INF 5300 - 5.2.2014
Energy functions for segmentation/classification

Anne Schistad Solberg

- Bayesian spatial models for classification
- Markov random field models for spatial context
- Other segmentation techniques:
  - EM-clustering
  - Mean shift segmentation
  - Graph-based segmentation (briefly)

Curriculum

- 3.7.2 in Szeliski
- 5.3, 5.4 and briefl 5.5 in Szeliski

- Additional reading:
  - Will use the notation from ”Random field models in image analysis” by Dubes and Jain, Journal of Applied Statistics, 1989, pp. 131-154, except section 2.3 and 2.4.
  - For the extension to using other types of constraints, more details can be found in ”A Markov random field model for classification of multisource satellite imagery”, by Solbert, Taxt and Jain.

- Bayesian modelling using prior models to constrain the segmentation/classification results are commonly used.
- They imply statistical models for data/measurements, and prior information about the likelihood of observing similar class labels for neighboring pixels.
- Statistical models allows also modelling of the uncertainty associated with both estimates and measured class labels.

Bayes rule

- Common notation:
- Measurements y
- Class labels x
- Posterior probability given data p(x|y)
- Prior model p(x)
- p(y) is a normalizing constant to scale to 1.
- This can be written as the log-likelihood:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

$$-\log p(x|y) = -\log p(y|x) - \log p(x) + C$$

- The maximum aposteriori solution x given data y is the minimum of this negative log-likelihood. This is called the energy function

$$E(x,y) = E_d(x,y) + E_p(x)$$

- $E_d(x,y)$ is the data term
- $E_p(x)$ is the prior term
- x is the set of class labels for all pixels in the image x=[f(0,0),…..f(m-1,n-1)]
- y is the set of feature vectors for all pixels in the image y=[d(0,0),…,d(m-1,n-1)]
- For Markov random fields the prior term must be expressed as a sum of local pairwise interactions

$$E_p = \sum_{(i,j),(k,l)\in N} V_{i,j,k,l}(f(i,j), f(k,l))$$

- N is a set of pixels in a neighborhood

Binary MRFs

- Binary MRF are e.g. used for denoising scanned images.
- We have two classes, background and foreground.
- Energy function, data term:

$$Ed(i,j) = w\partial\left(f(i,j), d(i,j)\right)$$

Seeks correspondence between the input and output image

- Energy functions, regularization term:

$$E_p(i,j) = E_x(i,j) + E_y(i,j) = s\partial(f(i,j), f(i+1,j)) + s\partial(f(i,j)\, f(i,j+1))$$

Seeks correspondence between neighboring pixels in the output image

Ordinal-valued MRFs

- Ordinal: labels have implicit ordering
- Used e.g. for denoising gray-level images
- Energy function, data term:

$$E_d(i,j) = w(i,j)\rho_d(f(i,j) - d(i,j))$$

- Energy function, regularization term:

$$E_p(i,j) = s_x(i,j)\rho_p(f(i,j) - f(i+1,j)) + s_y(i,j)\rho_p(f(i,j) - f(i,j+1))$$

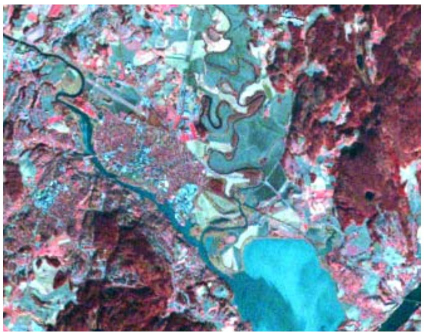
- Different forms of the penalty can be used, e.g. a hyper-Laplacian

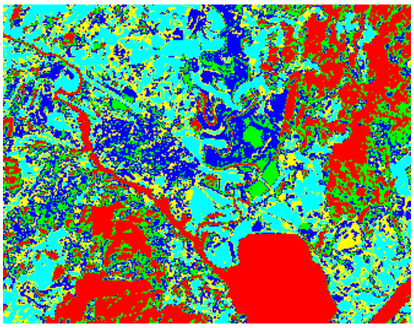
$$\rho_p(d) = |d|^p,\ p < 1$$

- If ρ is a quadratic function,the MRF is called a Gaussian MRF. ρ can also depend on the data.

Background – contextual classification

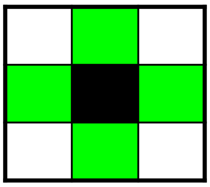
- An image normally contains areas of similar class
  - neighboring pixels tend to be similar.
- Classified images based on a non-contextual model often contain isolated misclassified pixels (or small regions).
- How can we get rid of this?
  - Majority filtering in a local neighborhood
  - Remove small regions by region area
  - Bayesian models for the joint distribution of pixel labels in a neighborhood.
- How do we know if the small regions are correct or not?
  - Look at the data, integrate spatial models in the classifier.



