INF 5300 – Repetition lecture

Anne Solberg (anne@ifi.uio.no)

Topics from Annes lectures

- Snakes
- Feature selection
- Feature transforms
  - Principal component transform
  - Fisher's linear discriminant
- Support vector machines classification
- Contextual classification

- Oral exam: 20min with questions from 2-3 of these topics.

SNAKES: The energy function

- Simple snake with only two terms (no termination energy):

$$E_{snake}(s) = E_{\text{int}}(v_s) + E_{image}(v_s)$$
$$= \alpha\left|\frac{dv_s}{ds}\right|^2 + \beta\left|\frac{d^2v_s}{ds^2}\right|^2 + \gamma E_{edge}$$

- We need to approximate both the first derivative and the second derivative of v_s, and specify how E_{edge} will be computed.
- The snakes is initialized from a starting position, e.g. a circle with given center and radius.
- How should the snake iterate from its initial position?

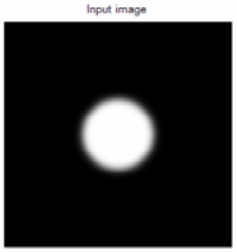
SNAKES: A simple image term

$$E_{image} = \int_0^1 P(v(s))ds$$

- A common way of defining P(x,y) is:

$$P(x,y) = -c\left|\nabla(G_\sigma * I(x,y))\right|$$

- c is a constant, ∇ is a gradient operator, G_σ is a Gaussian filter, and I(x,y) the input image. Note the minus sign as the gradient is high for edges.



SNAKES:
Approximating the first derivative of v_s

Consider a local point *s* on the curve.

Average distance between points on the contour

Distance between this point and the next point

$$\left|\frac{dv_s}{ds}\right|^2 = \left|\sum_{i=0}^{S-1}\left\|v_i - v_{i+1}\right\|/S - \left\|v_s - v_{s+1}\right\|\right|$$

$$= \left|\sum_{i=0}^{S-1}\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}/S - \sqrt{(x_s - x_{s+1})^2 + (y_s - y_{s+1})^2}\right|$$

SNAKES:
Approximating the second derivative of v_s

Why is this correct?
INF 2310 – Konvolusjon og ikke-lineære filtere (1.3.2011)
Hint: Check the derivation of the 2D Laplace operator

$$\left|\frac{d^2v_s}{ds^2}\right|^2 = \left|(v_{s+1} - 2v_s + v_{s-1})\right|^2$$

$$= (x_{s+1} - 2x_s + x_{s-1})^2 + (y_{s+1} - 2y_s - y_{s-1})^2$$

To answer a question regarding this foil
This is not a course in differential equations, so understanding this is not so important

- Assume that we seek an iterative solution.
- Assume that we have one solution

$$\hat{v}(s) = (\hat{x}(s), \hat{y}(s))$$

- If this solution is perturbated slightly by $\varepsilon\delta v(s)$, the solution that has minimum energy must satisfy:

$$\frac{dE_{snake}(\hat{v}(s) + \varepsilon\delta v(s))}{d\varepsilon} = 0$$

The new solution should be a minimum, so the derivative must be 0.

- The slight spatial perturbation is defined as $\delta v(s)=(\delta_x(s), \delta_y(s))$.
- The perturbed snake solution is:

$$\hat{v}(s) + \varepsilon\delta v(s) = (\hat{x}(s) + \varepsilon\delta_x(s), \hat{y}(s) + \varepsilon\delta_y(s))$$

SNAKES

- We have two equations

  $Ax=f_x(x,y)$

  $Ay=f_y(x,y)$
- These means that the snake energy should be balanced by the edge energy.
- We need an iterative approach to get a solution that is globally optimal (one single iteration by computing A^{-1} gives a local optimal solution).
- An iterative solution must have snake points that depend on time, a snake that can move.
- Let $x^{<i>},y^{<i>}$ denote the solution at time i.

