LUND UNIVERSITY

L-4: Linear classification methods

Ted Kronvall, Cristian Sminchisescu

Admin

Moodle:

- Register for ”Machine Learning 2019” in Moodle by the 14th of April.
- Use enrollment key ”clustering2019”.
- Sign up for the Q & A sessions for Assignment 1 - the first one is tomorrow!

Small recap from L-3

- In convex optimization, local minima are global. This is not true in general for non-convex problems.
- MSE, LASSO, ridge regression, SVM, logistic regression are convex problems, neural network training is not.
- One may check if a problem is convex using the definition, the second-order derivative, the epigraph, or by identifying it as a composition of convex functions.
- Optimization problems may be either unconstrained or constrained.
  - For an unconstrained problem, the optimal point has its derivative equal to zero.
  - For a constrained problem, the derivative of the Lagrangian function w.r.t. the primal variables is zero at the optimal point.
- The KKT conditions are necessary and sufficient for a point to be optimal in constrained optimization.
- Coordinate descent updates the coordinates in the variable on at a time, without needing to calculate the derivative.
- Gradient and steepest descent use the first-order derivative. Gradient descent is the most common approach in deep learning.

Today

- ~~Basic concepts; tasks, training, validation, regularization~~.
- ~~Probability theory and optimization~~
- ~~Linear regression~~
- Linear methods for classification

- Clustering methods
- Neural networks

- Convolutional and recurrent NNs
- Generative modeling

- Reinforcement learning
- Natural language processing
- Deep learning for computer vision

The classification model

Classification model

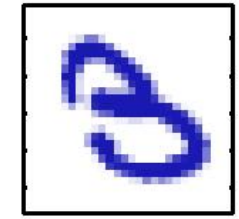
NOTATIONAL CONVENTIONS

- Recall how notation is used in machine learning problems:

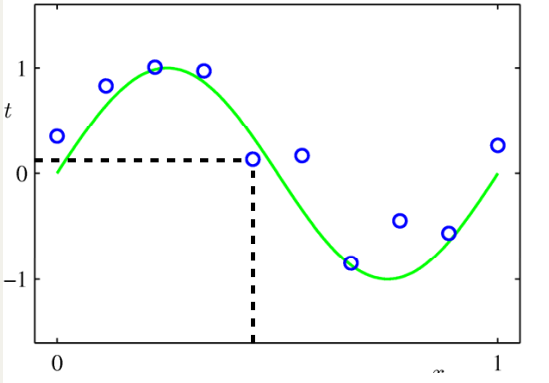
$t_i, \mathbf{t}, \mathbf{T}$	Targets
$x_i, \mathbf{x}, \mathbf{X}$	Inputs
$y_i, \mathbf{y}, \mathbf{Y}$	Model output
$w_i, \mathbf{w}, \mathbf{W}$	trainable variables
$\lambda, \mu, \rho, \ldots$	(capacity) hyperparameters

= Data

no targets = unsupervised data

$\mathbf{X_i} =$ 

$t_i = 3$

t_i 

x_i

Classification model

Input x_i $\xrightarrow{y(x_i)}$ Output t_i

$\mathbf{X_i} =$ 

$\mathbf{x}_i = [\text{weight}_i, \text{height}_i]$

$t_i = 3$ or

$\mathbf{t}_i = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

$t_i \in \{\text{man}, \text{woman}\}$ or

$\mathbf{t}_i \in \{[1, 0], [0, 1]\}$ or

$t_i \in \{-1, 1\}$

Classification model

Linear Classification

- Classification is intrinsically non-linear because of the training constraints that ***place non-identical inputs in the same class***
- Differences in the input vector sometimes causes 0 change in the answer
- Linear classification means that the adaptive part $\mathbf{w}$ is linear
  - The adaptive part is cascaded with a fixed non-linearity f
  - It may also be preceded by a fixed non-linearity ϕ when nonlinear basis functions are used

fixed non-linear functions

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0, \qquad c = f(y(\mathbf{x}))$$

adaptive linear parameters decision

Classification model

Approach 1: Discriminant Function

- Use discriminant functions directly, and do not compute probabilities
- Convert the input vector into one or more real values so that a simple process (thresholding, or a majority vote) can be applied to assign the input to the class
- The real values should be chosen to maximize the useable information about the class label present in the real value
- Given discriminant functions $f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})$
  Classify $\mathbf{x}$ as class C_k, **iff** $f_k(\mathbf{x}) > f_j(\mathbf{x}), \forall j \neq k$