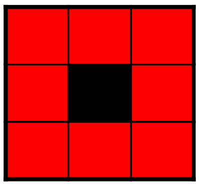


Relation between classes of neighboring pixels

- Consider a single pixel i.
- Consider a local neighborhood N_i centered around pixel i.
- The class label at position i depends on the class labels of neighboring pixels.
- Model the probability of class k at pixel i given the classes of the neighboring pixels.
- More complex neighborhoods can also be used.



4-neighborhood



8-neighborhood

Reminder – pixelwise classification

- Prior probabilities $P(\omega_r)$ for each class
- We have S classes.
- Bayes classification rule: classify a feature vector y_i (for pixel i) to the class with the highest posterior probability $P(\omega_r | y_i)$

$$P(\omega_r | y_i) = \max_{s=1,\ldots S} P(\omega_s | y_i)$$

- $P(\omega_s | y_i)$ is computed using Bayes formula

$$P(\omega_s \mid y_i) = \frac{p(y_i \mid \omega_s) P(\omega_s)}{p(y_i)}$$

$$p(y_i) = \sum_{s=1}^{R} p(y_i \mid \omega_s) P(\omega_s)$$

- $p(y_i | \omega_s)$ is the class-conditional probability density for a given class (e.g. Gaussian distribution)(corresponds to $p(y_i | x_i=\omega_s)$ here)
- This involves only one pixel i.

A Bayesian model for ALL pixels in the image

$Y = \{y_1,...,y_N\}$ Image of feature vectors to classify

$X = \{x_1,...x_N\}$ Class labels of pixels

- Classification consists choosing the class that maximizes the posterior probabilities for **ALL** pixels in the image

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{\sum_{\text{all classes}} \mathrm{P(Y \mid X)P(X)}}$$

- Maximizing P(X|Y) with respect to $x_1,.....x_N$ is equivalent to maximizing P(Y|X)P(X) since the denominator does not depend on the classes $x_1,.....x_N$.
- Note: we are now maximizing the class labels of ALL the pixels in the image simultaneously.
- This is a problem involving finding N class labels simuntaneously.
- P(X) is the prior model for the scene. It can be simple prior probabilities, or a model for the spatial relation between class labels in the scene.

Two kinds of pixel dependency

- Interpixel feature dependency:
  - Dependency between the feature vectors.
- Interpixel class dependency:
  - Dependency between class labels of neighboring pixels.

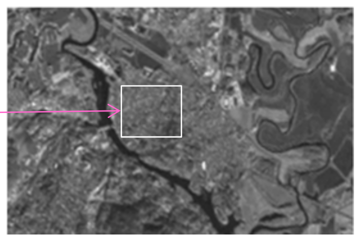
These two types will now be explained more formally.

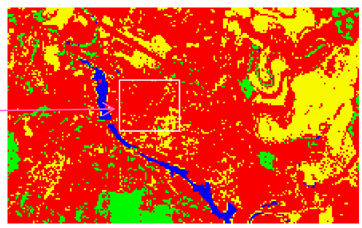
Model the joint distribution of the gray level of neighboring pixels $p(y_1,y_2|x_1,x_2)$

y_1,and y_2 are the feature vectors

x_1 and x_2 are the class labels

Model the probability for the class labels $p(x_1|x_2)$





Background: A little statistics

- Consider two events A and B.
- P(A) and P(B) is the probability of events A and B.
- P(B|A) is the conditional probability of B assuming A, and is defined as:

$$P(B \mid A) = \frac{P(B, A)}{P(A)}$$

$$P(B \mid A)P(A) = P(B, A) = P(A \mid B)P(B)$$

- P(A,B) is the joint probability of the two events A and B.

Interpixel feature dependency

- $P(y_1, y_2, \ldots y_N \mid x_1, x_2, \ldots, x_N)$ is generally the joint probability of observing feature vectors $y_1, \ldots y_N$ at pixel positions 1,...N given the underlying true class labels of the pixels.
- The observed feature vector for pixel *i* might depend on the observed feature vector for pixel *j* (neighboring pixels)
- We will not consider such models (If you are interested, see Dubes and Jain 1989).

