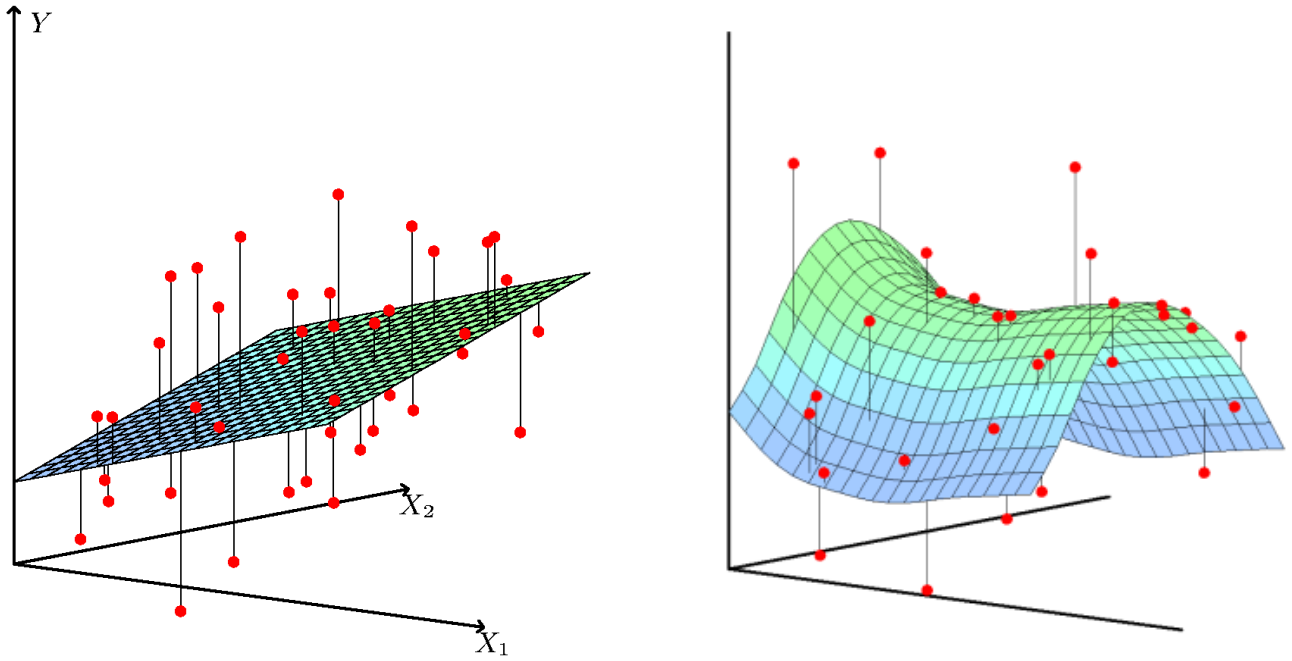
Introduction to Supervised Learning



Week 2: Maximum Likelihood Parameter Estimation, Linear Regression

Iasonas Kokkinos

i.kokkinos@cs.ucl.ac.uk

University College London

Practicalities

Deliverables: report + code

Due date: late October/early November

Week 1: recap

- Overview of course
- Review of probability
- Bayes' theorem

$$P(A = a|B = b) = \frac{P(B = b|A = a)P(A = a)}{\sum_{a'} P(B = b|A = a')P(A = a')}$$

- Bayes-Optimal classification rule (minimum probability of error)

Decide for class k whenever it has the greatest posterior probability of all classes:

$$p(\mathcal{C}_k|x) > p(\mathcal{C}_j|x) \quad \forall j \neq k$$

Generative vs. Discriminative classifiers

$$y_k(x) \propto p(x|\mathcal{C}_k)p(\mathcal{C}_k)$$

- First determine the class-conditional densities for each class individually and separately infer the prior class probabilities.
- Then use Bayes' theorem to determine class membership.

$\Rightarrow$ *Generative methods*

Generative vs. Discriminative classifiers

$$y_k(x) \propto p(x|\mathcal{C}_k)p(\mathcal{C}_k)$$

- First determine the class-conditional densities for each class individually and separately infer the prior class probabilities.
- Then use Bayes' theorem to determine class membership.

$\Rightarrow$ *Generative methods*

$$y_k(x) = p(\mathcal{C}_k|x)$$

- First solve the inference problem of determining the posterior class probabilities.
- Then use decision theory to assign each new x to its class.

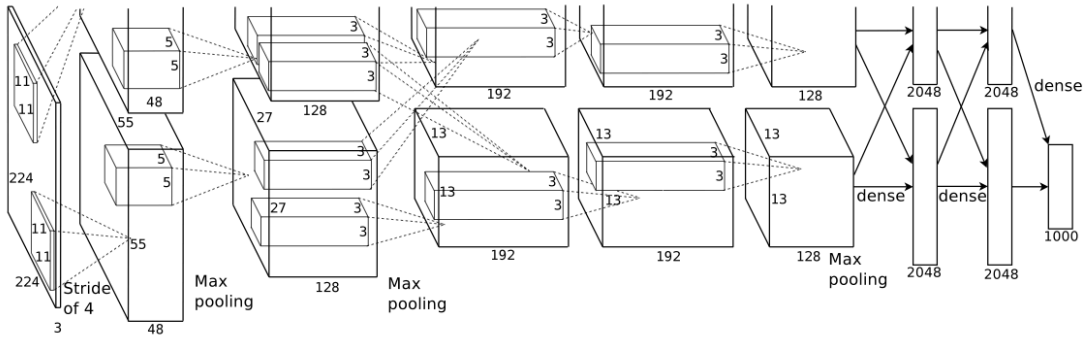
$\Rightarrow$ *Discriminative methods*

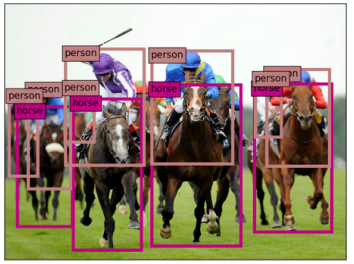
Tentative Course Schedule: generative & discriminative

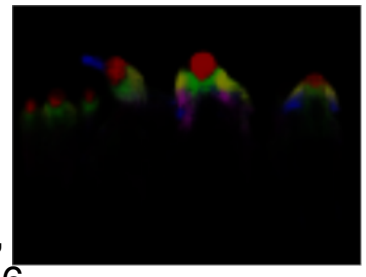
- 1st week: Introduction, fundamental concepts
- 2nd week: Linear Regression
- 3rd week: Logistic Regression
- 4th week: Support Vector Machines
- 5th week: Ensemble Models (Adaboost, Random Forests)
- 6th week: Unsupervised learning (K-means, PCA, Sparse Coding)
- 7th week: Deep Learning (neural networks, backpropagation, SGD)
- 8th week: Probabilistic modelling (hidden variable models, EM)
- 9th week: Intro to Structured Prediction (Random Fields, Graphical Models)
- 10th week: review and applications

Two Main Approaches

- Discriminative



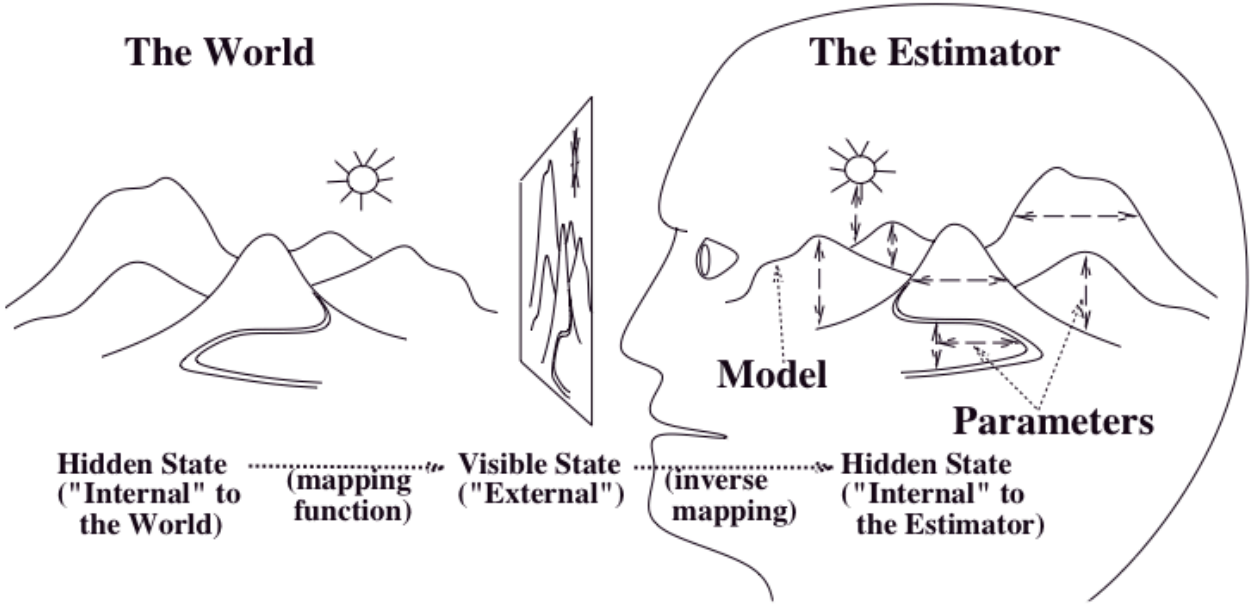




I. Kokkinos, UberNet : Training a ‘Universal’ Convolutional Neural Network for Low-, Mid-, and High-Level Vision using Diverse Datasets and Limited Memory, arxiv, 2016