LUND UNIVERSITY

Classification model

Approach 2: Class-conditional Probabilities

- Infer conditional class probabilities $p(C_k|\mathbf{x})$
- Use conditional distribution to make optimal decisions, e.g. by minimizing some loss function

- Example, 2 classes

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\phi), \qquad p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$$

$$\text{where } \sigma(a) = \frac{1}{1+\exp(-a)}$$

Classification model

Represent Target Values: *Binary* vs. *Multiclass*

- **Two classes (*N=2)*:** typically use a single real valued output that has target values of 1 for the *positive class* and 0 (or -1) for the *negative class*
  - For probabilistic class labels, the target can be the probability of the positive class and the output of the model can be the probability the model assigns to the positive class

$$t_i \in \{-1, 1\}$$

- **For the multiclass (*N>2*)**, we use a vector of *N* target values containing a single 1 for the correct class and zeros elsewhere
  - For probabilistic labels we can then use a vector of class probabilities as the target vector

$$\mathbf{t}_i = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$$

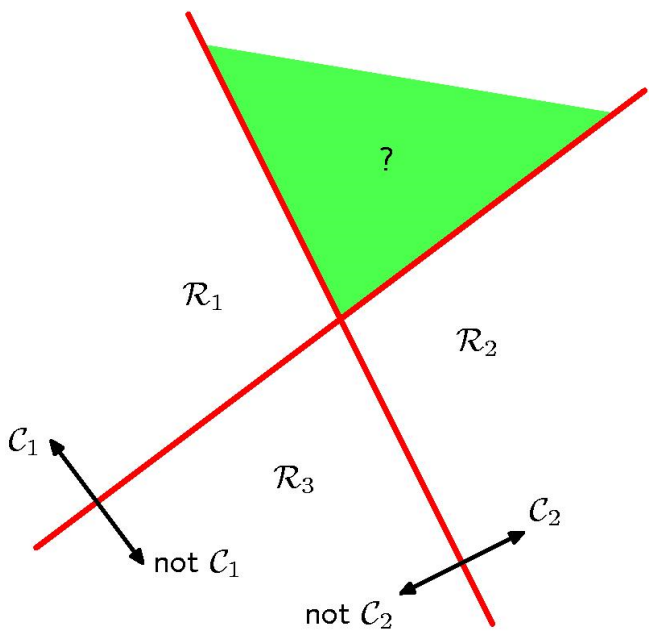
LUND UNIVERSITY

Discriminant Functions for Multiclass

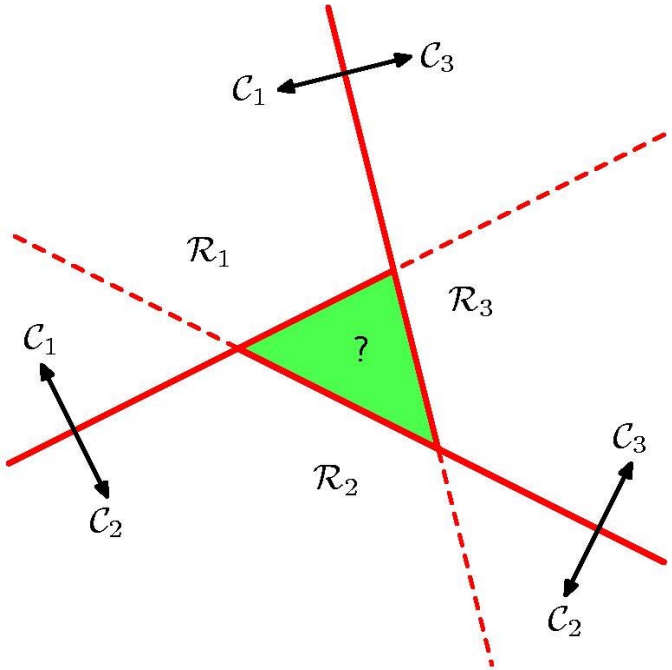
- One possibility is to use N binary (2-way) discriminants
  - Each function separates one class from the rest
- Another possibility is to use $\frac{N(N-1)}{2}$ binary (2-way) discriminants
  - Each function discriminates between two specific classes. We have 1 discriminant for each class pair
- Both methods have ambiguities