- If the feature vector for pixel i is independent of all the other pixels, this can be simplified as:

$$P(y_1, \ldots y_N \mid X) = \prod_{i=1}^{N} P(y_i \mid x_i) = P(y_1 \mid x_1) \cdot P(y_2 \mid x_2) \cdots P(y_N \mid x_N)$$

Interpixel class dependency

- The class labels for pixel *i* depends on the class labels of neighboring pixels, but not on the neighbors' observed feature vectors.
  - Such models are normally used for classification.
  - Reasonable if the features are not computed from overlapping windows
  - Reasonable if the sensor does not make correlated measurement errors

- What this means is that when we estimate the class label of pixel *i*, we think that it will be valuable to know the class labels of the neighboring pixels (the image consists of regions with partly continuous class type).

Introduction to Markov random field modelling

- Two elements are central in Markov modelling:
  - Gibbs random fields
  - Markov random fields
- There is an analogy between Gibbs and Markov random fields as we soon will see.
- This will result in an energy function minimization problem.

Discrete Gibbs random fields (GRF) - Global model

- A discrete Gibbs random field gives a global model for the pixel labels in an image:

$$P(\mathbf{X} = \mathbf{x}) = e^{-U(\mathbf{x})/Z}$$

- **X** is a random variable, **x** is a realization of **X**.
- U(**x**) is a function called energy function
- Z is a normalizing constant

Neighborhood definitions (for MRFs)

- Pixel site *j* is a neighbor of site $i \neq j$ if the probability

$$P(X_i = x_i \mid \text{all } X_k = x_k, k \neq i)$$

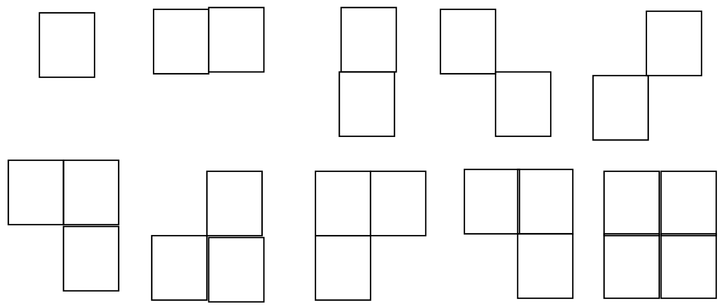
depends on x_j, the value of X_j.

- A clique is a set of sites in which all pairs of sites are mutual neighbors. The set of all cliques in a neighborhood is denoted Q.
- A potential function or clique function $V_c(\mathbf{x})$ is associated with each clique *c*.
- The energy function U(x) can be expressed as a sum of potential functions $U(\mathbf{x}) = \sum_{c \in Q} V_c(\mathbf{x})$
- 8-neighborhoods are commonly used, but more complex neighborhoods can also be defined.

Neighborhoods and cliques

2	1	2
1	t	1
2	1	2

1st and 2nd order neighbors of pixel t



Clique types for a 2nd order neighborhood

Remark: we normally only use cliques involving two sites. The model is then called a *pairwise interaction model. Then a clique is just a pair of neighboring pixels.*

Common simple potential functions

- Derin and Elliott's model:

$$V_c(\mathbf{x}) = \begin{cases} \xi_c \text{ if all sites in clique c have the same class} \\ -\xi_c \text{ otherwise} \end{cases}$$

- Ising's model:

$$V_c(\mathbf{x}) = \beta I(x_i, x_k)$$

  - β controls the degree of spatial smoothing
  - $I(c_i, c_k) = -1$ if $c_i = c_k$ and 0 otherwise
  - This corresponds to counting the number of pixels in the neighborhood assigned to the same class as pixel i.

- These two models are equivalent (except a different scale factor) for second order cliques

Discrete Markov random fields – local interaction models

- A Markov random field (MRF) is defined in terms of local properties.
- A random field is a discrete Markov random field with respect to a given neighborhood if the following properties are satisfied:
  1. Positivity: $P(\mathbf{X}=\mathbf{x})>0$ for all $\mathbf{x}$
  2. Markov property:

     $$P(X_t=x_t|\mathbf{X}_{S|t}=\mathbf{x}_{S|t})=P(X_t=x_t|\mathbf{X}_{\partial t} =\mathbf{x}_{\partial t})$$

     $S|t$ refers to all M pixel sites, except site t

     ∂t refers to all sites in the neighborhood of site t
  3. Homogeneity: $P(X_t=x_t|\mathbf{X}_{\partial t} =\mathbf{x}_{\partial t})$ is the same for all sites t.

Relationship between MRF and GRF

- A unique GRF exists for every MRF field and vice-versa if the Gibbs field is defines in terms of cliques of a neighborhood system.
- Advantage: a global model can be specified using local interactions only.