Two Main Approaches

- Generative



Likelihood Prior

Posterior

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

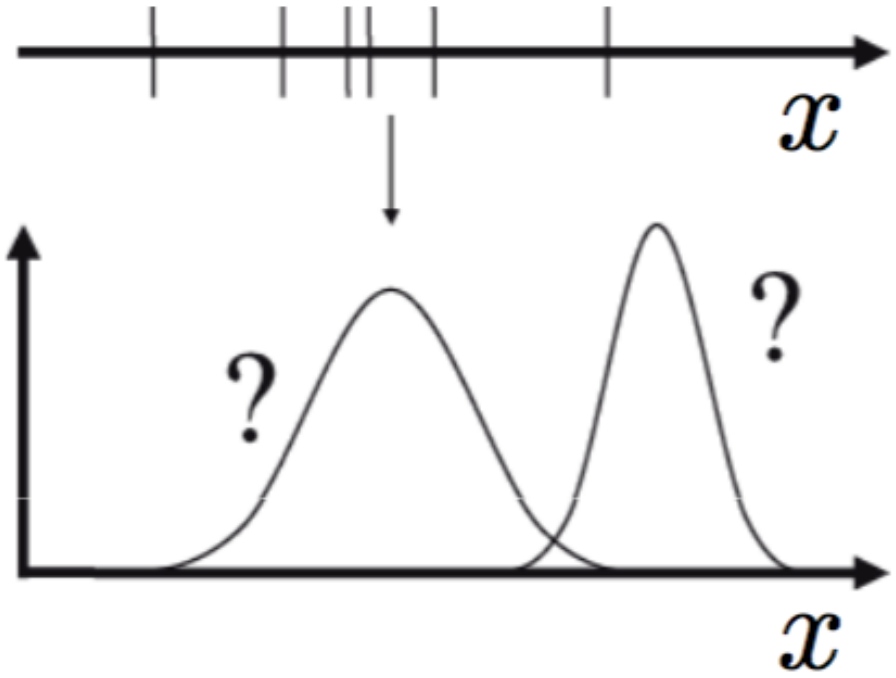
Question: where do all these functions come from?

Loss: user-specific

Probabilities? (prior, conditional, posterior)

Estimated from data!

Probability density estimation



Observed Input

Inferred distribution

Step 1: Decide about the form of the distribution

Step 2: Adapt the form to the data (i.e. estimate parameters)

Expectation & mean

- The average value of some function $f(x)$ under a probability distribution $p(x)$ is called its expectation

$$\mathbb{E}[f] = \sum_{x} p(x) f(x)$$

discrete case

$$\mathbb{E}[f] = \int p(x) f(x)\, \mathrm{d}x$$

continuous case

Expected value of x: mean value of x

$$\langle x \rangle \doteq \sum_{x} p(x) x = \mathbb{E}[x]$$

$$\langle x \rangle \doteq \int p(x) x dx = \mathbb{E}[x]$$

Variance and Covariance

- The variance provides a measure how much variability there is in $f(x)$ around its mean value $\mathbb{E}[f(x)]$.

$$\mathrm{var}[f] = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$$

Variance and Covariance

- The variance provides a measure how much variability there is in $f(x)$ around its mean value $\mathbb{E}[f(x)]$.

$$\mathrm{var}[f] = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$$

- For two random variables x and y, the covariance is defined by

$$\begin{aligned}\mathrm{cov}[x, y] &= \mathbb{E}_{x,y}\left[\{x - \mathbb{E}[x]\}\{y - \mathbb{E}[y]\}\right] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]\end{aligned}$$

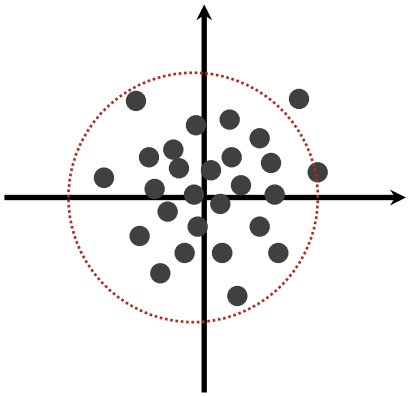
- If $\mathbf{x}$ and $\mathbf{y}$ are vectors, the result is a covariance matrix

$$\begin{aligned}\mathrm{cov}[\mathbf{x}, \mathbf{y}] &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\{\mathbf{x} - \mathbb{E}[\mathbf{x}]\}\{\mathbf{y}^{\mathrm{T}} - \mathbb{E}[\mathbf{y}^{\mathrm{T}}]\}\right] \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}}[\mathbf{x}\mathbf{y}^{\mathrm{T}}] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^{\mathrm{T}}]\end{aligned}$$

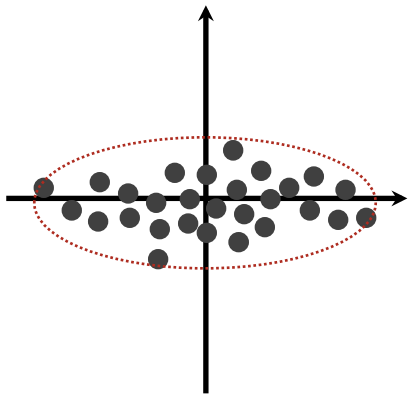
Covariance matrix reminder

Covariance matrix:

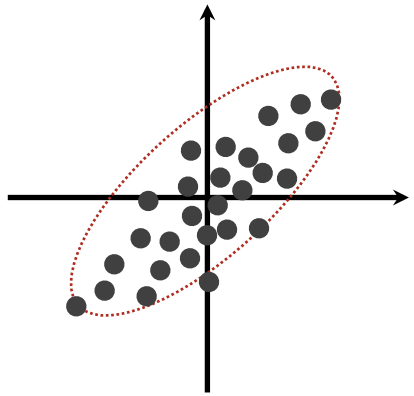
$$\Sigma_{i,j} = E\left((x_i - E(x_i))(x_j - E(x_j))\right)$$



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 4 & 0 \\ 0 & 0.5 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 2.45 & 1.2 \\ 1.2 & 2.1 \end{pmatrix}$$

Height, Income

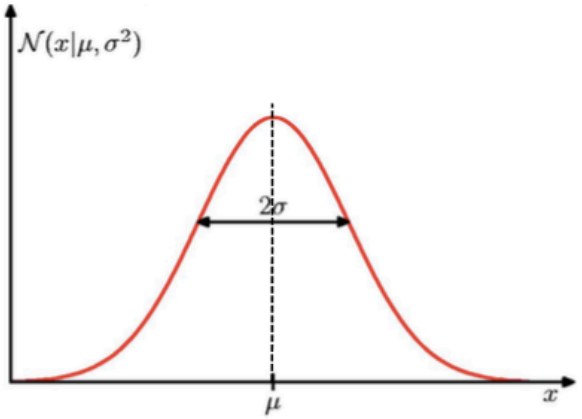
Height, Weight

Gaussian (or Normal) distribution

One-dimensional case

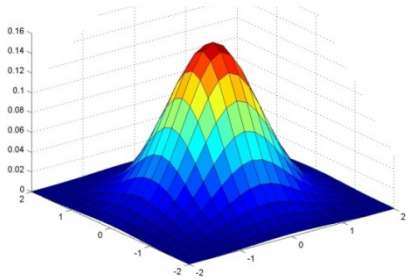
- Mean μ
- Variance σ^2

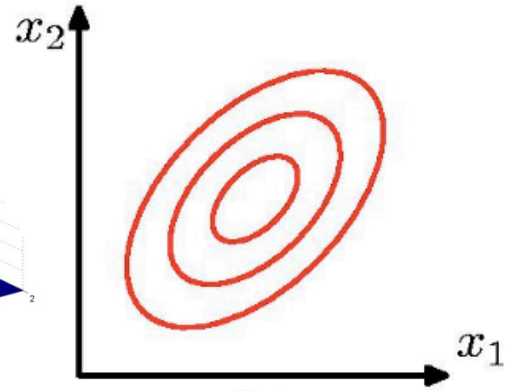
$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$