SNAKES:The Kass snake algorithm

- Initialize the snake by selecting an initial countour
- Compute the initial energy terms and the gradient.
- Select parameters
- Given the solution at iteration i $x^{<i>}$,$y^{<i>}$, compute $x^{<i+1>}$,$y^{<i+1>}$:

$$x^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda x^{<i>} + f_x(x^{<i>}, y^{<i>})\right)$$
$$y^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda y^{<i>} + f_y(x^{<i>}, y^{<i>})\right)$$

SNAKES:The Kass differential equations

- The coordinates of the snake should be found by solving the differential equations iteratively:

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\} + \frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\} + \frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}} = 0$$

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{y}(s)}{ds}\right\} + \frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{y}(s)}{ds^2}\right\} + \frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial y}\right|_{\hat{x},\hat{y}} = 0$$

- The iterative solution was given by

$$x^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda x^{<i>} + f_x(x^{<i>}, y^{<i>})\right)$$
$$y^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda y^{<i>} + f_y(x^{<i>}, y^{<i>})\right)$$

- λ is a step size

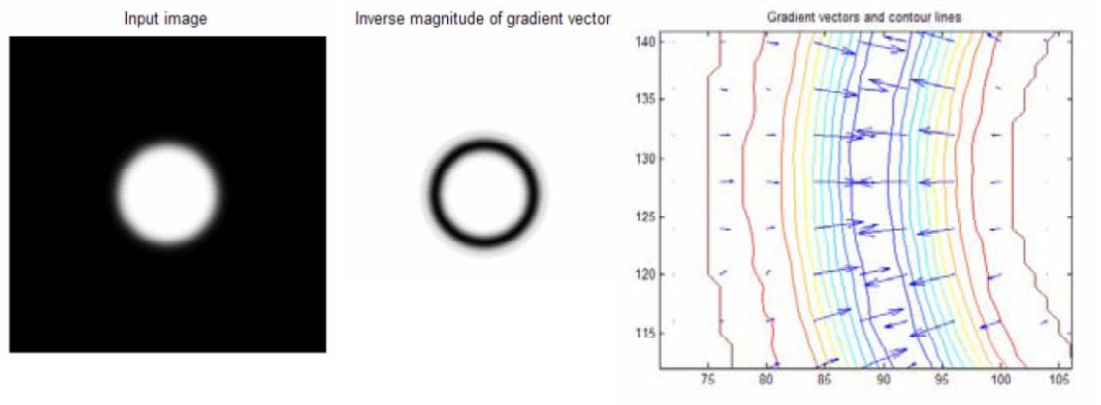
SNAKES: Capture range problems

- The snake must be initialized fairly close to the final target in order to get convergence.
- To make a really good initialization we need to have a very good estimate of the solution before starting the iterative process of adapting the snake.
- So we can find a good solution if we already know the solution. Obviously not very intersting...

SNAKES:Capture range problems

- One good way of visualizing this is by looking at the negative of the gradient of the external force field.
- These are the fources that pull the snake.
- The next slide shows this for a circle.
- Notice that outside the area in the immediate vicinity of the circle, these forces are negligible.

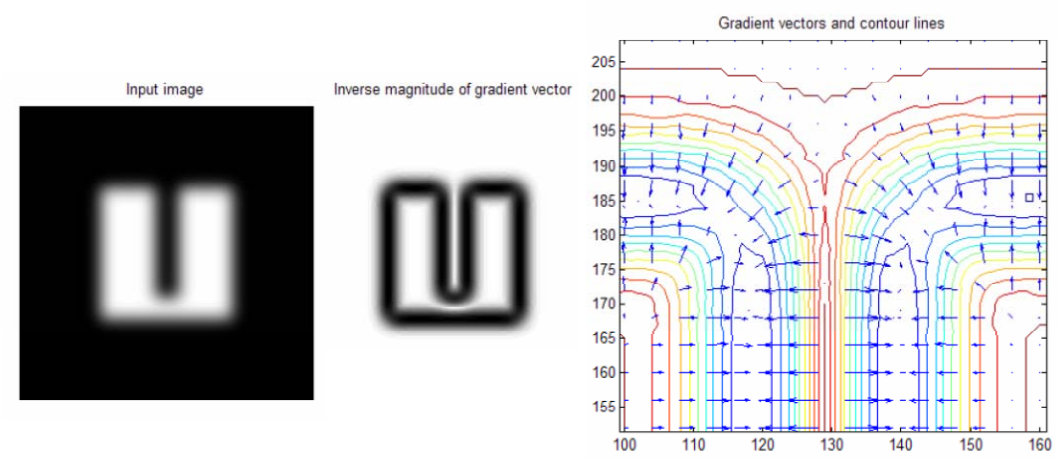
SNAKES: Capture range problems



SNAKES: Capture range problems

- This phenomenon is also the reason why you will not get convergence ito concavities, there are simply no forces to ”drag” the snake into the concavity.

