Class is 170.

Announcements

Matlab Grader homework, Binary graded.

168 (HW1), 166,165 (HW2) has done the homework. (**If you have not done HW talk to me/TA!**)

Homework 3 due **5 May TODAY**

Homework 4 (SVM +Dictionary Learning) due ~24 May, released soon

Jupiter "GPU" homework. Due 10 May

Today:

- Stanford CNN 12, K-means, EM (Bishop 9)
- Play with Tensorflow playground before class http://playground.tensorflow.org

  Solve the spiral problem

Monday

- Stanford CNN 13, Dictionary Learning,

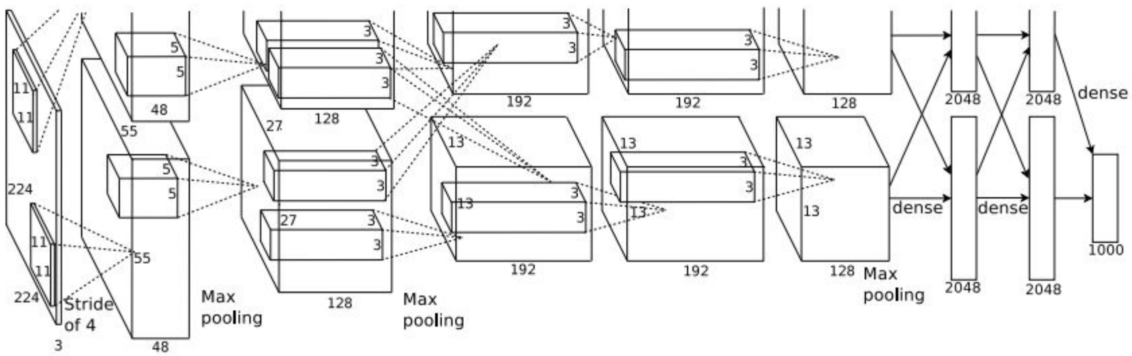
Projects

- **3-4** person groups preferred
- Deliverables: Poster, Report & main code (plus proposal, midterm slide)
- **Topics** your own or chose form suggested topics. Some **physics inspired.**
- **April 26 groups** due to TA. **5 students not signed up 44 Groups formed. Guidelines is on Piazza**
- **May 5** proposal due. use dropbox Format “Proposal”+groupNumber

https://www.dropbox.com/request/XGqCV0qXm9LBYz7J1msS

- **Wednesday May 22** Midterm slide presentation.
- **Project discussion, 22 May:** We split into 6 sub-classes. The purpose is to make sure your project is on track, good progress and good goals. Each group gives a ~10 min presentation by all members
  - (each person talks for ~2 min, ~1 slide)
  - Motivation & background, which data?
  - small Example,
  - final outcome, (focused on method and data)
  - difficulties,
- There are 7 Groups in each sub-class, thus we have 15 min in total/group. And will use the remaining time for discussion.
- **June 5, 5-8pm** poster. Atkinson Hall with Pizza. Upload June ~3
- Report and code due **Saturday 15 June.**

What’s going on inside ConvNets?



Input Image:
3 x 224 x 224

Class Scores:
1000 numbers

What are the intermediate features looking for?

Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

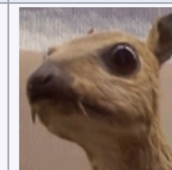
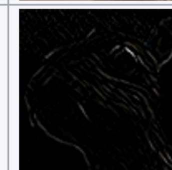
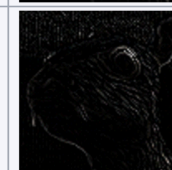
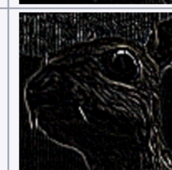
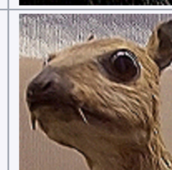
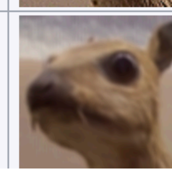
Image Processing

The general expression of a convolution is

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t),$$

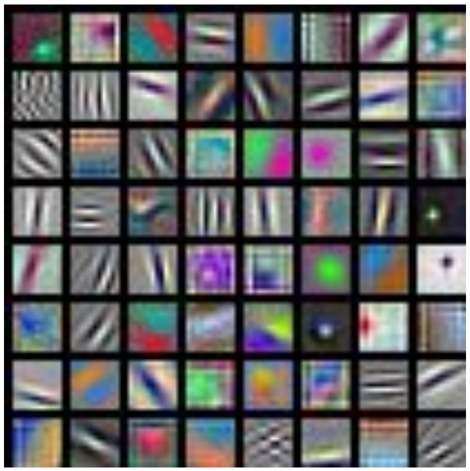
where $g(x,y)$ is the filtered image, $f(x,y)$is the original image, ω is the filter kernel. Every element of the filter kernel is considered by $-a \leq s \leq a$ and $-b \leq t \leq b$.

Depending on the element values, a kernel can cause a wide range of effects.

		Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Reverse engineering

First Layer: Visualize Filters

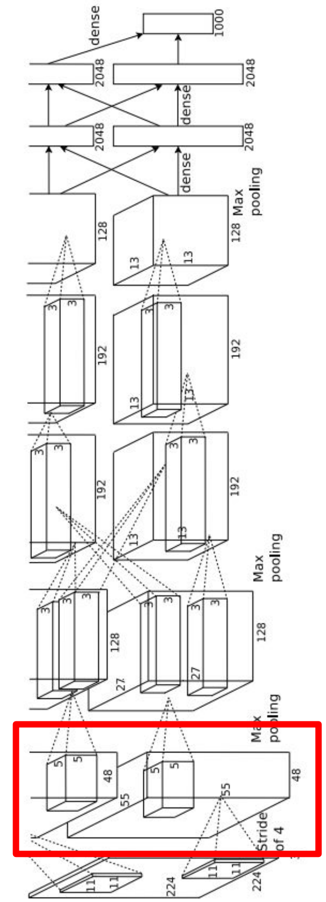


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



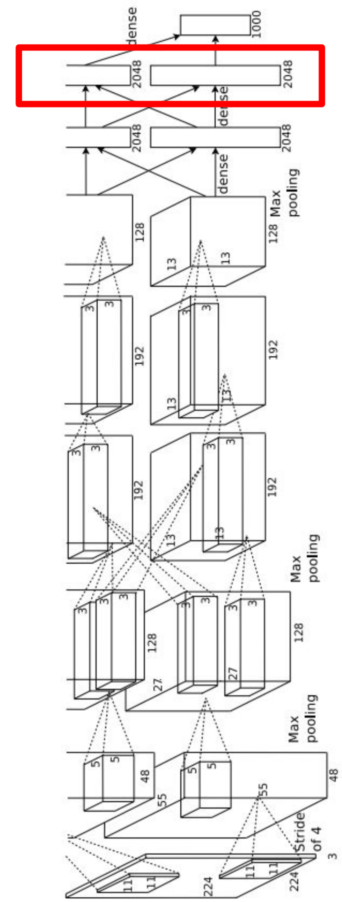
Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Similar to human neuron: First observe oriented edges

$w^T x$ subject to $||w|| = 1$

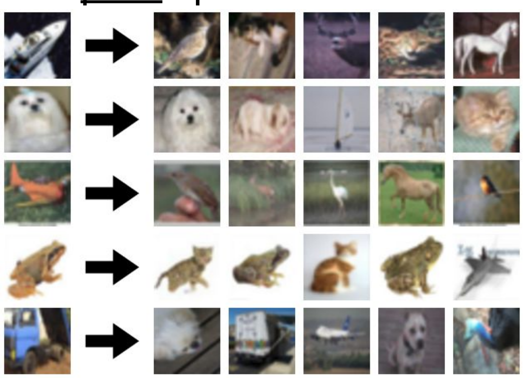
Last Layer: Nearest Neighbors

4096-dim vector



Test image L2 Nearest neighbors in feature space

Recall: Nearest neighbors in pixel space



Feature Inversion

Given a CNN feature vector for an image, find a new image that:

- Matches the given feature vector
- “looks natural” (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

Given feature vector

Features of new image

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

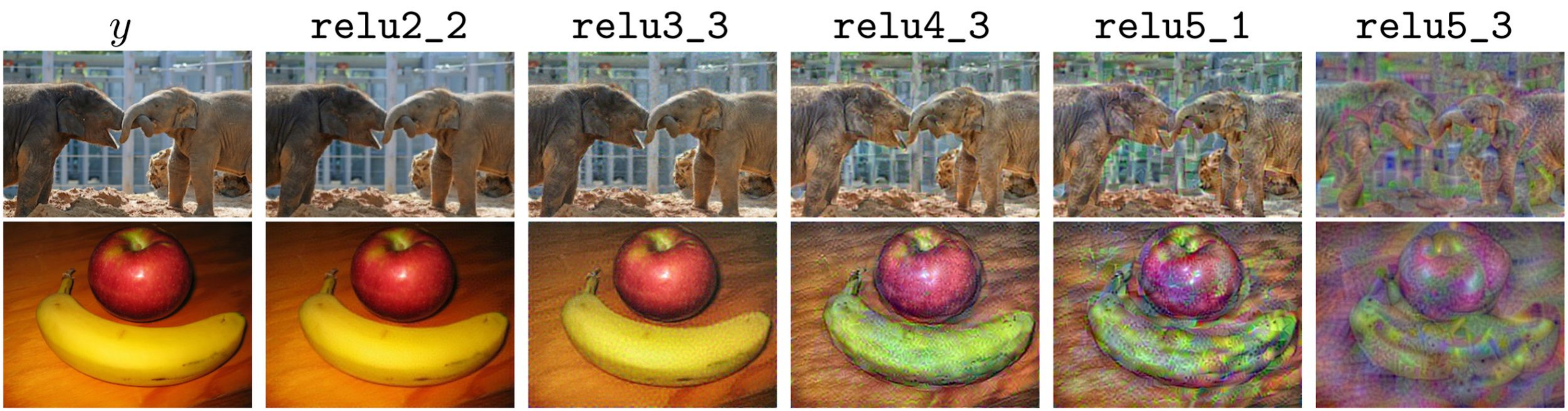
$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer (encourages spatial smoothness)

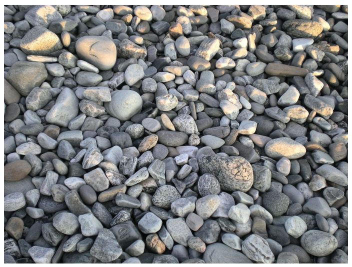
Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015

Feature Inversion

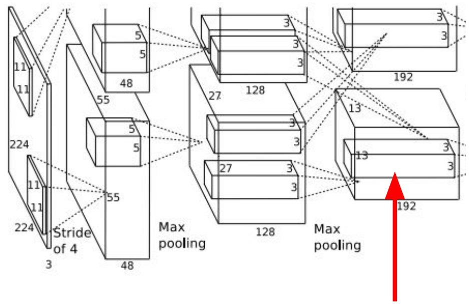
Reconstructing from different layers of VGG-16

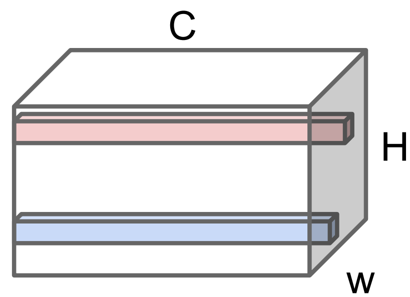


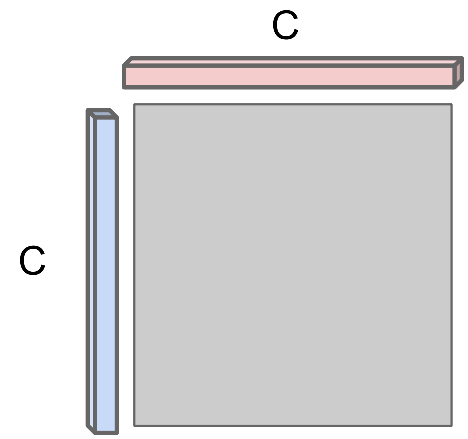
Neural Texture Synthesis: Gram Matrix



This image is in the public domain.







Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

Average over all HW pairs of vectors, giving **Gram matrix** of shape C x C

Efficient to compute; reshape features from

C x H x W to =C x HW

then compute $G = FF^T$

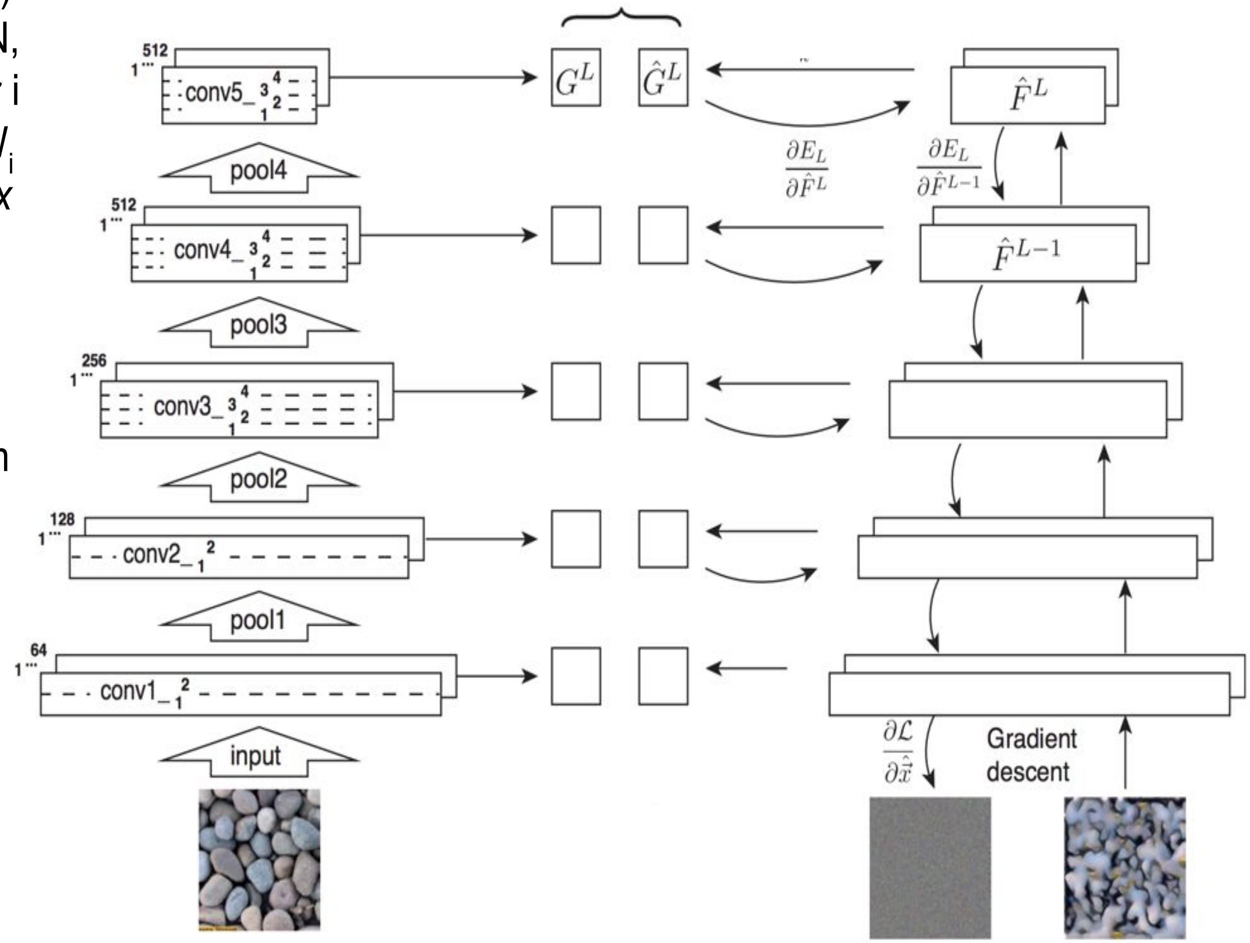
Neural Texture Synthesis

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

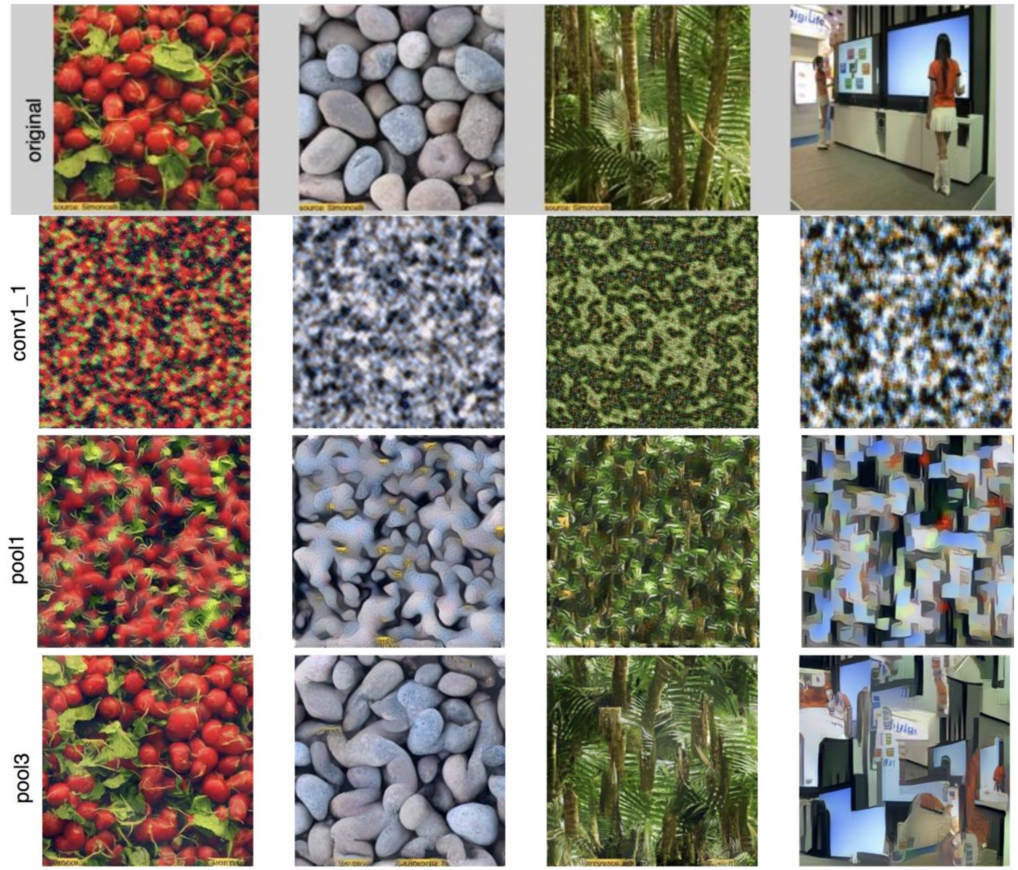
$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \text{ (shape } C_i \times C_i\text{)}$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5

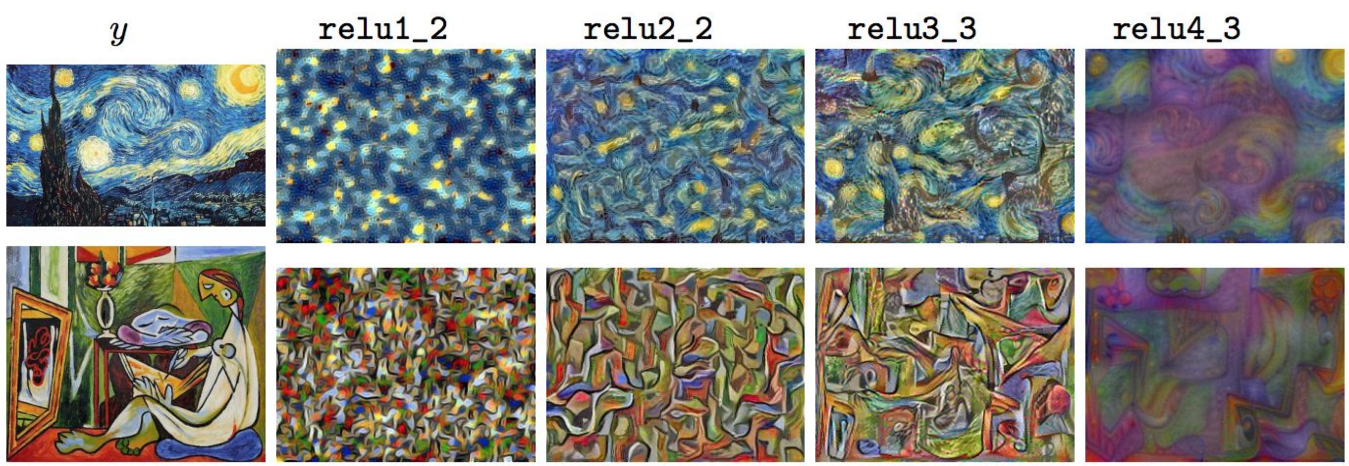


atys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Neural Texture Synthesis



Reconstructing texture from higher layers recovers larger features from the input texture



Texture synthesis (Gram reconstruction)

