Foundations of Machine Learning
CentraleSupélec — Fall 2017

6. Linear & logistic regressions

Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Inserm
Institut national
de la santé et de la recherche médicale

institutCurie

PSL
RESEARCH UNIVERSITY PARIS

MINES
ParisTech

Learning objectives

- **Density estimation:**
  - Define **parametric methods.**
  - Define the **maximum likelihood estimator** and compute it for **Bernouilli**, **multinomial** and **Gaussian** densities.
  - Define the **Bayes estimator** and compute it for **normal priors.**
- **Supervised learning:**
  - Compute the maximum likelihood estimator / least-square fit solution for **linear regression.**
  - Compute the maximum likelihood estimator for **logistic regression.**

Density estimation

Parametric methods

- $\mathcal{X} = \{\boldsymbol{x}^i\}_{i=1,\ldots,n}$ $\quad\quad\quad\quad\quad\quad$ $\boldsymbol{x}^i \sim p(\boldsymbol{x}|\theta)$
- **Parametric estimation:**
  - **assume a form for p(x|θ)**
    E.g. $p(x_j|\theta_j) \sim \mathcal{N}(\mu_j, \sigma_j^2)$ $\quad\quad$ $\theta = \{\mu_1, \sigma_1, \ldots, \mu_p, \sigma_p\}$
  - Goal: estimate θ using $\mathcal{X}$
  - usually assume that $\boldsymbol{x}^i$ **independent and identically distributed** (iid)

Maximum likelihood estimation

- **Find θ such that $\mathcal{X}$ is the most likely to be drawn.**
- **Likelihood** of θ given the i.i.d. sample $\mathcal{X}$:

$$\ell(\theta|\mathcal{X}) = p(\mathcal{X}|\theta) = p(\boldsymbol{x}^1|\theta)p(\boldsymbol{x}^2|\theta)\dots p(\boldsymbol{x}^n|\theta)$$

- **Log likelihood:**

$$\mathcal{L}(\theta|\mathcal{X}) = \log \ell(\theta|\mathcal{X}) = \log p(\boldsymbol{x}^1|\theta) + \dots + \log p(\boldsymbol{x}^n|\theta)$$

- **Maximum likelihood estimation (MLE):**

$$\theta^* = \arg\max_{\theta} \mathcal{L}(\theta|\mathcal{X})$$

Bernouilli density

- Two states: failure / success

  $x \in \{0, 1\}$

  $P(X = x|p_0) = {p_0}^x (1 - p_0)^{(1-x)}$

  $\mathcal{X} = \{x^i\}_{i=1,\ldots,n}$

  MLE estimate of p_0:

Bernouilli density

- Two states: failure / success

  $x \in \{0, 1\}$

  $P(X = x|p_0) = {p_0}^x (1 - p_0)^{(1-x)}$

  $\mathcal{X} = \{x^i\}_{i=1,\ldots,n}$

  MLE estimate of p_0:

- **Log likelihood:** 

Bernouilli density

- Two states: failure / success

  $x \in \{0, 1\}$

  $P(X = x|p_0) = {p_0}^x (1 - p_0)^{(1-x)}$

  $\mathcal{X} = \{x^i\}_{i=1,\ldots,n}$

  MLE estimate of p_0:

- **Log likelihood:**

$$L(p_0|\mathcal{X}) = \log P(\mathcal{X}|p_0) = \sum_{i=1}^{n} \left(x^i \log p_0 + (1 - x^i) \log(1 - p_0)\right)$$

- **Maximize the likelihood:** 

Bernouilli density

- Two states: failure / success

  $x \in \{0, 1\}$

  $P(X = x|p_0) = {p_0}^x (1 - p_0)^{(1-x)}$

  $\mathcal{X} = \{x^i\}_{i=1,\ldots,n}$

  MLE estimate of p_0:

- **Log likelihood:**

$$L(p_0|\mathcal{X}) = \log P(\mathcal{X}|p_0) = \sum_{i=1}^{n} \left(x^i \log p_0 + (1 - x^i) \log(1 - p_0)\right)$$

- **Maximize the likelihood:** set the gradient to 0.

Log likelihood (under iid assumption):

$$\log P(\{x^1, x^2, \dots, x^n\} \mid p_0) = \sum_{i=1}^{n} \log \left[p_0^{x^i} (1-p_0)^{(1-x^i)} \right]$$

$$= \left(\sum_{i=1}^{n} x^i \right) \log p_0 + \left(n - \sum_{i=1}^{n} x^i \right) \log (1-p_0)$$

to maximize: this is concave, set the derivative to 0

$$\frac{\partial L}{\partial p_0} = \left(\sum_{i=1}^{n} x^i \right) \frac{1}{p_0} + \left(n - \sum_{i=1}^{n} x^i \right) \frac{-1}{1-p_0}$$

(we assume $p_0 \notin \{0, 1\}$.)

$$\frac{\partial L}{\partial p_0} = 0 \iff (1-\hat{p}_0) \sum_{i=1}^{n} x^i - \hat{p}_0 \left(n - \sum_{i=1}^{n} x^i \right) = 0$$

hence $\boxed{\hat{p}_0 = \frac{1}{n} \sum_{i=1}^{n} x^i}$

Bernouilli density

- Two states: failure / success

  $x \in \{0, 1\}$

  $P(X = x|p_0) = {p_0}^x (1 - p_0)^{(1-x)}$

  $\mathcal{X} = \{x^i\}_{i=1,\ldots,n}$

  MLE estimate of p_0:

- **Log likelihood:**

$$L(p_0|\mathcal{X}) = \log P(\mathcal{X}|p_0) = \sum_{i=1}^{n} \left(x^i \log p_0 + (1 - x^i) \log(1 - p_0)\right)$$

- **Maximize the likelihood:** set its gradient to 0.