Multi-dimensional case

- Mean $\boldsymbol{\mu}$
- Covariance $\boldsymbol{\Sigma}$



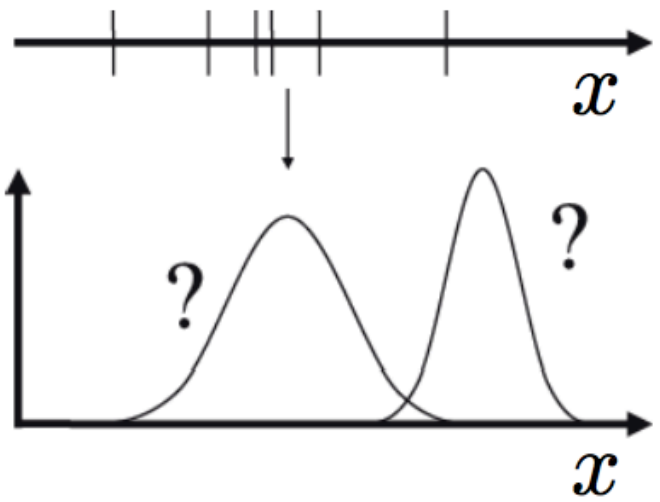


$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Parameter estimation

Given

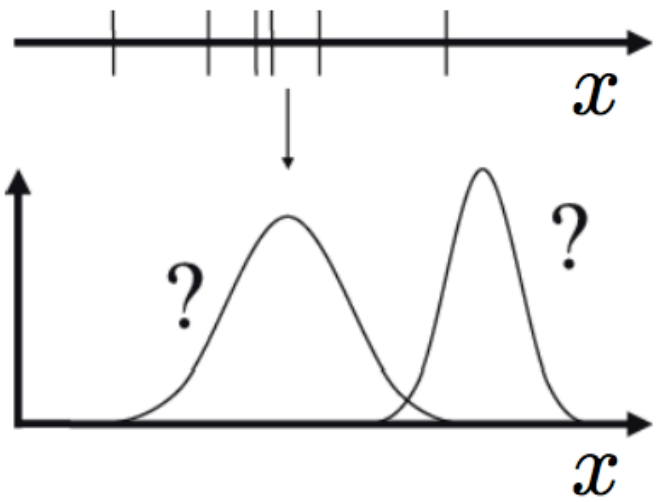
- Data $X = \{x_1, x_2, \ldots, x_N\}$
- Parametric form of the distribution with parameters θ
- E.g. for Gaussian distrib.: $\theta = (\mu, \sigma)$



Parameter estimation

Given

- Data $X = \{x_1, x_2, \ldots, x_N\}$
- Parametric form of the distribution with parameters θ
- E.g. for Gaussian distrib.: $\theta = (\mu, \sigma)$



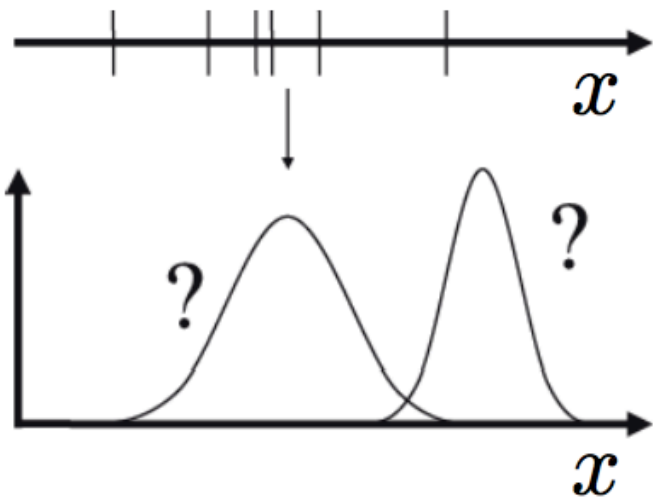
Learning

- Estimation of the parameters θ

Parameter estimation

Given

- Data $X = \{x_1, x_2, \ldots, x_N\}$
- Parametric form of the distribution with parameters θ
- E.g. for Gaussian distrib.: $\theta = (\mu, \sigma)$



Learning

- Estimation of the parameters θ

Likelihood of θ

- Probability that the data X have indeed been generated from a probability density with parameters θ

$$L(\theta) = p(X|\theta)$$

Computation of the likelihood

- **Single data point:** $p(x_n|\theta)$

Computation of the likelihood

- Single data point: $p(x_n|\theta)$

- Assumption: all data points are independent

$$L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

Computation of the likelihood

- Single data point: $p(x_n|\theta)$

- Assumption: all data points are independent

$$L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

- Negative (of) log-likelihood

$$E(\theta) = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(x_n|\theta)$$

Computation of the likelihood

- Single data point: $p(x_n|\theta)$

- Assumption: all data points are independent

$$L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

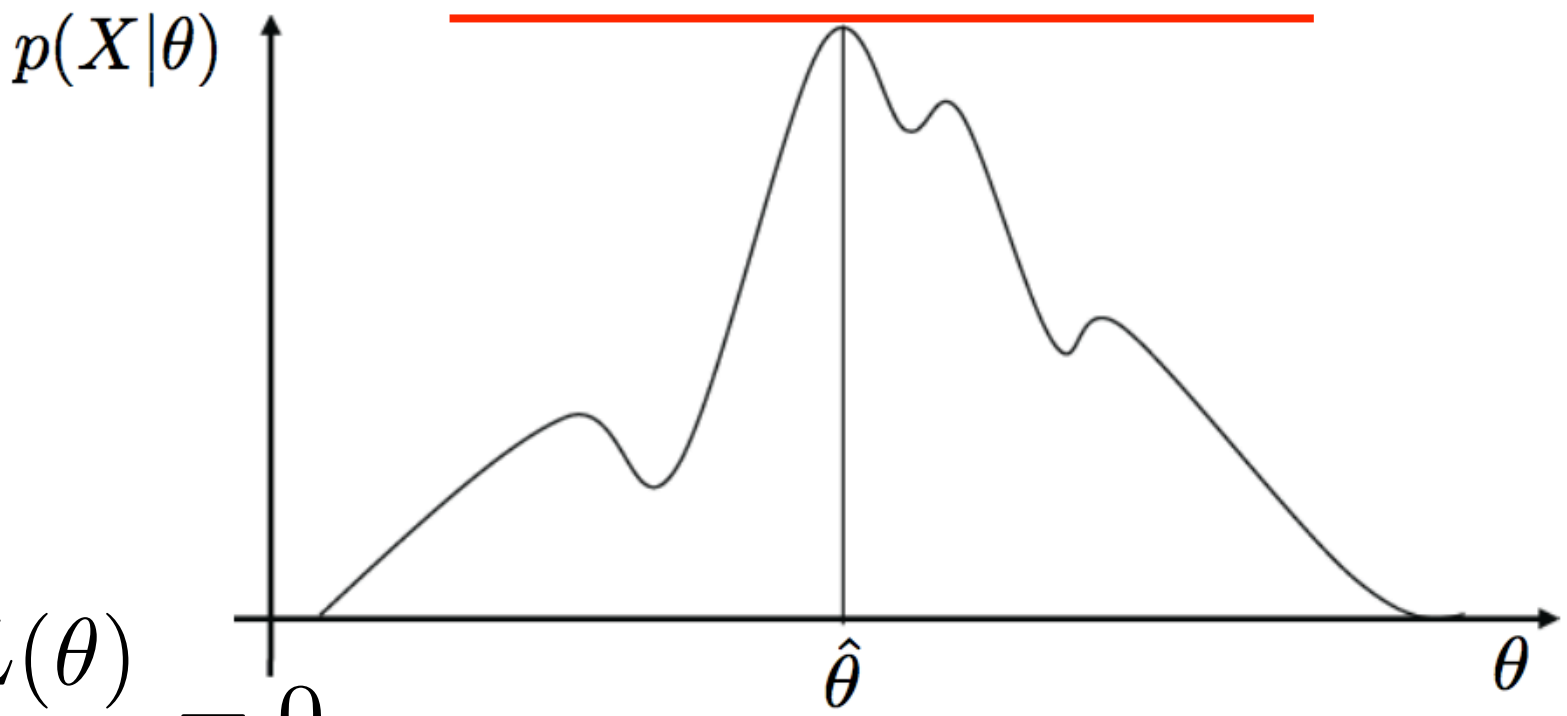
- Negative (of) log-likelihood

$$E(\theta) = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(x_n|\theta)$$

- Estimation of the parameters θ (Learning)
  - Maximize the likelihood
  - Minimize the negative log-likelihood

Likelihood: $L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$

We want to obtain $\hat{\theta}$ such that $L(\hat{\theta})$ is maximized.



