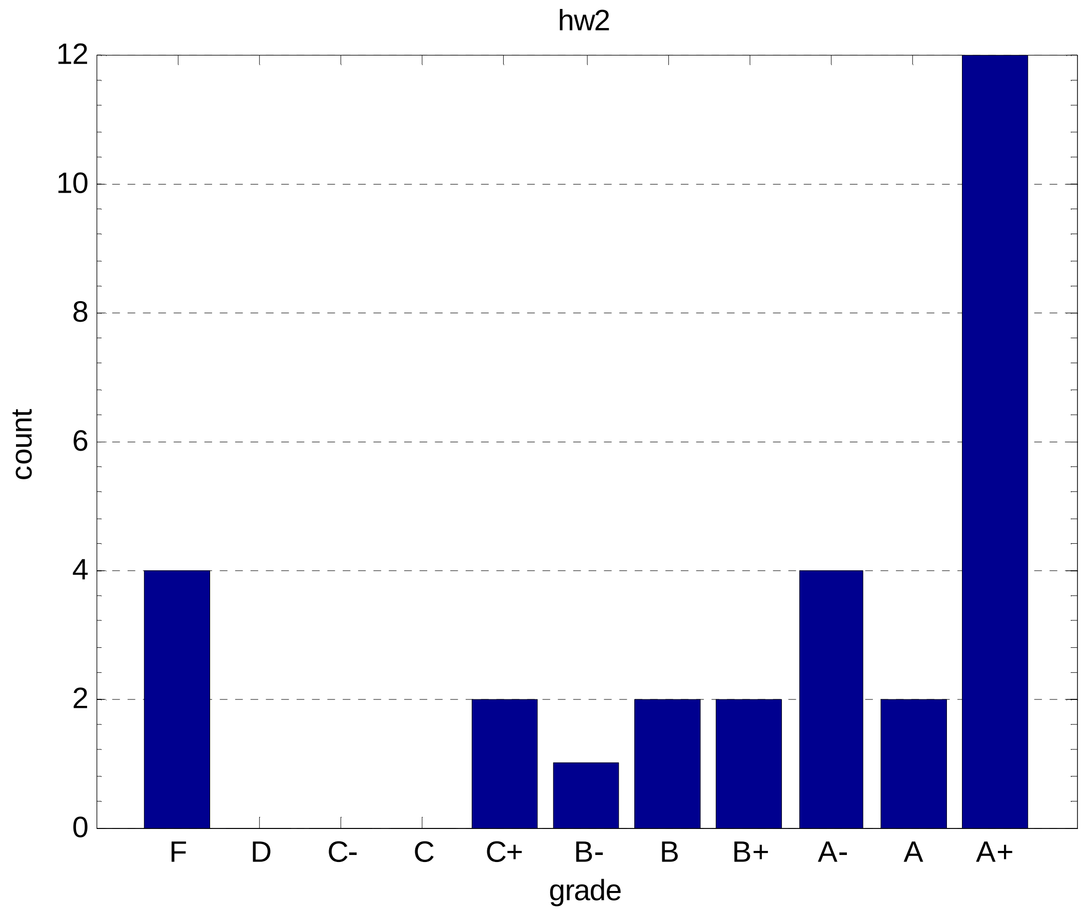
CS540 Machine learning
L7: MVN

hw2



Q1

* Lots of students don't realize that when X is rectangular, inv(X) doesn't exist. This is the most serious problem I saw with Q1, as most of students simplify by taking inv(X'X) = inv(X)*inv(X').

* Also most students still don't realize that a'b = b'a. They also have problems seeing the matrix dimensions, so sometime they will equate scalars to vectors without even realizing, or multiply matrices of incompatible dimensions. Some also didn't realize that AB not equal to BA. So a common mistake was X*inv(X'X)*X' = I, as it can be rearranged. Anyways, I think some more stress on linear algebra will help.

Q2

- * Very few students understood that J(W) can be split into J(w_i) and because of this
- separation of function into q number of functions depending only on w_i, it is possible
- to convert the problem into a bunch of OLS problems.

- * Many used the word "independent"
- ambiguously. Many students had the right idea but failed to clearly express it (and
- I didn't deduct marks whenever I saw even a glimpse of the right idea).

- * Few students
- also wrote that because norm is positive so that they can change "min of sum" to "sum of min"
- (which is wrong).

Q3

- * Many students didn't realize that J(w) is a scalar and when you
- differentiate wrt a scalar, you
- get a scalar. Most of these students got a vector at the end which
- was equated to zero, and then
- they tried to "magically" take the average and get the answer. I
- could see that only few students
- knew how to do matrix differentiation properly.

Q4

- * This question had least problems. It was easiest of all (although I
- thought it may be hard). I found
- that few students have very similar answers. I even found a bunch of
- answers with the same (less likely)
- mistake. I don't know if it was a coicidence or not. But anyway,
- this question was very easy for most of them.
- Only those people lost marks, who didn't attempt the last part.

Last time

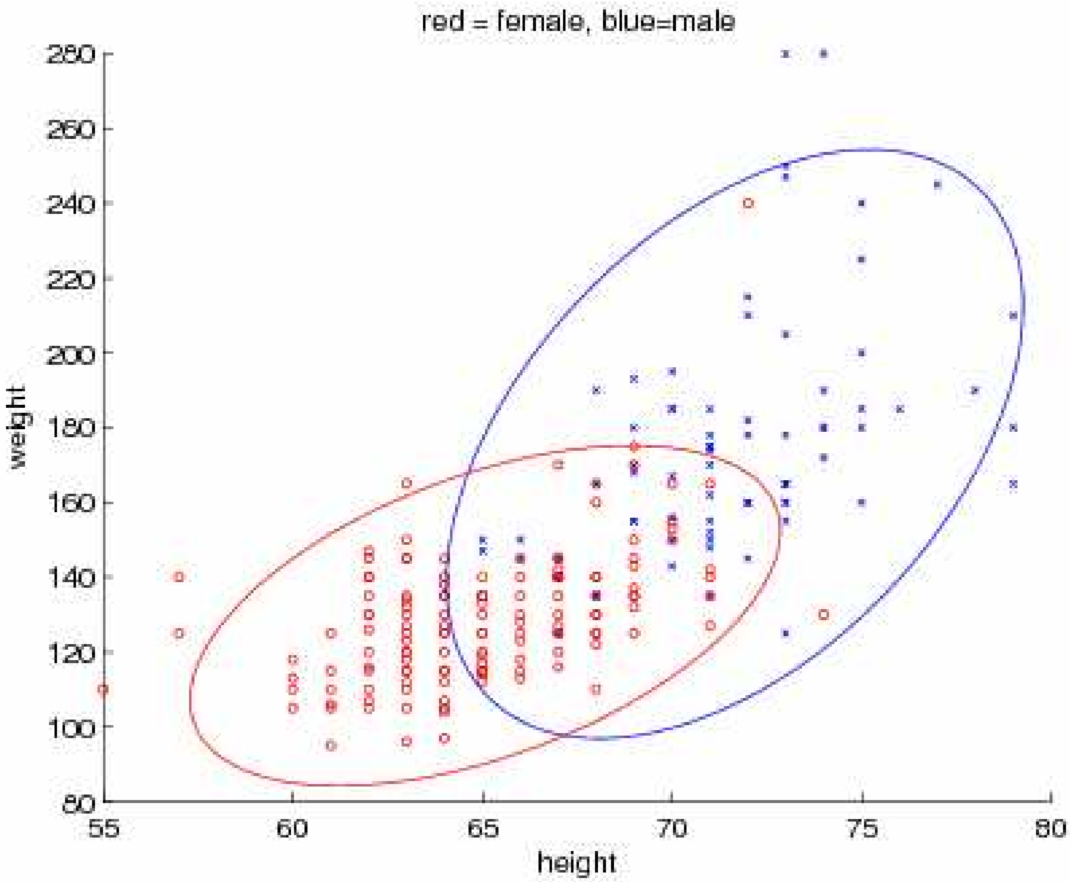
- Logistic regression p(y|x,theta)
- Perceptron algorithm
- IRLS (Newton’s algorithm)
- Multinomial logistic regression
- Why probabilistic classifiers?