$$\hat{p_0} = \frac{1}{n} \sum_{i=1}^{n} x^i$$

Multinomial density

- Consider **K mutually exclusive and exhaustive classes**
  - Each class occurs with probability p_k $\sum_{k=1}^{K} p_k = 1$
  - $x_1, x_2, \ldots, x_K$ indicator variables: $x_k=1$ if the outcome is class k and 0 otherwise

$$P(x_1^i, x_2^i, \ldots, x_K^i) = \prod_{k=1}^{K} p_k^{x_k^i}$$

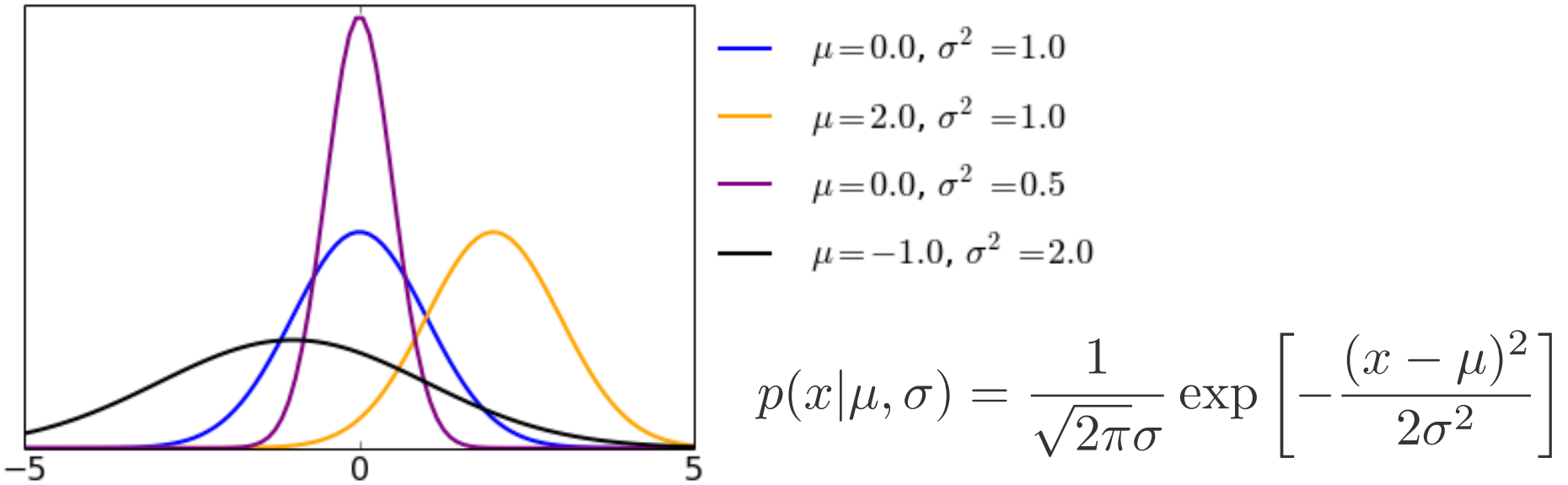
- The **MLE of p_k** is

$$\hat{p}_k = \frac{1}{n} \sum_{i=1}^{n} x_k^i$$

Gaussian distribution

- Gaussian distribution = normal distribution

$$x \sim \mathcal{N}(\mu, \sigma^2)$$



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

Compute the MLE estimates of μ and σ.

Log likelihood (under iid assumption):

$$p(\{x^1, x^2, \dots, x^n\} \mid \mu, \sigma) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x^i-\mu)^2}{\sigma^2}\right)$$

$$= \sum_{i=1}^{n} -\log(\sigma\sqrt{2\pi}) - \frac{(x^i-\mu)^2}{2\sigma^2}$$

This is concave. To maximize it, we compute the gradient and set it to 0.

- $$\frac{\partial L}{\partial \mu} = \sum_{i=1}^{n} -\frac{2\mu+2x^i}{2\sigma^2} = \frac{1}{\sigma^2}\left(n\mu - \sum_{i=1}^{n} x^i\right)$$

  $$\frac{\partial L}{\partial \mu}(\hat{\mu}) = 0 \Rightarrow \boxed{\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^i}$$ empirical mean

- $$\frac{\partial L}{\partial \sigma} = -\frac{n}{\sigma} + \sum_{i=1}^{n} \frac{(x^i-\mu)^2}{\sigma^3}$$

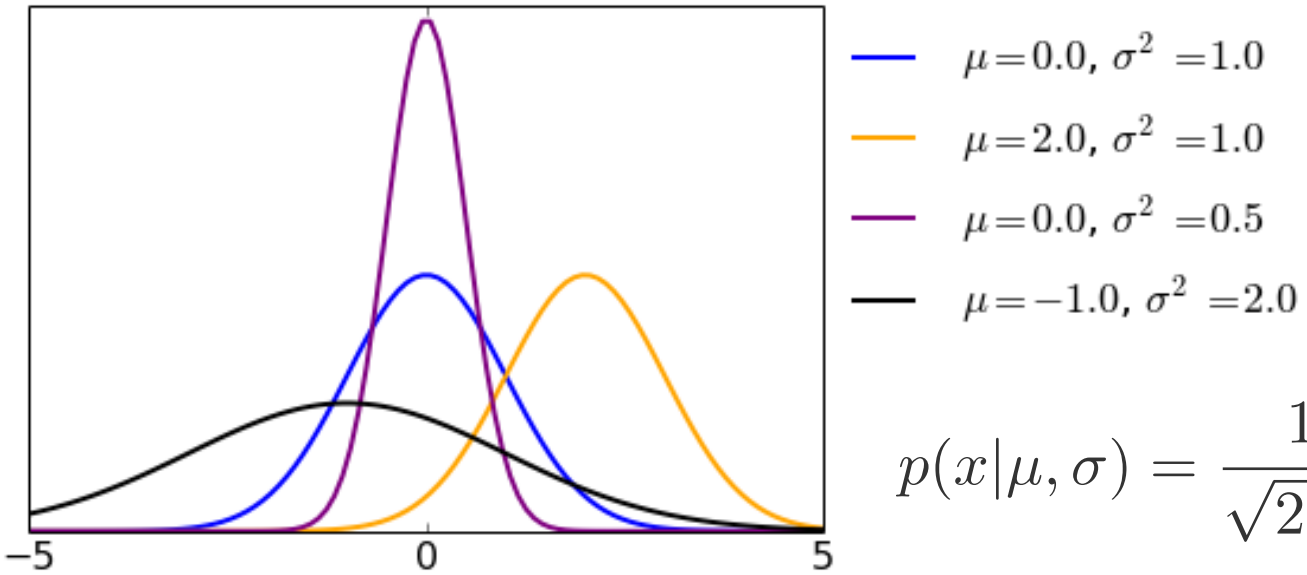
  $$\frac{\partial L}{\partial \sigma}(\hat{\mu}, \hat{\sigma}) = 0 \Rightarrow -n\hat{\sigma}^2 + \sum_{i=1}^{n} (x^i-\hat{\mu})^2 = 0$$

  hence $$\boxed{\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} (x^i-\hat{\mu})}$$ empirical variance

Gaussian distribution

- Gaussian distribution = normal distribution

$$x \sim \mathcal{N}(\mu, \sigma^2)$$



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

Compute the MLE estimates of μ and σ.