SNAKES: Capture range problems



SNAKES: Capture range problems

- Xu and Prince define the vector field:

$$\mathbf{v}(x,y) = (u(x,y), v(x,y))^T$$

- It is **v** that will be the GVF.
- The field **v** is the field that minimizes the following functional:

$$G = \iint \mu\left(u_x^{\ 2} + u_y^2 + v_x^2 + v_y^2\right) + \left|\nabla f\right|^2 \left|v - \nabla f\right|^2 dxdy$$

- v(x,y) is found by solving this equation.
- μ is a parameter that controls the amount of smoothing.

SNAKES: Capture range problems

- The first term will smooth the data, that is, far from edges the field will be kept as smooth as possible by imposing that the spatial derivatives be as small as possible.
- When $|\nabla f|$ is small, the vector field will be dominated by the partial derivatives of the vector field, yielding a smooth field.
- Close to edges (where $|\nabla f|$ is large) the field is forced to resemble the gradient of f itself.
- So **v** is smooth far from edges and nearly equal to the gradient of f close to edges.
- The term μ just defines the weight we give the different terms in the functional.
- The field **v** is computed iteratively

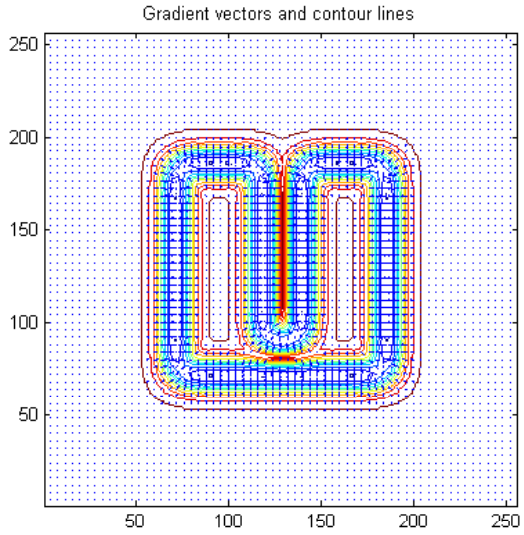
SNAKES:Capture range problems

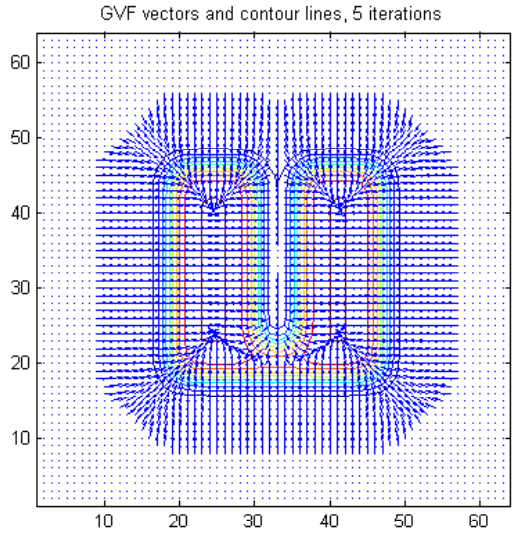
- This equation has a similar solution to the original differential equation.
- We treat u and v as functions of time and solve the equations iteratively.
  - Comparable to how we iteratively computed $x^{<i+1>},y^{<i+1>}$ from $x^{<i>},y^{<i>}$
- The solution is obviously a numerical one, we use two sets of iterations, one for u and one for v.
- After we have computed v(x,y), we replace E_{ext} (the edge magnitude term) by v(x,y)
- So an interative algorithm is first used to compute v(x,y)

Let us try to explain this….

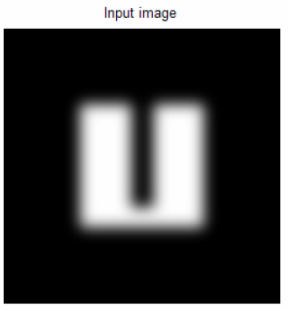
You may need to see this in high resolution in matlab

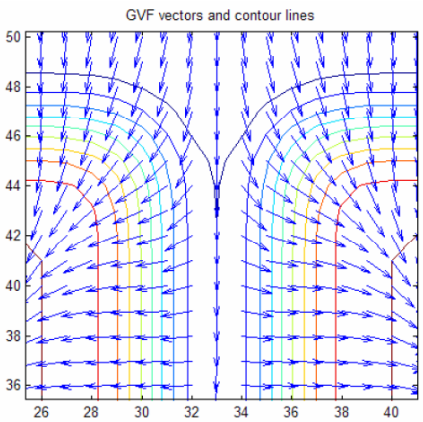
- Start with the gradient vector field and diffuse it over the image as we iterate





SNAKES: Capture range problems





FeatSel/Robust classification: Regularized covariance matrix estimation

- Let the covariance matrix be a weighted combination of a class-specific covariance matrix Σ_k and a common covariance matrix Σ:

$$\Sigma_k(\alpha) = \frac{(1-\alpha)n_k\Sigma_k + \alpha n\Sigma}{(1-\alpha)n_k + \alpha n}$$

  der $0 \le \alpha \le 1$ must be determined, and n_k and n is the number of training samples for class k and overall.
- Alternatively:

$$\Sigma_k(\beta) = (1-\beta)\Sigma_k + \beta I$$

  where the parameter $0 \le \beta \le 1$ must be determined.

