Probabilistic Graphical Models in Computer Vision (IN2329)

Csaba Domokos

Summer Semester 2017

4. Mixture of Gaussians & Graph cut

Agenda for today's lecture

In the **previous lecture** we learnt about the *energy function* corresponding to the problem of **binary image segmentation**:

$$E(\mathbf{y};\mathbf{x}) = \sum_{i\in\mathcal{V}} -\log\phi_i(y_i;x_i) + \sum_{(i,j)\in\mathcal{E}} w_{ij}[\![y_i \neq y_j]\!]$$

Today we are going to learn about

- the definition of the *unary energies* $E_i(y_i;x_i)$. In fact we estimate *mixtures of Gaussans* f_{fg} and f_{bg} for the foreground and background, respectively, by making use of the *EM algorithm*.

$$E_i(y_i;x_i) = \begin{cases} -\log f_{\text{bg}}(x_i) & \text{if } y_i = 0 \\ -\log f_{\text{fg}}(x_i) & \text{otherwise} \end{cases} = \begin{cases} 0 & \text{if } y_i = 0 \\ -\log \frac{f_{\text{fg}}(x_i)}{f_{\text{bg}}(x_i)} & \text{otherwise}\ . \end{cases}$$

- *Graph cuts*, which will be applied to **minimize** the *energy function*.

Mixture of Gaussians

Multivariate Gaussian distribution

Assume a *D-dimensional random vector* $\mathbf{X} = (X_1,\dots,X_D)$, i.e. a vector whose components are random variables, with the joint density function

$$p(x_1,\dots,x_D) = \frac{1}{\sqrt{|2\pi\mathbf{\Sigma}|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\ .$$

$\mathbf{X}$ is said to have **multivariate Gaussian (or Normal) distribution** with parameters $\boldsymbol{\mu}\in\mathbb{R}^D$ and $\mathbf{\Sigma}\in\mathbb{R}^{D\times D}$ assuming that $\mathbf{\Sigma}$ is *positive definite*.

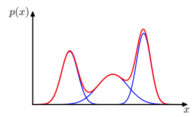
$\boldsymbol{\mu}$ is called the **mean vector** and $\mathbf{\Sigma}$ is called the **covariance matrix**. We often use the notation $\mathbf{X}\sim\mathcal{N}(\mathbf{x}\mid\boldsymbol{\mu},\mathbf{\Sigma})$ denoting $\mathbf{X}$ has Normal distribution.

Reminder. A symmetric $\mathbf{A}\in\mathbb{R}^{n\times n}$ matrix is said to be **positive definite**, if $\mathbf{u}^T\mathbf{A}\mathbf{u} > 0$ for all non-zero $\mathbf{u}\in\mathbb{R}^n$.

Mixture of Gaussians

Let us consider a superposition of K Gaussian densities

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



Mixture of three Gaussians
Source: C. Bishop: PRML, 2006.

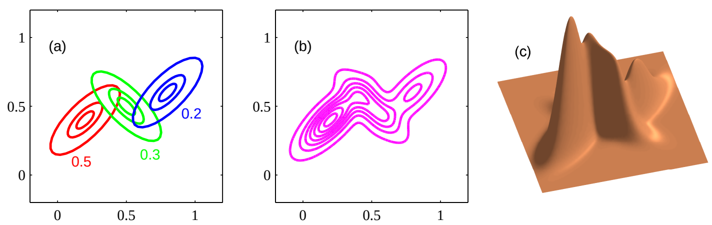
which is called a **mixture of Gaussians**.
The parameters π_k are called **mixing coefficients**.

$$\boxed{1} = \int_{\mathbb{R}^D} p(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{\mathbb{R}^D}\sum_{k=1}^{K}\pi_k\,\mathcal{N}(\mathbf{x}\mid\boldsymbol{\mu}_k,\mathbf{\Sigma}_k)\mathrm{d}\mathbf{x} = \boxed{\sum_{k=1}^{K}\pi_k}\ .$$

All the density functions are non-negative, hence $\pi_k \geqslant 0$ for $1\leqslant k\leqslant K$, therefore

$$0\leqslant\pi_k\leqslant 1 \quad \text{for all} \quad k=1,\dots,K\ .$$

Example: Mixture of three 2D Gaussians *



Iso-countours of each component

Iso-contours of $p(\mathbf{x}\mid\boldsymbol{\theta})$

Surface plot of $p(\mathbf{x}\mid\boldsymbol{\theta})$

Source: C. Bishop: Pattern Recognition and Machine Learning, 2006.