Neural Style Transfer

Content Image

+

Style Image

=

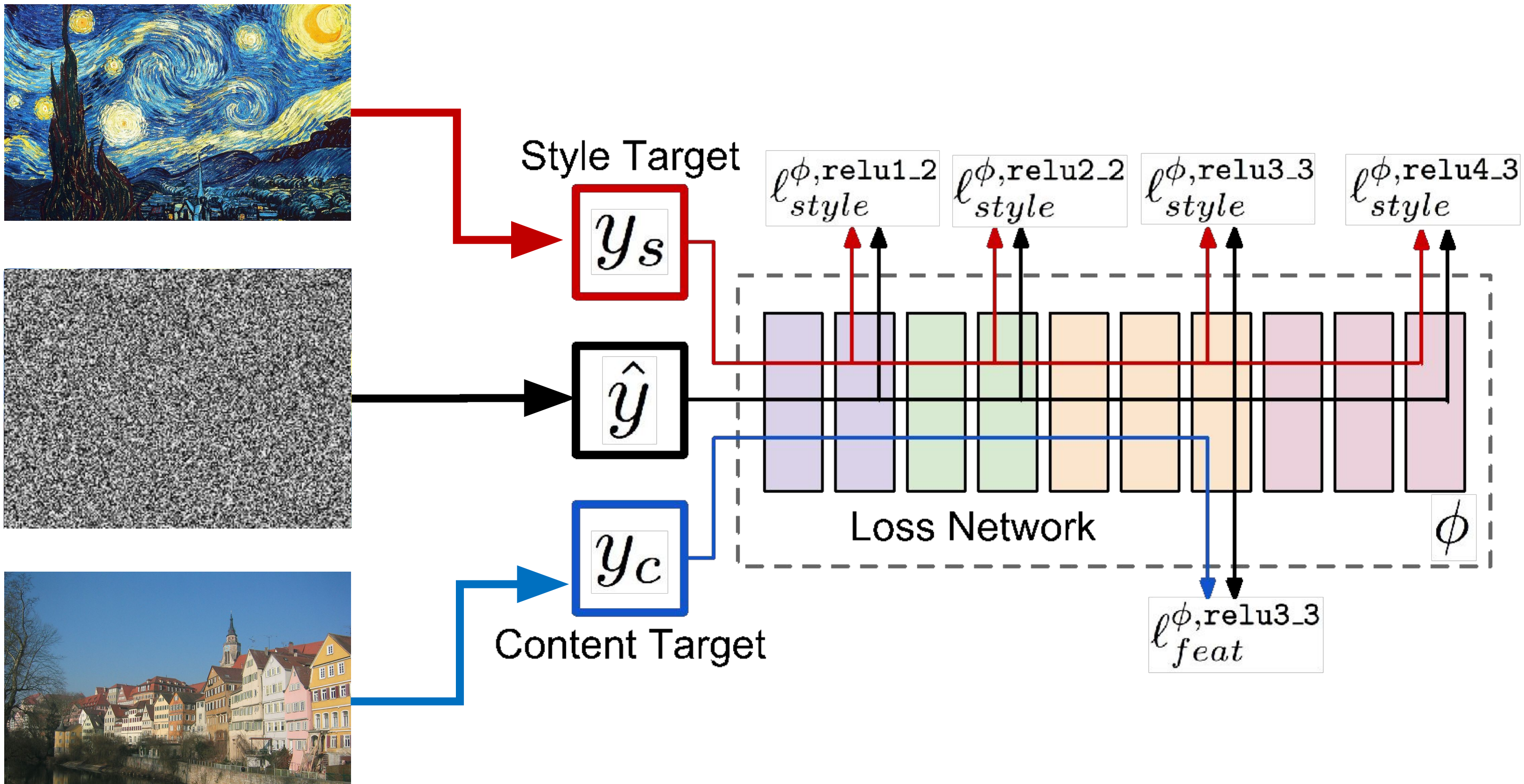
Style Transfer!

Style transfer=Feature reconstruction loss+ Gram matrix

Style image

Output image (Start with noise)

Content image



Gatys, Ecker, and Bethge, “Image style transfer using convolutional neural networks”, CVPR 2016

Output image

K-means and Expectation Maximization

Mike Bianco and Peter Gerstoft

ECE228

5/6/2019

K-means and expectation maximization (EM) can be considered unsupervised learning

- In *supervised learning*, we have desired machine learning (ML) model output or 'action' **y** based on inputs **x** (features), and model parameters $\boldsymbol{\theta}$
  - Probabilities of the form: p(**y**|**x**,$\boldsymbol{\theta}$)
  - Linear regression and classification, support vector machines, etc.

- In *unsupervised learning,* we are interested in discovering useful patterns in the features. This can be for discovering latent data 'causes' or significant 'groups'
  - Probabilities of the form: p(**x**|$\boldsymbol{\theta}$)
  - Principal components analysis (PCA), K-means, dictionary learning, etc.

Unsupervised learning

Unsupervised machine learning is inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning.

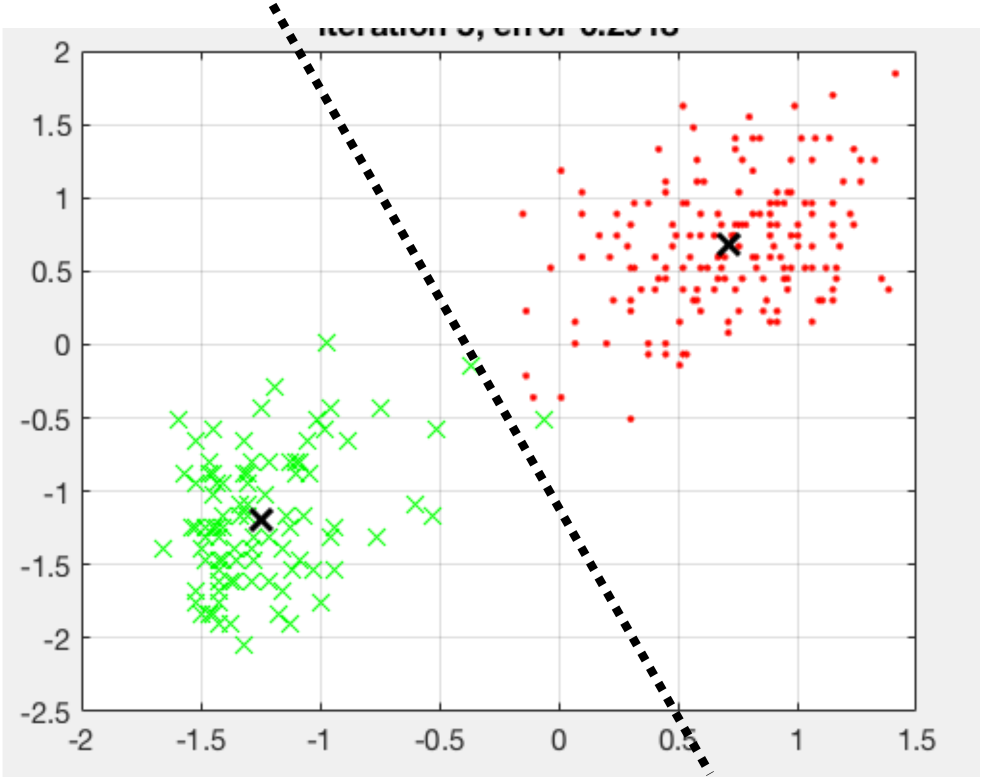
We are not interested in prediction

Supervised learning: all classification and regression.

$$\boldsymbol{Y} = \boldsymbol{w}^T \boldsymbol{X}$$

Prediction is important.

Supervised learning: least square classifier (binary)