Feature selection vs. Feature transformation

- Given a large set of N features, how do we select the best subset of m features?
  - How do we select m?
  - Finding the best combination of m features out a N possible is a large optimization problem.
  - Full search is normally not possible.
  - Suboptimal approaches are often used.
  - How many features are needed?
- Alternative: compute lower-dimensional projections of the N-dimensional space
  - PCA
  - Fisher's linear discriminant
  - Projection pursuit and other non-linear approaches

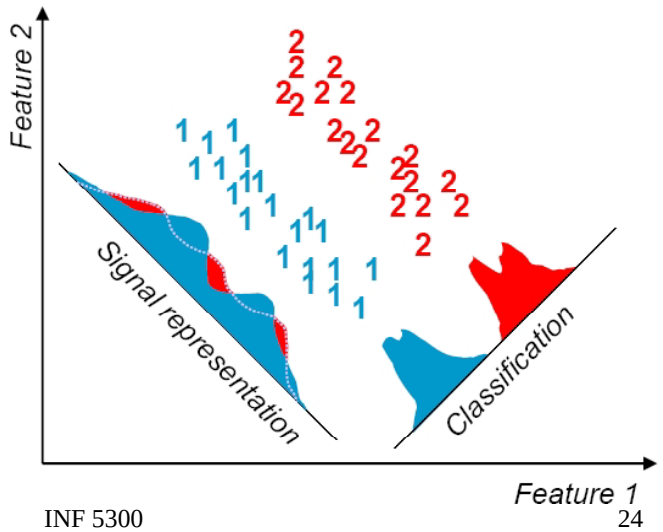
Linear feature transforms

- Feature extraction can be stated as
  - Given a feature space $x_i \in \mathbb{R}_n$ find an optimal mapping $y = f(x) : \mathbb{R}_n \rightarrow \mathbb{R}_m$ with $m < n$.
  - An optimal mapping in classification :the transformed feature vector y yield the same classification rate as x.
- The optimal mapping may be a non-linear function
  - Difficult to generate/optimize non-linear transforms
  - Feature extraction is therefore usually limited to linear transforms $y = A^T x$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{11} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

Signal representation vs classification

- Principal components analysis (PCA)
  - – signal representation, unsupervised
  - – Minimize the mean square representation error
- Linear discriminant analysis (LDA)
  - –classification, supervised
  - – Maximize the distance between the classes



PCA: Correlation matrix vs. covariance matrix

- Σ_x is the covariance matrix of x

$$\Sigma_x = E\left[(x-\mu)(x-\mu)^T\right]$$

- R_x is the correlation matrix of x

$$R_x = E\left[(x)(x)^T\right]$$

- $R_x=\Sigma_x$ if $\mu_x=0$.

Principal component or Karhunen-Loeve transform

- Let x be a feature vector.
- Features are often correlated, which might lead to redundancies.
- We now derive a transform which yields **uncorrelated** features.
- We seek a linear transform $y=A^Tx$, and the y_is should be uncorrelated.
- The y_is are uncorrelated if $E[y(i)y(j)^T]=0$, $i\neq j$.
- If we can express the information in x using uncorrelated features, we might need **fewer** coefficients.

Principal component transform

- The correlation of Y is described by the correlation matrix
  $R_Y=E[yy^T]=E[A^Txx^TA]=A^TR_xA$ R_x is the correlation matrix of X
  R_x is symmetric, thus all eigenvectors are orthogonal.

- We seek uncorrelated components of Y, thus R_y should be diagonal.

From linear algebra:

- R_y will be diagonal if A is formed by the orthogonal eigenvectors a_i, i=0,...,N-1 of R_x: $R_y=A^TR_xA=\Lambda$, where Λ is diagonal with the eigenvalues of R_x, λ_i, on the diagonal.
- We find A by solving the equation $A^TR_xA=\Lambda$ (using Singular Value Decomposition (SVD)).
- A is formed by computing the eigenvectors of R_x. Each eigenvector will be a column of A.

