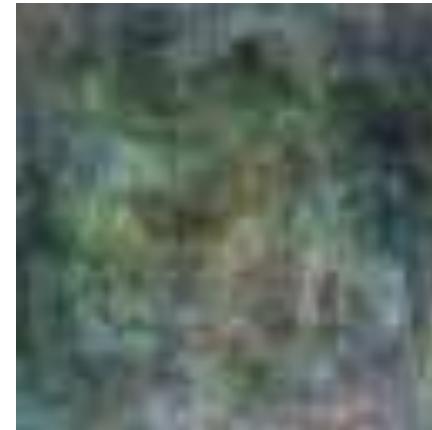
Reconstructed



airplane



car



cat



dog



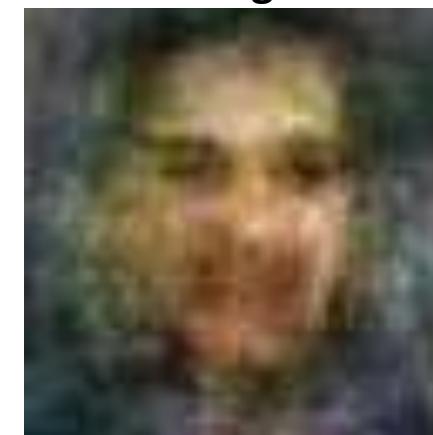
flower



fruit