LUND UNIVERSITY

Problems with Multi-class Discriminant Functions Constructed from Binary Classifiers



1 - vs.- all



1 vs. 1

If we base our decision on binary classifiers, we can encounter ambiguities

Simple Solution

Use N discriminant functions, $y_i, y_j, y_k, \ldots$, and

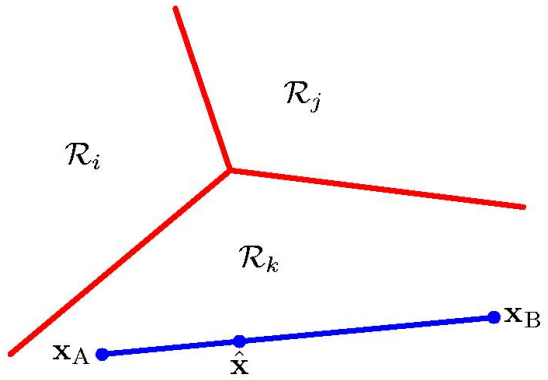
take the max over their response. Why?

- Consider linear discriminants y

$$y_k(\mathbf{x}) = w_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

- The decision boundary between class k and j is given by the $D-1$ hyperplane

$$(w_k^{\mathrm{T}} - w_j^{\mathrm{T}})\mathbf{x} + (w_{k0} - w_{j0}) = 0$$



In this linear case the decision regions are convex

$\mathbf{x}_A, \mathbf{x}_A \in R_K,\ \hat{\mathbf{x}} = \lambda\mathbf{x}_A + (1-\lambda)\mathbf{x}_B, 0 \le \lambda \le 1$

From the linearity of $y \Rightarrow y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1-\lambda)y_k(\mathbf{x}_B)$

But $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A)$ and $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$ $\quad \forall j \neq k \Rightarrow y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$

$\Rightarrow \hat{\mathbf{x}}$ also lies inside R_k

Hence R_k is convex

Least squares for classification

LUND UNIVERSITY

Least Squares Classification

Assume each class has its own linear model: $y_k(\mathbf{x}) = w_k^{\mathrm{T}}\mathbf{x} + w_{k0}$

Then we can write:

$y(\mathbf{x}) = \widetilde{\mathbf{W}}^T\tilde{\mathbf{x}}$, with k-th column a $D+1$-dim vector $\widetilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_K^T)^T$, $\tilde{\mathbf{x}} = (1, \mathbf{x}^T)^T$

decision surfaces are D-dimensional hyperplanes passing through the origin of the D + 1-dimensional expanded input space.