Parameter estimation *

We are interested in a method to find the *maximum likelihood estimator* of a **parameter** θ of a **probability distribution** $p(x\mid\theta)$.
Reminiscent of naming conventions:

$$p(\theta\mid x) = \frac{p(x\mid\theta)p(\theta)}{p(x)} \propto p(x\mid\theta)\ \ p(\theta)\ .$$

Posterior probability Likelihood Prior probability

We are given finite amount of **measurement** (i.e. observed data) $x_1,x_2,\dots$, and also know the probability distribution $p(x\mid\theta)$. The maximum likelihood estimate of θ is given by

$$\hat{\theta}\in\operatorname*{argmax}_{\theta} p(x\mid\theta)\ .$$

A possible solution: *Expectation–maximization algorithm*, which iteratively makes guesses about the data x, and iteratively maximizes $p(x\mid\theta)$ over θ.

Latent variables

We introduce a K-dimensional **binary random variable** $z \in \mathbb{B}^K$ having a *1-of-K representation*, i.e. $z_k = 1$ and all other elements are equal to 0. Let us define the *marginal distribution*

$$p(z_k = 1) = \pi_k \ ,$$

which is considered as the *prior probability* of picking the k^{th} component of a mixture of Gaussians. This distribution can be also written as a joint distribution

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \ .$$

Moreover, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$, i.e. *the likelihood*, can be written as

$$p(\mathbf{x} \mid z_k = 1) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \ , \quad \text{thus} \quad p(\mathbf{x} \mid \mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \ .$$

Responsibilities *

The **distribution of mixture of Gaussian**, specified by the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \in \mathbb{R}^K \times \mathbb{R}^{D \times K} \times \mathbb{R}D \times D \times K$, is given by

$$p(\mathbf{x}) \triangleq p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta}) = \sum_{\mathbf{z}} p(\mathbf{z} \mid \boldsymbol{\theta}) p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta})$$

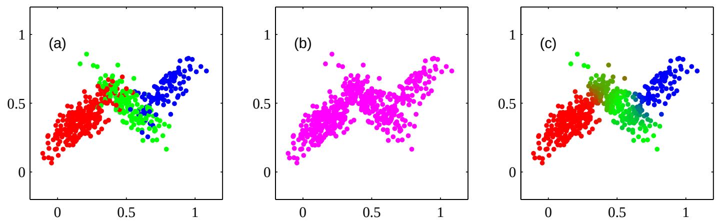
$$= \sum_{\mathbf{z}} \prod_{k=1}^{K} \left(\pi_k \ p(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)^{z_k} = \sum_{k=1}^{K} \pi_k \ \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \ .$$

The *posterior probabilities* $p(z_k = 1 \mid \mathbf{x})$, denoted by $\gamma_k(\mathbf{x})$, a.k.a. **responsibilities**, show the probability that a given sample $\mathbf{x}$ belongs to the k^{th} component.

$$\gamma_k(\mathbf{x}) \triangleq p(z_k = 1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid z_k = 1) p(z_k = 1)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid z_k = 1) p(z_k = 1)}{\sum_{l=1}^{K} p(z_l, \mathbf{x})}$$

$$= \frac{p(z_k = 1) p(\mathbf{x} \mid z_k = 1)}{\sum_{l=1}^{K} p(z_l = 1) p(\mathbf{x} \mid z_l = 1)} = \frac{\pi_k \ \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \ \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} \ .$$

Example: Mixture of three 2D Gaussians *



Samples from $p(\mathbf{z} \mid \boldsymbol{\theta}) p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) = p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta})$ — Samples from $p(\mathbf{x} \mid \boldsymbol{\theta})$ — Values of responsibilities coded by `RGB colors` $[\gamma_1(\mathbf{x}) \quad \gamma_2(\mathbf{x}) \quad \gamma_3(\mathbf{x})]$.