PCA: Mean square error approximation

- x can be expressed as a combination of all N basis vectors:

$$x=\sum_{i=0}^{N-1} y(i)a_i, \text{ where } y(i)=a_i^T x$$

- An approximation to x is found by using only m of the basis vectors:

$$\hat{x}=\sum_{i=0}^{m-1} y(i)a_i$$

  •a projection into the m-dimensional
  •subspace spanned by m eigenvectors

- The PC-transform is based on minimizing the mean square error associted with this approximation.
- The mean square error associated with this approximation is

$$E\left[\|x-\hat{x}\|^2\right]=E\left[\left\|\sum_{i=m}^{N-1} y(i)a_i\right\|^2\right]=E\left[\sum_i\sum_j\left(y(i)a_i^T\right)\left(y(j)a_j\right)\right]=$$
$$\sum_{i=m}^{N-1} E\left[y^2(i)\right]=\sum_{i=m}^{N-1} a_i^T E\left[xx^T\right]a_i$$

PCA

- Furthermore, we can find that

$$E\left[\left\|x-\hat{x}\right\|^2\right]=\sum_{i=m}^{N-1}a_i^T\lambda_i a_i=\sum_{i=m}^{N-1}\lambda_i$$

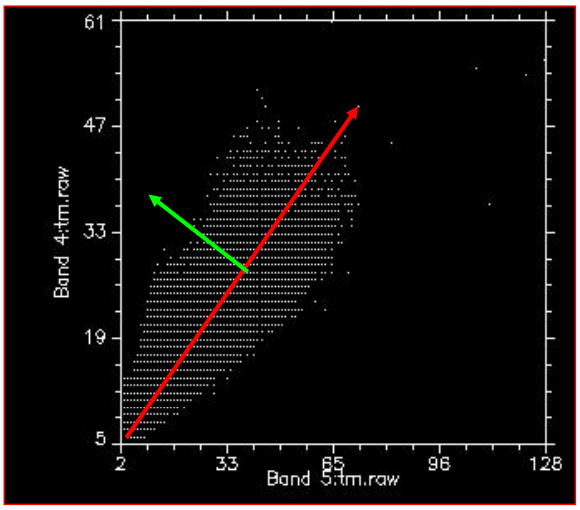
- The mean square error is thus

$$E\left[\left\|x-\hat{x}\right\|^2\right]=\sum_{i=1}^{N-1}\lambda_i-\sum_{i=1}^{m}\lambda_i=\sum_{i=m}^{N-1}\lambda_i$$

- The error is minimized if we select the eigenvectors corresponding to the *m* largest eigenvales of the correlation matrix R_x.
- The transformed vector y is called the principal components of x. The transform is called the principal component transform or Karhunen-Loeve-transform.

PCA: Geometrical interpretation of principal components

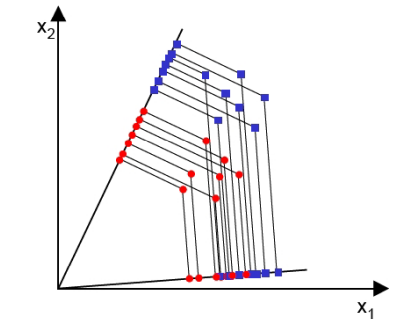
- The eigenvector corresponding to the largest eigenvalue is the direction in n-dimensional space with highest variance.
- The next principal component is orthogonal to the first, and along the direction with the second largest variance.

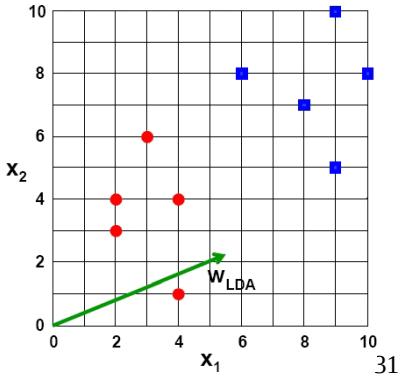


- Note that the direction with the highest variance is
- NOT related to separability between classes.