Given $\{\mathbf{x}_n, \mathbf{t}_n\}, n = 1, \ldots, N$; $n-th$ row of $\mathbf{T}$ is $\mathbf{t}_n^T$; $\widetilde{\mathbf{X}}$'s $n-th$ row is $\tilde{\mathbf{x}}_n^T$

The sum of squares error function for classification is:

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^T(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\}$$

$$\frac{\partial E_D}{\partial \widetilde{\mathbf{W}}} = 0 \Rightarrow \widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^T\mathbf{T} = \widetilde{\mathbf{X}}^+\mathbf{T}$$

$$\boldsymbol{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T\tilde{\mathbf{x}} = \mathbf{T}^T(\widetilde{\mathbf{X}}^+)^T\tilde{\mathbf{x}} \quad \leftarrow \text{Closed-form solution}$$

$\mathbf{X}^+$ is the pseudoinverse of $\mathbf{X}$

Property: every vector in the training set and the model prediction for any value of $\mathbf{x}$, satisfy some linear constraint:

$$\mathbf{a}^T\mathbf{t}_N + b = 0,\ \mathbf{a}^T\mathbf{y}(\mathbf{x}) + b = 0, \text{for some constants } \mathbf{a}, b.$$

LUND UNIVERSITY

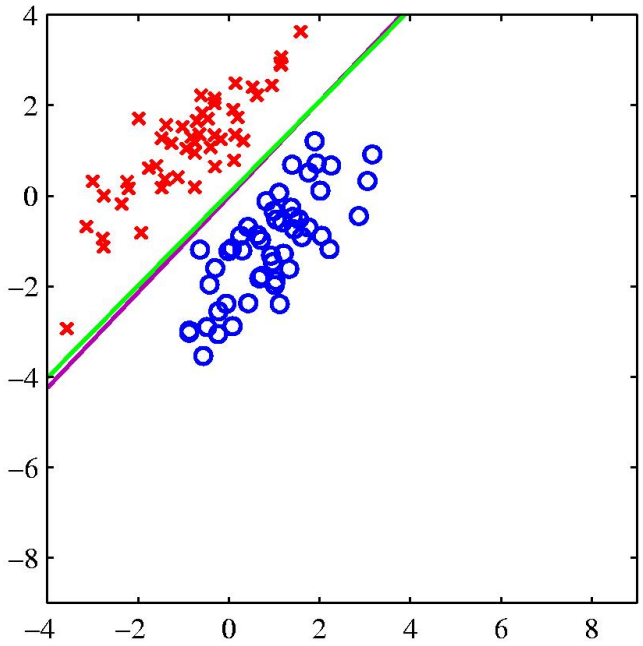
Classification with LS

[Matlab example]