Source: C. Bishop: Pattern Recognition and Machine Learning, 2006.

Estimation of a mixture of Gaussians

Suppose we have a set of *i.i.d.* data samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ drawn from a mixture of Gaussians. The data set is represented by $\mathbf{X} \in \mathbb{R}^{N \times D}$.

The goal is to find the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, specifying the model from which the samples $\mathbf{x}_n$ have most likely been drawn. We may find the parameters which maximize the *likelihood function* $p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta})$. To simplify the optimization we use the **log-likelihood function** $\mathcal{L}(\boldsymbol{\theta})$

$$\boldsymbol{\theta}^* \in \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \stackrel{i.i.d.}{=} \operatorname*{argmax}_{\boldsymbol{\theta}} \ln \prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{z}_n \mid \boldsymbol{\theta})$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \ln \prod_{n=1}^{N} p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta}) p(\mathbf{z}_n \mid \boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \ln \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\pi_k \ \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)^{z_{nk}}$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left(\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \ .$$

Note that there is no closed-form solution for this model $\Rightarrow$ iterative solution.

Recall the EM algorithm *

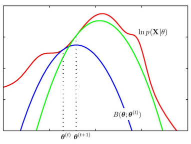
1: Choose an initial setting for the parameters $\boldsymbol{\theta}^{(0)}$
2: $t \rightarrow 0$
3: **repeat**
4: $\quad t \rightarrow t + 1$
5: $\quad$ **E step**. Evaluate $q^{(t-1)}(\mathbf{Z}) \triangleq p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{(t-1)})$
6: $\quad$ **M step**. Evaluate $\boldsymbol{\theta}^{(t)}$ given by

$$\boldsymbol{\theta}^{(t)} = \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) \ ,$$

where $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) \triangleq \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \mid \mathbf{X}, \boldsymbol{\theta}^{(t-1)}]$

$$= \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{(t-1)}) \ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$$

7: **until** convergence of either the parameters $\boldsymbol{\theta}$ or the log likelihood $\mathcal{L}(\boldsymbol{\theta}; \mathbf{X})$



Source: C. Bishop: PRML, 2006.

E step *

We need to calculate $p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$, which is calculated based on $p(\mathbf{z}_n \mid \mathbf{x}_n, \boldsymbol{\theta}^{\text{old}})$ for all $n = 1, \ldots, N$ as follows (see Exercise)

$$p(\mathbf{z}_n \mid \mathbf{x}_n, \boldsymbol{\theta}^{\text{old}}) = \frac{p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta}^{\text{old}}) \ p(\mathbf{z}_n \mid \boldsymbol{\theta}^{\text{old}})}{p(\mathbf{x}_n \mid \boldsymbol{\theta}^{\text{old}})}$$

$$= \frac{\pi_k \ \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

$$\triangleq \gamma_k(\mathbf{x}_n) \ .$$

Therefore, in the **E step** we need to calculate the *responsibilities* $\gamma_k(\mathbf{x}_n)$ for all data points $\mathbf{x}_n$ and components $k = 1, \ldots, K$.

M step for μ *

We have already known that $z_{nk} = \gamma_k(\mathbf{x}_n)$. Therefore, we may consider

$$\boldsymbol{\theta}^* \in \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_k(\mathbf{x}_n) \left(\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \ \text{ s.t. } \ \pi_k > 0 \ , \ \sum_{k=1}^{K} \pi_k = 1 \ .$$

Setting the derivative of $\mathcal{L}(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\mu}_k$ to 0, one can obtain that (see Exercise)

$$\frac{\sum_{n=1}^{N} \gamma_k(\mathbf{x}_n) \ \mathbf{x}_n}{\sum_{m=1}^{N} \gamma_k(\mathbf{x}_m)} = \boldsymbol{\mu}_k \ .$$

M step for Σ *

$$\boldsymbol{\theta}^* \in \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_k(\mathbf{x}_n) \left(\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \ \text{ s.t. } \ \pi_k > 0 \ , \ \sum_{k=1}^{K} \pi_k = 1 \ .$$

Setting the derivative of $\mathcal{L}(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\Sigma}_k$ to 0, one can obtain (see Exercise)

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{\sum_{m=1}^{N} \gamma_k(\mathbf{x}_m)} \ .$$

Remark: A $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ matrix, calculated as

$$\boldsymbol{\Sigma} = \frac{1}{N-1} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T \ ,$$

is called **sample covariance matrix** of data points $\{\mathbf{x}_n \in \mathbb{R}^D\}_{n=1}^{N}$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ is the **sample mean**.