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^i$$

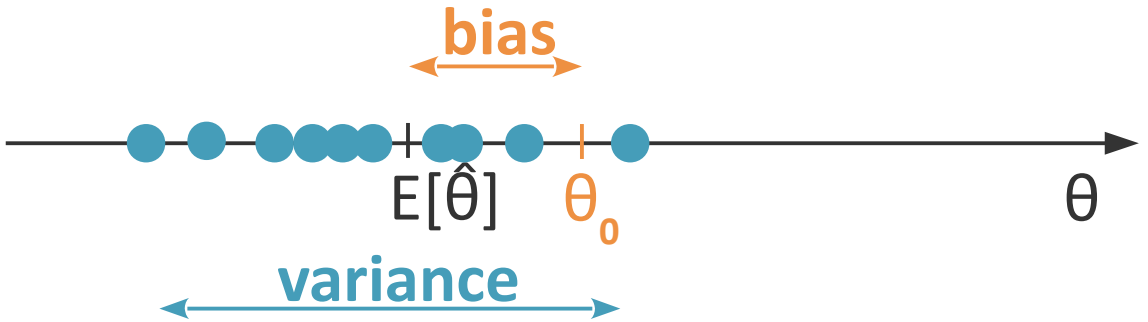
$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x^i - \hat{\mu})^2$$

Bias-variance tradeoff

- **Mean squared error of the estimator:**

$$\begin{aligned} \mathrm{MSE}(\hat{\theta}) &= \mathbb{E}[(\hat{\theta} - \theta_0)^2] \\ &= \mathrm{Var}(\hat{\theta}) + \mathrm{Bias}^2(\hat{\theta}) \end{aligned}$$

A biased estimator may achieve better MSE than an unbiased one.



Bayes estimator

$$P(y = c|\boldsymbol{x}) = \frac{P(y = c)p(\boldsymbol{x}|y = c)}{p(\boldsymbol{x})}$$

prior, likelihood, posterior, evidence

- Treat θ as a random variable with prior p(θ)
- **Bayes rule:**

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})}$$

- **Density estimation at x:**

$$p(\boldsymbol{x}|\mathcal{X}) = \int p(\boldsymbol{x}, \theta|\mathcal{X})d\theta = \int p(\boldsymbol{x}|\theta)p(\theta|\mathcal{X})d\theta.$$

Bayes estimator

- Treat θ as a random variable with prior p(θ)
- **Bayes rule:**

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})}$$

- **Density estimation**

$$p(\boldsymbol{x}|\mathcal{X}) = \int p(\boldsymbol{x}, \theta|\mathcal{X})d\theta = \int p(\boldsymbol{x}|\theta)p(\theta|\mathcal{X})d\theta.$$

- **Maximum likelihood estimate (MLE):**

$$\theta_{\text{MLE}} = \arg\max_{\theta} p(\mathcal{X}|\theta)$$

- **Bayes estimate:**

$$\theta_{\text{Bayes}} = \arg\min_{\hat{\theta}} \mathbb{E}[L(\hat{\theta}, \theta)]$$

$$L(\hat{\theta}, \theta) = (\hat{\theta} - \theta)^2 \quad \Rightarrow \quad \theta_{\text{Bayes}} = \mathbb{E}[\theta|\mathcal{X}] = \int \theta p(\theta|\mathcal{X})d\theta$$

Bayes estimator

$$\theta^*_{\text{Bayes}} = \underset{\hat{\theta}}{\operatorname{argmin}}\ \mathbb{E}\left[\text{MSE}(\hat{\theta}, \theta)\right]$$

↳ Squared Error

i.e. $(\hat{\theta} - \theta)^2$

$$\mathbb{E}\left[(\hat{\theta} - \theta)^2\right] = \hat{\theta}^2 - 2\hat{\theta}\,\mathbb{E}[\theta] + \mathbb{E}[\theta^2]$$

$\hat{\theta}$: deterministic; θ: random variable

$$= \underbrace{\left(\hat{\theta} - \mathbb{E}[\theta]\right)^2}_{\text{always} \geq 0,\ \text{hence minimal in 0 when } \hat{\theta} = \mathbb{E}[\theta].} - \underbrace{\mathbb{E}[\theta]^2 + \mathbb{E}[\theta^2]}_{\text{does not depend on } \hat{\theta}}$$

Hence $\theta^*_{\text{Bayes}} = \underset{\hat{\theta}}{\operatorname{argmin}}\ \mathbb{E}\left[(\hat{\theta} - \theta)^2\right] = \mathbb{E}[\hat{\theta}]$

$\hat{\theta}$ ← estimated from data $\mathcal{X}$.

Bayes estimator: Normal prior

- n data points (iid) $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$ $\theta \sim \mathcal{N}(\mu, \sigma^2)$
- MLE of θ: $\hat{\theta}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x^i$

Compute the Bayes estimator of θ $\theta_{\mathrm{Bayes}} = \mathbb{E}[\theta|\mathcal{X}]$

$$p(u|m,s) = \frac{1}{\sqrt{2\pi}s}\exp\left[-\frac{(u-m)^2}{2s^2}\right]$$

Hint:

Compute p(θ|$\mathcal{X}$) and show that it follows a normal distribution

Normal density estimation (Bayes)

n points $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$

prior belief: $\theta \sim \mathcal{N}(\mu, \sigma^2)$

$\theta_{Bayes} = \mathbb{E}[\theta|X]$

Bayes rule: $p(\theta|X) = \frac{p(X|\theta)\, p(\theta)}{p(X)}$

$X = \{x^1, x^2, \ldots, x^n\}$

$$p(X|\theta) = \text{likelihood} = \prod_{i=1}^{n} \frac{1}{\sigma_0\sqrt{2\pi}} \exp\left(-\frac{(x^i-\theta)^2}{2\sigma_0^2}\right)$$

$$= \left(\frac{1}{\sigma_0\sqrt{2\pi}}\right)^n \exp\left(-\sum_{i=1}^{n} \frac{(x^i-\theta)^2}{2\sigma_0^2}\right)$$

$$p(\theta) = \text{prior} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$$

$$\text{Hence } p(\theta|X) = \underbrace{\frac{1}{p(X)}\left(\frac{1}{\sigma_0\sqrt{2\pi}}\right)^n \frac{1}{\sigma\sqrt{2\pi}}}_{A} \exp\left(-\sum_{i=1}^{n}\frac{(x_i-\theta)^2}{2\sigma_0^2} - \frac{(\theta-\mu)^2}{2\sigma^2}\right)$$