This time

- Multivariate Gaussians
- Definition
- Eigenanalysis
- MLE
- Plug into a classifier

Correlated features

- Height and weight are not independent



Multivariate Gaussian

- Multivariate Normal (MVN)

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})]$$

- Exponent is the Mahalanobis distance between x and μ

$$\Delta = (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

Σ is the covariance matrix (positive definite)

$$\mathbf{x}^T\Sigma\mathbf{x} > 0 \ \forall\mathbf{x}$$

Bivariate Gaussian

- Covariance matrix is

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$$

where the correlation coefficient is

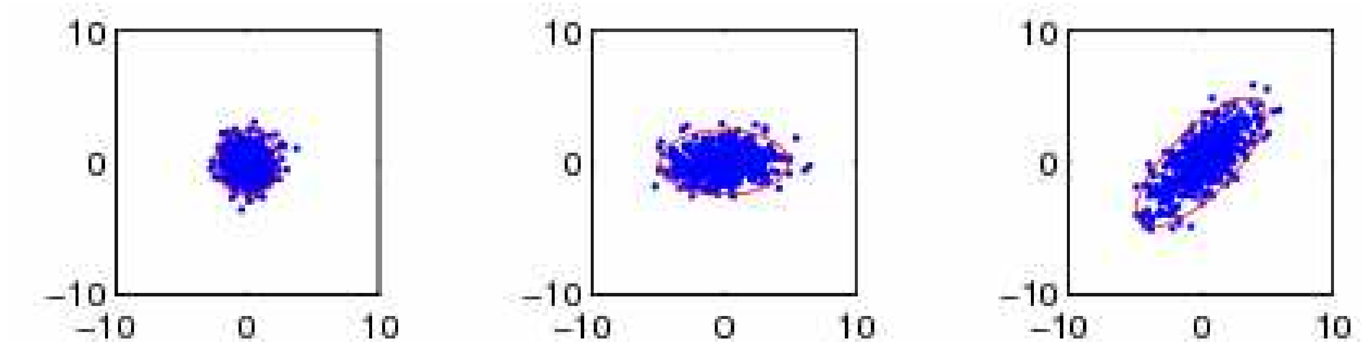
$$\rho = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$$

and satisfies -1 $\leq$ ρ $\leq$ 1

- Density is

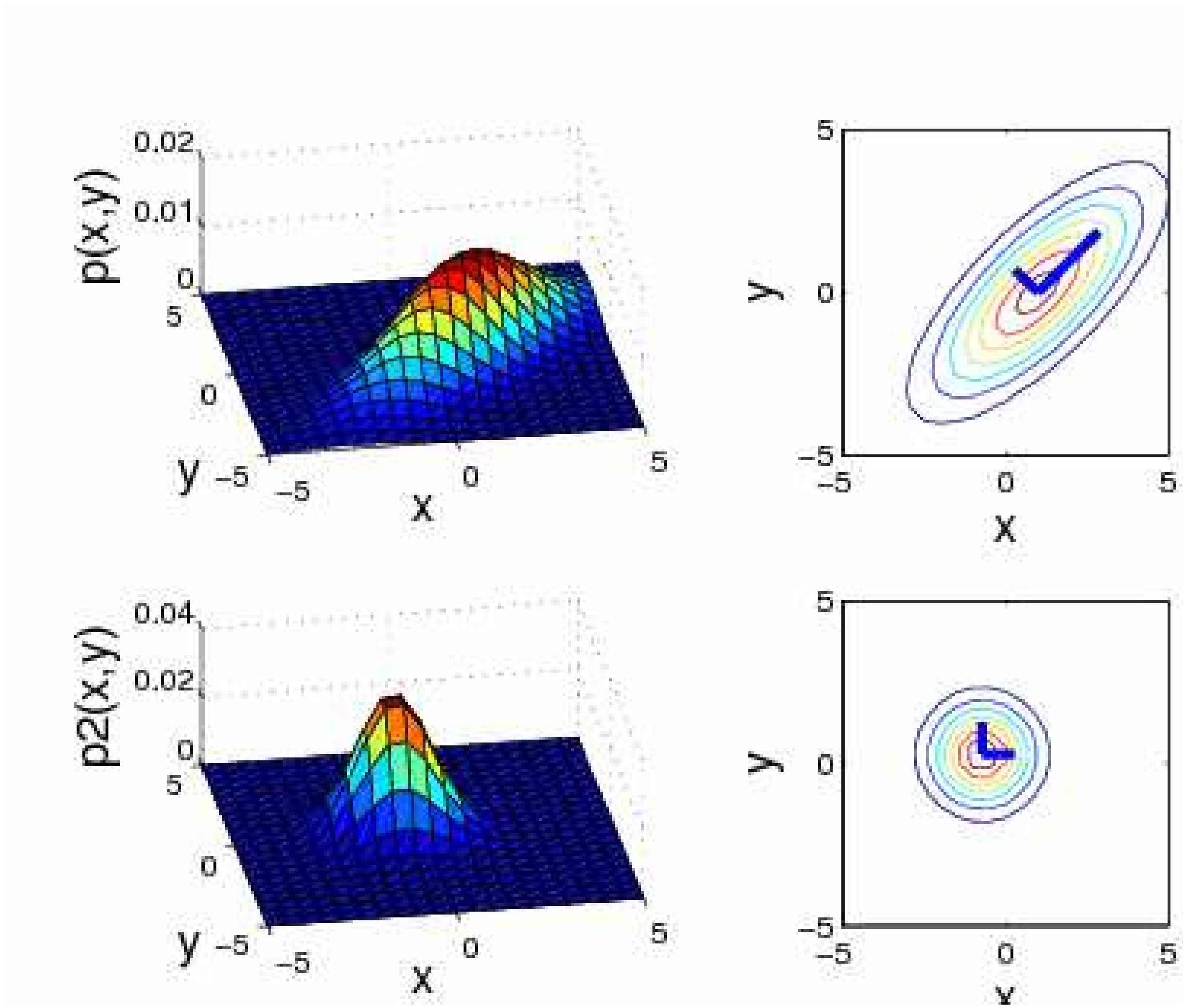
$$p(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{(\sigma_x\sigma_y)}\right)\right)$$

Spherical, diagonal, full covariance



$$\Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$$

Surface plots



Eigenanalysis

- We can show analytically that the contours of constant density will be ellipses by studying the eigenvectors / values of Σ.
- This analysis will prove useful for other things, too.

Eigenvectors and eigenvalues

- We can compute the evecs u_i and evals λ_i of any square m x m matrix A; these satisfy

$$A\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

- In matrix form, this becomes

$$\begin{aligned} AU &= U\Lambda \\ (A - \Lambda)U &= 0 \end{aligned}$$

where Λ is a diagonal matrix of evals.