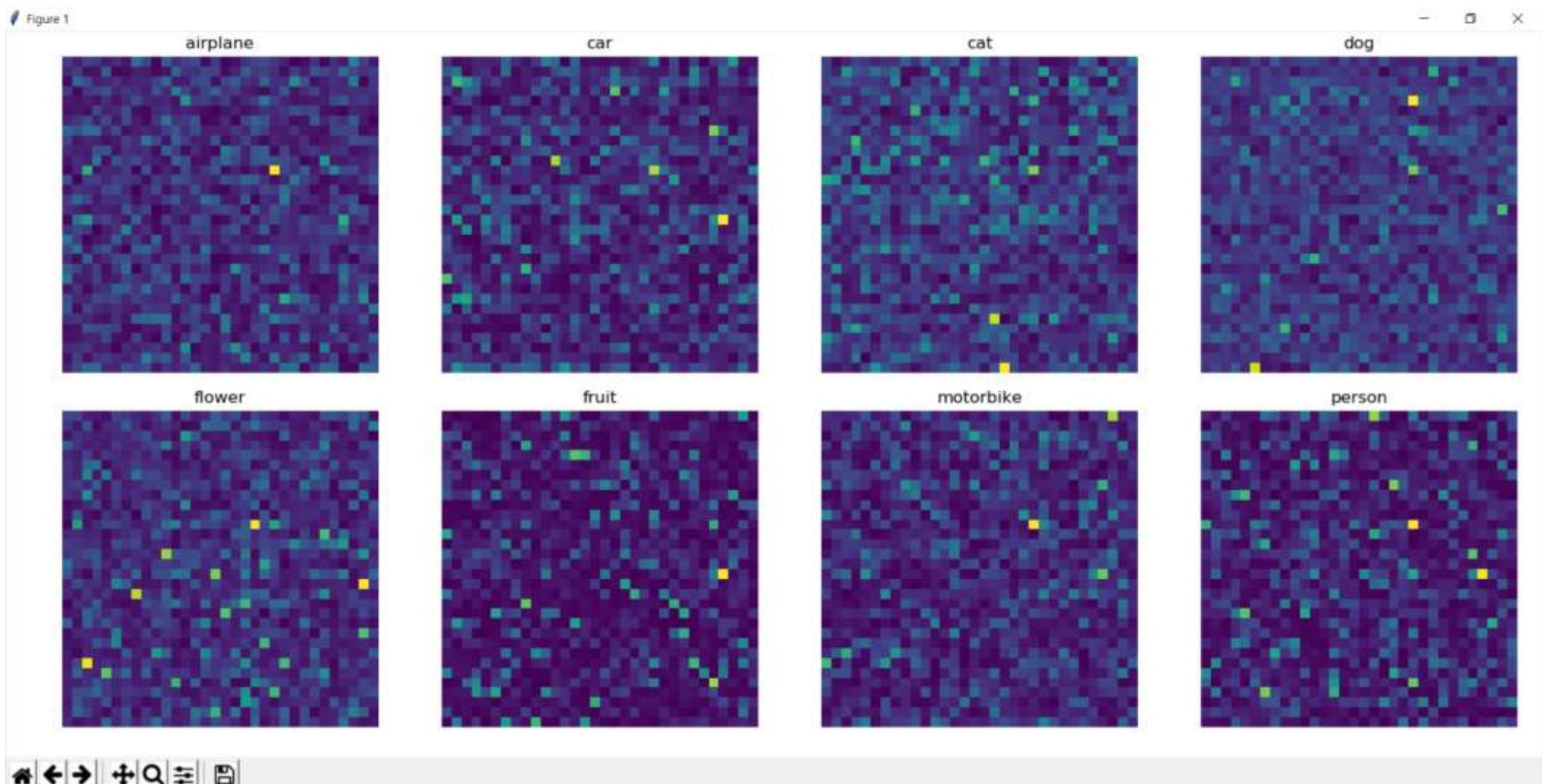


motorbike

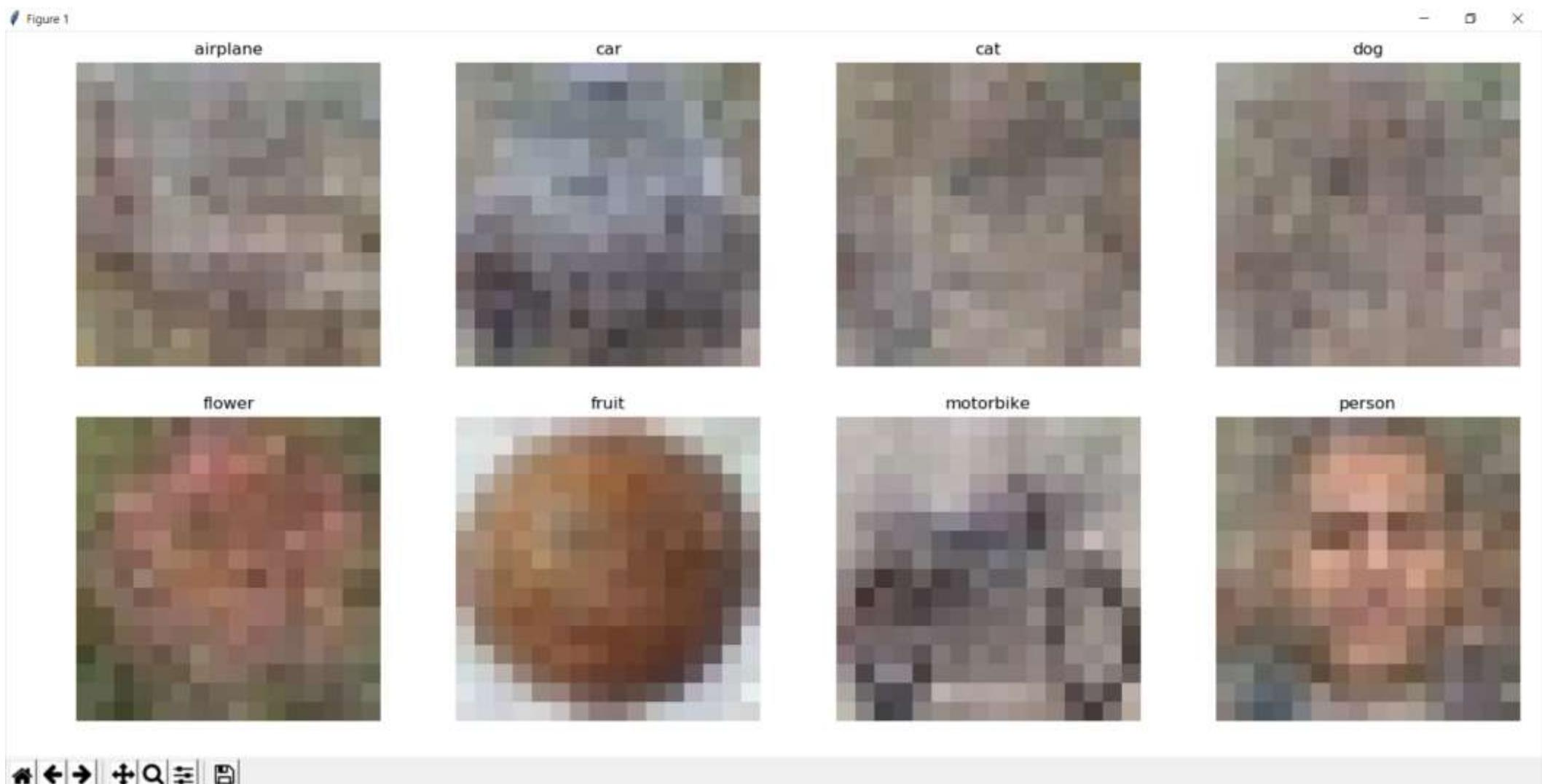


person

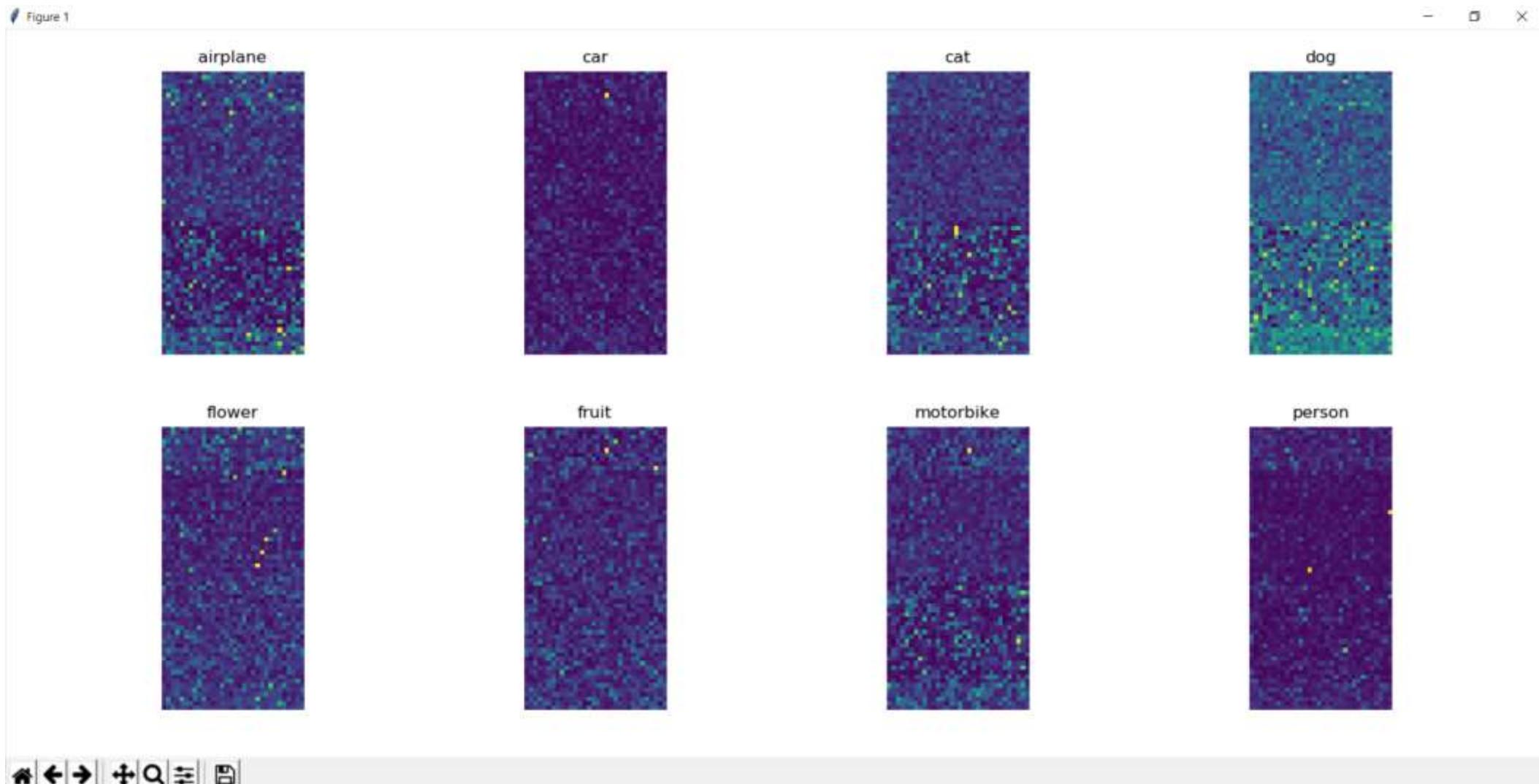
Reconstruct features back into images