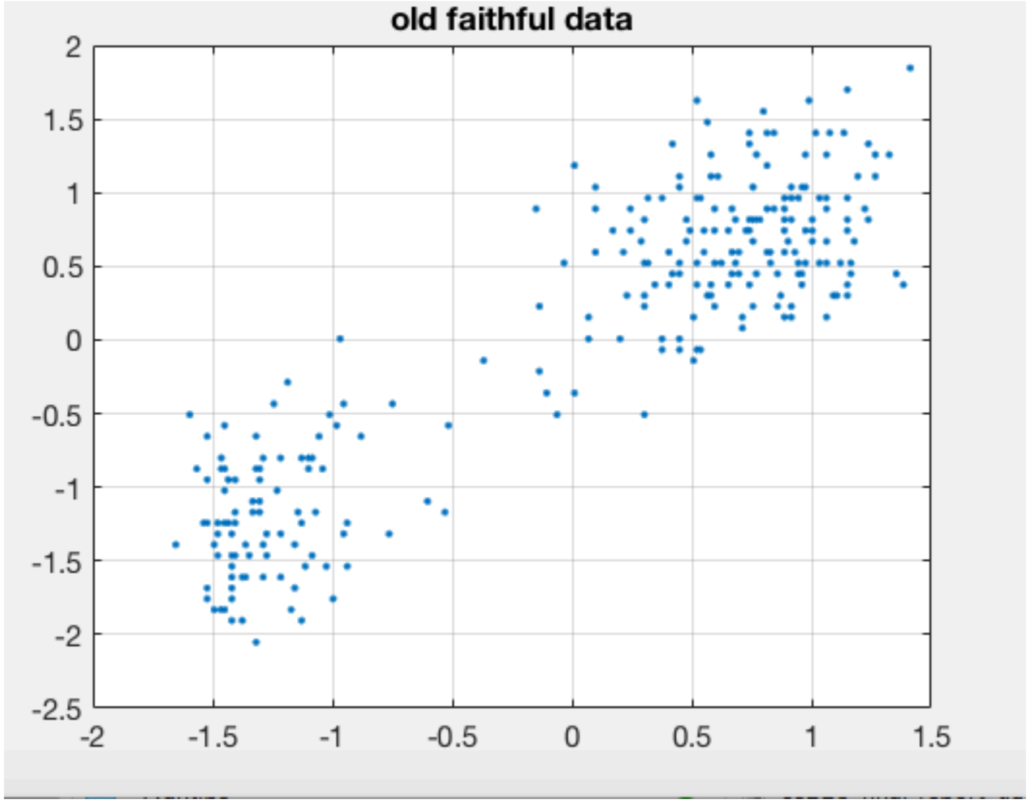
y=Xw

Training set $\{(x^1, y^1), (x^2, y^2), (x^3, y^3)\}$

We are given the two classes

(green = 0, red = 1)

Unsupervised learning: how are features best divided?



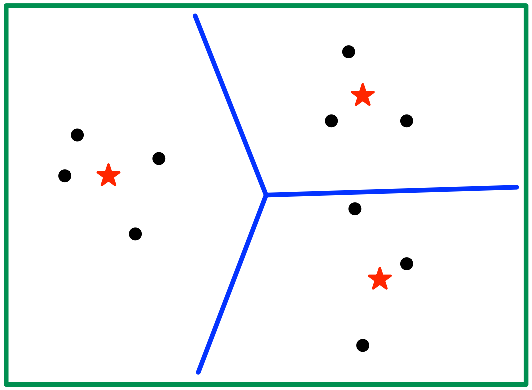
Just have features $\{(x_1^1, x_2^1), (x_1^2, x_2^2), (x_1^3, x_2^3)\}$

K-means

- **Input**: Points $\mathbf{x}_1,\ldots,\mathbf{x}_N \in R^p$; integer K
- **Output**: "Centers", or representatives, $\mu_1,\ldots, \mu_K \in R^p$
- Output also $\mathbf{z}_1,\ldots,\mathbf{z}_N \in R^K$

Goal: Minimize average squared distance between points and their nearest representatives:

- $cost(\mu_1, \ldots, \mu_K) = \sum_{n=1}^{N} \min_j \left\| x_n - \mu_j \right\|$



The centers carve $\mathbb{R}^p$ up into k convex regions: μ_j's region consists of points for which it is the closest center.

K-means

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \tag{9.1}$$

Solving for r_{nk}

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{9.2}$$

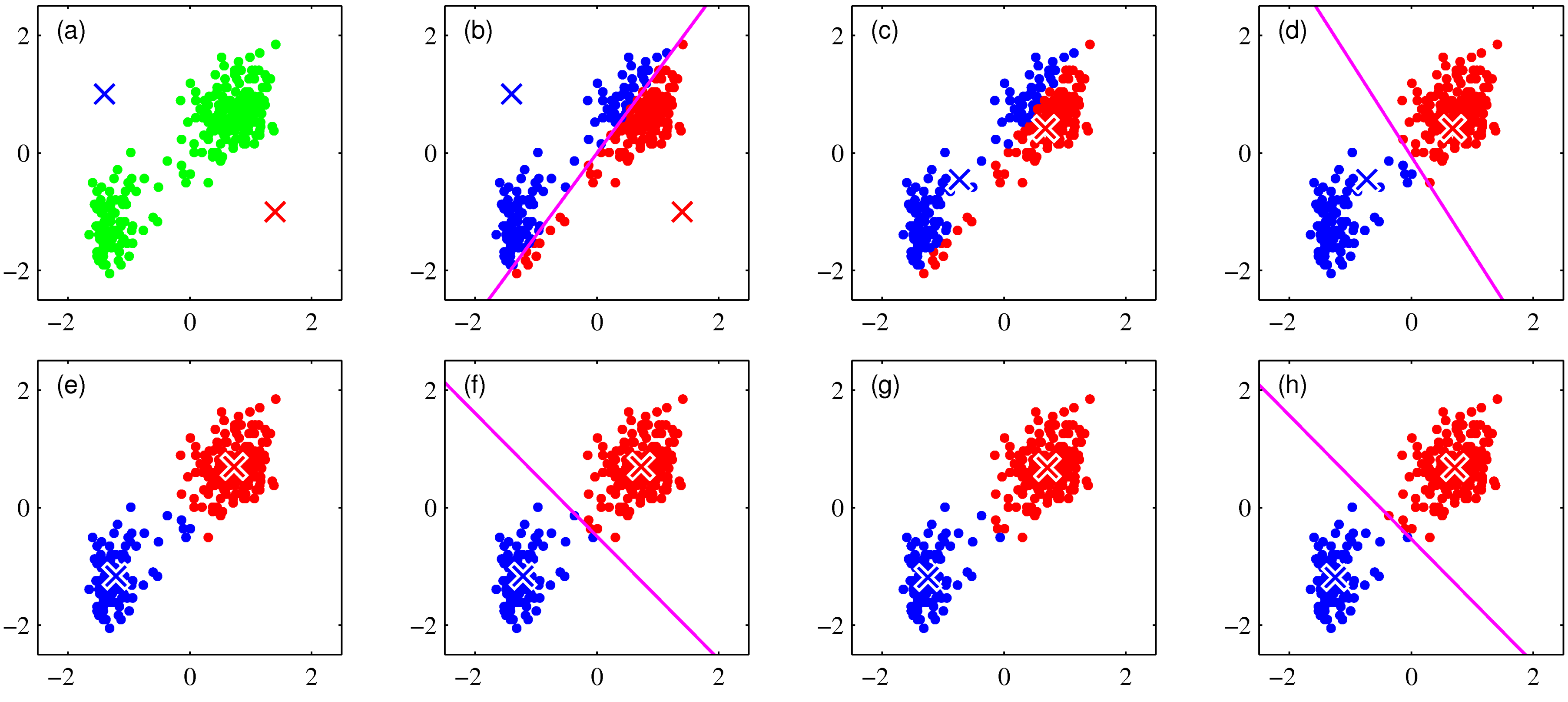
Differentiating for μ_k

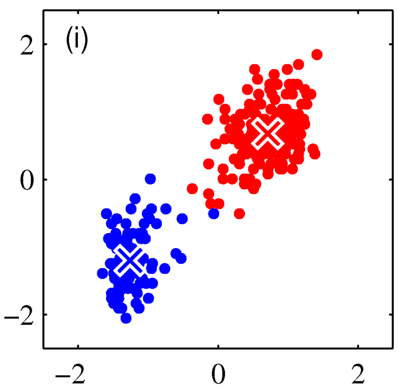
$$2 \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \tag{9.3}$$