- For this set of eqns to have a soln, we require

$$|A - \Lambda| = 0$$

- This is a polynomial of order m, so it has m solutions (though these need not all be distinct).
- In Matlab, just type

```
[U,Lam] = eig(A);
```

Real, symmetric matrices

- If $A_{ij} \in R$, then A is called real.
- If $A^T = A$, then A is called symmetric.
- Examples include: covariance matrices, kernel matrices and Hessian matrices.
- A^{-1} is also symmetric, since

$$\begin{aligned} A^{-1}A &= I \\ A^T(A^{-1})^T &= I^T \\ AA^{-T} &= I \\ A^{-T} &= A^{-1} \end{aligned}$$

Orthogonal matrices

- If A is real and symmetric (so $A^T = A$), then one can show that the evals are real and the evecs are orthonormal, i.e. $\mathbf{u}_i^T \mathbf{u}_j = \delta(i-j)$
- In matrix form this becomes $U^T U = I$
- We say U is an orthogonal matrix.
- The rows are also orthonormal since

$$
\begin{aligned}
U^T U &= I \\
U^T U U^{-1} &= U^{-1} \\
U U^{-1} &= U U^T = I
\end{aligned}
$$

Diagonalization

- If A is real and symmetric, then U is orthogonal.
- Hence we can express A as a sum of outer products of the evecs weighted by the evals

$$
\begin{aligned}
AU &= U\Lambda \\
A &= U\Lambda U^T = \sum_{i=1}^{p} \lambda_i \mathbf{u}_i \mathbf{u}_i^T
\end{aligned}
$$

$$
A = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_p \\ | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_p \end{pmatrix} \begin{pmatrix} - & u_1^T & - \\ & \vdots & \\ - & u_p^T & - \end{pmatrix}
$$

$$
= \lambda_1 \begin{pmatrix} | \\ u_1 \\ | \end{pmatrix} \begin{pmatrix} - & u_1^T & - \end{pmatrix} + \cdots + \lambda_p \begin{pmatrix} | \\ u_p \\ | \end{pmatrix} \begin{pmatrix} - & u_p^T & - \end{pmatrix}
$$

Transformation by an orthogonal matrix

- Consider a vector x transformed by the orthogonal matrix U to give

$$\tilde{\mathbf{x}} \;=\; U\mathbf{x}$$

- The length of the vector is preserved since

$$||\tilde{\mathbf{x}}||^2 = \tilde{\mathbf{x}}^T\tilde{\mathbf{x}} = \mathbf{x}^T U^T U^T \mathbf{x} = \mathbf{x}^T\mathbf{x} = ||\mathbf{x}||^2$$

- The angle between vectors is preserved

$$\tilde{\mathbf{x}}^T\tilde{\mathbf{y}} = \mathbf{x}^T U^U \mathbf{y} = \mathbf{x}^T\mathbf{y}$$

- Thus multiplication by U can be interpreted as a rigid rotation of the coordinate system.

Geometry of diagonalization

- Let A be a linear transformation. We can always decompose this into a rotation U, a scaling Λ, and a reverse rotation $U^T=U^{-1}$.
- Hence $A = U \Lambda U^T$.
- The inverse mapping is given by $A^{-1} = U \Lambda^{-1} U^T$

$$
\begin{aligned}
A &= \sum_{i=1}^{m} \lambda_i \mathbf{u}_i \mathbf{u}_i^T \\
A^{-1} &= \sum_{i=1}^{m} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T
\end{aligned}
$$

Positive definite matrices

- A matrix A is pd if $x^T A x > 0$ for any non-zero vector x.
- Hence all the evecs of a pd matrix are positive

$$
\begin{aligned}
A\mathbf{u}_i &= \lambda_i \mathbf{u}_i \\
\mathbf{u}_i^T A\mathbf{u}_i &= \lambda_i \mathbf{u}_i^T \mathbf{u}_i = \lambda_i > 0
\end{aligned}
$$

- A matrix is positive semi definite (psd) if $\lambda_i >= 0$.
- A matrix of all positive entries is not necessarily pd; conversely, a pd matrix can have negative entries

```
> [u,v] = eig([1 2; 3 4])
u =
   -0.8246   -0.4160
    0.5658   -0.9094
v =
   -0.3723         0
         0    5.3723
```

```
[u,v]=eig([2 -1; -1 2])
u =
   -0.7071   -0.7071
   -0.7071    0.7071
v =
     1     0
     0     3
```

Rank of a matrix

- The rank of a matrix is the number of non zero evalues.
- If the matrix is not full rank, it is not invertible, since

$$
\begin{aligned}
A &= U\Lambda U^T \\
|A| &= |U||\Lambda||U^T| = \prod_{i=1}^{m} \lambda_i
\end{aligned}
$$

Visualizing a covariance matrix

- Let $\Sigma = U \Lambda U^T$. Hence

$$\Sigma^{-1} = U^{-T}\Lambda^{-1}U^{-1} = U\Lambda^{-1}U = \sum_{i=1}^{p} \frac{1}{\lambda_i}\mathbf{u}_i\mathbf{u}_i^T$$

- Let $y = U(x-\mu)$ be a transformed coordinate system, translated by μ and rotated by U. Then

$$\begin{aligned} (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) &= (\mathbf{x}-\boldsymbol{\mu})^T\left(\sum_{i=1}^{p}\frac{1}{\lambda_i}\mathbf{u}_i\mathbf{u}_i^T\right)(\mathbf{x}-\boldsymbol{\mu}) \\ &= \sum_{i=1}^{p}\frac{1}{\lambda_i}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{u}_i\mathbf{u}_i^T(\mathbf{x}-\boldsymbol{\mu}) = \sum_{i=1}^{p}\frac{y_i^2}{\lambda_i} \end{aligned}$$

Visualizing a covariance matrix

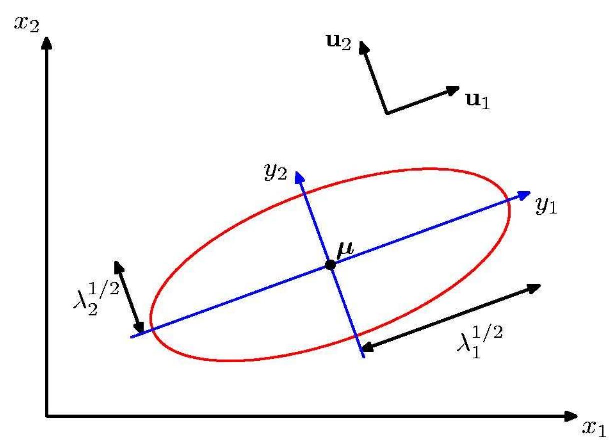
- From the previous slide

$$(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \sum_{i=1}^{p}\frac{y_i^2}{\lambda_i}$$

- Recall that the equation for an ellipse in 2D is

$$\frac{y_1^2}{\lambda_1}+\frac{y_2^2}{\lambda_2}=1$$

- Hence the contours of equiprobability are elliptical, with axes given by the evecs and scales given by the evals of Σ



Standardizing the data

- We can subtract off the mean and divide by the standard deviation of each dimension to get the following (for case i=1:n and dimension j=1:d)

$$y_{ij} = \frac{x_{ij} - \overline{x}_j}{\sigma_j}$$

- Then E[Y]=0 and Var[Y_j]=1.
- However, Cov[Y] might still be elliptical due to correlation amongst the dimensions.