Back to the initial model…

$Y = \{y_1,\ldots,y_N\}$ Image of feature vectors to classify

$X = \{x_1,\ldots x_N\}$ Class labels of pixels

Task: find the optimal estimate **x'** of the true labels **x*** for all pixels in the image

- Classification consists choosing the class labels **x'** that maximizes the posterior probabilities

$$P(\mathbf{X}=\mathbf{x} \mid \mathbf{Y}=\mathbf{y}) = \frac{P(\mathbf{Y}=\mathbf{y} \mid \mathbf{X}=\mathbf{x})P(\mathbf{X}=\mathbf{x})}{\sum_{\text{all classes}} \mathrm{P}(\mathbf{Y}=\mathbf{y} \mid \mathbf{X}=\mathbf{x})\mathrm{P}(\mathbf{X}=\mathbf{x})}$$

- We assume that the observed random variables are conditionally independent:

$$P(\mathbf{Y}=\mathbf{y} \mid \mathbf{X}=\mathbf{x}) = \prod_{i=1}^{M} P(Y_i = y_i \mid X_i = x_i)$$

- We use a Markov field to model the spatial interaction between the classes (the term P(**X**=**x**)).

$$P(\mathbf{X}=\mathbf{x}) = \mathrm{e}^{-\mathrm{U}(\mathbf{x})/\mathrm{Z}}$$

$$U(\mathbf{x}) = \sum_{c \in Q} V_c(\mathbf{x})$$

$$V_c(\mathbf{x}) = \beta I(x_i, x_k)$$

- Rewrite $P(Y_i=y_i|X_i=x_i)$ as

$$P(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}) = \frac{1}{Z_1} e^{-Udata(Y|X)}$$

$$Udata(Y \mid X) = \sum_{i=1}^{M} -\log P(Y_i = y_i \mid X_i = x_i)$$

- Then, $P(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y}) = \frac{1}{Z_2} e^{-Udata(Y|X)} e^{-U(X)}$

- Maximizing this is equivalent to minimizing

$$U_{data}(Y \mid X) + U(X)$$

Udata(X|C)

- Any kind of probability-based classifier can be used, for example a Gaussian classifier with a k classes, d-dimensional feature vector, mean μ_k and covariance matrix Σ_k:

$$Udata(x_i \mid c_i) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\left(\left|\Sigma_k\right|\right) - \frac{1}{2} x_i^T \Sigma_k^{-1} x_i + \mu_k^T \Sigma_k^{-1} x_i - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k$$

$$\propto -\frac{1}{2} x_i^T \Sigma_k^{-1} x_i + \mu_k^T \Sigma_k^{-1} x_i - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\log\left(\left|\Sigma_k\right|\right)$$

Finding the labels of ALL pixels in the image

- We still have to find an algorithm to find an estimate **x'** for all pixels.
- Alternative optimization algorithms are:
  - Simulated annealing (SA)
    - Can find a global optimum
    - Is very computationally heavy
  - Iterated Conditional Modes (ICM)
    - A computationally attractive alternative
    - Is only an approximation to the MAP estimate
  - Maximizing the Posterior Marginals (MPM)

- We will only study the ICM algorithm, which converges only to a local minima and is theoretically suboptimal, but computationally feasible.

ICM algorithm

1. Initilalize x_t, t=1,...N as the non-contextual classification by finding the class which maximize $P(Y_t=y_t|X_t=x_t)$.
2. For all pixels t in the image, update $\hat{x}_t$ with the class that maximizes

$$P(Y_t = y_t \mid X_t = x_t) P(X_t = x_t \mid \mathbf{X}_{\partial t} = \hat{x}_{\partial t})$$

3. Repeat 2 n times

Usually <10 iterations are sufficient

ICM in detail

Initilalize x_t, t=1,...N as the non-contextual classification by finding the class which maximize $P(Y_t=y_t|X_t=x_t)$, assign it to classified_image(i,j)

```
For iteration k=1:maxit do
        For i=i:N,j=1:N (all pixels) do
                minimum_energy=High_number;
                For class s=1:S do
                        Udata = -log (P(Y_t=y_t|X_t=x_t))
                        Ucontxt=0;
                        nof_similar_neighbors=0;
                        for neighb=1:nof_neighbors
                            if (classified_image(neighb)=s) //neighbor and s of same class
                                ++nof_similar_neighbors;
                        Ucontxt = -beta*nof_similar_neighbors;
                        energy = Udata + Ucontxt;
                        if (energy < minimum_energy)
                                minimum_energy = energy;
                                bestclass = s;
                new_classified_image(i,j) = bestclass;
                if (new_classified_image(i,j)!=classified_image(i,j))
                        ++nof_pixels_changed;
        if nof_pixels_changed<min-limit
                break;
```


ICM comments

- $P(Y_t=y_t|X_t=x_t)$ can be computed based on various software packages, stored, and used in the ICM algorithm.
- For an image with S classes, this can be stored in a S-band image.
- For each iteration, only the labels x_i change.