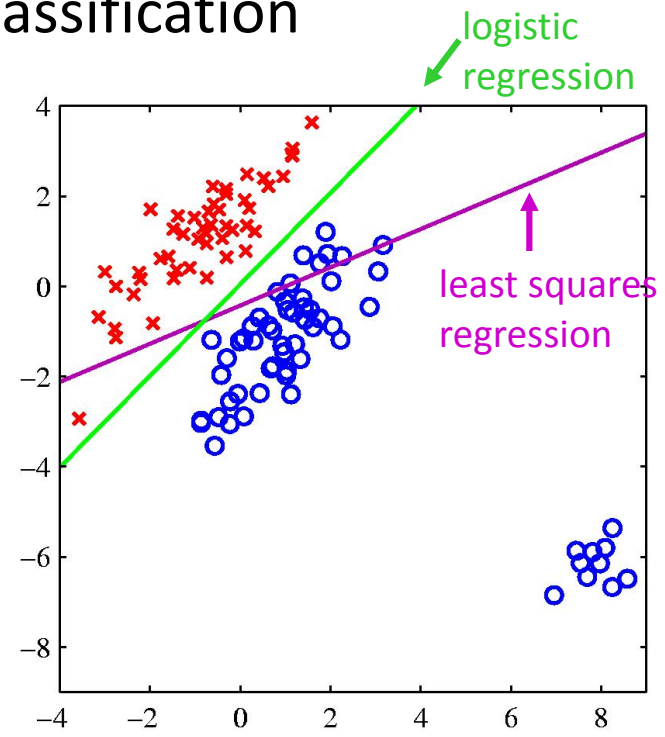
LUND UNIVERSITY

Classification with LS

Problems with using least squares for classification



Least squares solutions lack robustness to outliers

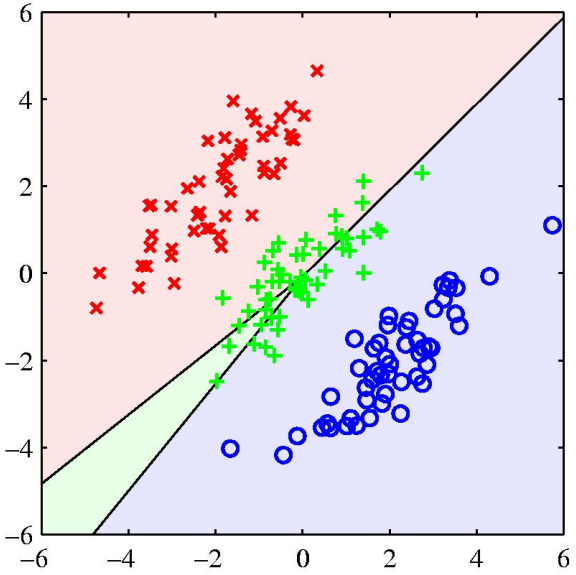


If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being “too correct”

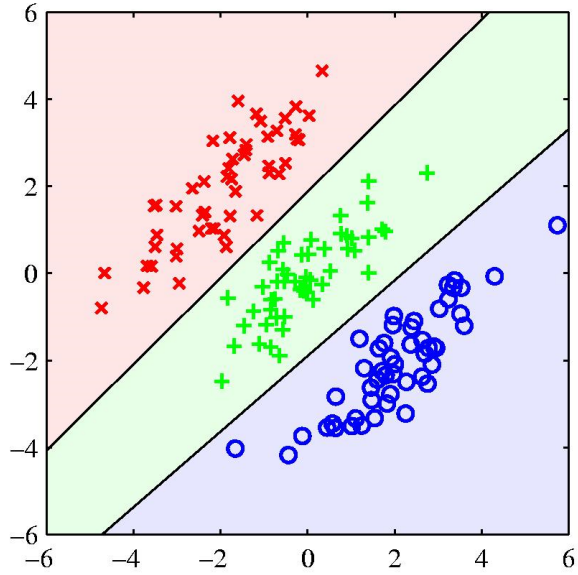
Classification with LS

For non-Gaussian targets, least squares regression gives poor decision surfaces

Least Squares



Logistic Regression



Remember that least squares corresponds to the Maximum Likelihood under a Gaussian conditional distribution

Clearly the binary target vectors have a distribution that is far from Gaussian

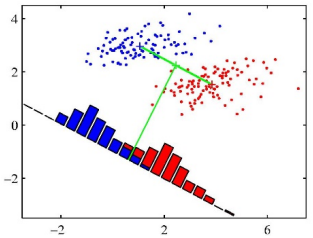
Fisher's linear discriminant (LDA)

LUND UNIVERSITY

Linear discriminant analysis

Fisher's Linear Discriminant

- We can view classification in terms of dimensionality reduction
- A simple linear discriminant function is a projection of the $D-$ dimensional data $\mathbf{x}$ down to 1 dimension



Project: $y = \boldsymbol{w}^T\mathbf{x}$;

Classify: if $y \geq -w_0$ then C_1 else C_2

- However projection results in loss of information. Classes well separated in the original input space may strongly overlap in 1d
- We will adjust the projection (weight vector $\mathbf{w}$) to achieve the best separation among classes. But what do we mean by *best separation*?

