INF 5300 - 4.3.2014
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

Curriculum

- 3.7.2 in Szeliski
- 5.3
- 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 Solberg, Taxt and Jain.

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 ++(in a later lecture)

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:

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

- 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

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

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

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

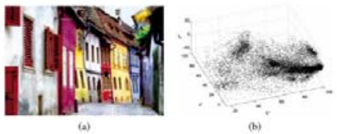
Mean shift clustering/segmentation algorithm

- 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

- Assume that we want a clustering algorithm that does not assume any particular distribution of the feature vectors.
- We want to fit a non-parametric probability density to the scatter plot
- 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:

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


(a) (b)

Scatter plots in L*u*v* space

Mean shift segmentation – main idea

- Want to avoid estimating f(x) at all locations.
- Main ideas: select random locations in the scatter plot as cluster centers. Move the cluster centers in the direction of a peak in the feature space/scatter plot.
- If we move in the direction of the negative gradient, we will move towards a local peak.
- It uses the so-called multiple restart gradient descent algorithm: start at many points y_k and take a step up-hill from these point.
- Locally, we estimate f(x) using a kernel function K(x-x_i). Typically a Gaussian kernel is used (h is the width of the kernel):

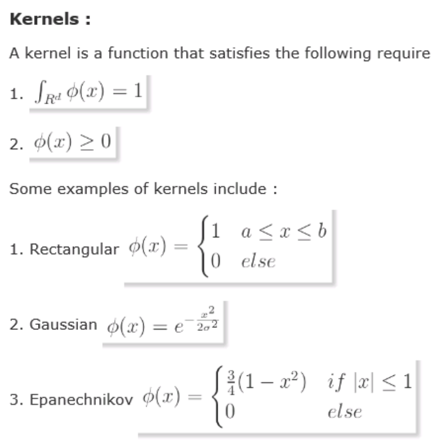
$$K\left(\frac{\|x-x_i\|^2}{h^2}\right)$$

- The estimate of f(x) is then:

$$f(\bar{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} k\left(\frac{\|x-x_i\|^2}{h^2}\right)$$

- d is the number of features.

What is a kernel?



Kernels :

A kernel is a function that satisfies the following requirements :

1. $\int_{R^d} \phi(x) = 1$

2. $\phi(x) \geq 0$

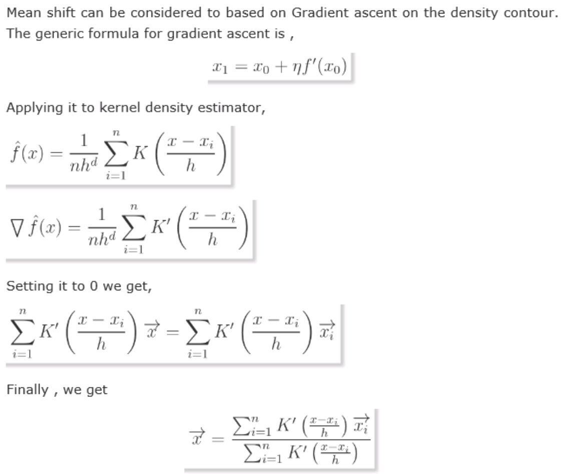
Some examples of kernels include :

1. Rectangular $\phi(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & else \end{cases}$

2. Gaussian $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$

3. Epanechnikov $\phi(x) = \begin{cases} \frac{3}{4}(1-x^2) & if\ |x| \leq 1 \\ 0 & else \end{cases}$

Gradient ascent



Mean shift can be considered to based on Gradient ascent on the density contour.
The generic formula for gradient ascent is ,

$$x_1 = x_0 + \eta f'(x_0)$$

Applying it to kernel density estimator,

$$\hat{f}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)$$

$$\nabla \hat{f}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K'\left(\frac{x-x_i}{h}\right)$$

Setting it to 0 we get,

$$\sum_{i=1}^{n} K'\left(\frac{x-x_i}{h}\right)\vec{x} = \sum_{i=1}^{n} K'\left(\frac{x-x_i}{h}\right)\vec{x_i}$$

Finally , we get

$$\vec{x} = \frac{\sum_{i=1}^{n} K'\left(\frac{x-x_i}{h}\right)\vec{x_i}}{\sum_{i=1}^{n} K'\left(\frac{x-x_i}{h}\right)}$$

Mean shift segmentation

- The gradient of f is (let 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

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

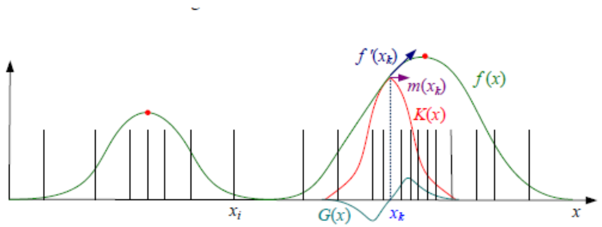
$$m(x) = \frac{\sum_i x_i G(x-x_i)}{\sum_i G(x-xi)} - x$$

- Start at location y_0. The current estimate of y_k is replaced with its locally shifted weighted mean:

$$y_{k+1} = y_k + m(y_k)$$

- 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.