  - Why should you use a temporal array to store the updated labels at iteration k, and a separate array for the labels at the next iteration k+1?
    - Hint: try this on a checkerboard image.

How to choose the smoothing parameter β

- β controls the degree of spatial smoothing
- β normally lies in the range $1 \leq \beta \leq 2.5$
- The value of β can be estimated based on formal parameter estimation procedures (heavy statistics, but the best way!)
- Another approach is to try different values of β, and choose the one that produces the best classification rate on the training data set.

An energy function for preserving edges

- When β is large, the Ising model tends to smooth the image across edges.
- We can add another energy term to penalize smoothing edges by introducing line processes (Geman and Geman 1984).
- Consider a model where edges can occur between neigbhboring pixels and let l(i,j) represent if there is an edge between pixel i and pixel j :

Line processes

- l(i,j)=0 if there is no edge between pixel i and j, and 1 of there is an edge
- There is an edge if pixels i and j belong to diffent classes, if $c_i \neq c_j$
- We can define an energy function penalizing the number of edges in a neihborhood

$$U_{line}(i) = \beta_l \sum_{k \in N_i} l(i, j)$$

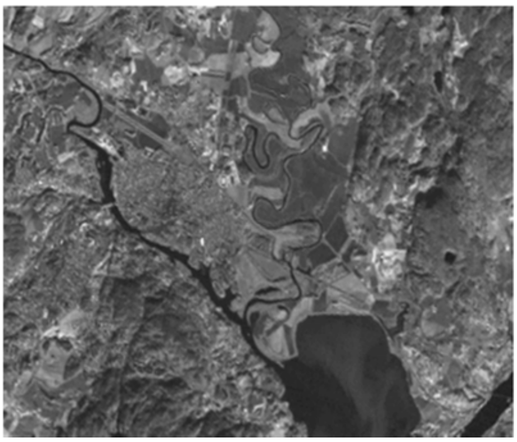
- and let

$$U = Udata(X \mid C) + Uspatial(C) + U_{line}(C)$$

- This will smooth the image, but preserve edges much better.

Test image 1

- A Landsat TM image
- Five classes:
  - Water
  - Urban areas
  - Forest
  - Agricultural fields
  - Vegetation-free areas
- The image is expected to be fairly well approximated by a Gaussian model

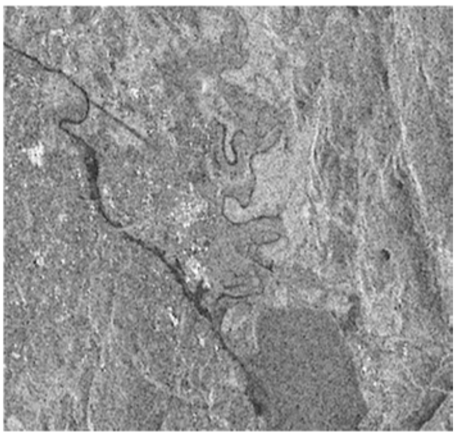


Classification results, Landsat TM image

Method	Training data, Noncontextual	Test data, Noncontextual	Test data, contextual
Gaussian	90.1	90.5	96.3
Multilayer perceptron	89.7	90.0	95.5

Data set 2

- ERS SAR image
- 5 texture features from a lognormal texture model used
- 5 classes:
  - Water
  - Urban areas
  - Forest
  - Agricultural fields
  - Vegetation-free areas



Classification results, SAR image

Method	Training data, Noncontextual	Test data, Noncontextua l	Test data, contextual
Gaussian	63.7	63.4	67.1
Multilayer perceptron	66.6	66.9	70.8
Tree classifier	70.3	65.0	76.1

More on different energy functions

- MRF local energy terms can be used to model other types of context to (see Solberg 1996)
  - Multitemporal classification
  - Consistency with an existing map or previous classification
  - Consistency with other types of GIS data

An energy function for fusion with a thematic map

- Assume that a map or previous classification of the scene exists.
- This map can be partly inaccurate and needs to be updated.
- Let $C^g=\{c^g_1,\ldots,c^g_N\}$ be an old map of the area.
- Consider a set of S different classes. The probability for a change from class s_1 to s_2 can be specified as a table of transitions (next page) $\Pr(x_i|c^g_i)$.
- An additional energy term can be

$$U_G = -\beta_g \sum_{neighborhood} \Pr(x_i \mid c_i^g)$$

Example of allowed transitions

	Urban	Forest	Agricultural	Bare soil	Water
Urban	1.0	0.0	0.0	0.0	0.0
Forest	0.1	0.7	0.1	0.1	0.0
Agricultral	0.1	0.1	0.7	0.1	0.0
Bare soil	0.1	0.1	0.1	0.7	0.0
Water	0.0	0.0	0.0	0.0	0.1