Fisher's View of Class Separation (I)

- The simplest measure of class separation when projected onto $\mathbf{w}$ is the separation of the projected class means. This suggests choosing $\mathbf{w}$ so to maximize

$$m_2 - m_1 = \mathbf{w}^{\mathbf{T}}(\mathbf{m}_2 - \mathbf{m}_1), m_k = \mathbf{w}^{\mathbf{T}}\mathbf{m}_{\mathrm{k}}, \mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in C_1}\mathbf{x}_{\boldsymbol{n}}, \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in C_2}\mathbf{x}_{\boldsymbol{n}}$$

- This can be made arbitrarily large by increasing $\left|\left|\mathbf{w}\right|\right|$. We could handle this by imposing unit norm constraints using Lagrange multipliers. We get

$$\max_{\mathbf{w}}\mathbf{w}^{\mathbf{T}}(\mathbf{m}_2 - \mathbf{m}_1), \text{ s.th. } \left|\left|\mathbf{w}\right|\right| = \mathbf{1}$$

$$\Rightarrow \mathbf{w} = -\frac{\mathbf{1}}{\mathbf{2}\boldsymbol{\lambda}}(\mathbf{m_2}-\mathbf{m_1}) \propto \mathbf{m_2} - \mathbf{m_1}$$

- However, if the main direction of variance in each class is not orthogonal to the direction between means, we will not get good separation (see next slide)

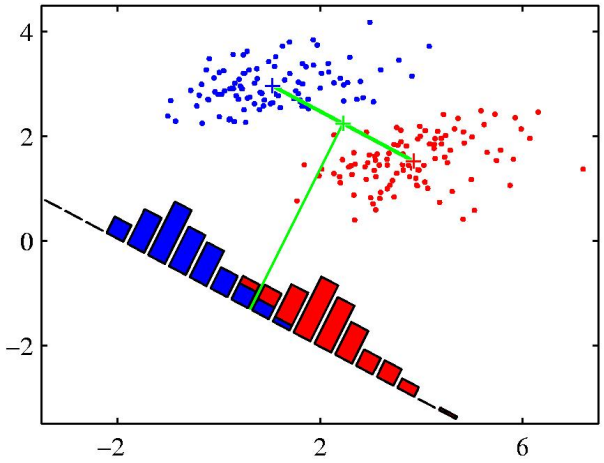
LUND UNIVERSITY

Linear discriminant analysis

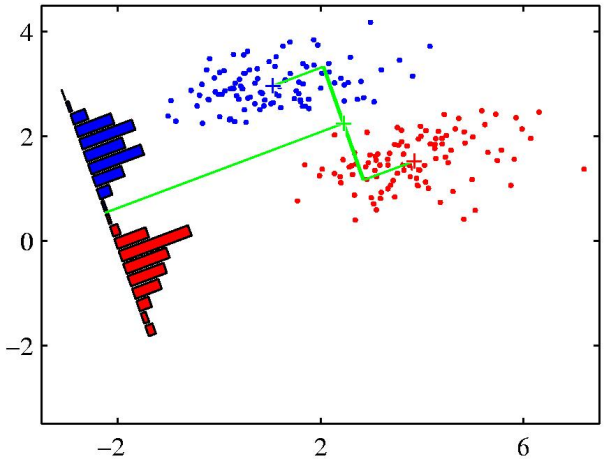
[Matlab example]

Linear discriminant analysis

Advantage of using Fisher's Criterion



When projected onto the line joining the class means, the classes are not well separated



Fisher chooses a direction that makes the projected classes much tighter, even though their projected means are less far apart

Fisher's View of Class Separation (II)

- **Fisher**
  - maximize a function that gives a large separation between the projected class means,
  - while also giving a small variance within each class
  - thereby minimizing class overlap

  - Choose direction maximizing the **ratio** of **between class** variance to **within class** variance
  - This is the direction in which the projected points contain the most information about class membership (under Gaussian assumptions)

Linear discriminant analysis

Fisher's Linear Discriminant

- We seek a linear transformation that is best for discrimination

$$y = \mathbf{w}^T \mathbf{x}$$

- The projection onto the vector separating the class means seems right

$$\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$$

- But we also want small variance within each class

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2, \qquad m_k = \mathbf{w}^{\mathbf{T}} \mathbf{m}_{\mathrm{k}}$$