M step for π *

To integrate the conditions on $\boldsymbol{\pi}$ we use the **Lagrange multiplier method**

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_k(\mathbf{x}_n)\big(\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\big) + \lambda(1 - \sum_{k=1}^{K} \pi_k) .$$

Setting the derivative w.r.t. π_k to 0, we obtain

$$\sum_{n=1}^{N} \frac{\gamma_k(\mathbf{x}_n)}{\pi_k} - \lambda = 0$$

$$\sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_k(\mathbf{x}_n) = \lambda \sum_{k=1}^{K} \pi_k \quad \Rightarrow \quad N = \lambda$$

therefore

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)}{N} .$$

The EM Algorithm for mixtures of Gaussians

1: Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k for all $k = 1, \ldots, K$

2: **repeat**

3: **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma_k(\mathbf{x}_n) = \frac{\pi_k \, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} \quad \text{for } 1 \leqslant n \leqslant N \text{ and } 1 \leqslant k \leqslant K .$$

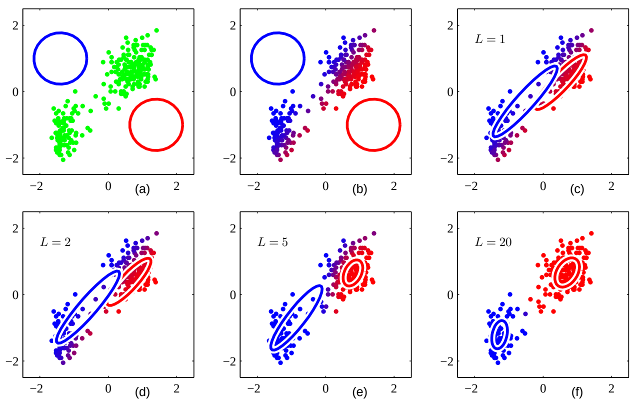
4: **M step**. Re-estimate the parameters $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for all $k = 1, \ldots, K$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)\mathbf{x}_n}{\sum_{m=1}^{N} \gamma_k(\mathbf{x}_m)} , \quad \boldsymbol{\Sigma}_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T}{\sum_{m=1}^{N} \gamma_k(\mathbf{x}_m)}$$

$$\pi_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma_k(\mathbf{x}_n)}{N}$$

5: **until** convergence of either the parameters $\boldsymbol{\theta}$ or the log likelihood $\mathcal{L}(\boldsymbol{\theta})$

Example *



Remarks

- The `EM algorithm` is **not limited** to mixture of Gaussians, but it can also be applied to *other probability distributions*.
- The algorithm does **not** necessary yield global maxima. In practice, it is restarted with *different initializations* and after convergence the result with the highest log-likelihood is chosen.
- One can think the `EM algorithm` as an **alternating minimization** procedure. Considering $f(\boldsymbol{\theta}, q)$ as the objective function, one iteration of the EM algorithm can be reformulated as

$$\text{E-step:} \quad q^{(t+1)} \in \underset{q}{\operatorname{argmax}} f(\boldsymbol{\theta}^{(t)}, q)$$

$$\text{M-step:} \quad \boldsymbol{\theta}^{(t+1)} \in \underset{\boldsymbol{\theta}}{\operatorname{argmax}} f(\boldsymbol{\theta}, q^{(t)})$$

Graph Cut

Graph cut

Assume a **weighted directed graph** $G = (\mathcal{V}, \mathcal{E}, c)$

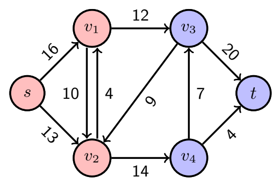
- $\mathcal{V} = \{1, \ldots, n\}$ is a finite set of nodes,
- $\mathcal{E} \subseteq \{(i,j) \in \mathcal{V} \times \mathcal{V} \mid i \neq j\}$ is the set of edges,
- $c : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a weight function. (For any $(i,j) \notin \mathcal{E}$, $c(i,j) = 0$.)