A: not a function of θ

$$= -\frac{1}{2}\left[\frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_i^2+\theta^2-2x_i\theta) + \frac{1}{\sigma^2}(\theta^2+\mu^2-2\mu\theta)\right]$$

$$= -\frac{1}{2}\left[\left(\frac{n}{\sigma_0^2}+\frac{1}{\sigma^2}\right)\theta^2 - 2\left(\frac{\sum x_i}{\sigma_0^2}+\frac{\mu}{\sigma^2}\right)\theta + K\right]$$

K: not a function of θ

$$= -\frac{1}{2}\,\frac{n\sigma^2+\sigma_0^2}{\sigma_0^2\sigma^2}\left(\theta - \frac{\sum x_i\,\sigma^2+\mu\sigma_0^2}{n\sigma^2+\sigma_0^2}\right)^2 + K_2$$

$K - \frac{\sum x_i\,\sigma^2+\mu\sigma_0^2}{n\sigma^2+\sigma_0^2}$: constant wrt θ

Hence

$$p(\theta|X) = A \exp\left(-\frac{1}{2} \frac{\left(\theta - \frac{\sum x^i \sigma^2 + \mu\sigma_0^2}{n\sigma^2+\sigma_0^2}\right)^2}{\frac{\sigma_0^2\sigma^2}{n\sigma^2+\sigma_0^2}} \right) \exp\left(-\frac{1}{2} K_2\right)$$

$$p(\theta|X) = \text{Cte} \times \exp\left(-\frac{1}{2} \frac{(\theta - m)^2}{\Delta^2}\right)$$

where $m = \frac{\sigma^2 \sum x^i + \mu\sigma_0^2}{n\sigma^2+\sigma_0^2}$ and $\Delta^2 = \frac{\sigma_0^2\sigma^2}{n\sigma^2+\sigma_0^2} > 0$

$\sum_{i=1}^{n} x^i = n\hat{\theta}_{MLE}$

and $\text{Cte} = 1$ because $\int p(\theta|X)\, d\theta = 1$ and $\int -\frac{1}{2} \frac{(\theta-m)^2}{\Delta^2} d\theta = 1$

(we know because the density of a normal distr. must sum to 1.)

Hence $\theta|X$ is normally distributed, hence $\mathbb{E}[\theta|X]$ is the mean parameter of its distribution: $\boxed{\theta_{Bayes} = m}$

Bayes estimator: Normal prior

- n data points (iid) $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$ $\theta \sim \mathcal{N}(\mu, \sigma^2)$
- MLE of θ: $\hat{\theta}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x^i$

  Compute the Bayes estimator of θ $\theta_{\mathrm{Bayes}} = \mathbb{E}[\theta|\mathcal{X}]$

  p(θ|$\mathcal{X}$) follows a normal distribution with

  - mean

$$\frac{n\hat{\theta}_{\mathrm{MLE}}\sigma^2 + \mu\sigma_0^2}{n\sigma^2 + \sigma_0^2} = \frac{1/\sigma_0^2}{1/\sigma_0^2 + 1/n\sigma^2}\hat{\theta}_{\mathrm{MLE}} + \frac{1/\sigma^2}{n/\sigma_0^2 + 1/\sigma^2}\mu$$

  - variance $\frac{\sigma^2\sigma_0^2}{n\sigma^2 + \sigma_0^2}$

$$p(\theta|\mathcal{X}) = \frac{1}{\sqrt{2\pi}s}\exp\left[-\frac{(\theta - m)^2}{2s^2}\right]$$

Bayes estimator: Normal prior

- n data points (iid) $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$ $\theta \sim \mathcal{N}(\mu, \sigma^2)$
- MLE of θ: $\hat{\theta}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x^i$

Compute the Bayes estimator of θ $\theta_{\mathrm{Bayes}} = \mathbb{E}[\theta|\mathcal{X}]$

p(θ|$\mathcal{X}$) follows a normal distribution with

- mean

$$\frac{n\hat{\theta}_{\mathrm{MLE}}\sigma^2 + \mu\sigma_0^2}{n\sigma^2 + \sigma_0^2} = \frac{1/\sigma_0^2}{1/\sigma_0^2 + 1/n\sigma^2}\hat{\theta}_{\mathrm{MLE}} + \frac{1/\sigma^2}{n/\sigma_0^2 + 1/\sigma^2}\mu$$

- variance $\frac{\sigma^2\sigma_0^2}{n\sigma^2 + \sigma_0^2}$

$$p(\theta|\mathcal{X}) = \frac{1}{\sqrt{2\pi}s}\exp\left[-\frac{(\theta - m)^2}{2s^2}\right]$$

Bayes estimator: Normal prior

- n data points (iid) $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$ $\theta \sim \mathcal{N}(\mu, \sigma^2)$
- MLE of θ: $\hat{\theta}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x^i$
- **Bayes estimator:**

$$\theta_{\mathrm{Bayes}} = \frac{n/\sigma_0^2}{n/\sigma_0^2 + 1/\sigma^2}\hat{\theta}_{\mathrm{MLE}} + \frac{1/\sigma^2}{n/\sigma_0^2 + 1/\sigma^2}\mu$$

sample mean

prior mean

Bayes estimator: Normal prior

- n data points (iid) $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$ $\theta \sim \mathcal{N}(\mu, \sigma^2)$
- MLE of θ: $\hat{\theta}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x^i$
- **Bayes estimator:**

$$\theta_{\mathrm{Bayes}} = \frac{n/\sigma_0^2}{n/\sigma_0^2 + 1/\sigma^2}\hat{\theta}_{\mathrm{MLE}} + \frac{1/\sigma^2}{n/\sigma_0^2 + 1/\sigma^2}\mu$$

sample mean

prior mean

large when σ is ?

large when n is ?

Bayes estimator: Normal prior

- n data points (iid) $x^i \sim \mathcal{N}(\theta, \sigma_0^2)$ $\theta \sim \mathcal{N}(\mu, \sigma^2)$
- MLE of θ: $\hat{\theta}_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x^i$
- **Bayes estimator:**

$$\theta_{\mathrm{Bayes}} = \frac{n/\sigma_0^2}{n/\sigma_0^2 + 1/\sigma^2}\hat{\theta}_{\mathrm{MLE}} + \frac{1/\sigma^2}{n/\sigma_0^2 + 1/\sigma^2}\mu$$

- When n ↗: θ_{Bayes} gets closer to the sample average (uses information from the sample).
- When σ is small, θ_{Bayes} gets closer to μ (little uncertainty about the prior).