Whitening the data

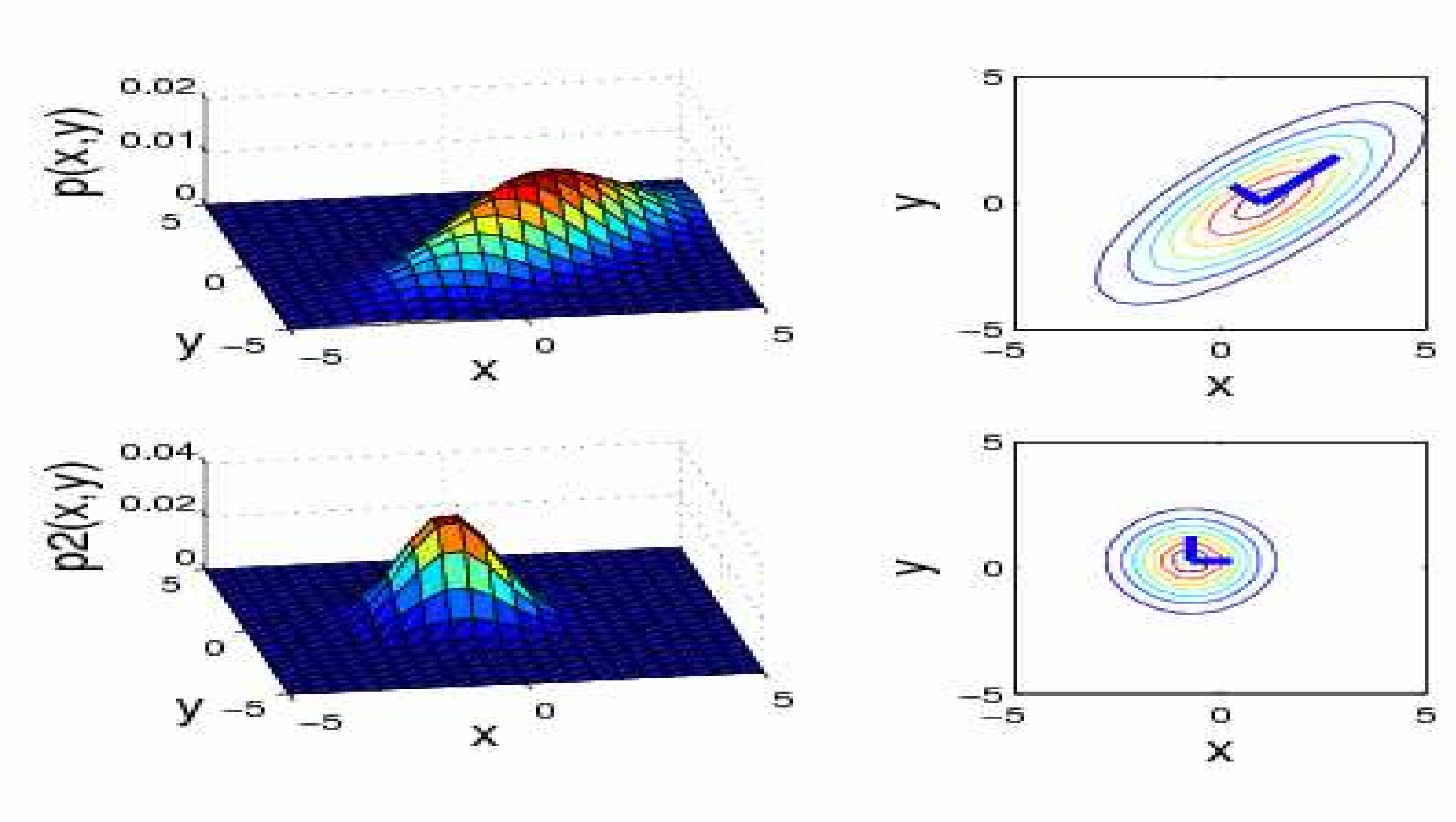
- Let X ~ N(μ,Σ) and Σ = U Λ U^T.
- To remove any correlation, we can apply the following linear transformation

$$
\begin{aligned}
Y &= \Lambda^{-\frac{1}{2}} U^T X \\
\Lambda^{-\frac{1}{2}} &= \mathrm{diag}(1/\sqrt{\Lambda_{ii}})
\end{aligned}
$$

- In Matlab

```
[U,D] = eig(cov(X));
Y = sqrt(inv(D)) * U' * X;
```

Whitening: example



Whitening: proof

- Let

$$
\begin{aligned}
Y &= \Lambda^{-\frac{1}{2}} U^T X \\
\Lambda^{-\frac{1}{2}} &= \mathrm{diag}(1/\sqrt{\Lambda_{ii}})
\end{aligned}
$$

- Using

$$\mathrm{Cov}[AX] = A\mathrm{Cov}[X]A^T$$

we have

$$
\begin{aligned}
\mathrm{Cov}[Y] &= \Lambda^{-\frac{1}{2}} U^T \Sigma U \Lambda^{-\frac{1}{2}} \\
&= \Lambda^{-\frac{1}{2}} U^T (U \Lambda U^T) U \Lambda^{-\frac{1}{2}} \\
&= \Lambda^{-\frac{1}{2}} \Lambda \Lambda^{-\frac{1}{2}} = I
\end{aligned}
$$

and

$$EY = \Lambda^{-\frac{1}{2}} U^T E[X]$$

Linear transformations of Gaussian RVs

- If X ~ N(μ, Σ) and Y = A X, then one can show that

$$Y \sim N(A\mu, A\Sigma A^T)$$

- Hence the whitened data is also Gaussian

$$Y \sim \mathcal{N}(\Lambda^{-\frac{1}{2}} U^T \mu, I)$$

Maximum likelihood estimation

- Log likelihood

$$\log p(X|\mu,\Sigma) = -\frac{Np}{2}\log(2\pi) - \frac{N}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)^T\Sigma^{-1}(\mathbf{x}_i - \mu)$$

- Let $\Lambda = \Sigma^{-1}$ be the precision matrix

$$\log p(X|\mu,\Sigma) = -\frac{Np}{2}\log(2\pi)\frac{N}{2}\log|\Lambda| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)^T\Lambda(\mathbf{x}_i - \mu)$$

- Just solve

$$\frac{\partial}{\partial\boldsymbol{\mu}}\log p(X|\boldsymbol{\mu},\Lambda) = 0, \frac{\partial}{\partial\Lambda}\log p(X|\boldsymbol{\mu},\Lambda) = 0$$

MLE for mean

- Log likelihood

$$\log p(X|\mu,\Sigma) = -\frac{Np}{2}\log(2\pi)\frac{N}{2}\log|\Lambda| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i-\mu)^T\Lambda(\mathbf{x}_i-\mu)$$

- Taking derivatives wrt a vector

$$\begin{aligned}\frac{\partial(\mathbf{a}^T\mathbf{y})}{\partial\mathbf{y}} &= \mathbf{a}\\ \frac{\partial(\mathbf{y}^TA\mathbf{y})}{\partial\mathbf{y}} &= (A+A^T)\mathbf{y}\end{aligned}$$

- Let $\mathbf{y}_i = \mathbf{x}_i - \boldsymbol{\mu}$ Then

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\mu}}(\mathbf{x}_i-\mu)^T\Sigma^{-1}(\mathbf{x}_i-\mu) &= \frac{\partial}{\partial\mathbf{y}_i}\frac{\partial\mathbf{y}_i}{\partial\boldsymbol{\mu}}\mathbf{y}_i^T\Sigma^{-1}\mathbf{y}_i\\ &= -1(\Sigma^{-1}+\Sigma^{-T})\mathbf{y}_i\end{aligned}$$

MLE for mean

- Log likelihood

$$\log p(X|\mu, \Sigma) = -\frac{Np}{2}\log(2\pi)\frac{N}{2}\log|\Lambda| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)^T\Lambda(\mathbf{x}_i - \mu)$$

- From before

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}}(\mathbf{x}_i - \mu)^T\Sigma^{-1}(\mathbf{x}_i - \mu) &= \frac{\partial}{\partial \mathbf{y}_i}\frac{\partial \mathbf{y}_i}{\partial \boldsymbol{\mu}}\mathbf{y}_i^T\Sigma^{-1}\mathbf{y}_i \\
&= -1(\Sigma^{-1} + \Sigma^{-T})\mathbf{y}_i
\end{aligned}$$