A **cut** $(\mathcal{S}, \mathcal{T})$ of G is a *disjoint* partition of $\mathcal{V}$ into $\mathcal{S}$ and $\mathcal{T} = \mathcal{V} \backslash \mathcal{S}$.

The **capacity** of the cut $(\mathcal{S}, \mathcal{T})$ is defined as

$$\text{cut}(\mathcal{S}, \mathcal{T}) = \sum_{(i,j) \in \mathcal{S} \times \mathcal{T}} c(i,j) .$$

Assume distinct nodes $s, t \in \mathcal{V}$, a cut $(\mathcal{S}, \mathcal{T})$ is called $s - t$ **cut** if $s \in \mathcal{S}$ and $t \in \mathcal{T}$.



The **minimum** $s - t$ **cut problem** is to find an $s - t$ cut with the lowest cost.

Example: $\text{cut}(\mathcal{S}, \mathcal{T}) = c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$.

Flow network

Flow network and flow

Let $G = (\mathcal{V}, \mathcal{E}, c)$ be a *directed weighted graph* with **non-negative** edge weights. Given two distinct nodes, a **source** s and a **sink** t, we call $(\mathcal{V}, \mathcal{E}, c, s, t)$ a **flow network**.

Let $(\mathcal{V}, \mathcal{E}, c, s, t)$ be a *flow network*. A function $f : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is called a **flow** if it satisfies the following properties:

1. **Capacity constraint**:

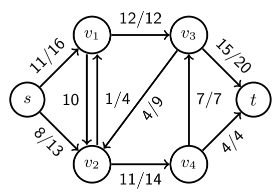
$$f(i,j) \leqslant c(i,j) \text{ for all } i, j \in \mathcal{V} .$$

2. **Skew-symmetry**:

$$f(i,j) = -f(j,i) \text{ for all } i, j \in \mathcal{V} .$$

3. **Flow conservation**:

$$\sum_{j \in \mathcal{V}} f(i,j) = 0 \text{ for all } i \in \mathcal{V} \backslash \{s, t\} .$$



The edges are labeled by $f(i,j)/c(i,j)$.

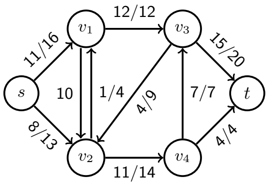
Only positive $f(i,j)$ are shown.

The value of a flow *

The **value** of a flow f is defined as

$$|f| \triangleq \sum_{(s,i)\in\mathcal{E}} f(s,i) = -\sum_{(i,t)\in\mathcal{E}} f(i,t) \,.$$

The **maximum-flow problem** is to find a *flow* f with the highest cost for a given flow network G.



The edges are labeled by $f(i,j)/c(i,j)$.
$|f| = 19$.

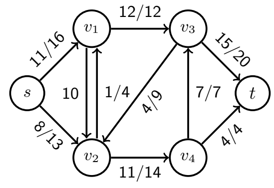
An equivalent definition of flows *

Now we give a *more intuitive definition* of *flows*. We will see that the previous definition is more helpful for the analysis of the *maximum-flow algorithm*.

Let $(\mathcal{V},\mathcal{E},c,s,t)$ be a flow network. A function $f:\mathcal{E}\to\mathbb{R}^+$ is called a **flow** if it satisfies the following two properties:

1. $f(i,j)\leqslant c(i,j)$ for all $(i,j)\in\mathcal{E}$.
2. For all $i\in\mathcal{V}\backslash\{s,t\}$

$$\sum_{(i,j)\in\mathcal{E}} f(i,j) = \sum_{(j,i)\in\mathcal{E}} f(j,i) \,.$$



The edges are labeled by $f(i,j)/c(i,j)$.