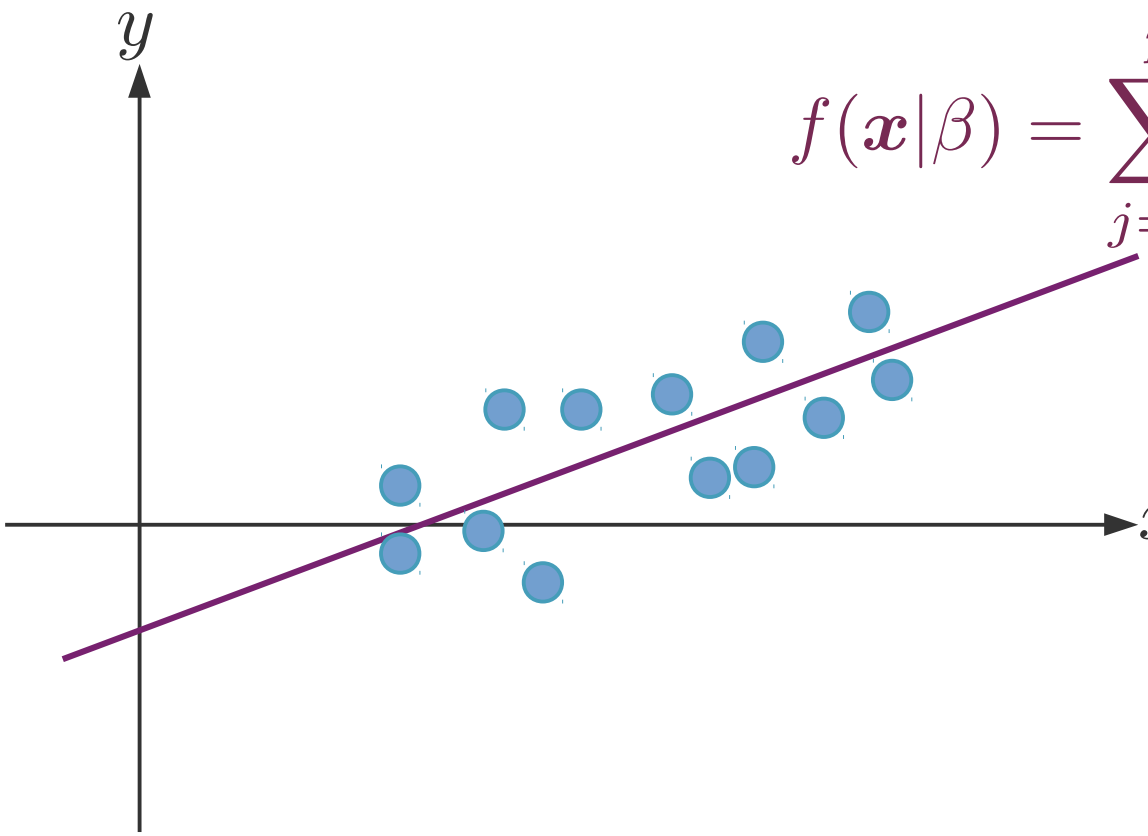
Linear regression

x_1

Linear regression

$\boldsymbol{x} \in \mathbb{R}^p, y \in \mathbb{R}$

$\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\ldots,n}$



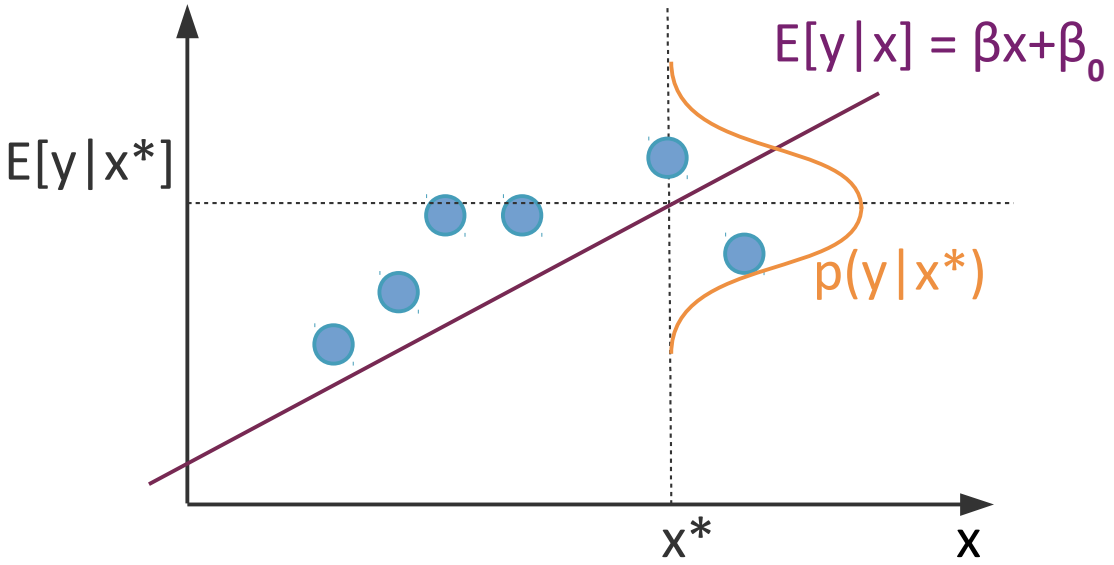
$$f(\boldsymbol{x}|\beta) = \sum_{j=1}^{p} \beta_j x_j + \beta_0$$

Linear regression: MLE

- Assume **error is Gaussian distributed**

$$y = g(\boldsymbol{x}) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- Replace g with its estimator f $f(\boldsymbol{x}|\beta) = \sum_{j=1}^{p} \beta_j x_j + \beta_0$



$$p(y|\boldsymbol{x}) \sim \mathcal{N}(f(\boldsymbol{x}|\beta), \sigma^2)$$

MLE under Gaussian noise

- **Maximize (log) likelihood**

$$\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\dots,n}$$

$$\begin{aligned}\mathcal{L}(\beta|\mathcal{D}) &= \log \prod_{i=1}^{n} p(y^i|\boldsymbol{x}^i)p(\boldsymbol{x}^i) \\ &= \log \prod_{i=1}^{n} p(y^i|\boldsymbol{x}^i) + \boxed{\log \prod_{i=1}^{n} p(\boldsymbol{x}^i)}\end{aligned}$$

independent of β

$$p(y|\boldsymbol{x}) \sim \mathcal{N}(f(\boldsymbol{x}|\beta), \sigma^2)$$

$$\begin{aligned}\mathcal{L}(\beta|\mathcal{D}) &= \log \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y^i - f(\boldsymbol{x}^i|\beta))^2}{2\sigma^2} \right] + \text{Cte} \right) \\ &= \text{Cte} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i|\beta))^2\end{aligned}$$