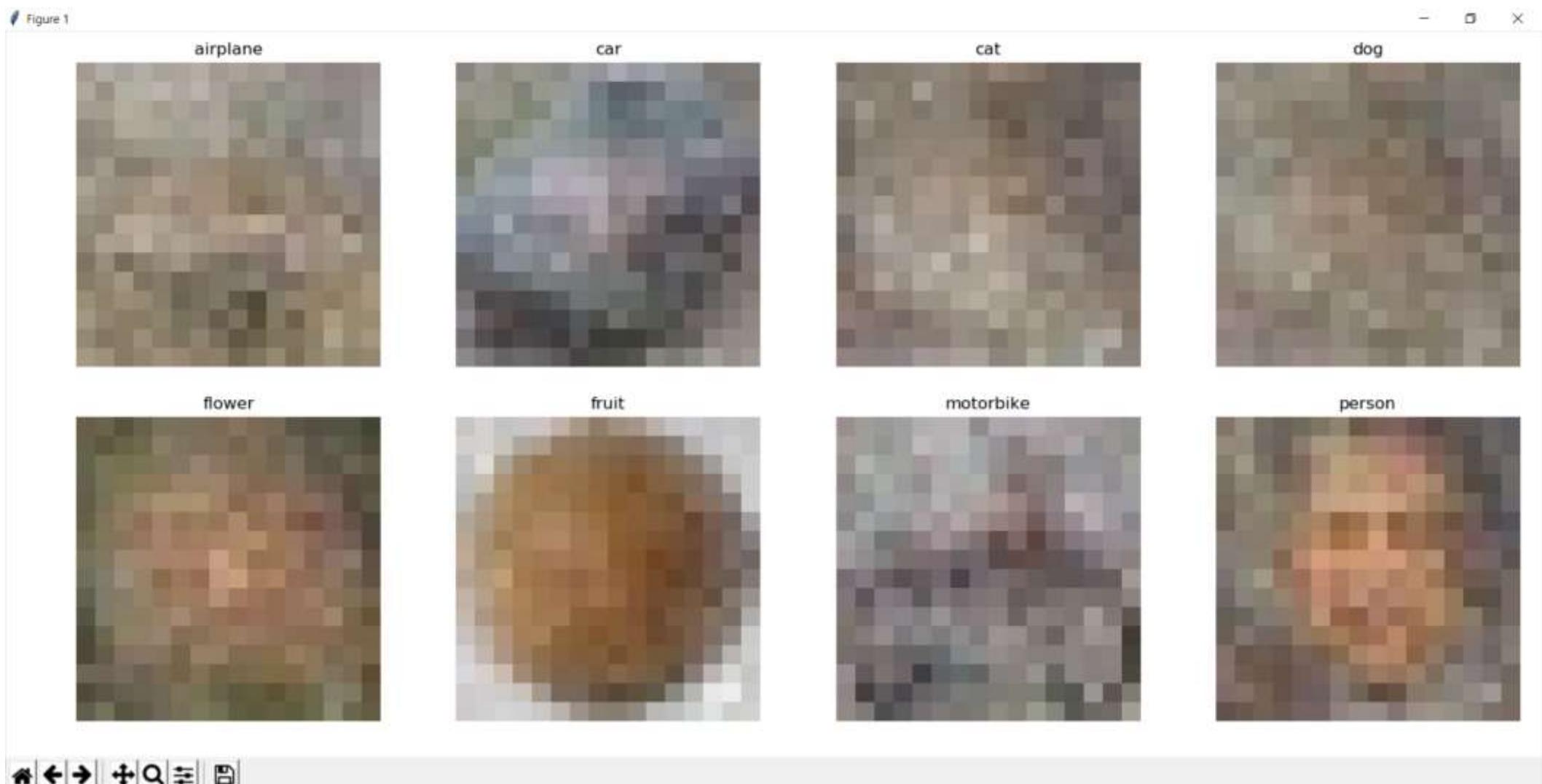
Reconstruct features back into images



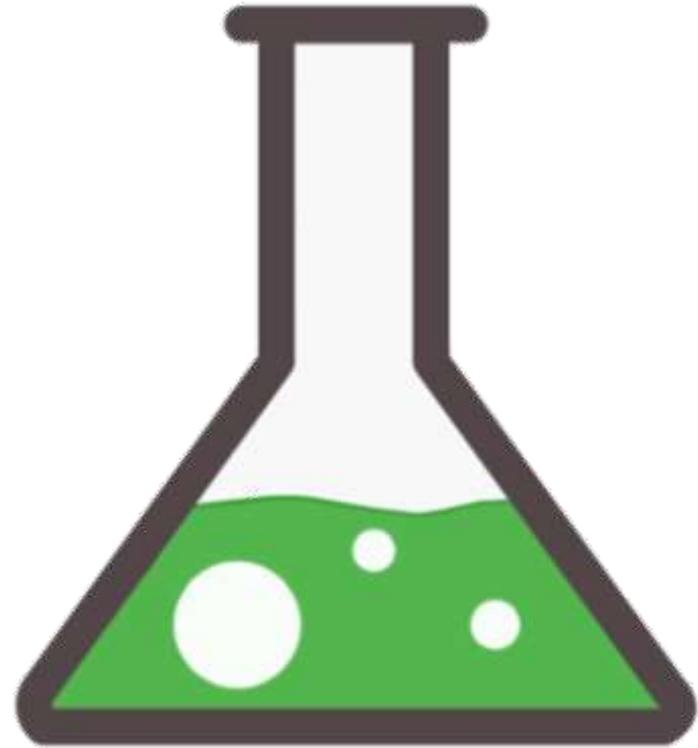
Reconstruct features back into images



Reconstruct features back into images



3.1. Assignments





Assignments

Supervised learning
Unsupervised learning
Object detection

Assignment 1

40%

Feature extraction and visualization
Transfer learning
Finetuning

Assignment 2

60%

Style transfer
Colorization
Texture analysis
GAN
Identification of human faces
Stereovision