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)

Adding spatial context

- Including location information :
  - Add the coordiates x_s= (x,y) in the kernel:

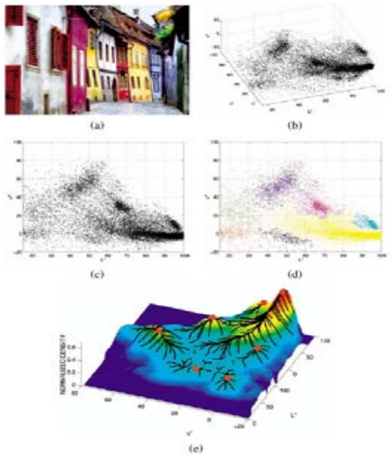
$$K(x_j)=k\left(\frac{\|x_r\|^2}{h_r^2}\right)k\left(\frac{\|x_s\|^2}{h_s^2}\right)$$

Spatial kernel involving the difference in coordinates between current pixel and the cluster center pixel

Spectral feature kernel

  - 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.

Mean shift in feature space



- The black lines illustrate the shifts in cluster centers as the iterations propagate.
- The red dots are the cluster centers we end up with.

Applications of mean shift

- The mean shift principle can be used for different applications:
  - Clustering/segmentation
  - Noise filtering/image restoration
  - Tracking objects

Regularization by Bayesian modelling

- 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:

$$p(x \mid y) = \frac{p(y \mid x)\, p(x)}{p(y)}$$

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

- 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)]
- We will use this to classify a feature vector based on the value of feature vectors or class labels of neighboring pixels.
  - Specifying interaction of features is easy on the local level.
- 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 - restoration

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

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

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 - restoration

- 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:

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

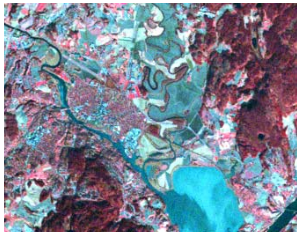
- Different forms of the penalty can be used, e.g. a hyper-Laplacian

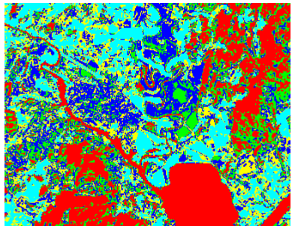
$$\rho_p(d) = |d|^p,\ p < 1$$

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

Background – contextual classification

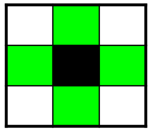
- An image normally contains areas of similar class
  - neighboring pixels tend to be similar.
- Classified images based on a non-contextual model often contain isolated misclassified pixels (or small regions).
- How can we get rid of this?
  - Majority filtering in a local neighborhood
  - Remove small 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.



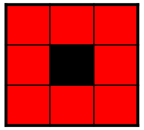


Relation between classes of neighboring pixels

- 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 $P(\omega_r | y_i)$

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

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

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

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

- $p(y_i | \omega_s)$ 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

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

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

- We want to contrain X so the resulting classified image is smooth and neighboring pixels are likely to have the same class.

- 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 $x_1,\ldots.x_N$ is equivalent to maximizing P(Y|X)P(X) since the denominator does not depend on the classes $x_1,\ldots.x_N$.
- 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.

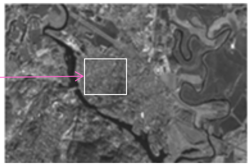
Two kinds of pixel dependency

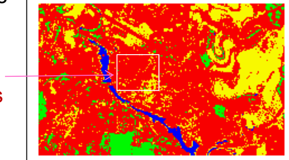
- Interpixel feature dependency:
  - Dependency between the feature vectors.
- Interpixel class dependency:
  - Dependency between class labels of neighboring pixels.
  - Results in smoothness in the classified image.

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 vector 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).
  - The resulting energy function will be computationally complex.
- 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. Even if this is the case, this simplification does not influence the result too much.
  - 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 transform a problem of finding the highest posterior probability to a problem of finding the minimum of a simple energy function.
- 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:

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

- **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)

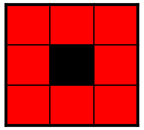
- Pixel site *j* is a neighbor of site *i≠j* if the probability

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

depends on x_j, the value of X_j.

- The energy function U(x) can be expressed as a sum of potential functions V_c(x) involving pixels in a neighborhood Q

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



- 8-neighborhoods are commonly used, but more complex neighborhoods can also be defined.