- Fisher's objective function

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

← Between class

← Within class

Linear discriminant analysis

Criterion Derivation

$$
\begin{aligned}
(m_2 - m_1)^2 &= \left(\mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)\right)^2 \\
&= \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}\mathbf{w} \\
&= \mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}.
\end{aligned}
$$

$$
\begin{aligned}
s_1^2 + s_2^2 &= \sum_{n \in \mathcal{C}_1} (y_n - m_1)^2 + \sum_{k \in \mathcal{C}_2} (y_k - m_2)^2 \\
&= \sum_{n \in \mathcal{C}_1} \left(\mathbf{w}^{\mathrm{T}}(\mathbf{x}_n - \mathbf{m}_1)\right)^2 + \sum_{k \in \mathcal{C}_2} \left(\mathbf{w}^{\mathrm{T}}(\mathbf{x}_k - \mathbf{m}_2)\right)^2 \\
&= \sum_{n \in \mathcal{C}_1} \mathbf{w}^{\mathrm{T}}(\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}}\mathbf{w} \\
&\quad + \sum_{k \in \mathcal{C}_2} \mathbf{w}^{\mathrm{T}}(\mathbf{x}_k - \mathbf{m}_2)(\mathbf{x}_k - \mathbf{m}_2)^{\mathrm{T}}\mathbf{w} \\
&= \mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}.
\end{aligned}
$$

LUND UNIVERSITY

Linear discriminant analysis

Fisher's Linear Discriminant Derivations

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

where

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

LUND UNIVERSITY

Fischer's Linear Discriminant Computation

However, the objective $J(\mathbf{w})$ is invariant to rescaling $\mathbf{w} \rightarrow \alpha\mathbf{w}$. We can chose the denominator to be unity. We can then minimize

$$\min_{\mathbf{w}} -\frac{1}{2}\mathbf{w}^T\mathbf{S}_B\mathbf{w}$$

$$\mathbf{w}^T\mathbf{S}_W\mathbf{w} = 1$$

This corresponds to the (primal) Lagrangian

$$L_P = -\frac{1}{2}\mathbf{w}^T\mathbf{S}_B\mathbf{w} + \frac{1}{2}\lambda(\ \mathbf{w}^T\mathbf{S}_W\mathbf{w} - 1)$$

From the *KKT conditions*

$$\mathbf{S}_B\mathbf{w} = \boldsymbol{\lambda}\mathbf{S}_W\mathbf{w} \Rightarrow \mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \boldsymbol{\lambda}\mathbf{w}$$

Generalized eigenvalue problem, as $\mathbf{S}_W^{-1}\mathbf{S}_B$ not symmetric

LUND UNIVERSITY

Linear discriminant analysis

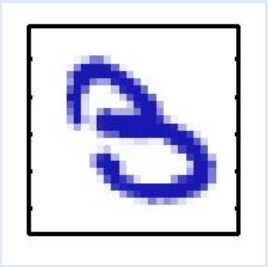
[Matlab example]

Logistic regression

LUND UNIVERSITY

Logistic regression

Input x_i $\xrightarrow{y(x_i)}$ Output t_i

$\mathbf{X_i} =$ 

$t_i = 3$ or

$\mathbf{t}_i = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

- Let's model the outputs as the probabilities for each class!
- One class can be removed, as probabilities should sum to one.

Logistic regression

Probabilistic Generative Models

- Use a class prior and a separate generative model of the input vectors for each class, and compute which model makes a test input vector most probable
- The posterior probability of class 1 is given by:

Logistic sigmoid

$$p(C_1 \mid \mathbf{x}) = \frac{p(C_1)p(\mathbf{x} \mid C_1)}{p(C_1)p(\mathbf{x} \mid C_1) + p(C_2)p(\mathbf{x} \mid C_2)} = \frac{1}{1+e^{-a}}$$

$$\textit{where} \quad a = \ln \frac{p(C_1)p(\mathbf{x} \mid C_1)}{p(C_2)p(\mathbf{x} \mid C_2)} = \ln \frac{p(C_1 \mid \mathbf{x})}{1 - p(C_1 \mid \mathbf{x})}$$

z is called the logit and is given by the log odds

Logistic regression

The Logistic **S**igmoid (due to **S**-shape)