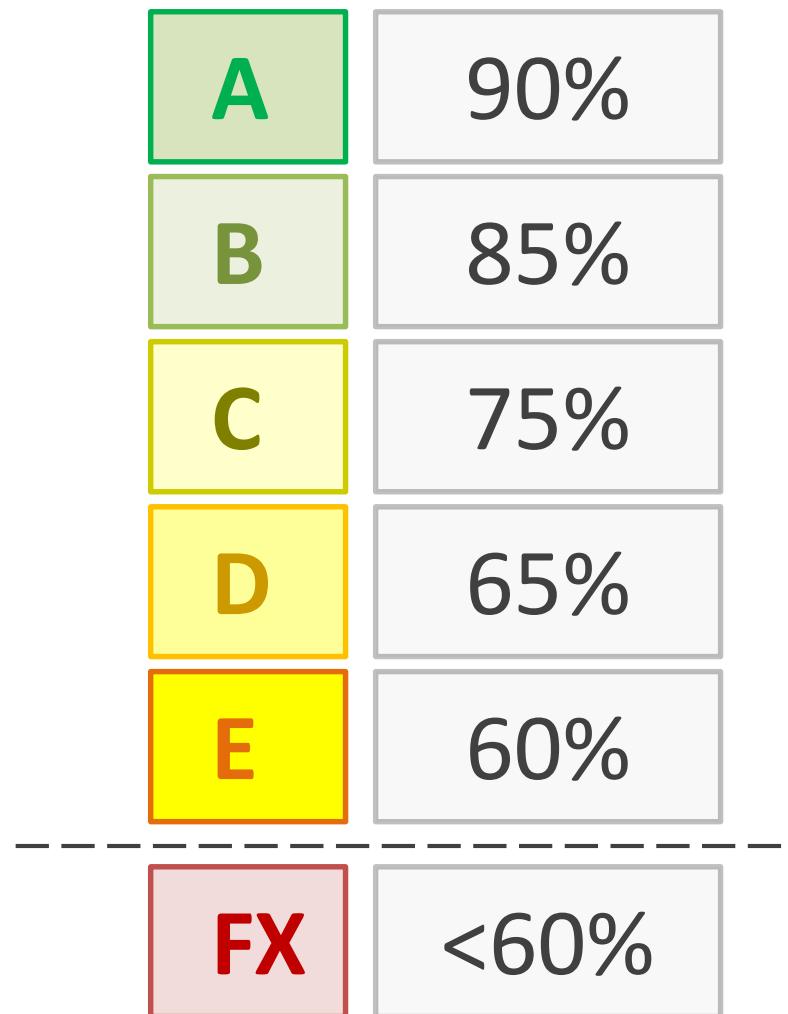
Tech talk

20%

Extra



Assignments





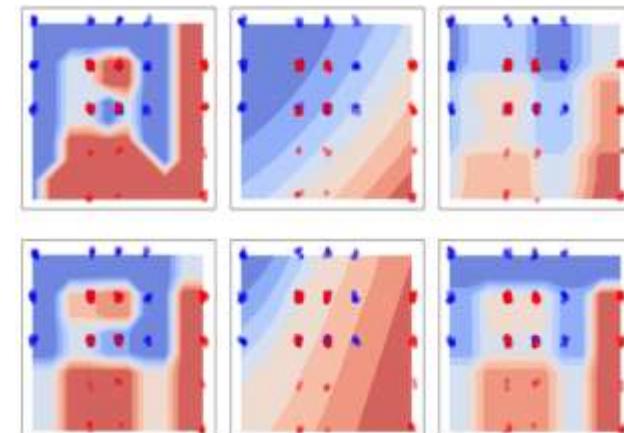
Assignment: Supervised learning

Description

- Get train/test data sample from open sources or generate it by yourself. Focus on low-dimension data, preferably 2 dimensions, 2 classes
- Visualize the distribution of sample data
- Estimate accuracy of data classification for 3-4 classifiers studied over lecture course
- Build ROC curves and hit maps

Assessment criteria

- In-house classifier: 40%
 - Out-of-box classifier: 20%
 - Data visualization charts: 20%
 - ROC curve charts: 20%
- * Extra: multi class championship 30%





Assignment: Unsupervised learning

Description

- Get train/test data sample from open sources or generate it by yourself. Focus on low-dimension data, preferably 2 dimensions, 3-5 classes
- Visualize the distribution of sample data
- Visualize iteration of EM/KNN algorithms
- Estimate the accuracy of classification

Assessment criteria

- In-house unsupervised classifier: 40%
 - Out-of-box classifier: 20%
 - Data visualization charts: 20%
 - Visualize the learning iterations: 20%
- * Benchmark results with some clustering algorithms 30%





Assignment: Object detection

Description

- Get train/test data sample from open sources or generate it yourself. Try on human faces, human shapes, cars, etc. Any type of object that you can capture with your laptop camera is preferable.
- Train the cascade detector and estimate the accuracy against the test set
- Run detector in live mode, detect objects in video stream captured by a laptop camera

Assessment criteria

- Build own detector: 40%
 - Use pre-trained detector: 20%
 - Accuracy charts: 20%
 - Detecting objects in live mode: 20%
- * Benchmark results with some alternative detectors 30%





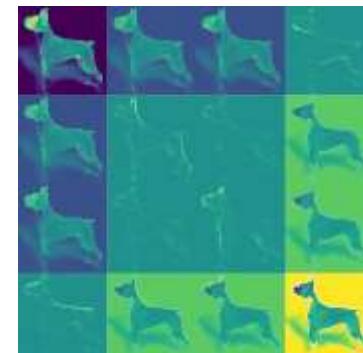
Assignment: Feature extraction

Description

- Get train/test data sample from open sources. Try both scenarios when classless are well separable and are not.
- Run data thru various feature extractors and visualize the results, estimate the quality of clustering
- Visualize the intermediate data of CNN: filters, layers