One can see that the two definitions of the flow are equivalent. (See Exercise)

Working with flows *

Let $G=(\mathcal{V},\mathcal{E},c,s,t)$ be a *flow network* and let f be a *flow* in G. We will use the following notation for $A,B\subseteq\mathcal{V}$

$$f(A,B) = \sum_{a\in A}\sum_{b\in B} f(a,b) \,.$$

It is easy to see that $|f| = f(\mathcal{V},\{t\})$, and $f(\{i\},\mathcal{V}) = 0$ for all $i\in\mathcal{V}\backslash\{s,t\}$ due to *flow conservation*.

Let $G=(\mathcal{V},\mathcal{E},c,s,t)$ be a *flow network* and let f be a *flow* in G. Then the following equalities hold:

i) For all $A\subseteq\mathcal{V}$, we have $f(A,A)=0$.
ii) For all $A,B\subseteq\mathcal{V}$, we have $f(A,B)=-f(B,A)$.
iii) For all $A,B,C\subseteq\mathcal{V}$ with $A\cap B=\emptyset$, we have

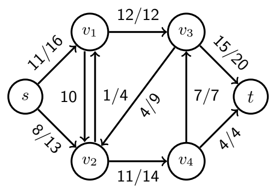
$$f(A\cup B,C) = f(A,C)+f(B,C) \text{ and } f(C,A\cup B) = f(C,A)+f(C,B) \,.$$

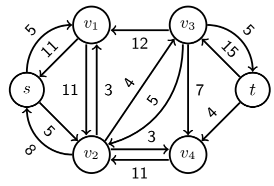
Proof. Exercise. □

Residual network

Let $G=(\mathcal{V},\mathcal{E},c,s,t)$ be a *flow network* and let f be a *flow* in G. The *weighted directed graph* $G_f=(\mathcal{V},\mathcal{E}_f,c_f)$ is called **residual network** of G induced by f, where

$$c_f(i,j) = c(i,j) - f(i,j) \,,$$
$$\mathcal{E}_f = \{(i,j)\in\mathcal{V}\times\mathcal{V} : c_f(i,j)>0\} \,.$$





A path p from s to t in G_f is called an **augmenting path**.

Max-flow–min-cut theorem

Let f be a *flow* in a *flow network* $G=(\mathcal{V},\mathcal{E},c,s,t)$. Then the following conditions are equivalent:

1) f is a maximal flow in G.
2) The residual graph G_f contains no augmenting paths.
3) $|f| = \text{cut}(\mathcal{S},\mathcal{T})$ for some $s-t$ cut of G.

Proof of Max-flow–min-cut theorem $1)\Rightarrow 2)$ *

Suppose that f is *maximum flow* in G, but and there exists an *augmenting path* p in the *residual graph* G_f.

The maximum amount by which we can **increase** the flow in p is the **residual capacity** of p, given by

$$c_f(p) = \min\{c_f(i,j) : (i,j) \text{ is on } p\} \,.$$

Furthermore, let us define $f_p:\mathcal{E}\to\mathbb{R}$ as follows:

$$f_p(i,j) = \begin{cases} c_f(p) & \text{if } (i,j) \text{ is on } p \\ -c_f(p) & \text{if } (j,i) \text{ is on } p \\ 0 & \text{otherwise.} \end{cases}$$

One can see that f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$. Therefore the flow $f+f_p$ has the value $|f|+|f_p|>|f|$, which contradicts the optimality of f.

Proof of Max-flow–min-cut theorem $2)\Rightarrow 3)$ *

Suppose that G_f *has no augmenting path*, i.e. s and t are *disconnected* in G_f. Define

$$\mathcal{S} := \{v\in\mathcal{V} : \text{ there exists a path from } s \text{ to } v \text{ in } G_f\} \,.$$

Obviously, $(\mathcal{S},\mathcal{T})$ is a *cut* of G, where $\mathcal{T} = \mathcal{V}\backslash\mathcal{S}$.

For each pair of $(i,j)\in\mathcal{S}\times\mathcal{T}$, we have $f(i,j)=c(i,j)$, otherwise $(i,j)\in\mathcal{E}_f$ would be held, which would imply that $j\in\mathcal{S}$.