- Hence

$$\begin{aligned}
\frac{\partial}{\partial \mu}\log p(X|\mu, \Sigma) &= -\frac{1}{2}\sum_{i=1}^{N} -2\Sigma^{-1}(\mathbf{x}_i - \mu) \\
&= \Sigma^{-1}\sum_{i=1}^{N}(\mathbf{x}_i - \mu) = 0
\end{aligned}$$

- So finally

$$\mu_{ML} = \frac{1}{N}\sum_i \mathbf{x}_i$$

MLE for Σ

- Log likelihood

$$\log p(X|\mu,\Sigma) = -\frac{Np}{2}\log(2\pi)\frac{N}{2}\log|\Lambda| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i-\mu)^T\Lambda(\mathbf{x}_i-\mu)$$

- Trace of matrix is sum of diagonal entries

$$\mathrm{tr}(A) = \sum_i A_{ii}$$

- Cyclic permutation property of trace

$$\mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA)$$

- “Trace trick”

$$x^T A x = \mathrm{tr}(x^T A x) = \mathrm{tr}(x x^T A)$$

- Derivatives wrt a matrix

$$\begin{aligned}\frac{\partial}{\partial A}\mathrm{tr}(BA) &= B^T \\ \frac{\partial}{\partial A}\log|A| &= A^{-T}\end{aligned}$$

MLE for Σ

- Log likelihood

$$
\begin{aligned}
\ell(\mathcal{D}|\Lambda, \hat{\mu}) &= \frac{N}{2}\log|\Lambda| - \tfrac{1}{2}\sum_i (x_i-\mu)^T\Lambda(x_i-\mu) \\
&= \frac{N}{2}\log|\Lambda| - \tfrac{1}{2}\sum_i \mathrm{tr}[(x_i-\mu)(x_i-\mu)^T\Lambda] \\
&= \frac{N}{2}\log|\Lambda| - \tfrac{1}{2}\sum_i \mathrm{tr}[S\Lambda] \\
S &\stackrel{\text{def}}{=} \sum_i (\mathbf{x}_i-\overline{x})(\mathbf{x}_i-\overline{x})^T = (\sum_i \mathbf{x}_i\mathbf{x}_i^T) - N\overline{x}\overline{x}^T
\end{aligned}
$$

- Derivative

$$
\begin{aligned}
\frac{\partial \ell(\mathcal{D}|\Sigma, \hat{\mu})}{\partial \Lambda} &= \frac{N}{2}\Lambda^{-T} - \frac{1}{2}S^T = 0 \\
\Lambda^{-T} &= \Sigma = \frac{1}{N}S
\end{aligned}
$$

MLE for Σ

- MLE

$$\Sigma_{ML} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{x})(\mathbf{x}_i - \overline{x})^T$$

- In Matlab

```
Sigma = cov(X, 1)
```

- In 1d

$$\sigma^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

- Unbiased estimate

$$\Sigma_{unb} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{x})(\mathbf{x}_i - \overline{x})^T$$

- In Matlab

```
Sigma = cov(X)
```

Bayes rule for classifiers

Class-conditional density Class prior

Class posterior

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

Normalization constant

Gaussian classifiers

- Class posterior (using plug-in rule)

$$
\begin{aligned}
p(Y=c|\mathbf{x}) &= \frac{p(\mathbf{x}|Y=c)p(Y=c)}{\sum_{c'=1}^{C} p(\mathbf{x}|Y=c')p(Y=c')} \\
&= \frac{\pi_c |2\pi\Sigma_c|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_c)^T \Sigma_c^{-1}(\mathbf{x}-\mu_c)\right]}{\sum_{c'} \pi_{c'} |2\pi\Sigma_{c'}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_{c'})^T \Sigma_{c'}^{-1}(\mathbf{x}-\mu_{c'})\right]}
\end{aligned}
$$

- We will consider the form of this equation for various special cases:
- $\Sigma_1=\Sigma_0$,
- Σ_c tied, many classes
- General case

Linear/ quadratic discriminant analysis

$\Sigma_1 = \Sigma_0$

- Class posterior simplifies to

$$
\begin{aligned}
p(Y=1|\mathbf{x}) &= \frac{p(\mathbf{x}|Y=1)p(Y=1)}{p(\mathbf{x}|Y=1)p(Y=1)+p(\mathbf{x}|Y=0)p(Y=0)} \\
&= \frac{\pi_1 \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\right]}{\pi_1 \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\right] + \pi_0 \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0)\right]} \\
&= \frac{\pi_1 e^{a_1}}{\pi_1 e^{a_1} + \pi_0 e^{a_0}} = \frac{1}{1+\frac{\pi_0}{\pi_1}e^{a_0-a_1}} \\
a_c &\stackrel{\text{def}}{=} -\frac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma(\mathbf{x}-\mu_c)
\end{aligned}
$$

$\Sigma_1 = \Sigma_0$

- Class posterior simplifies to

$$
\begin{aligned}
p(Y=1|\mathbf{x}) &= \frac{1}{1+\exp\left[-\log\frac{\pi_1}{\pi_0}+a_0-a_1\right]} \\
a_0-a_1 &= -\tfrac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0)+\tfrac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1) \\
&= -(\mu_1-\mu_0)^T\Sigma^{-1}\mathbf{x}+\tfrac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0)
\end{aligned}
$$

so

Linear function of x

$$
\begin{aligned}
p(Y=1|\mathbf{x}) &= \frac{1}{1+\exp\left[-\beta^T\mathbf{x}-\gamma\right]} = \sigma(\beta^T\mathbf{x}+\gamma) \\
\beta &\stackrel{\text{def}}{=} \Sigma^{-1}(\mu_1-\mu_0) \\
\gamma &\stackrel{\text{def}}{=} -\tfrac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0)+\log\frac{\pi_1}{\pi_0} \\
\sigma(\eta) &\stackrel{\text{def}}{=} \frac{1}{1+e^{-\eta}} = \frac{e^{\eta}}{e^{\eta}+1}
\end{aligned}
$$