Fisher's Linear Discriminant

- Goal:
  - Reduce dimension while preserving class discriminatory information
- Strategy (2 classes):
  - We have a set of samples $x=\{x_1, x_2, \ldots, x_n\}$ where n_1 belong to class ω_1 and the rest n_2 to class ω_2. Obtain a scalar value by projecting x onto a line y: $y=w^T x$
  - **Challenge: find *w* that maximizes the separability of the classes**
  - **Maximize the criterion J**





Fisher: Introducing general scatter matrices

- In M-dimensional space, let us now consider matrices describing the variance:
  - Variance INSIDE each class
  - Variance BETWEEN the classes (how well separated are the classes)
  - The total variance in the data set is constant and independent of any class labels

Fisher: Scatter matrices – M classes

- Within-class scatter matrix:

$$S_w = \sum_{i=1}^{M} P(\omega_i) S_i$$

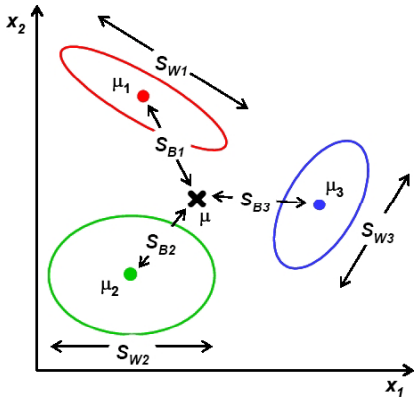
$$S_i = E\left[(x-\mu_i)(x-\mu_i)_T\right]$$

•Variance within each class

- Between-class scatter matrix:

$$S_b = \sum_{i=1}^{M} P(\omega_i)(\mu_i-\mu_0)(\mu_i-\mu_0)^T$$

$$\mu_0 = \sum_{i=1}^{M} \mu_i$$ •Distance between the classes



- Mixture or total scatter matrix:

$$S_m = E\left[(x-\mu_0)(x-\mu_0)^T\right]$$

•Variance of feature with

•respect to the global mean

Fisher: Relation between eigenvalues and the scatter of a matrix

- The eigenvalues associated with an eigenvector tells how strong the contribution along this direction is.
- A scalar measure of the scatter matrix M is its determinant (the product of the eigenvalues). This gives us ONE measure of the scatter in the matrix.
- If M is a covariance matrix, |M| is a measure of the l-dimensional hypervolume of the data. If the data lies in a subspace, |M| will be zero.
- For a covariance matrix M, trace(M) is the sum of the eigenvalues and thus a measure of the spread or scatter in A.

Fisher's linear discriminant

- Fisher's linear discriminant is a transform that uses the information in the training data set to find a linear combination that best separates the classes.
- It is based on the criterion J_3:

$$J_3 = trace\left\{S_w^{-1} S_b\right\}$$
$$S_w = \sum_{i=1}^{M} P(\omega_i) S_i - \text{within-class scatter}$$
$$S_b = \sum_{i=1}^{M} P(\omega_i)(\mu_i - \mu_0)(\mu_i - \mu_0)^T - \text{between-class scatter}$$

- From the feature vector x, let S_{xw} and S_{xb} be the within-class and between-class scatter matrix.
- The scatter matrices for the transformed variable $y=A^Tx$ are:

$S_{yw}=A^TS_{xw}A$ $S_{yb}=A^TS_{xb}A$

Fisher

- In subspace y, J_3 becomes:

$$J_3 = trace\left\{\left(A^T S_w A\right)^{-1}\left(A^T S_b A\right)\right\}$$

- Problem: find A such that J_3 is maximized.
- Solution: set

$$\frac{\partial J_3(A)}{\partial A} = 0$$
$$\Updownarrow$$
$$\frac{\partial J_3(A)}{\partial A} = -2S_{xw}A\left(A^T S_{xw}A\right)^{-1}\left(A^T S_{xb}A\right)\left(A^T S_{xw}A\right)^{-1} + 2S_{xb}A\left(A^T S_{xw}A\right)^{-1} = 0$$
$$\Updownarrow$$
$$S_{xw}^{-1}S_{xb}A = A(S_{yw}^{-1}S_{yb})$$

Computing Fishers linear discriminant

- For l=M-1:
  - Form a matrix C such that its columns are the M-1 eigenvectors of $S_{xw}^{-1}S_{xb}$
  - Set $\hat{y} = C^T x$
  - This gives us the maximum J_3 value.
  - This means that we can reduce the dimension from m to M-1 without loss in class separability power (but only if J_3 is a correct measure of class separability.)
  - Alternative view: with a Bayesian model we compute the probabilities $P(\omega_i|x)$ for each class (i=1,...M). Once M-1 probabilities are found, the remaining $P(\omega_M|x)$ is given because the $P(\omega_i|x)$'s sum to one.