which we can easily solve for $\boldsymbol{\mu}_k$ to give

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}. \tag{9.4}$$

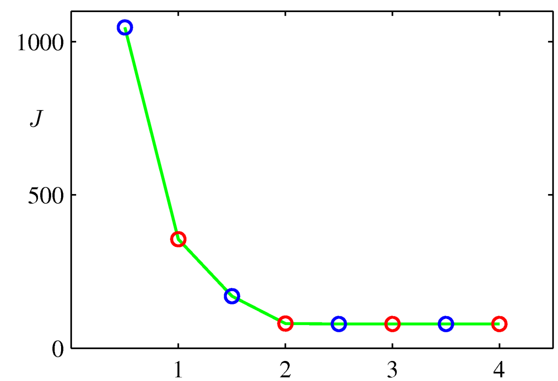
K-means



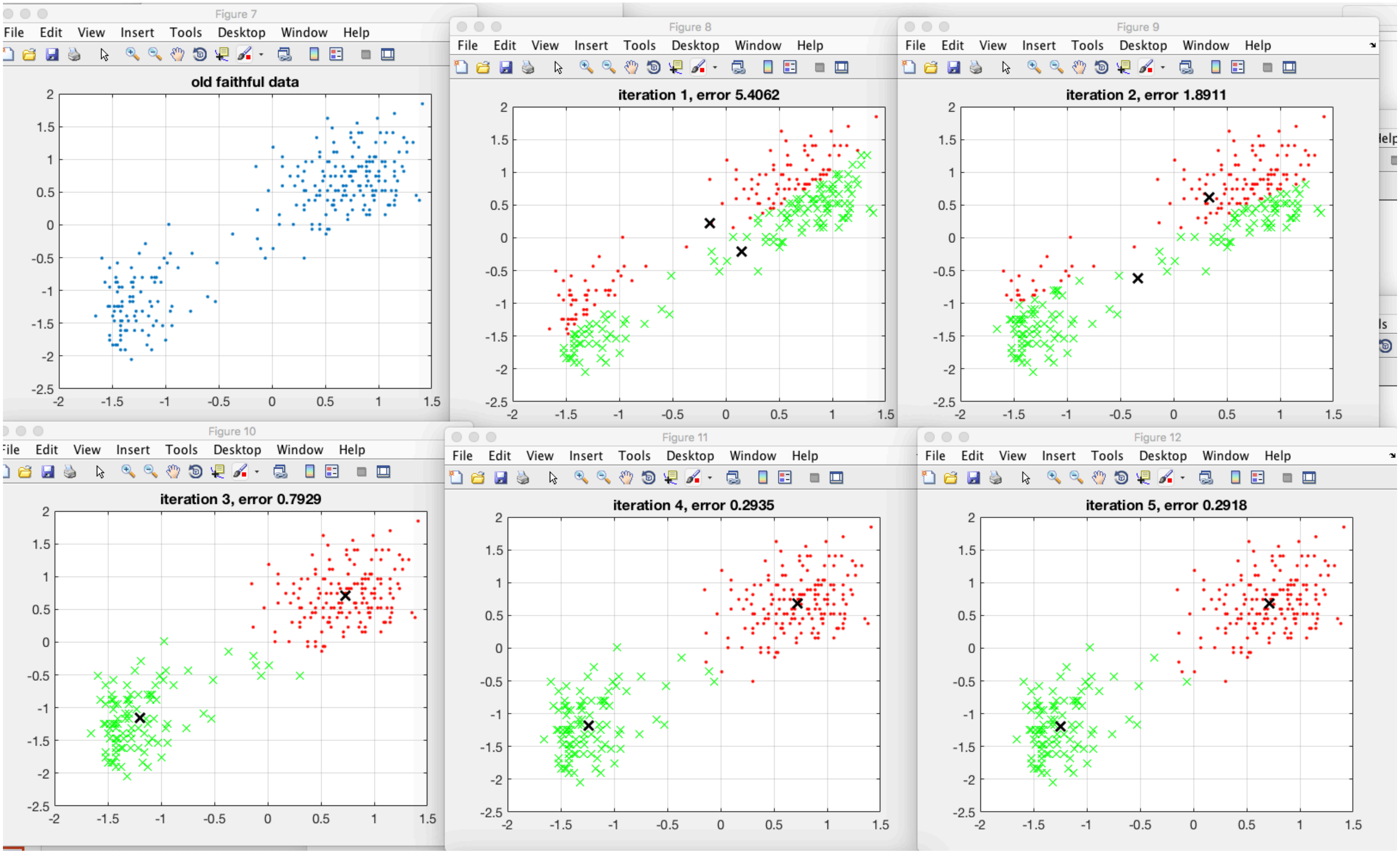


$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}.$$



Old Faithful, Kmeans from Murphy



Application of K-means to data compression: Vector Quantization

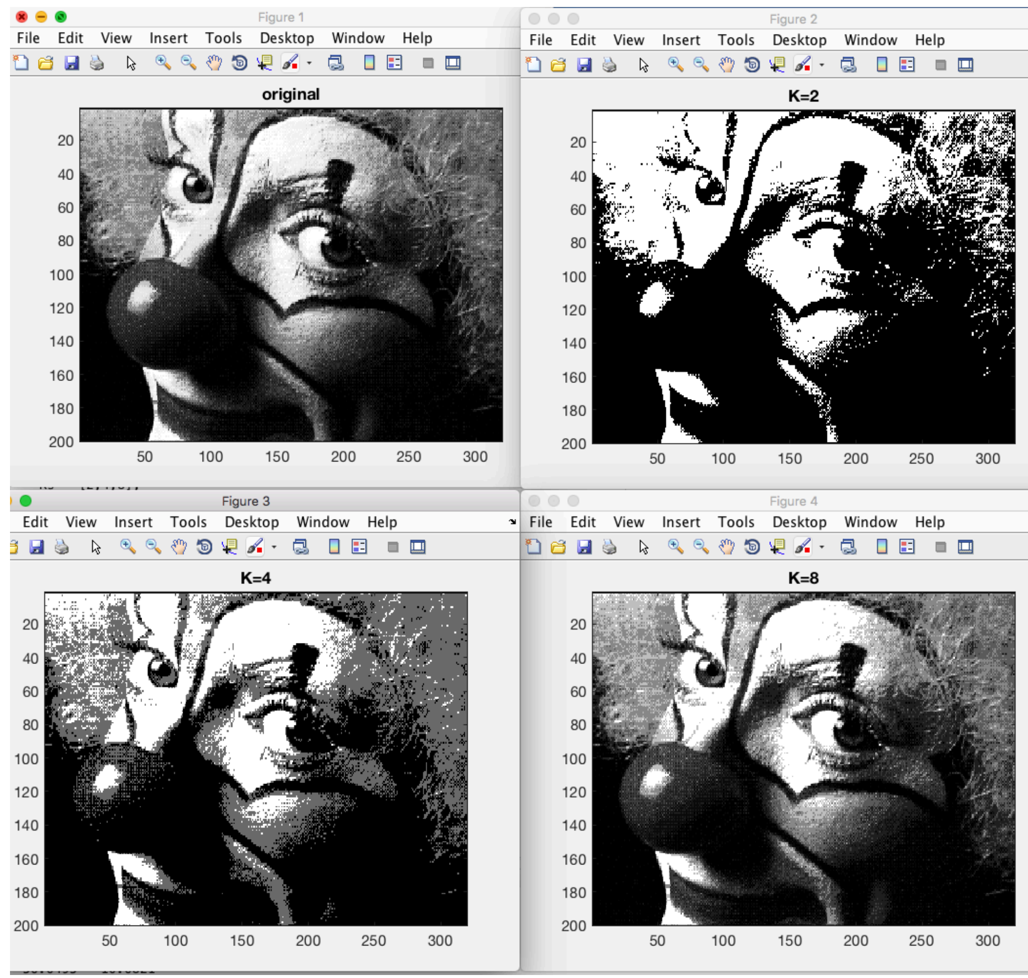
Each pixel $\mathbf{x}_i$ is represented
By codebook of K entries μ_k