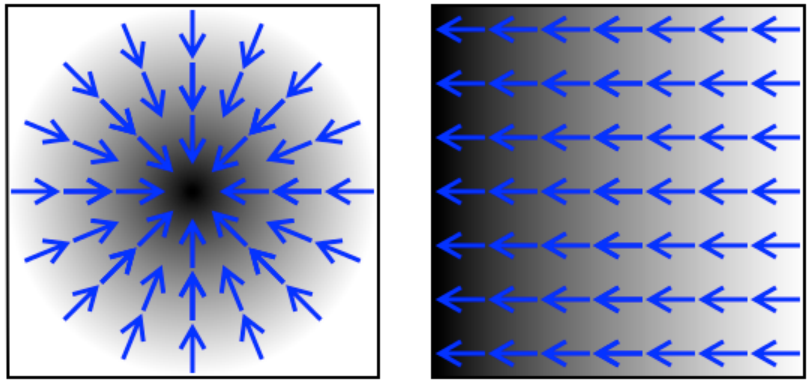
$$\frac{\partial L(\theta)}{\partial \theta} = 0$$

Gradient reminder

multivariate function: $f(x_1, \ldots, x_N)$

gradient: $$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix}$$

at extremum: $$\nabla f = \mathbf{0}$$



Maximum Likelihood Parameter estimation

- How do we minimize a function?

⇒ Take the derivative and set it to zero.

$$\frac{\partial}{\partial\theta}E(\theta) = -\frac{\partial}{\partial\theta}\sum_{n=1}^{N}\ln p(x_n|\theta) = -\sum_{n=1}^{N}\frac{\frac{\partial}{\partial\theta}p(x_n|\theta)}{p(x_n|\theta)} \stackrel{!}{=} 0$$

Maximum Likelihood Parameter estimation

- How do we minimize a function?

⇒ Take the derivative and set it to zero.

$$\frac{\partial}{\partial\theta}E(\theta) = -\frac{\partial}{\partial\theta}\sum_{n=1}^{N}\ln p(x_n|\theta) = -\sum_{n=1}^{N}\frac{\frac{\partial}{\partial\theta}p(x_n|\theta)}{p(x_n|\theta)} \stackrel{!}{=} 0$$

Negative log-likelihood for 1D Gaussian

$$E(\theta) = -\sum_{n=1}^{N}\ln p(x_n|\mu,\sigma)$$

Maximum Likelihood Parameter estimation

- How do we minimize a function?

⇒ Take the derivative and set it to zero.

$$\frac{\partial}{\partial\theta}E(\theta) = -\frac{\partial}{\partial\theta}\sum_{n=1}^{N}\ln p(x_n|\theta) = -\sum_{n=1}^{N}\frac{\frac{\partial}{\partial\theta}p(x_n|\theta)}{p(x_n|\theta)} \stackrel{!}{=} 0$$

Negative log-likelihood for 1D Gaussian

$$\begin{aligned} E(\theta) &= -\sum_{n=1}^{N}\ln p(x_n|\mu,\sigma) \\ &= -\sum_{n=1}^{N}\ln\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{||x_n-\mu||^2}{2\sigma^2}\right\}\right) \end{aligned}$$

$$\frac{\partial}{\partial \mu} E(\mu, \sigma) = -\sum_{n=1}^{N} \frac{\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)}$$

$$p(x_n|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{||x_n - \mu||^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial \mu} E(\mu, \sigma) = -\sum_{n=1}^{N} \frac{\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)}$$

$$= -\sum_{n=1}^{N} -\frac{2(x_n - \mu)}{2\sigma^2}$$

$$p(x_n|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{||x_n - \mu||^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial \mu} E(\mu, \sigma) = -\sum_{n=1}^{N} \frac{\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)}$$

$$= -\sum_{n=1}^{N} -\frac{2(x_n - \mu)}{2\sigma^2}$$

$$= \frac{1}{\sigma^2} \sum_{n=1}^{N} (x_n - \mu)$$

$$p(x_n|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{||x_n - \mu||^2}{2\sigma^2}}$$

$$
\begin{aligned}
\frac{\partial}{\partial \mu} E(\mu, \sigma) &= -\sum_{n=1}^{N} \frac{\frac{\partial}{\partial \mu} p(x_n|\mu, \sigma)}{p(x_n|\mu, \sigma)} \\
&= -\sum_{n=1}^{N} -\frac{2(x_n - \mu)}{2\sigma^2} \\
&= \frac{1}{\sigma^2} \sum_{n=1}^{N} (x_n - \mu) \\
&= \frac{1}{\sigma^2} \left(\sum_{n=1}^{N} x_n - N\mu \right)
\end{aligned}
$$

$$
p(x_n|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{||x_n - \mu||^2}{2\sigma^2}}
$$

$$p(x_n|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{||x_n-\mu||^2}{2\sigma^2}}$$

$$\begin{aligned}
\frac{\partial}{\partial\mu}E(\mu,\sigma) &= -\sum_{n=1}^{N}\frac{\frac{\partial}{\partial\mu}p(x_n|\mu,\sigma)}{p(x_n|\mu,\sigma)} \\
&= -\sum_{n=1}^{N} -\frac{2(x_n-\mu)}{2\sigma^2} \\
&= \frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu) \\
&= \frac{1}{\sigma^2}\left(\sum_{n=1}^{N}x_n - N\mu\right)
\end{aligned}$$

$$\frac{\partial}{\partial\mu}E(\mu,\sigma) \stackrel{!}{=} 0 \quad \Leftrightarrow \quad \hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}x_n$$

Maximum Likelihood estimates for Gaussian

We thus obtain

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

"sample mean"

Maximum Likelihood estimates for Gaussian

We thus obtain

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

"sample mean"

In a similar fashion, we get

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N} (x_n - \hat{\mu})^2$$

"sample variance"

$\hat{\theta} = (\hat{\mu}, \hat{\sigma})$ is the Maximum Likelihood estimate for the parameters of a Gaussian distribution.

Vector-valued case

Sample mean:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

Sample covariance:

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T$$

Unbiased estimate:

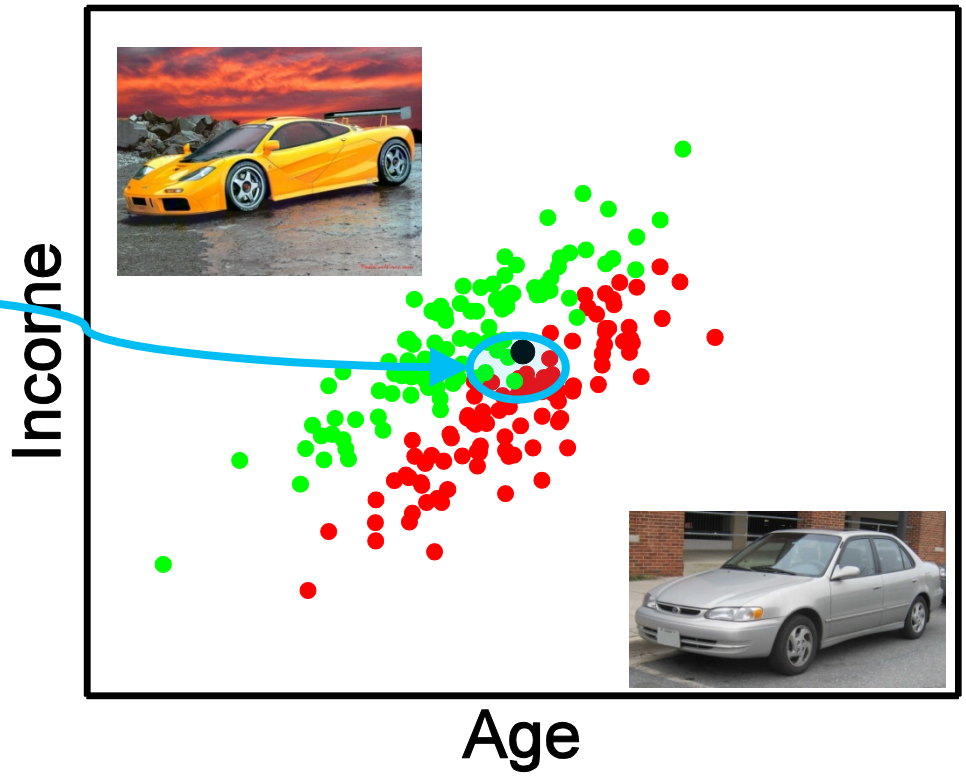
$$\hat{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T$$