MLE under Gaussian noise

- **Maximize (log) likelihood**

$$\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\dots,n}$$

$$\begin{aligned}\mathcal{L}(\beta|\mathcal{D}) &= \log \prod_{i=1}^{n} p(y^i|\boldsymbol{x}^i)p(\boldsymbol{x}^i) \\ &= \log \prod_{i=1}^{n} p(y^i|\boldsymbol{x}^i) + \log \prod_{i=1}^{n} p(\boldsymbol{x}^i)\end{aligned}$$

independent of β

$$p(y|\boldsymbol{x}) \sim \mathcal{N}(f(\boldsymbol{x}|\beta), \sigma^2)$$

$$\begin{aligned}\mathcal{L}(\beta|\mathcal{D}) &= \log \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y^i - f(\boldsymbol{x}^i|\beta))^2}{2\sigma^2} \right] + \text{Cte} \right) \\ &= \text{Cte} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i|\beta))^2\end{aligned}$$

?

MLE under Gaussian noise

- **Maximize (log) likelihood**

$$\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\ldots,n}$$

$$\begin{aligned}\mathcal{L}(\beta|\mathcal{D}) &= \log \prod_{i=1}^{n} p(y^i|\boldsymbol{x}^i)p(\boldsymbol{x}^i) \\ &= \log \prod_{i=1}^{n} p(y^i|\boldsymbol{x}^i) + \boxed{\log \prod_{i=1}^{n} p(\boldsymbol{x}^i)}\end{aligned}$$

independent of β

$$p(y|\boldsymbol{x}) \sim \mathcal{N}(f(\boldsymbol{x}|\beta), \sigma^2)$$

$$\begin{aligned}\mathcal{L}(\beta|\mathcal{D}) &= \log \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y^i - f(\boldsymbol{x}^i|\beta))^2}{2\sigma^2} \right] + \text{Cte} \right) \\ &= \text{Cte} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i|\beta))^2\end{aligned}$$

- **Assuming Gaussian error, maximizing the likelihood is equivalent to minimizing the sum of squared residuals.**

Linear regression least-squares fit

- Minimize the **residual sum of squares**

$$\begin{aligned}
\mathrm{RSS}(\beta) &= \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i))^2 \\
&= \sum_{i=1}^{n} \left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j \right)^2 \\
&= (y - X\beta)^\top (y - X\beta)
\end{aligned}$$

$$X = \begin{pmatrix}
1 & x_1^1 & x_2^1 & \cdots & x_p^1 \\
1 & x_1^2 & x_2^2 & \cdots & x_p^2 \\
\vdots & \vdots & \cdots & \vdots & \\
1 & x_1^n & x_2^n & \cdots & x_p^n
\end{pmatrix}$$

Linear regression least-squares fit

- Minimize the **residual sum of squares**

$$\begin{aligned}\mathrm{RSS}(\beta) &= \sum_{i=1}^{n}(y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n}\left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j\right)^2 \\ &= (y - X\beta)^\top (y - X\beta)\end{aligned}$$

Historically:

- Carl Friedrich Gauss (to predict the location of Ceres)
- Adrien Marie Legendre

Linear regression least-squares fit

- Minimize the **residual sum of squares**

$$\begin{aligned}\mathrm{RSS}(\beta) &= \sum_{i=1}^{n}(y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n}\left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j\right)^2 \\ &= (y - X\beta)^\top (y - X\beta)\end{aligned}$$

Estimate β. What condition do you need to verify?

$$\nabla_\beta \mathrm{RSS} = -2X^\top (y - X\beta)$$

$$X^\top (y - X\hat{\beta}) = 0$$

$$X^\top X\hat{\beta} = X^\top y$$

Linear regression least-squares fit

- Minimize the **residual sum of squares**

$$\begin{aligned}\mathrm{RSS}(\beta) &= \sum_{i=1}^{n}(y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n}\left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j\right)^2 \\ &= (y - X\beta)^\top (y - X\beta)\end{aligned}$$

- Assuming **X has full column rank** (and hence X^TX invertible):

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

Linear regression least-squares fit

- Minimize the **residual sum of squares**

$$\begin{aligned}\mathrm{RSS}(\beta) &= \sum_{i=1}^{n}(y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n}\left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j\right)^2 \\ &= (y - X\beta)^\top (y - X\beta)\end{aligned}$$

- Assuming **X has full column rank** (and hence X^TX invertible):

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

- If X is rank-deficient, use a **pseudo-inverse.**

A **pseudo-inverse** of A
is a matrix G s. t. AGA = A

Gauss-Markov Theorem

- Under the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2)$

  the least-squares estimator of β is its (unique) best linear unbiased estimator.

Gauss-Markov Theorem

- Under the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2)$

  the least-squares estimator of β is its (unique) best linear unbiased estimator.

- **Best Linear Unbiased Estimator (BLUE):**

  Var($\hat{\beta}$) < Var(β*) for any β* that is a linear unbiased estimator of β

Gauss-Markov Theorem

- Under the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2)$

  the least-squares estimator of β is its (unique) best linear unbiased estimator.

- **Best Linear Unbiased Estimator (BLUE):**

  Var($\hat{\beta}$) < Var(β*) for any β* that is a linear unbiased estimator of β

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

$$\beta^* = Ay$$

Gauss-Markov Theorem

- Under the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2)$

  the least-squares estimator of β is its (unique) best linear unbiased estimator.

- **Best Linear Unbiased Estimator (BLUE):**

  Var(β̂) < Var(β*) for any β* that is a linear unbiased estimator of β

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

$$\mathbb{E}[\beta^*] = \beta$$

$$\begin{aligned} \mathbb{E}[\hat{\beta}] &= \mathbb{E}[(X^\top X)^{-1} X^\top (X\beta + \epsilon)] \\ &= \beta \end{aligned}$$

Gauss-Markov Theorem

- Under the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2)$

  the least-squares estimator of β is its (unique) best linear unbiased estimator.