Encode($\mathbf{x}_i$)=$\underset{k}{\operatorname{argmin}} ||x_i - \mu_k||$

Consider N=64k observations, of D=1 (b/w) dimension, C=8 bit

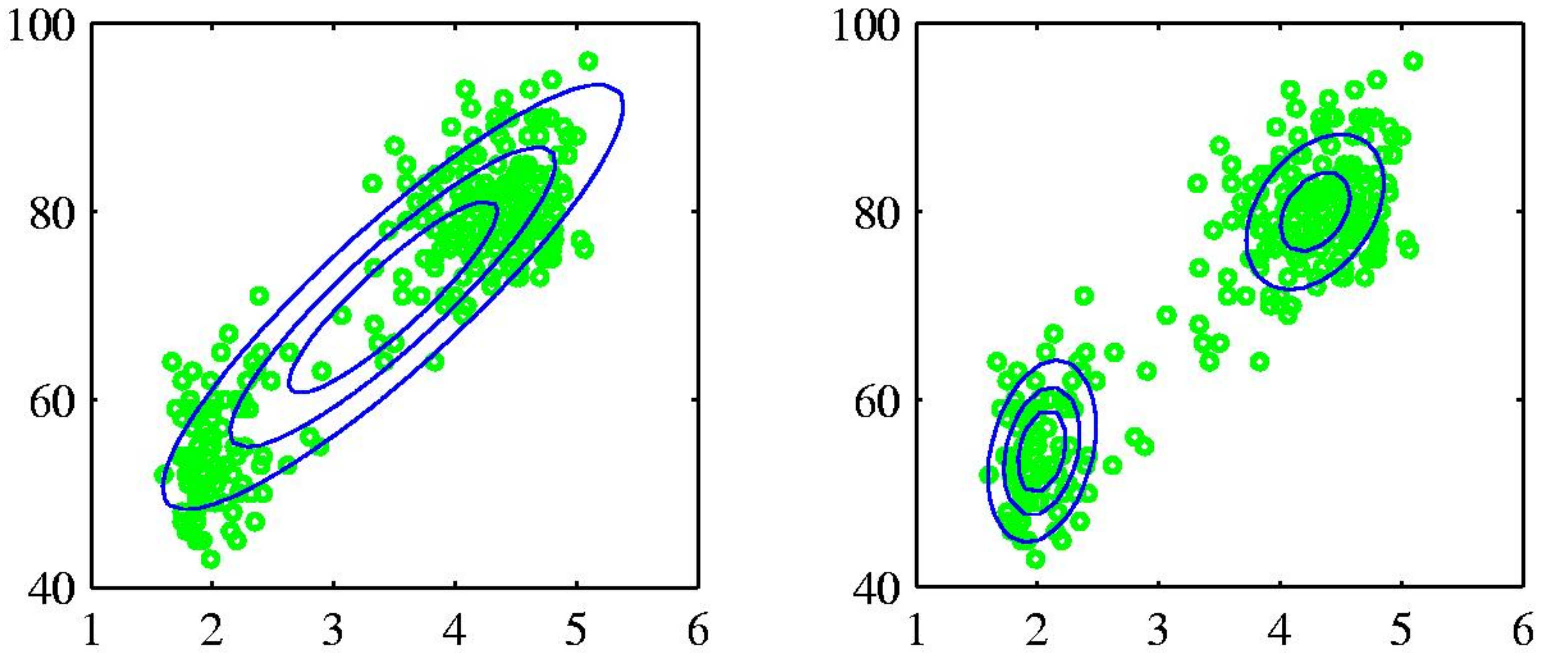
NC=513k

$N\log_2 K$+KC bits is needed
K=4 gives 128k a factor 4.



Mixtures of Gaussians (1)

Old Faithful geyser:
The time between eruptions has a bimodal distribution, with the mean interval being either 65 or 91 minutes, and is dependent on the length of the prior eruption. Within a margin of error of ±10 minutes, Old Faithful will erupt either 65 minutes after an eruption lasting less than 2 ½ minutes, or 91 minutes after an eruption lasting more than 2 ½ minutes.



Single Gaussian

Mixture of two Gaussians

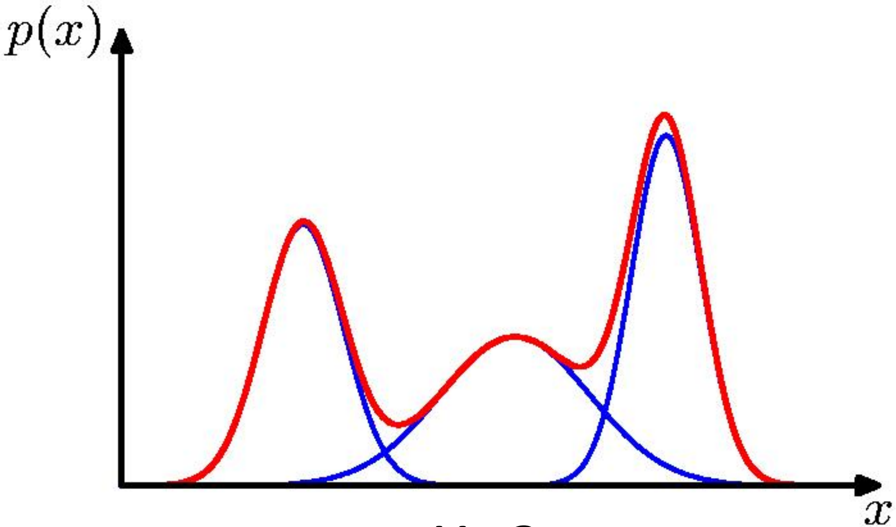
Mixtures of Gaussians (2)

Combine simple models into a complex model:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Component

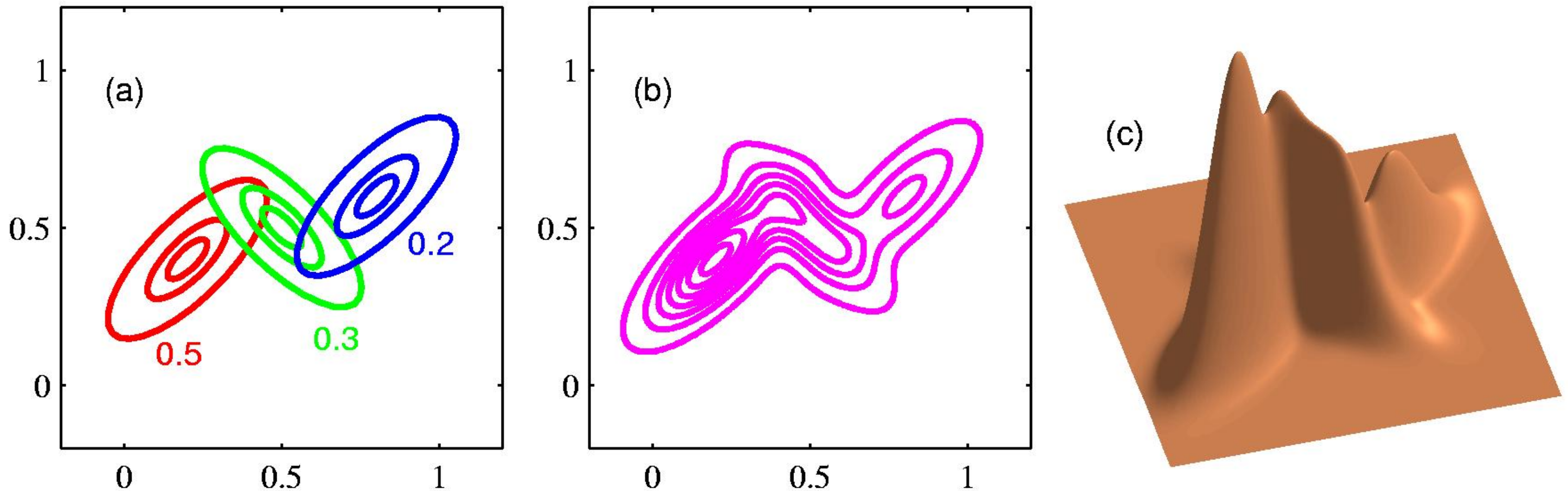
Mixing coefficient



K=3

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$

Mixtures of Gaussians (3)