Decision boundary

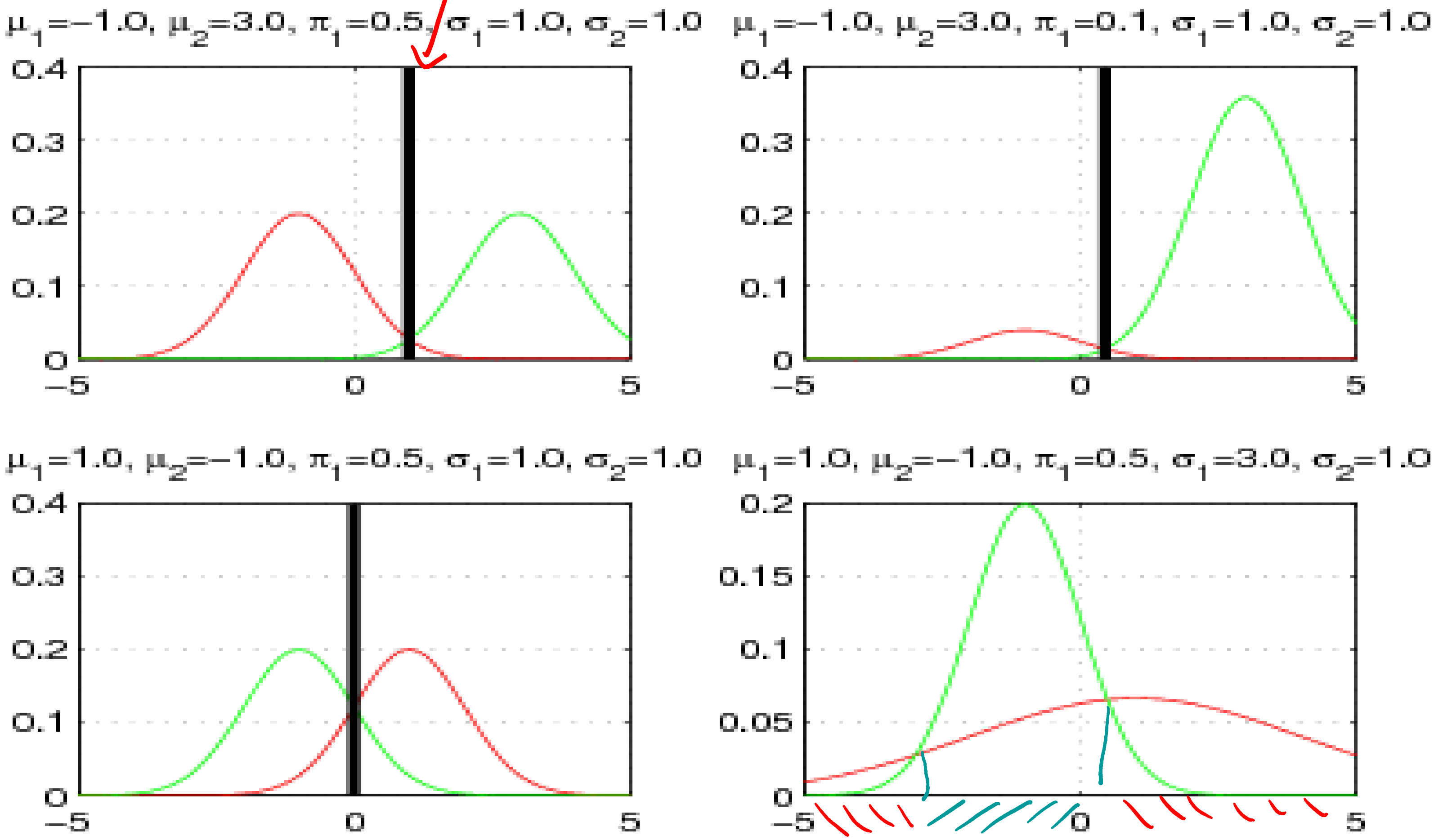
- Rewrite class posterior as

$$
\begin{aligned}
p(Y=1|\mathbf{x}) &= \sigma(\boldsymbol{\beta}^T\mathbf{x}+\gamma) = \sigma(\mathbf{w}^T(\mathbf{x}-\mathbf{x}_0)) \\
\mathbf{w} &= \beta = \Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0) \\
\mathbf{x}_0 &= -\frac{\gamma}{\boldsymbol{\beta}} = \tfrac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_0) - \frac{\log(\pi_1/\pi_0)}{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^T\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)
\end{aligned}
$$

- If $\Sigma=I$, then $w=(\mu_1-\mu_0)$ is in the direction of $\mu_1-\mu_0$, so the hyperplane is orthogonal to the line between the two means, and intersects it at x_0
- If $\pi_1=\pi_0$, then $x_0 = 0.5(\mu_1+\mu_0)$ is midway between the two means
- If π_1 increases, x_0 decreases, so the boundary shifts toward μ_0 (so more space gets mapped to class 1)