Assessment criteria

- Build own extractor : 40%
 - Use existing feature extractors: 20%
 - PCA and TSNE charts: 20%
 - Benchmark quality of extractors: 20%
- * Illustrate de-convolutions: 30%





Assignment: Transfer learning

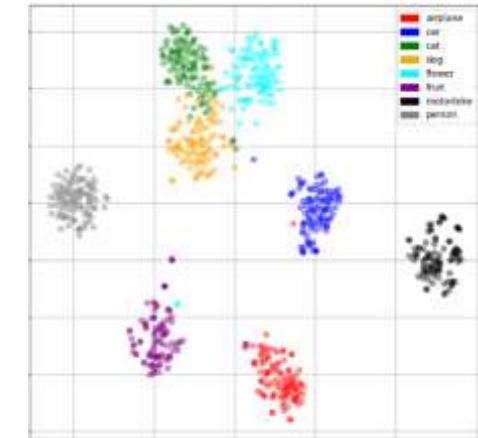
Description

- Benchmark accuracy of multiple classifiers (SVM, RF, LM, etc) applied on top of several feature extractors (AlexNet, VGG, ImageNet etc)
- Research how accuracy very depending on the depth of the frozen layers, visualize this dependency
- Build/visualize confusion matrices
- Illustrate most typical classification errors

Assessment criteria

- Build confusion matrices: 60%
- Build the accuracy trend: 20%
- Illustrate typical errors: 20%

fruit	99	0	0	0	0	0	0	1
motorbike	0	95	1	1	1	0	0	2
person	1	0	95	0	0	1	1	2
airplane	0	1	0	92	2	4	0	1
car	0	1	0	2	91	1	4	1
flower	1	1	0	2	2	87	3	4
cat	1	0	1	0	2	7	73	16
dog	1	0	1	1	1	12	24	60





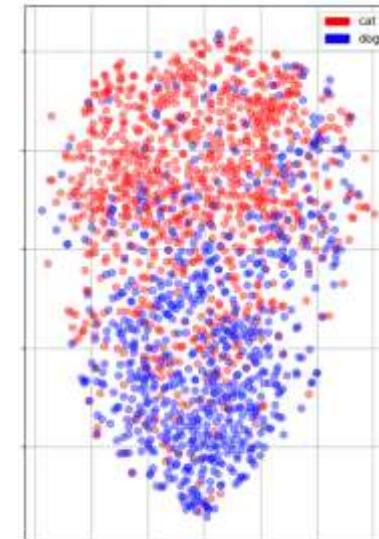
Assignment: Finetuning

Description

- Benchmark accuracy of multiple CNNs finetuned on specific classes of images (e.g faces, cars)
- Illustrate how internal parameters were optimized to maximize the classification accuracy
- Build/visualize confusion matrices
- Illustrate most typical classification errors

Assessment criteria

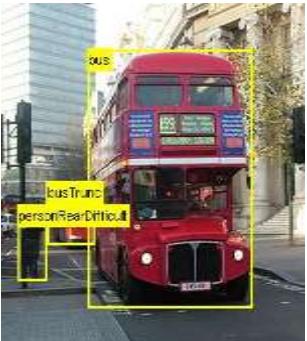
- Build confusion matrices: 60%
- Illustrate internal CNN data: 20%
- Illustrate typical errors: 20%





Assignment: Tech talks

1. Object detection



2. Style transfer



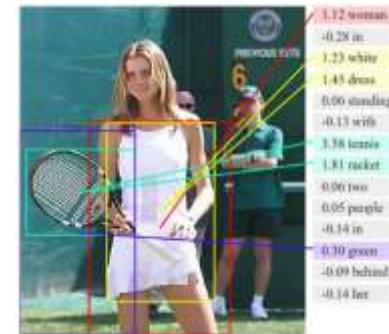
3. GAN



3D object reconstruction



Image
to text



Segmentation



- | | |
|---|----------|
| ■ | null |
| ■ | shorts |
| ■ | shoes |
| ■ | purse |
| ■ | top |
| ■ | necklace |
| ■ | hair |
| ■ | skin |



Assignment: Tech talks