Fisher:Computation: Case 2: l<M-1

- Form C by selecting the eigenvectors corresponding to the l largest eigenvalues of

$$S_{xw}^{-1}S_{xb}$$

- We now have a loss of discriminating power since

$$J_{3,\hat{y}} < J_{3,x}$$

Support Vector Machines
Two linear separable classes

- Let x_i, i=1,...N be all the l-dimensional feature vectors in a training set with N samples.
- These belong to one of two classes, ω_1 and ω_2.
- We assume that the classes are **linearly separable**.
- This means that a hyperplane

  $g(x)=w^Tx+w_0=0$

  correctly classifies all these training samples.
- $w=[w_1,...w_l]$ is called a weight vector, and w_0 is the threshold.

SVM

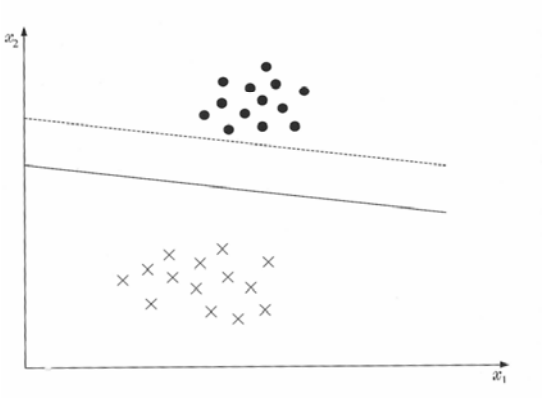
- If the classes are linearly separable, there exist a hyperplane $w^{*T}x=0$ such that:

$$w^{*T}\, x > 0 \quad \forall x \in \omega_1$$
$$w^{*T}\, x < 0 \quad \forall x \in \omega_2$$

- The above also covers the situation where the hyperplane does not cross the origin, $w^{*T}x+w_0=0$, since this can reformulated as $x'=[x,1]^T$, $w'=[w^{*T},w_0]^T$. Then $w^{*T}x+w_0=w'^Tx'$.
- Remember from 4300 that the decision boundary was defined as the surface where the discriminant function $g_1(x)-g_2(x)=0$ ($g_1(x)=g_2(x)$).

SVM

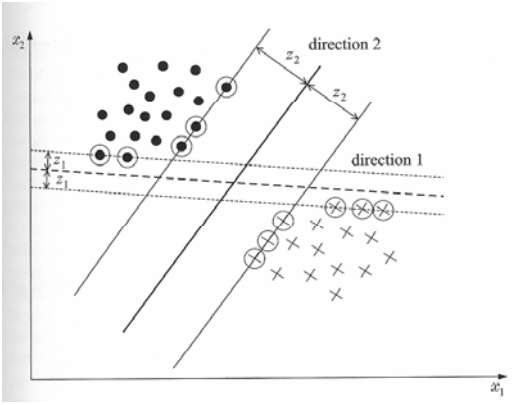
- There can be many such hyperplanes.
- Which of these two is best, and why?



SVM:Hyperplanes and margins

- A hyperplane is defined by its direction (w) and exact position (w_0).
- If both classes are equally probable, the **distance from the hyperplane to the closest points** in both classes should be equal. This is called the margin.
- The margin for direction 1 is $2z_1$, and for direction 2 it is $2z_2$.
- The distance from a point to a hyperplane is

$$z = \frac{|g(x)|}{\|w\|}$$

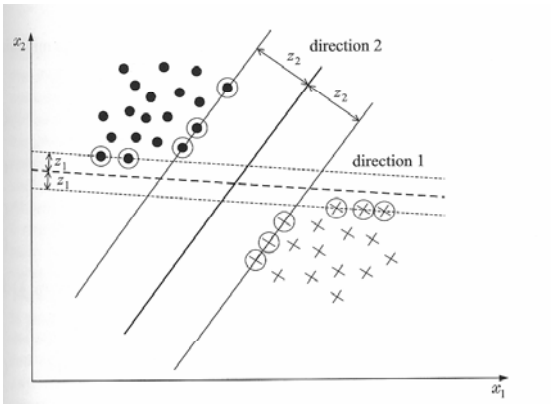


SVM: Hyperplanes and margins

- We can scale w and w_0 such that g(x) will be equal to 1 at the closest points in the two classes. This is equivalent to:

1. Have a margin of $\frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|}$
2. Require that

$$w^T x + w_0 \geq 1, \quad \forall x \in \omega_1$$
$$w^T x + w_0 \leq -1, \quad \forall x \in \omega_2$$