Binary Classification

- Training data $\{(x^i, y^i)\}, i = 1 \ldots M$

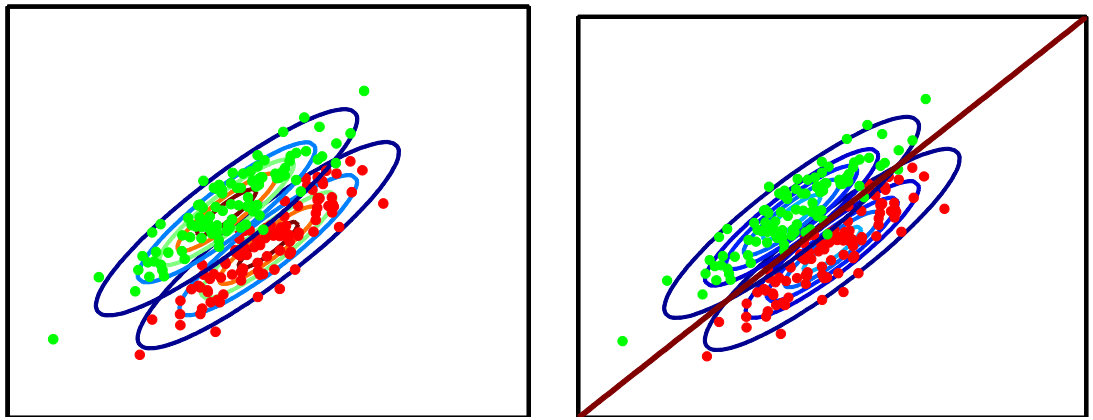
$$x \in R^N : \text{Input}, \; y \in \{0, 1\} : \text{Label}$$

- Classifier: $f : R^N \rightarrow \{0, 1\}$
  - Which car to advertise?
- Test datum: x
  (new client)



Step 1: Density Estimation

- Assumption: within each class the observations follow a multivariate Gaussian distribution

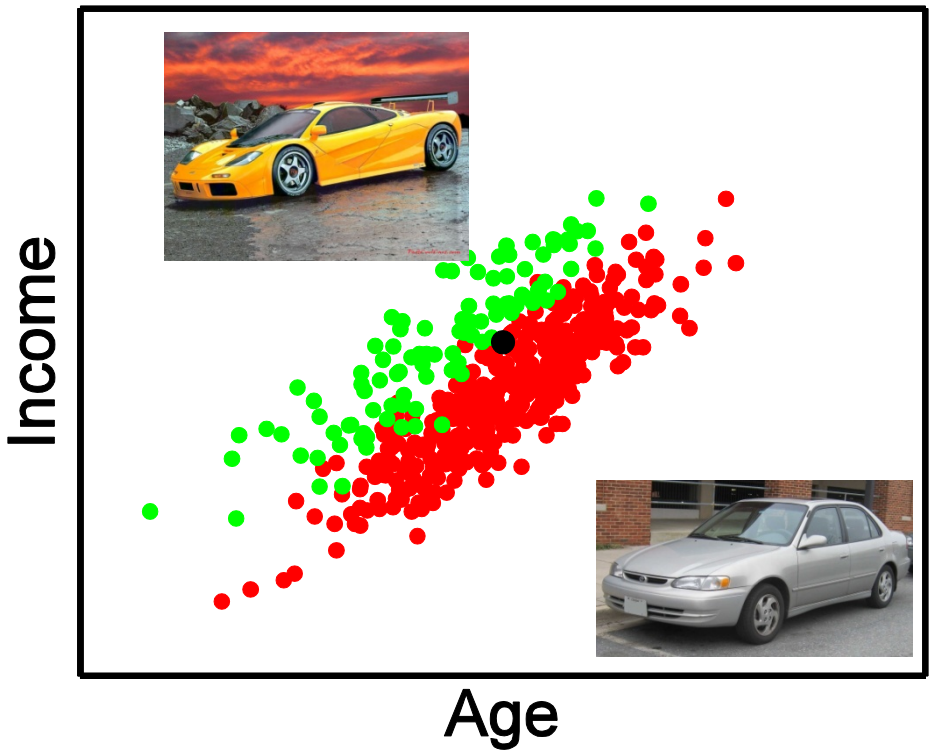


- Estimate class parameters using Maximum Likelihood
- Using these parameters assign to class according to whether

$$P(x|\mu_1^*, \Sigma_1^*) > P(x|\mu_0^*, \Sigma_0^*)$$

Not equally likely classes!

- What happens if one class is more probable?



Step 1, Continued: Class priors

- Parameters to estimate: $\pi_i = P(y = i)$
- Maximum likelihood:

$$\pi_k = \sum_{i=1}^{N} \frac{[y_i = k]}{N}$$

$$\pi_1 = \sum_{i=1}^{N} \frac{[y_i = 1]}{N}$$

- Two-class case:

$$\pi_0 = 1 - \pi_1$$

Step 2: Form Posterior (Bayes' Rule)

$$P(Y=1|X=x) = \frac{P(X=x|Y=1)P(Y=1)}{P(X=x)}$$

$$= \frac{P(X=x|Y=1)P(Y=1)}{\sum_{y\in\{0,1\}} P(X=x|Y=y)P(Y=y)}$$

Substitute:

$$P(X=x|Y=c) = \frac{1}{\sqrt{(2\pi)^N|\Sigma_c|}} \exp\left(-\frac{1}{2}(x-\mu_c)^T\Sigma_c^{-1}(x-\mu_c)\right)$$

$$P(Y=1)=\pi, \quad P(Y=0)=1-\pi$$

Posterior: $$P(Y=1|x) = \frac{1}{1+\exp(x^TAx + x^Tb + c)}$$

$$A = \frac{1}{2}\Sigma_1^{-1} - \frac{1}{2}\Sigma_0^{-1} \qquad b = \Sigma_0^{-1}\mu_0 - \Sigma_1^{-1}\mu_1$$

$$c = \frac{1}{2}\left[\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0\Sigma_0^{-1}\mu_0\right] + \log\left(\frac{1-\pi}{\pi}\sqrt{\frac{|\Sigma_0|}{|\Sigma_1|}}\right)$$

Step 3: Classify

Optimal classifier (Lecture 1):

$$c^* = \arg\max_c P(Y = c|X = x)$$

Decision boundary between two classes:

$$P(Y = 1|x) = \frac{1}{1 + \exp(x^T Ax + x^T b + c)} = \frac{1}{2}$$

Equivalently:

$$x^T Ax + x^T b + c = 0$$

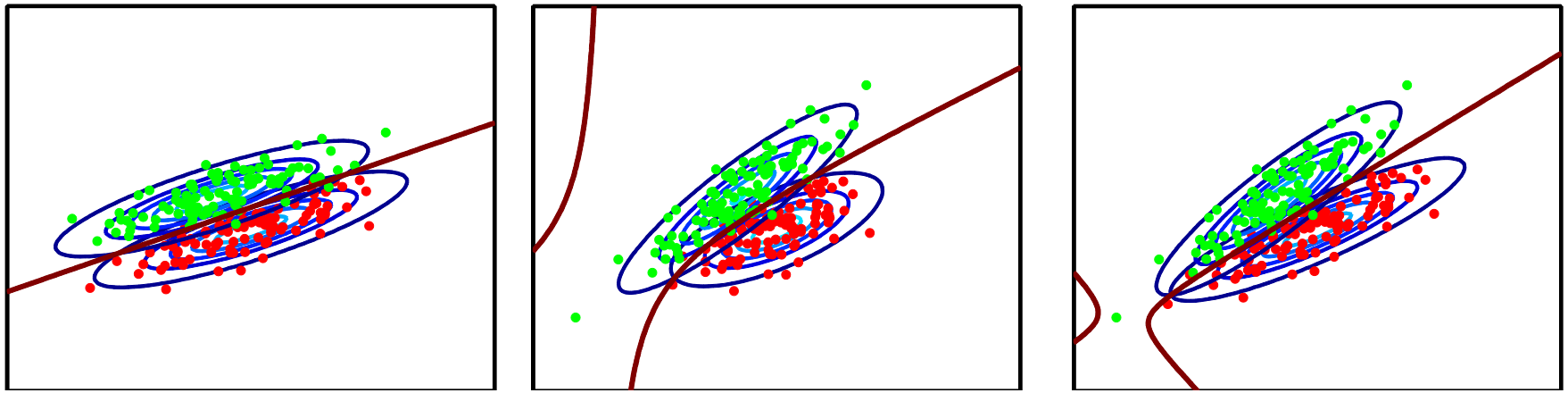
Quadratic Decision Boundary

where:

$$A = \frac{1}{2}\Sigma_1^{-1} - \frac{1}{2}\Sigma_0^{-1} \qquad b = \Sigma_0^{-1}\mu_0 - \Sigma_1^{-1}\mu_1$$

$$c = \frac{1}{2}\left[\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0\Sigma_0^{-1}\mu_0\right] + \log\left(\frac{1-\pi}{\pi}\sqrt{\frac{|\Sigma_0|}{|\Sigma_1|}}\right)$$