Decision boundary in 1d

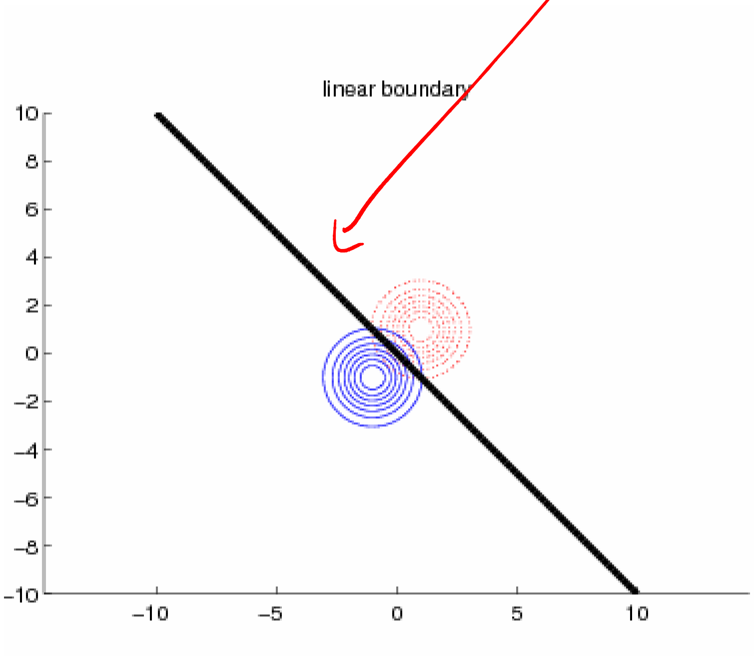
$p(y=1|x) = p(y=0|x)$

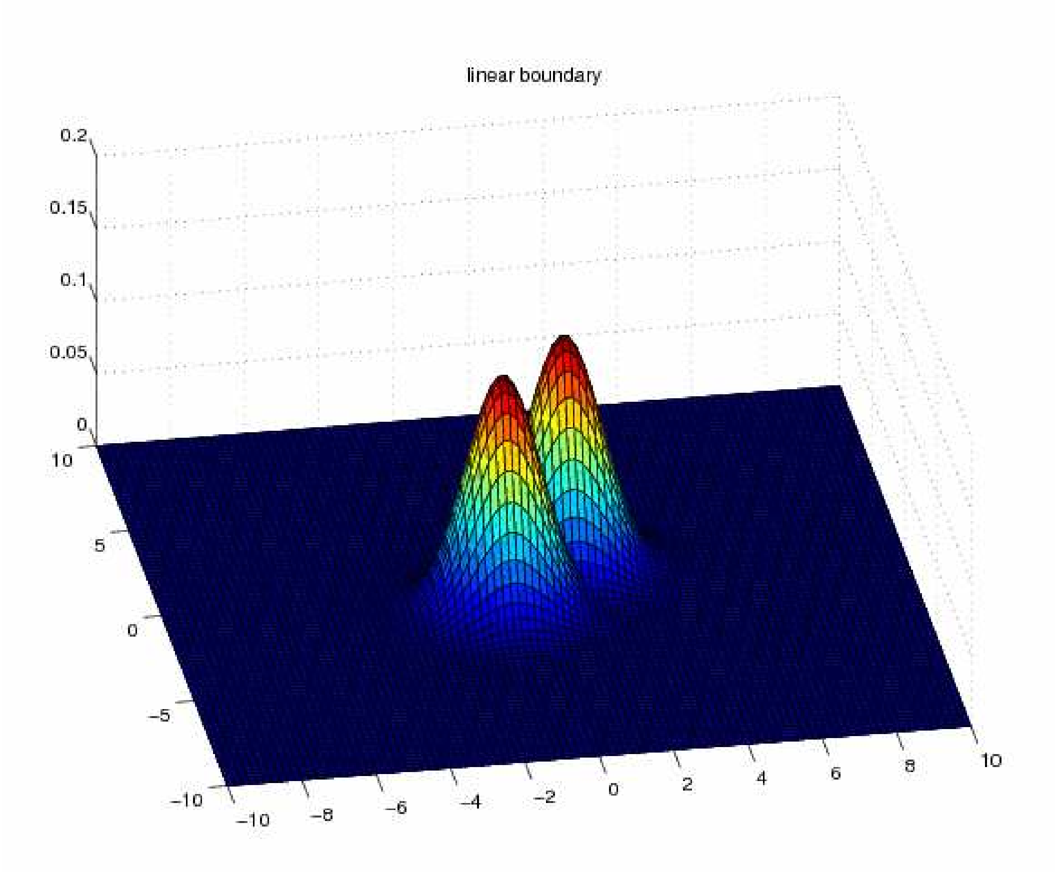


Discontinuous decision region

Decision boundary in 2d

$p(y=1|x) = p(y=0|x)$





Tied Σ, many classes

- Similarly to before

$$
\begin{aligned}
p(Y=c|\mathbf{x}) &= \frac{\pi_c \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_c)^T \Sigma_c^{-1}(\mathbf{x}-\mu_c)\right]}{\sum_{c'} \pi_{c'} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_{c'})^T \Sigma_{c'}^{-1}(\mathbf{x}-\mu_{c'})\right]} \\
&= \frac{\exp\left[\mu_{'}^T \Sigma^{-1} x - \frac{1}{2}\mu_c^T \Sigma^{-1}\mu_c + \log \pi_c\right]}{\sum_{c'} \exp\left[\mu_{c'}^T \Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_{c'}^T \Sigma^{-1}\mu_{c'} + \log \pi_{c'}\right]} \\
\theta_c &\stackrel{\text{def}}{=} \begin{pmatrix} -\mu_c^T \Sigma^{-1}\mu_c + \log \pi_c \\ \Sigma^{-1}\mu_c \end{pmatrix} = \begin{pmatrix} \gamma_c \\ \beta_c \end{pmatrix} \\
p(Y=c|\mathbf{x}) &= \frac{e^{\theta_c^T \mathbf{x}}}{\sum_{c'} e^{\theta_{c'}^T \mathbf{x}}} = \frac{e^{\beta_c^T \mathbf{x} + \gamma_c}}{\sum_{c'} e^{\beta_{c'}^T \mathbf{x} + \gamma_{c'}}}
\end{aligned}
$$

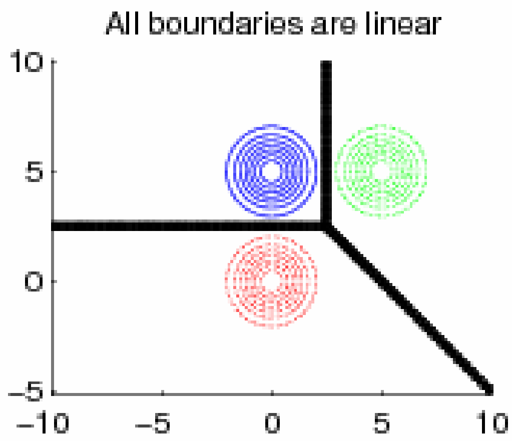
- This is the multinomial logit or softmax function

Tied Σ, many classes

- Discriminant function

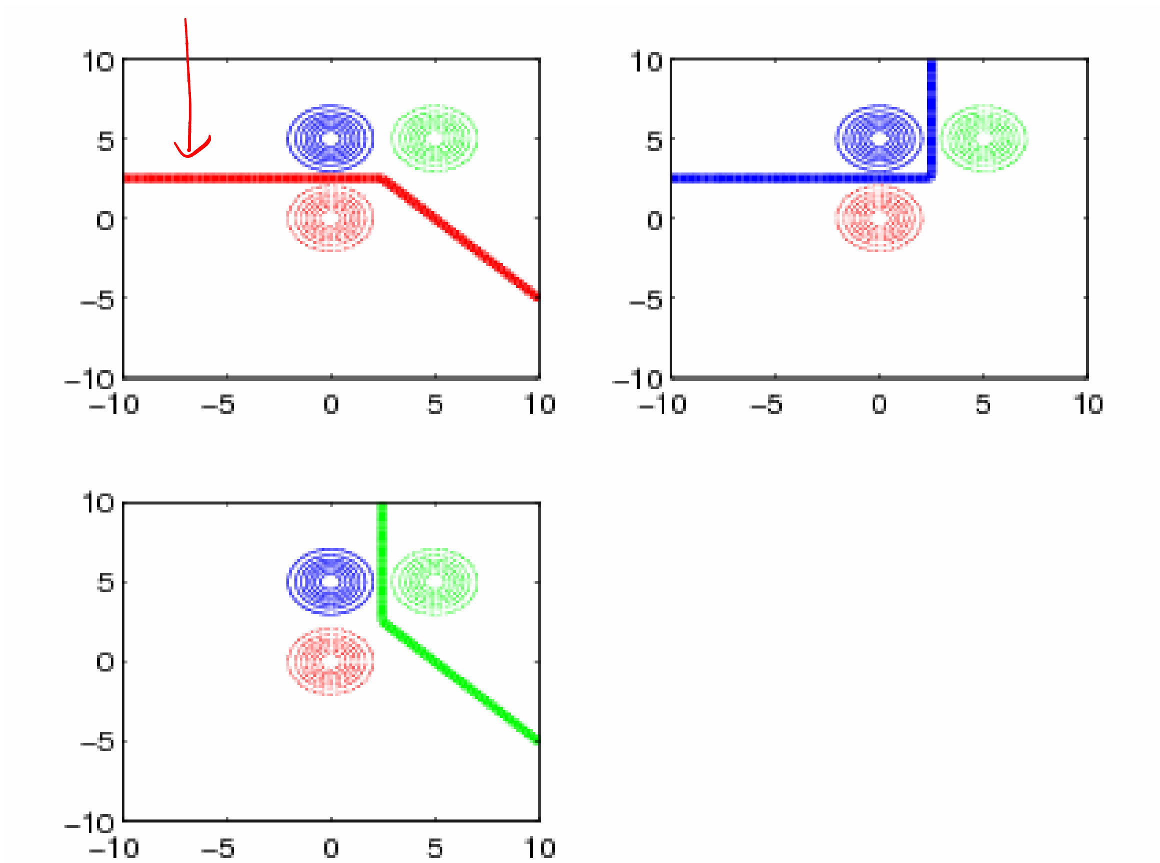
$$
\begin{aligned}
g_c(\mathbf{x}) &= -\tfrac{1}{2}(\mathbf{x}-\mu_c)^T \Sigma^{-1}(\mathbf{x}-\mu_c) + \log p(Y=c) = \beta_c^T \mathbf{x} + \beta_{c0} \\
\beta_c &= \Sigma^{-1}\mu_c \\
\beta_{c0} &= -\tfrac{1}{2}\mu_c^T \Sigma^{-1}\mu_c + \log \pi_c
\end{aligned}
$$

- Decision boundary is again linear, since $x^T \Sigma x$ terms cancel
- If $\Sigma = I$, then the decision boundaries are orthogonal to $\mu_i - \mu_j$, otherwise skewed