- **Best Linear Unbiased Estimator (BLUE):**

  $\mathrm{Var}(\hat{\beta}) < \mathrm{Var}(\beta^*)$ for any β* that is a linear unbiased estimator of β

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

$$\begin{aligned} Var(\hat{\beta}) &= \mathbb{E}[(X^\top X)^{-1} X^\top \epsilon\epsilon^\top X (X^\top X)^{-1}] \\ &= (X^\top X)^{-1} X^\top \sigma^2 I X (X^\top X)^{-1} \\ &= \sigma^2 (X^\top X)^{-1} \end{aligned}$$

$$\beta^* = Ay$$

$$Var(\beta^*) = \sigma^2 DD^\top + Var(\hat{\beta})$$

psd and minimal for D=0

$$D = A - (X^\top X)^{-1} X^\top$$

Proof of Gauss-Markov theorem:

Let $\hat{\beta} = (X^T X)^{-1} X^T y$ be the least-squares estimator of the linear regression btw X and y.

* $\hat{\beta}$ is linear
  ↖ definition: estimator of the form Ay

* $\hat{\beta}$ is unbiased:

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}[(X^T X)^{-1} X^T y]$$

↑ $y = X\beta + \varepsilon$ $\quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$

X, y, β are fixed $\quad \mathbb{E}[\varepsilon] = 0$

hence

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}[\underbrace{(X^T X)^{-1} X^T X}_{= I} \beta] = \beta \quad \blacksquare$$

* Let β^* be another linear, unbiased estimator of β.

$\beta^* = Ay$ ← linear

and $\mathbb{E}[\beta^*] = \beta$ ← unbiased, by definition.

Let $D = A - (X^TX)^{-1}X^T$ $\quad (\beta^* - \hat{\beta} = Dy)$.

$\text{Var}(\beta^*) = \text{Var}(Ay) = \text{Var}(A\varepsilon) = AA^T \times \sigma^2$

$$AA^T = \left(D + (X^TX)^{-1}X^T\right)\underbrace{\left(D + (X^TX)^{-1}X^T\right)^T}_{= D^T + \underbrace{(X^T)^T}_{X}\underbrace{\left((X^TX)^{-1}\right)^T}_{(X^TX)^{-1}}}$$

because X^TX is symmetric hence $(X^TX)^{-1}$ as well.

$$AA^T = DD^T + DX(X^TX)^{-1} + (X^TX)^{-1}X^TD^T + \underbrace{(X^TX)^{-1}X^TX}_{=I}(X^TX)^{-1}$$

$$= (X^TX)^{-1}$$

Because $\mathbb{E}[\beta^*] = \beta$ we can write

$\mathbb{E}[A(X\beta + \varepsilon)] = \beta$ i.e. $\mathbb{E}[AX\beta] = \beta$

hence $AX\beta = \beta$ (true for all β)

meaning $AX = I$

Because $DX = AX - (X^TX)^{-1}X^TX = AX - I$, this means $DX = 0$

$DX = 0$ also means $X^T D^T = 0$

Hence the two terms $DX(X^TX)^{-1}$ and $(X^TX)^{-1}X^TD^T$ are 0

and $AA^T = DD^T + (X^TX)^{-1}$

let us now compute $Var(\hat{\beta})$.

$$Var(\hat{\beta}) = Var((X^TX)^{-1}X^Ty)$$

$$= Var((X^TX)^{-1}X^T(X\beta+\varepsilon))$$

$$= Var((X^TX)^{-1}X^T\varepsilon)$$

$$= (X^TX)^{-1}X^T\underbrace{((X^TX)^{-1}X^T)^T}\,\underbrace{Var(\varepsilon)}_{=\sigma^2}$$

$$= X(X^TX)^{-1}$$

$$Var(\hat{\beta}) = (X^TX)^{-1}\sigma^2$$

Hence $Var(\beta^*) = \sigma^2 AA^T$

$$= \sigma^2 DD^T + \sigma^2(X^TX)^{-1}$$

$$\boxed{Var(\beta^*) = \sigma^2 DD^T + Var(\hat{\beta})}$$

Because $\sigma^2 > 0$

and DD^T positive semi definite

$\sigma^2 DD^T > 0$ unless $D = 0$

i.e. $\beta^* = \hat{\beta}$

We have thus proven that $\text{Var}(\beta^*) > \text{Var}(\hat{\beta})$ unless $\beta^* = \hat{\beta}$

Hence $\hat{\beta}$ is the Best Linear Unbiased Estimator of β ■.

Correlated variables

- If the variables are **decorrelated**:
  - Each coefficient can be estimated separately;
  - **Interpretation** is easy:

    "A change of 1 in x_j is associated with a change of β_j in Y, while everything else stays the same."
- **Correlations between variables cause problems:**
  - The **variance** of all coefficients tend to increase;
  - Interpretation is much harder

    when x_j changes, so does everything else.

Logistic regression

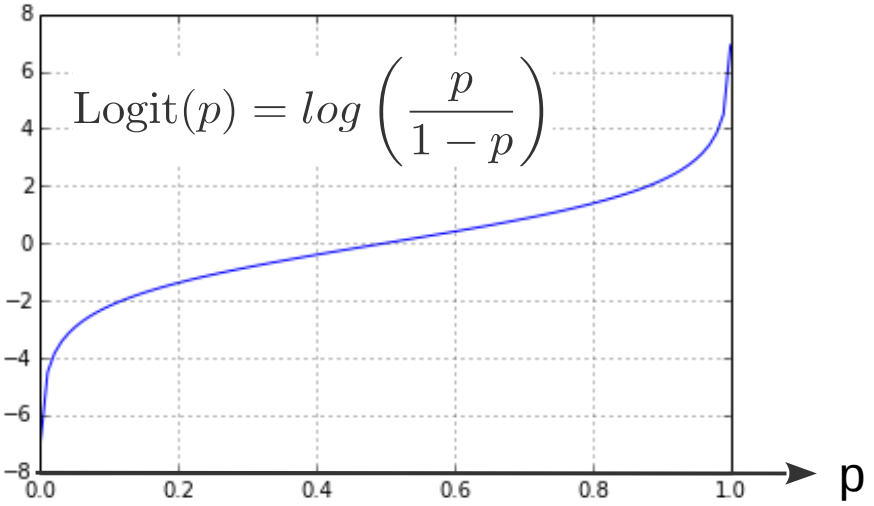
What about classification?