One can see that the flow across $(\mathcal{S},\mathcal{T})$ is $|f|$:

$$\begin{aligned} f(\mathcal{S},\mathcal{T}) &\overset{\text{iii)}}{=} f(\mathcal{S},\mathcal{V}) - f(\mathcal{S},\mathcal{S}) \overset{\text{i)}}{=} f(\mathcal{S},\mathcal{V}) \overset{\text{iii)}}{=} f(\{s\},\mathcal{V}) + f(\mathcal{S}\backslash\{s\},\mathcal{V}) \\ &= f(\{s\},\mathcal{V}) = |f| \,. \end{aligned}$$

Therefore $|f| = f(\mathcal{S},\mathcal{T}) = \text{cut}(\mathcal{S},\mathcal{T})$.

Proof of Max-flow–min-cut theorem $3)\Rightarrow 1)$ *

Let f be a *flow* in G such that $|f| = \text{cut}(\mathcal{S},\mathcal{T})$. In general, for **any** flow f in G the following holds:

$$|f| = f(\mathcal{S},\mathcal{T}) = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{T}} f(i,j) \leqslant \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{T}} c(i,j) = \text{cut}(\mathcal{S},\mathcal{T}) \,.$$

Hence $|f| = \text{cut}(\mathcal{S},\mathcal{T})$ is maximal (equivalently $\text{cut}(\mathcal{S},\mathcal{T})$ is minimal). □

Ford-Fulkerson algorithm *

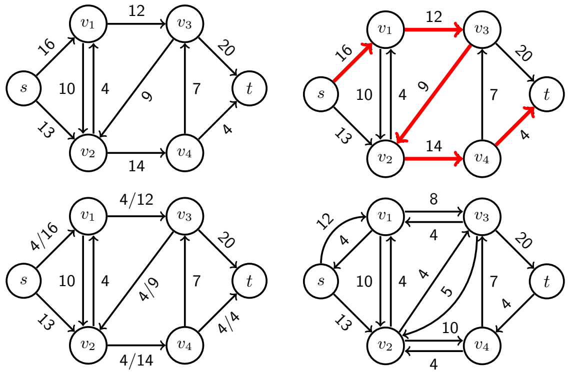
Input: A flow network $G = (\mathcal{V}, \mathcal{E}, c, s, t)$
Output: A minumum $s-t$ cut $(\mathcal{S}, \mathcal{T})$ of G

```
for all (i,j) ∈ E do
    f(i,j) ← 0 and f(j,i) ← 0
end for
while there exists a path p from s to t in the residual network G_f do
    c_f(p) ← min{c_f(i,j) : (i,j) is in p}
    for all (i,j) in p do
        f(i,j) ← f(i,j) + c_f(p)
        f(j,i) ← −f(i,j)
    end for
end while
S ← {v ∈ V : there exists a path from s to v in G_f} and T ← V\S
```

The complexity of this algorithm is $\mathcal{O}(|\mathcal{E}| \cdot \|f^*\|)$, where f^* is the value of the maximal flow.