Effect of different covariance matrices



Decision Boundaries: set of points such that $x^T Ax + x^T b + c = 0$

$$A = \frac{1}{2}\Sigma_1^{-1} - \frac{1}{2}\Sigma_0^{-1} \qquad b = \Sigma_0^{-1}\mu_0 - \Sigma_1^{-1}\mu_1$$

$$c = \frac{1}{2}\left[\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0\Sigma_0^{-1}\mu_0\right] + \log\left(\frac{1-\pi}{\pi}\sqrt{\frac{|\Sigma_0|}{|\Sigma_1|}}\right)$$

Step 3: Classify

Optimal classifier (Lecture 1):

$$c^* = \arg\max_c P(Y = c|X = x)$$

Special case: $\Sigma_0 = \Sigma_1$

$$P(Y = 1|X = x) = \frac{1}{1 + \exp(x^T b + c)} = \frac{1}{2}$$

$$x^T b + c = 0$$

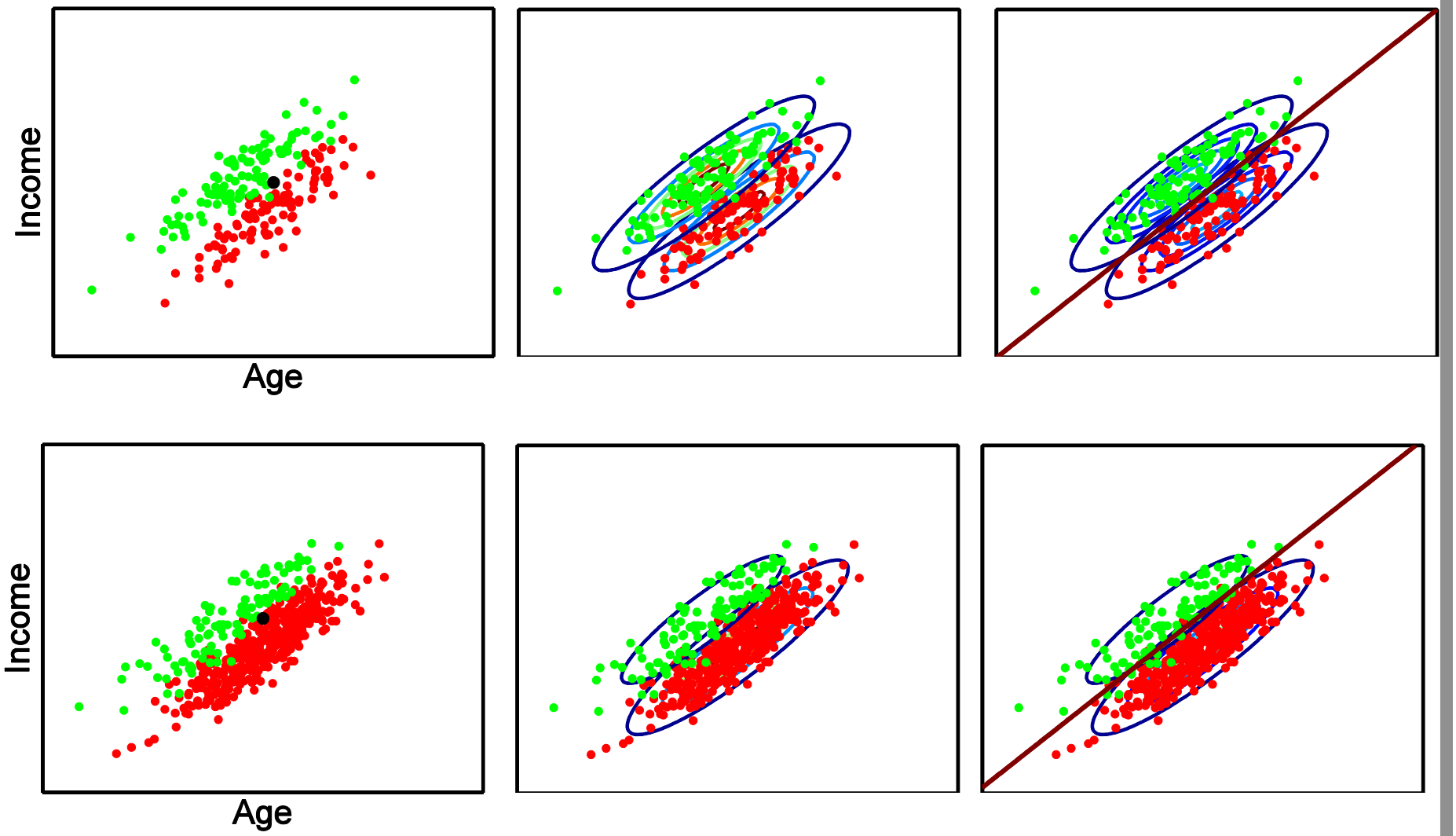
Linear Decision Boundary

$$b = \Sigma^{-1}(\mu_0 - \mu_1)$$

$$c = \frac{1}{2}\left[\mu_1^T \Sigma^{-1} \mu_1 - \mu_0^T \Sigma^{-1} \mu_0\right] + \log\left(\frac{1-\pi}{\pi}\right)$$

Back to car example: effect of class priors

- Decision boundary is shifted by $\log(\frac{1-\pi}{\pi}\sqrt{\frac{|\Sigma_0|}{|\Sigma_1|}})$



From generative to discriminative classification

Bayesian Decision Theory

$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)}$$

- Model conditional probability densities $p(x|\mathcal{C}_k)$ and priors $p(\mathcal{C}_k)$
- Compute posteriors $p(\mathcal{C}_k|x)$ (using Bayes' rule)
- Minimize probability of misclassification by maximizing $p(\mathcal{C}|x)$.

New approach

- Directly encode decision boundary
- Without explicit modeling of probability densities
- Minimize misclassification probability directly.

Classification through discriminant functions

Formulate classification in terms of comparisons

- Discriminant functions

$$y_1(x), \ldots, y_K(x)$$

- Classify x as class C_k if

$$y_k(x) > y_j(x) \quad \forall j \neq k$$

Classification through discriminant functions

Formulate classification in terms of comparisons

- Discriminant functions

$$y_1(x), \ldots, y_K(x)$$

- Classify x as class C_k if

$$y_k(x) > y_j(x) \quad \forall j \neq k$$

Examples (Bayes Decision Theory)

A
$$y_k(x) = p(\mathcal{C}_k|x)$$

B
$$y_k(x) = p(x|\mathcal{C}_k)p(\mathcal{C}_k)$$

C
$$y_k(x) = \log p(x|\mathcal{C}_k) + \log p(\mathcal{C}_k)$$

Different discriminant functions, same classification result!

Discriminant functions, continued

Example: 2 classes

$$
\begin{aligned}
& y_1(x) > y_2(x) \\
\Leftrightarrow \quad & y_1(x) - y_2(x) > 0 \\
\Leftrightarrow \quad & \mathbf{y}(x) > 0
\end{aligned}
$$

Discriminant functions, continued

Example: 2 classes

$$
\begin{aligned}
& y_1(x) > y_2(x) \\
\Leftrightarrow \quad & y_1(x) - y_2(x) > 0 \\
\Leftrightarrow \quad & \mathbf{y}(x) > 0
\end{aligned}
$$

Decision functions (from Bayes Decision Theory)

$$y(x) = p(\mathcal{C}_1|x) - p(\mathcal{C}_2|x)$$

$$y(x) = \ln \frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

Linear discriminant functions

2-class problem

- $y(x) > 0$: Decide for class C_1, else for class C_2

In the following, we focus on linear discriminant functions

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

weight vector

"bias" (= threshold)

- If a data set can be perfectly classified by a linear discriminant, then we call it linearly separable.