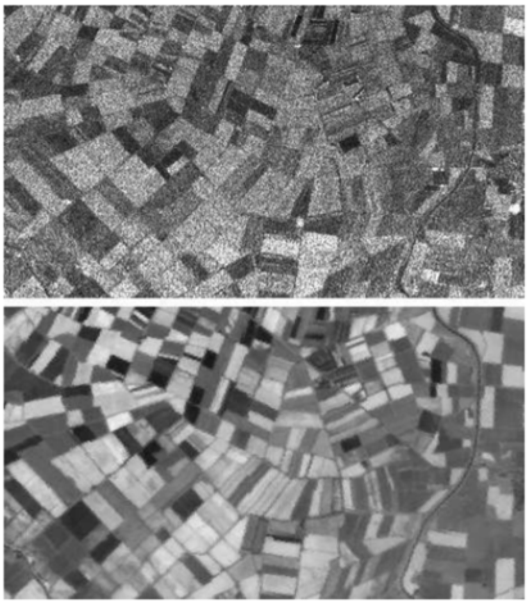
An energy term for crop ownership data

- På norsk: jordskiftekart eller bestandskart av grenser regioner som er en naturlig enhet og som ofte drives likt. Let a line process l(i,j) define if pixels i and j are assigned to the same class (l(i,j)=0) or not (l(i,j)=1) in the class label image.
- Let the crop ownership map be represented by a line process.
- An edge site in this map indicates if the two pixels (i,j) it involves are on the same region ($l_g(i,j)=0$) or not (l(i,j)=1).
- An energy term seeking consistency with the crop ownership map is:
- $$U_{map} = -\beta_{map} \sum_{neighborhood} W(x_i, l(i,j))$$

  where $W(x_i, l(i,j)) = 0$ if $l(i,j) = l_g(i,j)$ and 1 otherwise

Example agricultural classification

- Optical (Landsat) and SAR image of agricultural site.
- Classes: wheat, sugar beet, potatoes, carrots, grass, stubble, bare soil.
- Field border map also available.



SAR on top, Landsat bottom

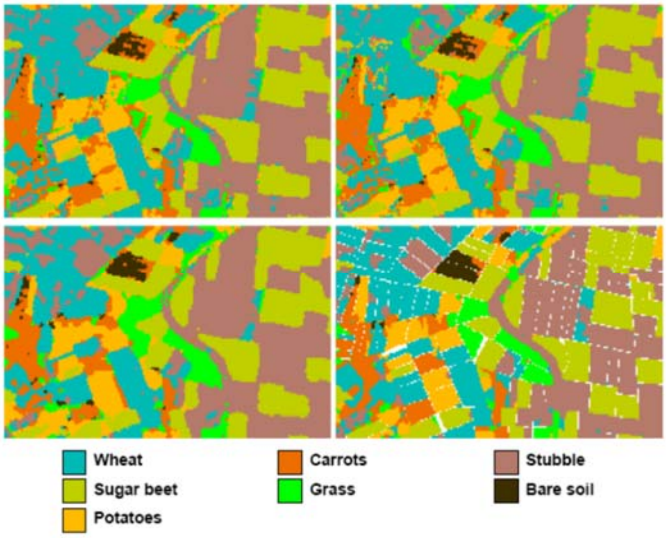
Example agricultural classification

Result based on only optical

Result based on optical+SAR No context

Result based on optical+SAR MRF

Result based on optical+SAR MRF with field borders Field borders overlaid in white



SAR	Optical	Combined – noncontextual	Combined – MRF	Combined – MRF with field border map
59.9	70.3	71.3	73.0	79.6

Segmentation methods covered

- Watershed segmentation (INF 4300)
- Split-and-merge/region growing (INF 4300)
- K-means clustering (INF 4300)
  - We extend this to mixtures of Gaussian now
- Mean shift segmentation
- Graph-cut algorithms

K-means clustering (Repetition)

- Note: K-means algorithm normally means ISODATA, but different definitions are found in different books
- K is assumed to be known

1. Start with assigning K cluster centers
   – k random data points, or the first K points, or K equally spaces points
   – For k=1:K, Set μ_k equal to the feature vector x_k for these points.
2. Assign each object/pixel x_i in the image to the closest cluster center using Euclidean distance.
   - Compute for each sample the distance r2 to each cluster center:

     $$r^2 = (x_i - \mu_k)^T (x_i - \mu_k) = \|x_i - \mu_k\|^2$$

   - Assign x_i to the closest cluster (with minimum r value)
3. Recompute the cluster centers based on the new labels.
4. Repeat from 2 until #changes<limit.