Mixture of Gaussians

- Mixtures of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

- Expressed with latent variable **z**

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\boldsymbol{x}|z_k = 1) = N(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(z_k = 1) =$$

$$p(\boldsymbol{x}|\boldsymbol{z}) =$$

$$p(\boldsymbol{z}) =$$

$$p(\boldsymbol{x}, \boldsymbol{z}) =$$

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$

Want to estimate the latent variables for data X

- Probability of data given latent representation

$$p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) =$$

- Log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) =$$

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}.$$

Can't we just solve for the latent variables by maximimizing log likelihood?

- Log likelihood $$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}.$$

- Take derivative w.r.t. $\boldsymbol{\mu}_k$:

Can't we just solve for the latent variables by maximimizing log likelihood?

- Log likelihood $$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}.$$
- Take derivative w.r.t. $\boldsymbol{\mu}_k$:

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

"responsibility", from Bayes's rule:

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

Solving for μ_k, Σ_k

Take derivative w.r.t. $\boldsymbol{\mu}_k$:

Solving for π_k

Use Lagrange multipliers with constraint $\sum_{k=1}^{K} \pi_k = 1$

EM Gauss Mix

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.

2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{9.23}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

where

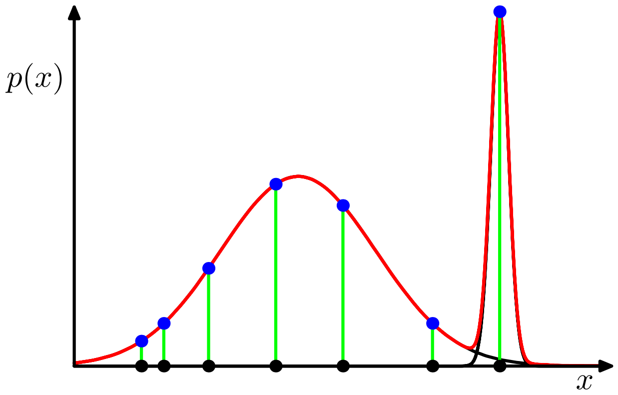
$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{9.28}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Important not to have singularities



General EM

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables $\mathbf{X}$ and latent variables $\mathbf{Z}$, governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \tag{9.32}$$

   where

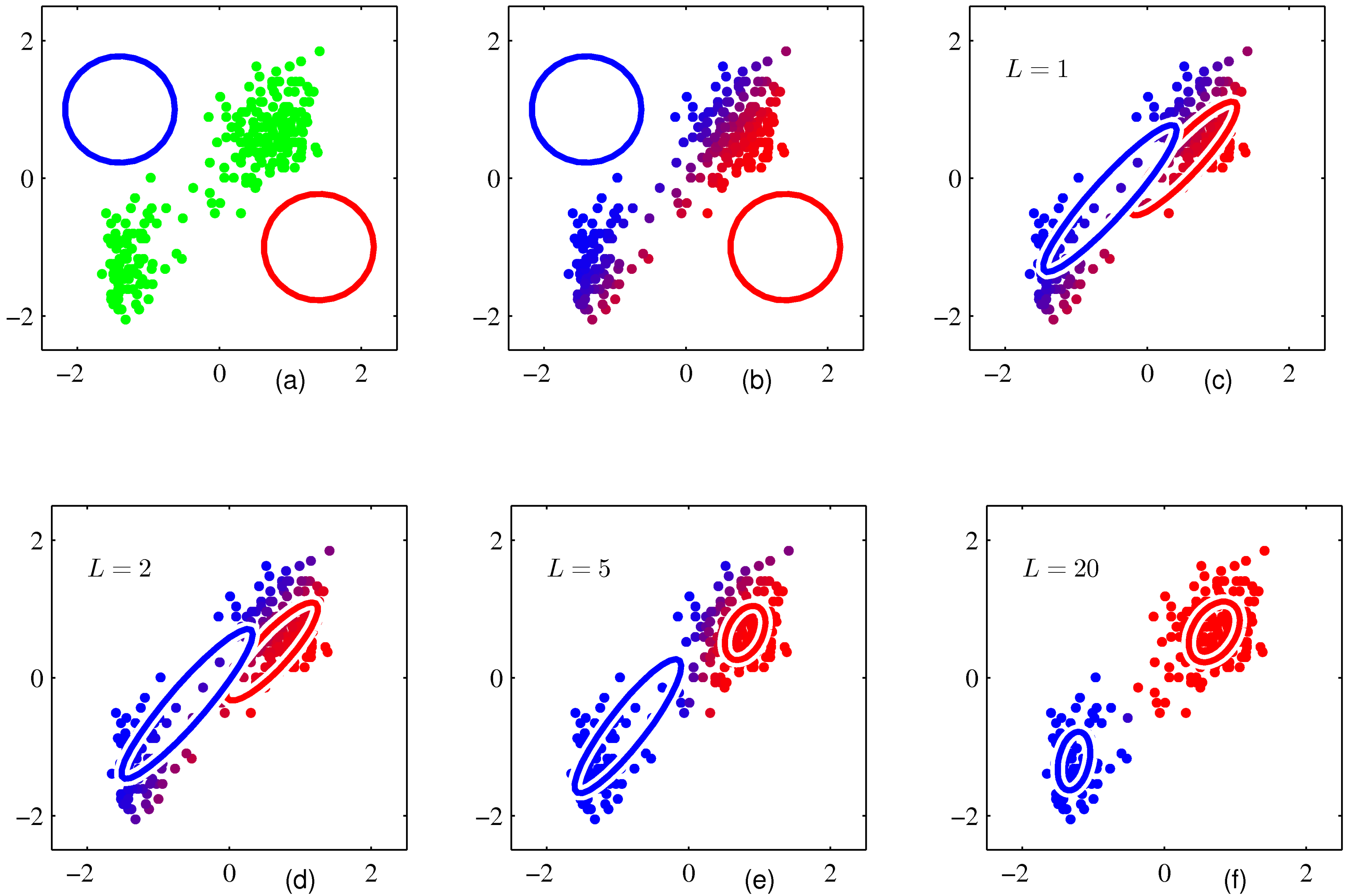
$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{9.33}$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \tag{9.34}$$

   and return to step 2.

Gaussian Mixtures



Kmeans and EM (9.3.2)

$$\boldsymbol{\Sigma}_k = \epsilon \boldsymbol{I}$$

$$p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp\left\{-\frac{1}{2\epsilon}\|\mathbf{x} - \boldsymbol{\mu}_k\|^2\right\}. \quad (9.41)$$

Whereby the responsibilities

$$\gamma(z_{nk}) = \frac{\pi_k \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2/2\epsilon\right\}}{\sum_j \pi_j \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2/2\epsilon\right\}}. \quad (9.42)$$

Becomes delta functions.

And the EM means approach the Kmeans

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n \quad (9.17)$$