SVM: The optimization problem with margins

- The class indicator for pattern i, y_i, is defined as 1 if y_i belongs to class ω_1 and -1 if it belongs to ω_2.
- The best hyperplane with margin can be found by solving the optimization problem with respect to w:

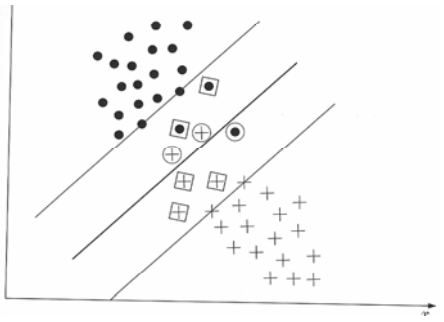
$$\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$$
$$\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,...N$$

- Checkpoint: do you understand this formulation?
- How is this criterion related to maximizing the margin?

SVM: The nonseparable case

- If the two classes are nonseparable, a hyperplane satisfying the conditions $w^Tx-w_0=\pm 1$ cannot be found.
- The feature vectors in the training set are now either:

1. Vectors that fall outside the band and are correctly classified.
2. Vectors that are inside the band and are correctly classified. They satisfy $0 \leq y_i(w^Tx+w_0)<1$
3. Vectors that are misclassified – expressed as $y_i(w^Tx+w_0)<0$



Correctly classified

Erroneously classified

SVM

- The three cases can be treated under a single type of contraints if we introduce slack variables ξ_i:

$$y_i[w^T x + w_0] \geq 1 - \xi_i$$

  - The first category (outside, correct classified) have $\xi_i=0$
  - The second category (inside, correct classified) have $0 \leq \xi_i \leq 1$
  - The third category (inside, misclassified) have $\xi_i > 1$

- The optimization goal is now to keep the margin as large as possible and the number of points with $\xi_i > 0$ as small as possible.

SVM: Cost function – nonseparable case

- The cost function to minimize is now

$$J(w, w_0, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

where $I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases}$

and ξ is the vector of parameters ξ_i.

- C is a parameter that controls how much misclassified training samples is weighted.
- We skip the mathematics and present an alternative formulation:

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j\right)$$

subject to $\sum_{i=1}^{N}\lambda_i y_i = 0$ and $0 \le \lambda_i \le C \;\; \forall i$

- All points between the two hyperplanes ($\xi_i>0$) can be shown to have $\lambda_i=C$.

SVMs: The nonlinear case

- We have now found a classifier that is not defined in terms of the class centres or the distributions, **but in terms of patterns close to the borders between classes, the support vectors.**
- It gives us a solution in terms of a hyperplane. This hyperplane can be expressed as a inner product between the training samples:

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j\right)$$

subject to $\sum_{i=1}^{N}\lambda_i y_i = 0$ and $0 \le \lambda_i \le C \;\; \forall i$

- The training samples are l-dimensional vectors.
- What if the classes overlap in l-dimensional space:
  - Can we find a mapping to a higher dimensional space, and use the SVM framework in this higher dimensional space?

SVM: Learning goals

- Understand enough of SVM classifiers to be able to use it for a classification application.
  - Understand the basic linear separable problem and what the meaning of the solution with the largest margin means.
  - Understand how SVMs work in the non-separable case using a cost for misclassification.
  - Accept the kernel trick: that the original feature vectors can be transformed into a higher dimensional space, and that linear SVM is applied in this space.
  - Know briefly how to extend from 2 to M classes.
  - Know which parameters (C,) the user must specify and how to perform a grid search for these.
  - Be able to find a SVM library and use it correctly ☺

SVM:The optimization problem with margins

$$\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,...N$$

- This is a quadratic optimization task with a set of linear inequality contraints.
- It can be shown that the solution has the form:

$$\mathrm{w} = \sum_{i=1}^{N} \lambda_i y_i x_i \quad \text{where} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

- The λ_i's are called Lagrange multipliers.
- The λ_i's can be either 0 or positive.
- We see that the solution w is a linear combination of $N_s \leq N$ feature vectors associated with a $\lambda_i > 0$.

MRF

$Y = \{y_1,...,y_N\}$ 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

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

MRF

- 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**)).

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

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

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

MRF

- 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

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

MRF-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:

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

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

MRF: 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.

MRF: 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

MRF: 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(Yt=yt|Xt=s))
                        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;
```


SVM: Learning goals

- Understand the energy function combining the data term and the contextual term.
- Understand the Ising model
- Understand the ICM algorithm
- Realize that other types of spatial constrains can be added by modifying the energy function.