ISODATA K-means: splitting and merging of clusters are included in the algorithm

Clustering by mixtures of Gaussians

- Euclidean distance can be replaced by Mahalanobis distance from point x_i to cluster center k:

$$d(x_i, \mu_k, \Sigma_k) = (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)$$

- We could just modify the K-means algorithm to use this measure after the first iteration.
- Mixtures of Gaussian considers that samples can be softly assigned to several nearby cluster centers:

$$p(x \mid \pi_k, \mu_k, \Sigma_k) = \sum_k \pi_k \frac{1}{|\Sigma_k|} e^{-d(x, \mu_k, \Sigma_k)}$$

- π_k is the mixing coefficient for cluster with mean μ_k and covariance Σ_k.

The EM-algoritm for clustering

- The EM-algoritm iteratively estimate the mixture parameters:

1. Expectation step (E-step): compute

$$z_{ik} = \frac{1}{Z_i} \pi_k \frac{1}{|\Sigma_k|} e^{-d(x,\mu_k,\Sigma_k)} \text{ with } \sum_k z_{ik} = 1$$

An estimate of the probability that xi belongs to the kth Gaussian

2. Maximation stage (M-step): update

$$\mu_k = \frac{1}{N_k} \sum_i z_{ik} x_i$$

$$\Sigma_k = \frac{1}{N_k} \sum_i z_{ik} (x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}$$

Mean shift clustering/segmentation algorithm

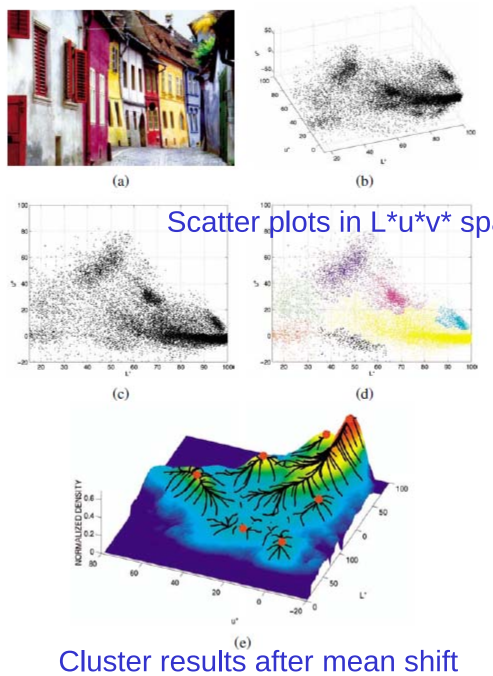
- K-means and mixtures of Gaussian are based on a parametric probability function.
- An alternative is to use a non-parametric smooth function that fits the data.
- The mean shift algoritms efficiently finds peaks in a distribution without estimating the entire distribution.
- It can be seen as the «inverse» of the watershed algorithm, which clims downhill.

The mean shift - background

- To estimate a density function for the scatter plots, we could use a Parzen window estimator, which smooths the data by convolving it with a kernel k() of width h:

$$f(x) = \sum_i K(x - x_i) = \sum_i k\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

- When we have computed f(x), we could find peaks by gradient descent.
- Drawback: does not work well with sparse data points.
- Solution: finding the peaks WITHOUT estimating the entire distribution.



Scatter plots in L*u*v* space

Cluster results after mean shift clustering, peaks marked in red

Mean shift segmentation

- Multiple restart gradient descent algorithm: start at many points y_k and take a step up-hill from these point.
- The gradient of f is (g(r)=-k'(r)):

$$\nabla f(x) = \sum_i (x_i - x) G(x - x_i) = \sum_i (x_i - x) g\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