Decision boundaries

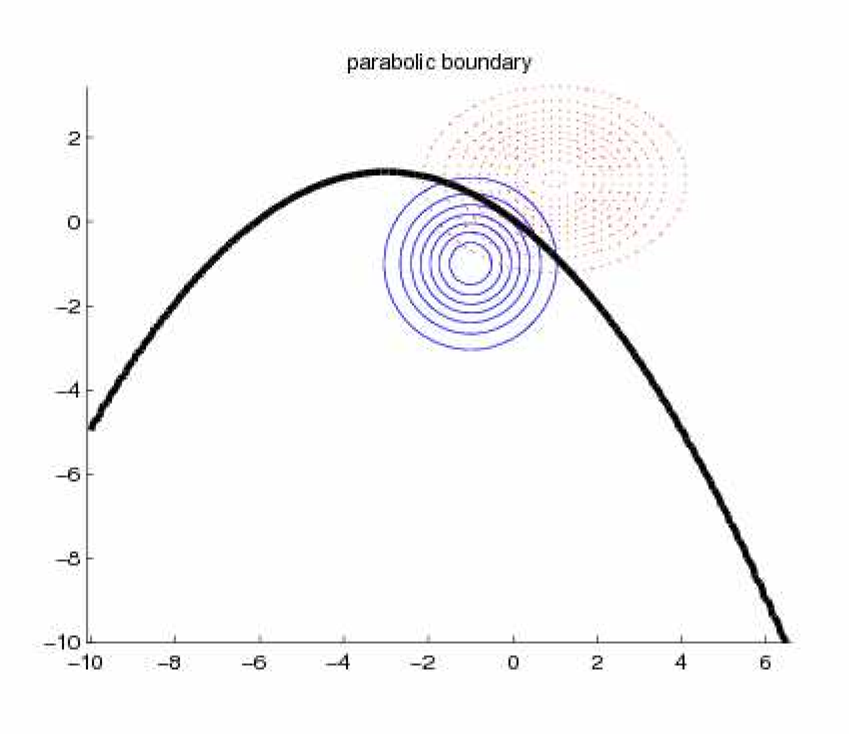
$g_1(x) - \max(g_2(x), g_3(x)) = 0$

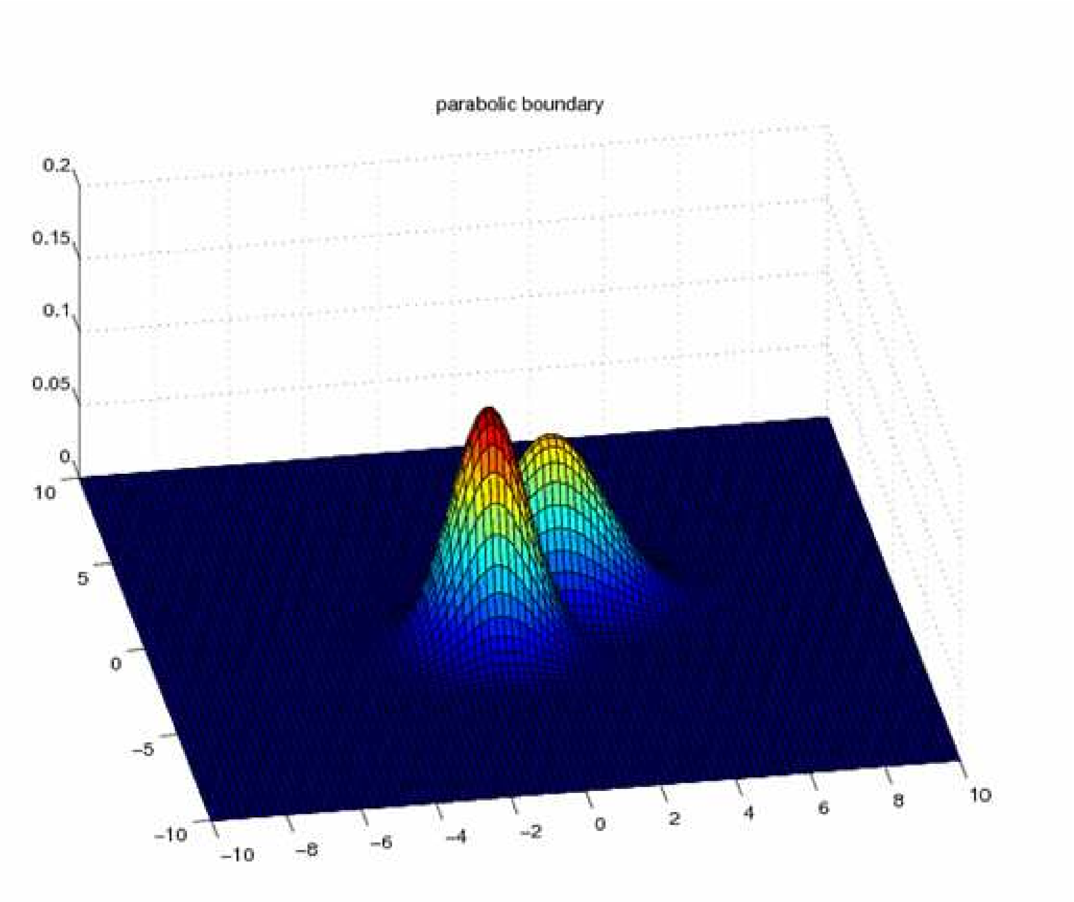



```
[x,y] = meshgrid(linspace(-10,10,100), linspace(-10,10,100));
g1 = reshape(mvnpdf(X, mu1(:)', S1), [m n]); ...
contour(x,y,g2*p2-max(g1*p1, g3*p3),[0 0],'-k');
```

Σ_0, Σ_1 arbitrary

- If the Σ are unconstrained, we end up with cross product terms, leading to quadratic decision boundaries



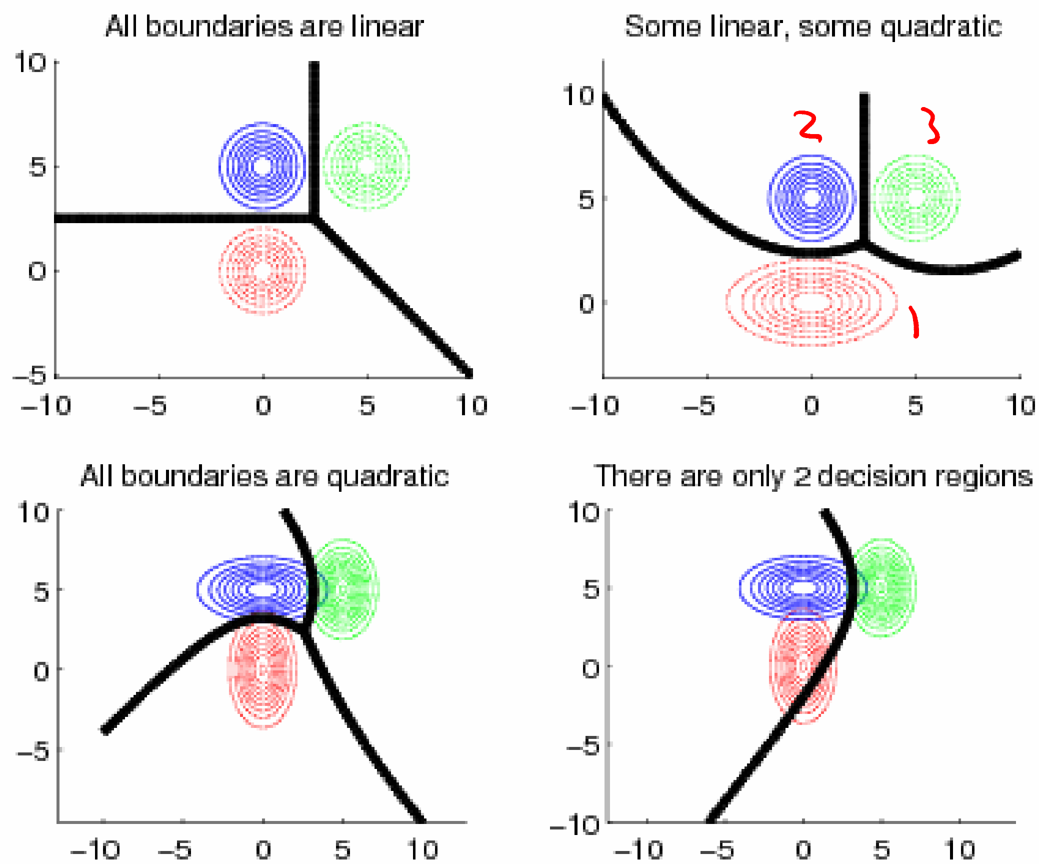


General case

$$\mu_1 = (0,0), \mu_2 = (0,5), \mu_3 = (5,5), \pi = (1/3, 1/3, 1/3)$$

$\Sigma_c = I$

$\Sigma_1 = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}, \Sigma_2 = \Sigma_3 = I$



$\Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$

$\Sigma_2 = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$

$\Sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$

$\Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$

$\Sigma_2 = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$

$\Sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$

$\pi = (0, 1/2, 1/2)$