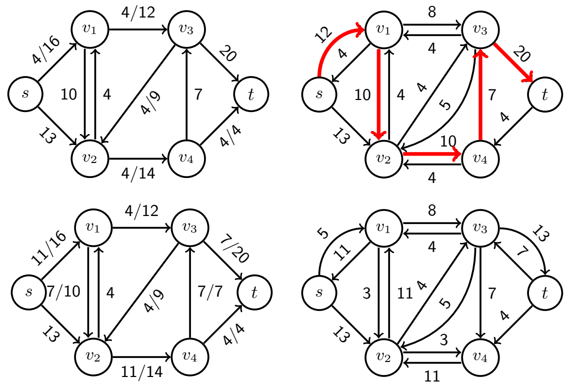
Example: iteration 1 *

Flow Residual network



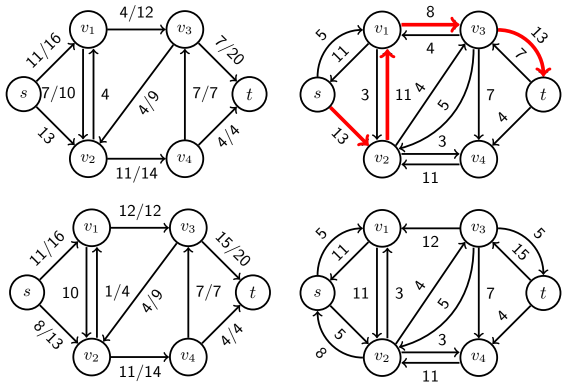
Example: iteration 2 *

Flow Residual network



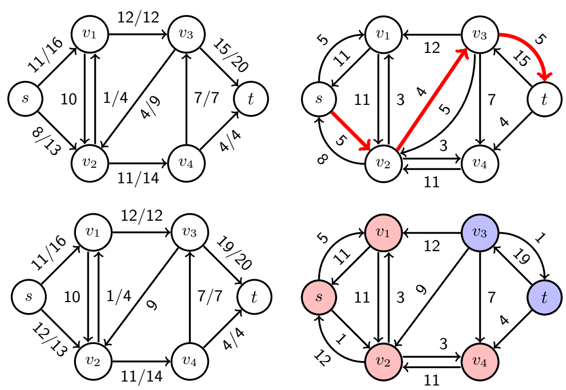
Example: iteration 3 *

Flow Residual network



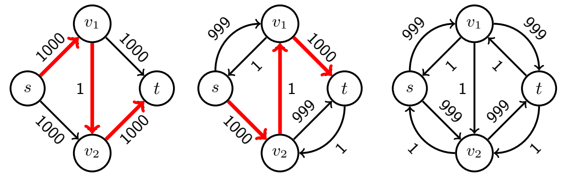
Example: iteration 4 *

Flow Residual network



A “bad” example *

Flow Residual network Residual network



Note that there exists an example, where the flow, computed by the Ford–Fulkerson algorithm, does not even converge to the maximum flow.

More precisely, if a flow network has integer $(\mathbb{N}_0)$ or rational $(\mathbb{Q}_0^+)$ capacities, then the Ford–Fulkerson algorithm terminates and it computes a maximum flow.

Edmonds–Karp algorithm

Input: A flow network $G = (\mathcal{V}, \mathcal{E}, c, s, t)$
Output: A minumum $s-t$ cut $(\mathcal{S}, \mathcal{T})$ of G

```
for all (i,j) ∈ E do
    f(i,j) ← 0 and f(j,i) ← 0
end for
while there exists a path p from s to t in the residual network G_f do
    p ← shorthestPath(G_f, s, t)
    c_f(p) ← min{c_f(i,j) : (i,j) is in p}
    for all (i,j) in p do
        f(i,j) ← f(i,j) + c_f(p)
        f(j,i) ← −f(i,j)
    end for
end while
S ← {v ∈ V : there exists a path from s to v in G_f} and T ← V\S
```

The complexity of this algorithm is $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{E}|^2)$. There exist more efficient algorithms for maximum flow calculation with complexity $\mathcal{O}(|\mathcal{V}|^2 \cdot |\mathcal{E}|)$ and $\mathcal{O}(|\mathcal{V}|^3)$.

Summary *

- **Max-flow–min-cut theorem** tells us that the minimum cut problem can be solved via maximum flow.
  These two problems are dual to each other, moreover strong duality holds.
- **Edmonds–Karp algorithm**: The `Ford-Fulkerson algorithm` becomes polynomial, if the shortest path is used as augmented path.

In the **next lecture** we will learn about

- Exact solution for *binary image segmentation* via *graph cut*
- *Boykov–Kolmogorov algorithm*
- Multi-label problem (e.g., stereo matching)

The EM algorithm for mixture of Gaussians

1. Frank Dellaert. The expectation maximization algorithm. Technical Report GIT-GVU-02-20, Georgia Institute of Technology, Atlanta, GA, USA, 2002
2. Shane M. Haas. The expectation-maximization and alternating minimization algorithms. Unpublished, 2002
3. Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006

Graph cut, Maximum flow

4. Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009
5. Tanmay Verma and Dhruv Batra. MaxFlow revisited: An empirical comparison of MaxFlow algorithms for dense vision problems. In Richard Bowden, John Collomosse, and Krystian Mikolajczyk, editors, *In Proceedings of British Machine Vision Conference*, pages 61.1–61.12, Surrey, UK, September 2012. BMVA Press