Notation

- D : Number of dimensions

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

Notation

- D : Number of dimensions

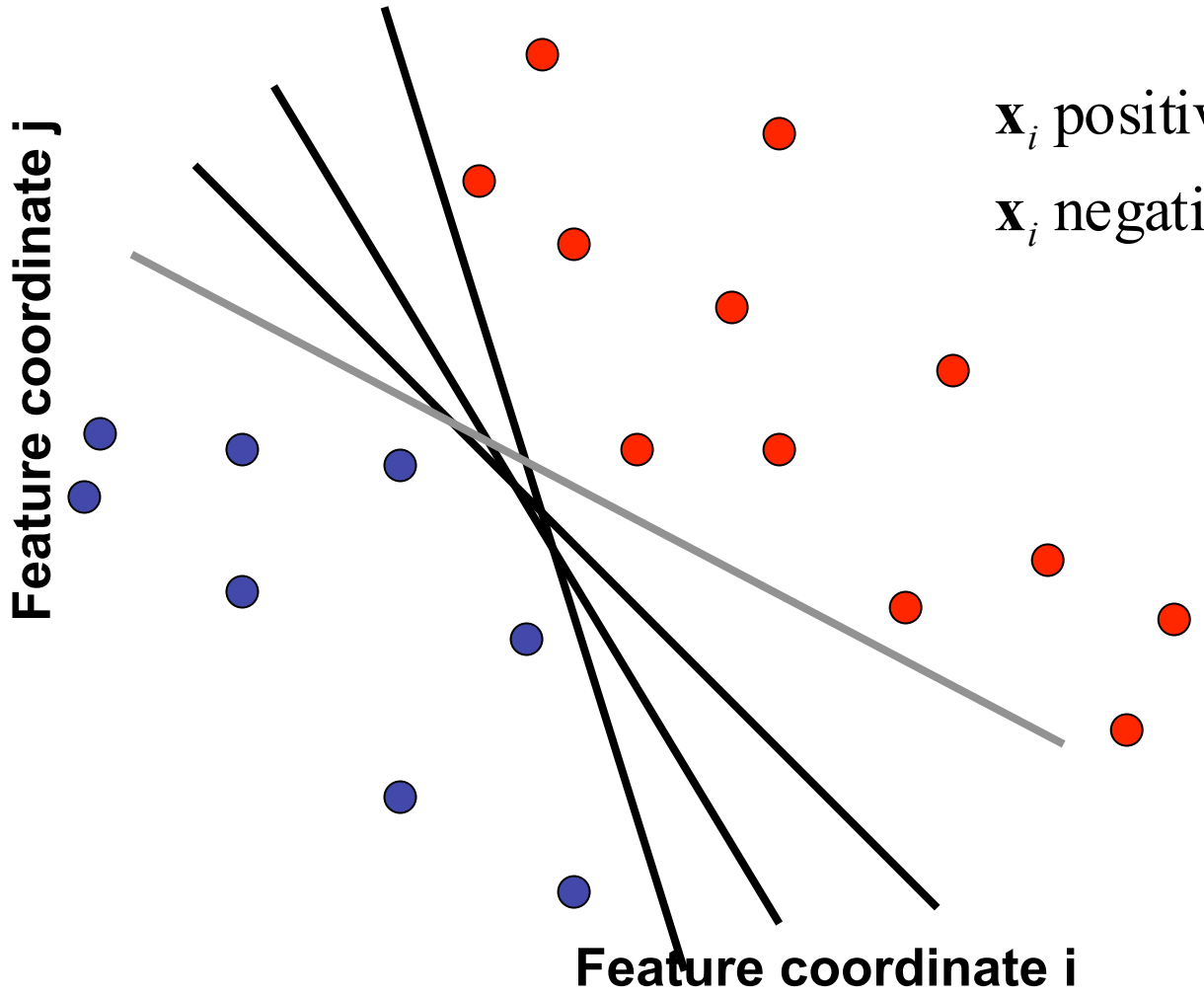
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \\ &= \sum_{i=1}^{D} w_i x_i + w_0 \\ &= \sum_{i=0}^{D} w_i x_i \end{aligned}$$

with $x_0 = 1$ constant

$y_{\mathbf{w}}(\mathbf{x})$ when we want to make explicit dependence on w

Linear classifiers



$\mathbf{x}_i$ positive: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative: $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

Each data point has a class label:

$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$

Linear regression

Classifier: mapping from features $x^i \in \mathcal{X}$ to labels $y^i \in \{0, 1\}$

$$f : \mathcal{X} \to \{0, 1\}$$

Linear regression: linear $f : \mathbb{R}^D \to \mathbb{R}$

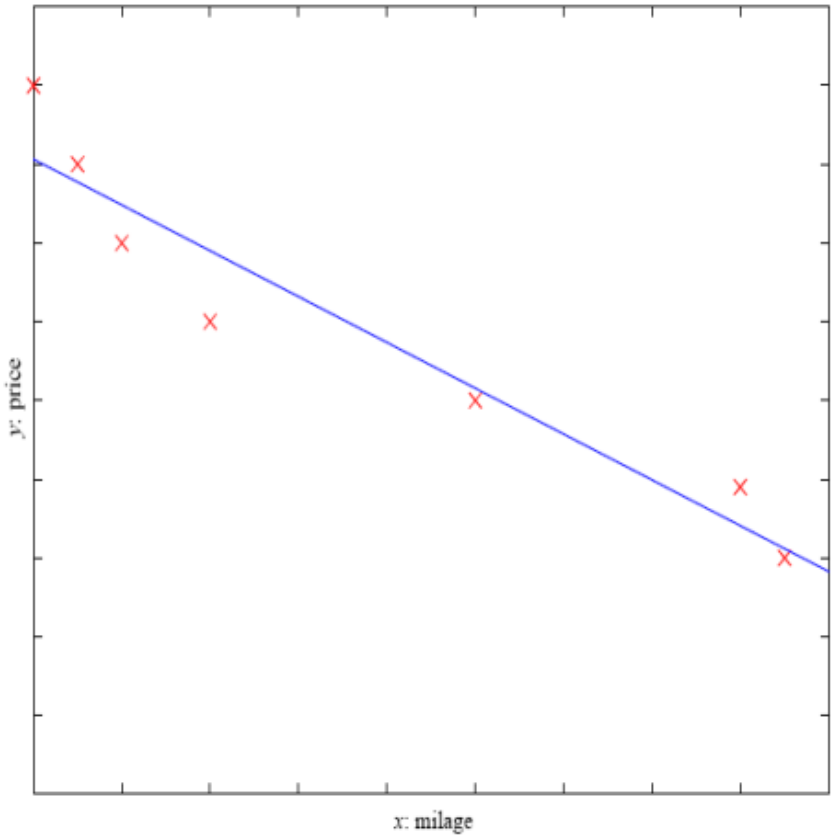
$$y = f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

binary decision can be obtained based on the sign of f

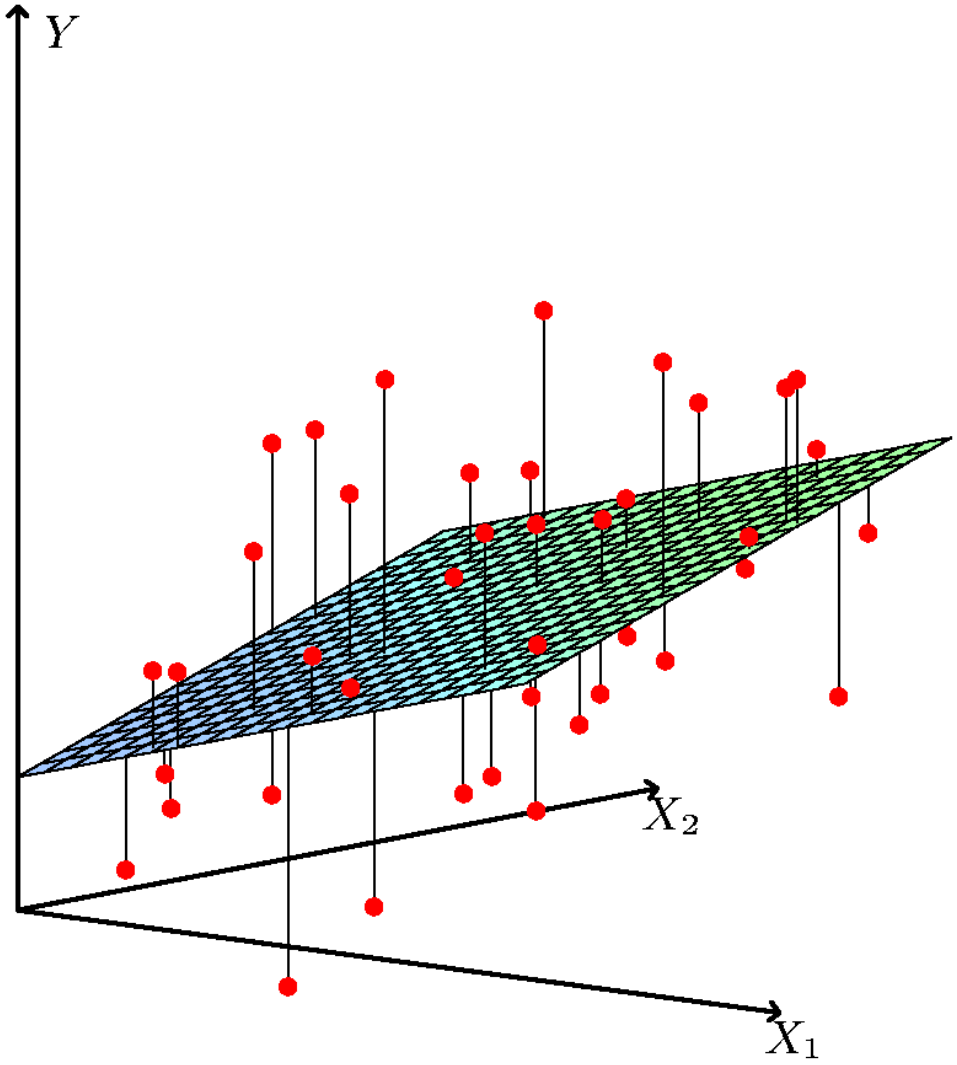
$$c = \text{sign}\left[\mathbf{w}^T \mathbf{x}\right]$$