- This is also called a squashing function because it maps the entire real axis into a finite interval

- For classification, the output a is a smooth function of the inputs and the weights $\mathbf{w}$ $\longrightarrow$ $a = \mathbf{w}^T \mathbf{x} + w_0$

- Properties

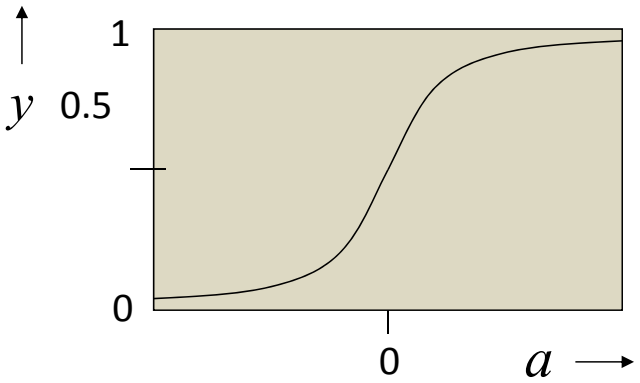
$$\sigma(-a) = 1 - \sigma(a), \qquad a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

logit function

$$y = \sigma(a) = \frac{1}{1+e^{-a}}$$

$$\frac{\partial a}{\partial w_i} = x_i \qquad \frac{\partial a}{\partial x_i} = w_i$$

$$\frac{dy}{da} = y\,(1-y)$$



Logistic regression

Multiclass Model (Softmax)

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)}$$

$$= \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\mathbf{x}|C_k)p(C_k)$

- This is known as the *normalized exponential*
- Can be viewed as a multiclass generalization of the logistic sigmoid
- It is also called a *softmax function* (it is a smoothed version of `max')

$$\text{if } a_k \gg a_j \; \forall j \neq k, \text{then } p(C_k|x) \simeq 1 \text{ and } p(C_j|\, x) \simeq 0$$

LUND UNIVERSITY

Logistic regression

Probabilistic Discriminative Models
Logistic Regression

- In our discussion of generative approaches, we saw that under general assumptions, the class posterior for C_1 can be written as a logistic sigmoid acting on a linear function of the feature vector ϕ
- In logistic regression, we use the functional form of the generalized linear model explicitly

$$p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi), \text{where } \sigma(a) = \frac{1}{1+\exp(-a)}$$

Logistic regression

Maximum Likelihood for *Logistic Regression*

$$p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi);\ \sigma(a) = \frac{1}{1+\exp(-a)}, \frac{d\sigma}{da} = \sigma(1-\sigma)$$

For dataset $\{\phi_n \equiv \phi(\mathbf{x}_n), t_n\}$, with $t_n \in \{0,1\}, n = 1, \ldots, N$,

$$\mathbf{t} = (t_1, t_2, \ldots, t_N)^T, \qquad y_n = p(C_1|\phi_n) = \sigma(a_n), a_n = \mathbf{w}^T\phi_n$$

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

Cross-entropy error

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

Similar form as the gradient of the sum-of-squares regression model

LUND UNIVERSITY

Logistic regression

[Matlab example]

Self study:

- **Goodfellow et al 2016 (Deep Learning):**
  - -
- **Bishop 2006 (Pattern Recognition and Machine Learning)**
  - Chapter 4: 4.1 (except 4.1.7), 4.3 up to 4.3.2, and 4.3.4.
- **Hastie et al. 2019 (Elements of Statistical Learning)**
  - Chapter 4: 4.1-4.4

LUND UNIVERSITY

Small recap from L-3

[Blackboard examples]

LUND UNIVERSITY

LUND UNIVERSITY

L-4: Linear classification methods

Ted Kronvall, Cristian Sminchisescu