The Ising potential function

- 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.

Discrete Markov random field definitions – 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(**X**=**x**)>0 for all **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

     - This means that the probability for observing class c in pixel t depends only on the class labels of neighboring pixels, not on pixels further apart.
  3. Homogeneity: $P(X_t=x_t|\mathbf{X}_{\partial t}=\mathbf{x}_{\partial t})$ is the same for all sites t.
     - The probability for observing class c should be the same for all pixels in the image.

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 pixels in a neighborhood system.
- Advantage: a global model can be specified using local interactions only.

Back to the initial model…

Y = {y_1,…,y_N} Image of feature vectors to classify

X = {x_1,…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

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

- 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**)).

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

$$U(\mathbf{x}_i)=\sum_{k\in\mathrm{neighborhood}Q}\beta I(x_i,x_k)$$

A sum over class labels in the neighborhood, Counting the number of neighboring pixels with class label equal to x_i.

- Rewrite $P(Y_i=y_i|X_i=x_i)$ (the data term) 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

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

Regularization term

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:

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

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

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.
- In practise, the error rate is often not much lower using more compext algorithms.

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

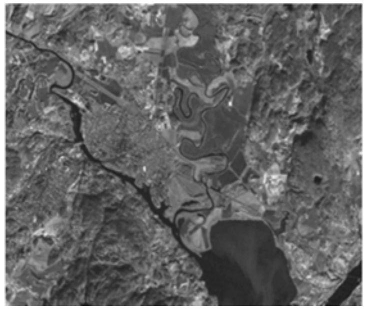
- $P(Y_t=y_t|X_t=x_t)$ can be computed based on various software packages, stored, and used in the ICM algorithm.
- For an image with S classes, this can be stored in a S-band image.
- For each iteration, only the labels x_i change.
  - – Why should you use a temporal array to store the updated labels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

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

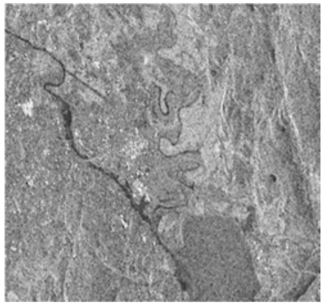


Classification results, Landsat TM image

Method	Training data, Noncontextual	Test data, Noncontextual	Test data, contextual
Gaussian	90.1	90.5	96.3
Multilayer perceptron neural network	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
- This data might deviate from Gaussian.



Classification results, SAR image

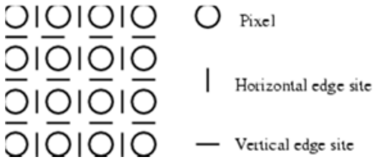
Method	Training data, Noncontextual	Test data, Noncontextual	Test data, contextual
Gaussian	63.7	63.4	67.1
Multilayer perceptron	66.6	66.9	70.8
Tree classifier	70.3	65.0	76.1

Extensions of this model

- The follwing slides shows example of extensions to include other types of prior constraints in the regularization.

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.

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 geographical 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

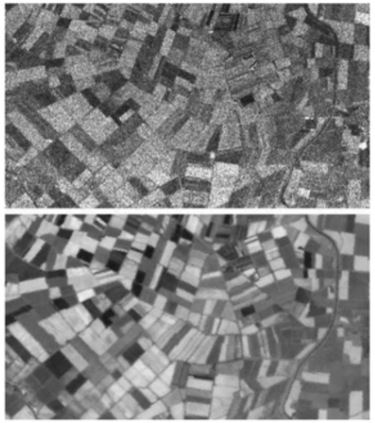
An energy term for crop ownership data

- 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

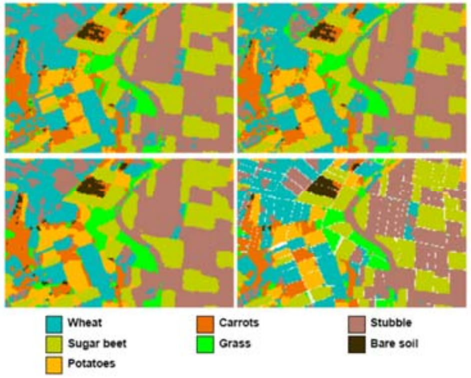
Example agricultural classification

- 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

Learning goals

- Understand the EM clustering algorithm for mixtures of Gaussians
- Understand the mean shift clustering algorithm and how it uses spatial information
- Understand Bayes rule for contextual classification and how this is connected to local energy functions.
- Understand the ICM algorithm.
- Know that this can be extended to other types of constraints and other data types.

Next week

- Lab on energy functions and segmentation
- Location: DSB Lab, Room 4270