- This can be written as

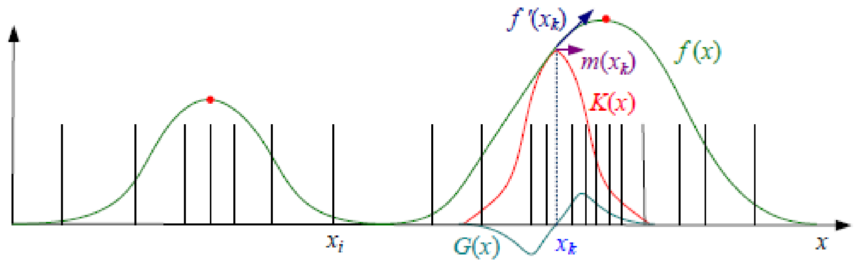
$$\nabla f(x) = \left[\sum_i G(x - x_i)\right] m(x)$$

$$m(x) = \frac{\sum_i x_i G(x - x_i)}{\sum_i G(x - xi)} - x$$

- The current estimate of y_k is replaced with its locally weighted mean:

$$y_{k+1} = y_k + m(y_k) = \frac{\sum_i x_i G(y_k - x_i)}{\sum_i G(y_k - x_i)}$$

Illustration of mean shift



- The kernel K is convolved with the image.
- The derivative of the kernel is computed by convolving the image with the derivative of the kernel
- The mean shift change m(x) is found from the derivative f′(x)

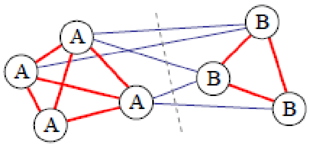
- Simple but slow algorithm: start a separate mean shift estimate y at every input point x, and iteration until only small changes.
- Faster: start at random points.
- Including location information:
  - Add the coordiates x_s= (x,y)in the kernel:

$$K(x_j) = k\left(\frac{\|x_r\|^2}{h_r^2}\right)k\left(\frac{\|x_s\|^2}{h_s^2}\right)$$

  - x_r is the spectral feature vector and h_r and h_s the bandwidth in the spectral and spatial domain.
  - The effect of this is that the algoritm step will take both spectral and spatial information and e.g. use larger steps in space between pixels with similar color.

Normalized cut segmentation

- Many segmentation algorithms are based on graphs and finding the cut of a graph that minimizes a criteria.



(a)

	A	B	sum
A	$assoc(A, A)$	$cut(A, B)$	$assoc(A, V)$
B	$cut(B, A)$	$assoc(B, B)$	$assoc(B, V)$
sum	$assoc(A, V)$	$assoc(B, v)$	

(b)

- All pixels are joined by edges with weights w_{ij} that measure their similarity.
- The cut between group A and B is the sum of all weights being cut:

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

- Minimizing the cut is not an optimal because the solution then would be to have one cluster per pixel.

Normalized cut

- Normalized cut:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

- assoc(A,A) is the sum of all weights within cluster A.
- assoc(A,V) = assoc(A,A)+cut(A,B)is the sum of all weights associated with pixels in cluster A.
- Let $W=[w_{ij}]$ be all weights sorted so that all nodes in A comes first and nodes in B second.
- To find the cut, Shi and Malik suggested using a real-valued assignment of nodes to groups.
- x is an indicator vector ($x_i=1$ if $x \in A$ and -1 if $x \in B$)

- Let d=W1 be the row sums of W
- Let D=diag(d)
- Minimizing the normalized cut is equivalent to minimizing:

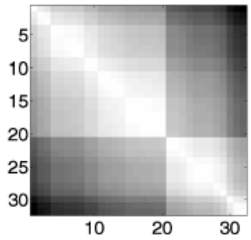
$$\min_y \frac{y^T(D-W)y}{y^T D y}$$

- y=((1+x)-b(1-x))/2 is a vector consisiting of 1s and –bs such that yd=0.
- This is a generalized eigenvalue system

$(D-W)=\lambda Dy$ which can be written as

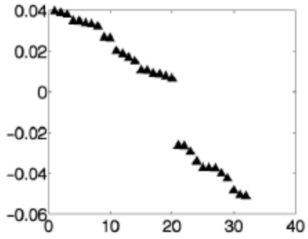
$(\mathrm{I}-\mathrm{N})z=\lambda z,\quad N=D^{-\frac{1}{2}}WD^{-\frac{1}{2}}\quad$ and $\;z=D^{\frac{1}{2}}y$

- N is calles the affinity matrix.
- The sign of the eigenvalues gives the cluster.



(a)

Sample W



(b)

Corresponding second smallest Eigenvalue

The weight function

- Many different weight functions can be used, a simple one is:

$$w_{ij} = \exp\left(-\frac{\left\|F_i - F_j\right\|^2}{\sigma_F^2} - \frac{\left\|x_i - x_j\right\|^2}{\sigma_s^2}\right)$$

- F is a feature vector that only considers pixels within a radius abs(x_i-x_j)<r

Graph cuts and energy-based methods

- If we restrict the neighborhoods to local neighborhoodsand compute region membership by summing pixels, the graph-cut can be written as a MRF or energy-minimizing problem.

- We will not og into detail of this.

Next week

- Lab on energy functions and segmentation
- Check course webpage for room