“perceptron” – 1960’s

Linear regression in 1D



Linear regression in 2D



What we would like to be happening

Training: $S = \{(\mathbf{x}^i, y^i)\}, i =,1, \ldots, N$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N$$

If N>D (e.g. 30 points, 2 dimensions) we have more equations than unknowns: **overdetermined** system!

Input-output relations can only hold approximately!

What is happening: approximations

Training: $S = \{(\mathbf{x}^i, y^i)\}, i =,1, \ldots, N$

$$y^1 \simeq w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1$$

$$y^2 \simeq w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2$$

$$\vdots$$

$$y^N \simeq w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N$$

If N>D (e.g. 30 points, 2 dimensions) we have more equations than unknowns: **overdetermined** system!

Input-output relations can only hold approximately!

What is happening: residuals

Training: $S = \{(\mathbf{x}^i, y^i)\}, i =,1, \ldots, N$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1 + \epsilon^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2 + \epsilon^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N + \epsilon^N$$

$$\epsilon^i = y^i - \mathbf{w}^T \mathbf{x}^i, \quad i = 1, \ldots, N$$

Objective: minimize residuals

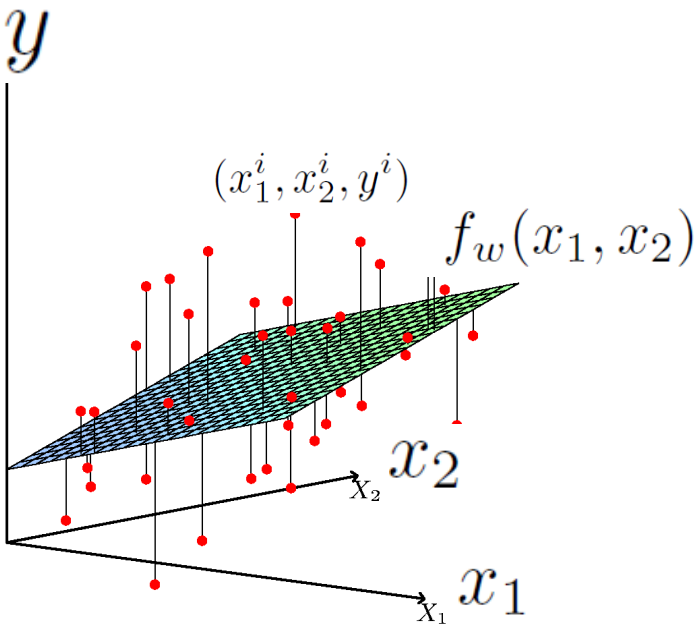
Loss function for linear regression

Training: given $S = \{(\mathbf{x}^i, y^i)\}, i = 1, \ldots, N$, estimate optimal $\mathbf{w}$

Loss function: quantify appropriateness of $\mathbf{w}$

sum of individual errors (`additive')

$$L(S, \mathbf{w}) = \sum_{i=1}^{N} l(y^i, f_{\mathbf{w}}(\mathbf{x}^i))$$



quadratic

$$= \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2$$

$$= \sum_{i=1}^{N} (\epsilon^i)^2$$

Why this loss function?

Easy to optimize!

Data science

https://en.wikipedia.org/wiki/Least_squares

The first clear and concise exposition of the method of least squares was published by Legendre in **1805.**

The technique is described as an algebraic procedure for **fitting linear equations to data** and Legendre demonstrates the new method by analyzing the same data as Laplace for the shape of the earth.

The value of Legendre's method of least squares was immediately recognized by leading astronomers and geodesists of the time

Data science

https://en.wikipedia.org/wiki/Least_squares

The first clear and concise exposition of the method of least squares was published by Legendre in **1805.**

The technique is described as an **algebraic procedure** for **fitting linear equations to data** and Legendre demonstrates the new method by analyzing the same data as Laplace for the shape of the earth.

The value of Legendre's method of least squares was immediately recognized by leading astronomers and geodesists of the time

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \ldots + w_D x_D^1 + \epsilon^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \ldots + w_D x_D^2 + \epsilon^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \ldots + w_D x_D^N + \epsilon^N$$

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} x_0^1 & x_1^1 & \dots & x_D^1 \\ x_0^2 & x_2^2 & \dots & x_D^2 \\ \vdots & & & \\ x_0^N & x_2^N & \dots & x_D^N \end{bmatrix} \begin{bmatrix} w^1 \\ w^2 \\ \vdots \\ w^D \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Nx1 **NxD** **Dx1** **Nx1**

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

Nx1 **NxD** **Dx1** **Nx1**

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \dots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \dots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Least squares solution for linear regression

Loss function: $L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^{N} (\epsilon^i)^2$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

Nx1 NxD Dx1 Nx1

$$\begin{aligned} L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \\ &= c + \mathbf{a}^T \mathbf{w} + \mathbf{w}^T \mathbf{B}\mathbf{w} \end{aligned}$$

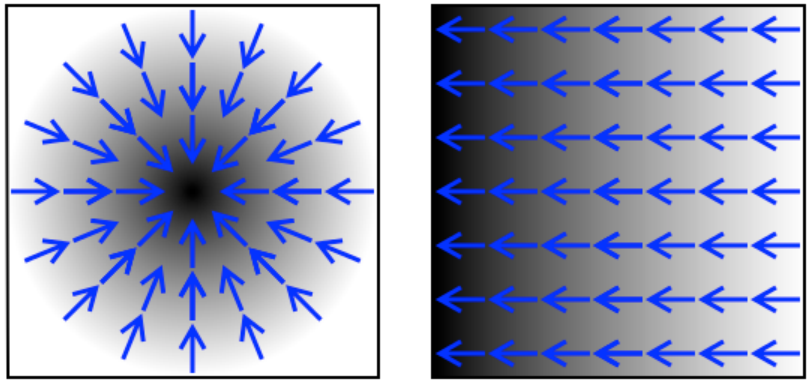
where: $\mathbf{a}^T = -2\mathbf{y}^T \mathbf{X}, \quad \mathbf{B} = \mathbf{X}^T \mathbf{X}$

Gradient reminder

multivariate function: $f(x_1, \ldots, x_N)$

gradient: $$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix}$$

at extremum: $$\nabla f = \mathbf{0}$$



least-squares solution

Loss function: $L(\mathbf{w}) = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} \quad = c + \mathbf{a}^T\mathbf{w} + \mathbf{w}^T\mathbf{B}\mathbf{w}$

where: $\mathbf{a}^T = -2\mathbf{y}^T\mathbf{X}, \quad \mathbf{B} = \mathbf{X}^T\mathbf{X}$

Gradient of loss: $\nabla L(\mathbf{w}) = \mathbf{a} + 2\mathbf{B}\mathbf{w}$

At extremum: $\nabla L(\mathbf{w}^*) = \mathbf{0}$

$$\Leftrightarrow \quad \mathbf{w}^* = -(2\mathbf{B})^{-1}\mathbf{a}$$

substitute

$$= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$