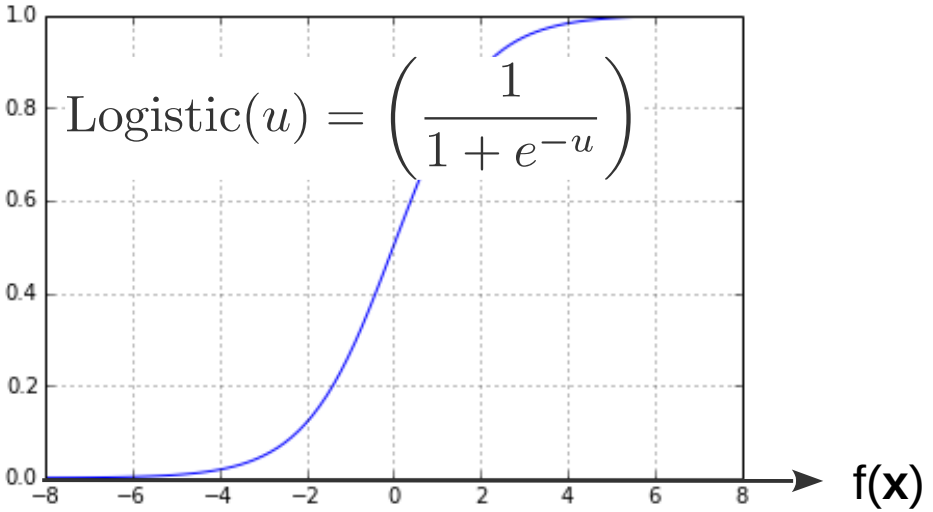
What about classification?

- Model P(Y=1|x) as a linear function?

What about classification?

- **Model P(Y=1|x) as a linear function?**
  - Problem: P(Y=1|**x**) **must be between 0 and 1.**
  - **Non-linearity:**
    - If P(Y=1|**x**) close to +1 or 0, **x** must change a lot for y to change;
    - If P(Y=1|**x**) close to 0.5, that's not the case.
  - Hence: use a **logit transformation**





→ **Logistic regression.**

Maximum likelihood estimation of logistic regression coefficients

$$\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0$$

- **Log likelihood for n observations** $\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\ldots,n}$

$$P(y=1|x) = \frac{1}{1+e^{-(\beta^T x+\beta_0)}}$$

We observe n datapoints (x^i, y^i), which we suppose iid

Likelihood: $\ell(\beta|X) = \prod_{i=1}^{n} P((x^i, y^i)|\beta)$

We replace $\beta = (\beta_1, \beta_2, \dots, \beta_p)^T$ with $(\beta_0, \beta_1, \beta_2, \dots, \beta_p)^T$ and X with $\begin{pmatrix} 1 & \\ \vdots & X \\ 1 & \end{pmatrix}$ (n rows, $p+1$ columns)

Log likelihood:

$$\mathcal{L}(\beta|X) = \sum_{i=1}^{n} \log P(x^i, y^i|\beta)$$

$$= \underbrace{P(x^i|\beta)}_{\text{constant w.r.t. } \beta} \; P(y^i|x^i, \beta)$$

$$= \begin{cases} P(y^i=1|x^i,\beta) & \text{if } y^i=1 \\ 1-P(y^i=1|x^i,\beta) & \text{if } y^i=0 \end{cases}$$

this can be equivalently written as

$$P(y^i=1|x^i,\beta)^{y^i} \times \left(1-P(y^i=1|x^i,\beta)\right)^{(1-y^i)}.$$

Let us write $g(x,\beta) = \frac{1}{1+e^{-\beta^T x}}$

then $\mathcal{L}(\beta|X) = \sum_{i=1}^{n} y^i \log g(x^i,\beta) + (1-y^i)\log\left(1-g(x^i,\beta)\right)$

Maximum likelihood estimation of logistic regression coefficients

$$\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0$$

- **Log likelihood for n observations** $\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\ldots,n}$

$$g = P(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-(\beta^\top \boldsymbol{x})}}$$

$$\begin{aligned} \boldsymbol{x} &\leftarrow [1, x_1, \ldots, x_p] \\ \beta &\leftarrow [\beta_0, \beta_1, \ldots, \beta_p] \end{aligned}$$

$$\begin{aligned} \mathcal{L}(\beta|\mathcal{D}) &= \sum_{i=1}^n \log P(y^i|\boldsymbol{x}^i) + \text{Cte} \\ &= \sum_{i=1}^n \left(y^i \log g^i + (1-y^i)\log(1-g^i)\right) \end{aligned}$$

Maximum likelihood estimation of logistic regression coefficients

$$\mathcal{L}(\beta|\mathcal{D}) = \sum_{i=1}^{n} \left(y^i \log g^i + (1 - y^i) \log(1 - g^i) \right)$$

$$g = P(y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-(\beta^\top \boldsymbol{x})}}$$

$$\begin{aligned} \boldsymbol{x} &\leftarrow [1, x_1, \ldots, x_p] \\ \beta &\leftarrow [\beta_0, \beta_1, \ldots, \beta_p] \end{aligned}$$

- **Gradient of the log likelihood** $\nabla_\beta \mathcal{L}$ 

To maximize the likelihood, we want to set its gradient to 0:

$$\nabla_\beta \mathcal{L} = 0$$

$$\nabla_\beta \mathcal{L} = \sum_{i=1}^{n} \frac{y^i}{g^i} \nabla_\beta g^i + (1-y^i) \frac{-1}{1-g^i} \nabla_\beta g^i \qquad \text{with } g^i = g(x^i, \beta)$$

and $$\nabla_\beta g^i = \frac{x^i \exp(-\beta^T x^i)}{(1+\exp(-\beta^T x^i))^2}$$

$$\nabla_\beta g^i = x^i g^i (1-g^i)$$

Hence
$$\nabla_\beta \mathcal{L} = \sum_{i=1}^{n} \frac{y^i}{g^i} x^i g^i (1-g^i) + (1-y^i) \frac{-x^i}{1-g^i} g^i (1-g^i)$$

$$= \sum_{i=1}^{n} x^i \left(y^i (1-g^i) - (1-y^i) g^i \right)$$

$$= \sum_{i=1}^{n} (y^i - g^i) x^i$$

Finally
$$\boxed{\nabla_\beta \mathcal{L} = \sum_{i=1}^{n} \left(y^i - \frac{1}{1+e^{-\beta^T x^i}} \right) x^i}$$

$\nabla_\beta \mathcal{L} = 0$ cannot be solved analytically ■

Maximum likelihood estimation of logistic regression coefficients

$$\mathcal{L}(\beta|\mathcal{D}) = \sum_{i=1}^{n} \left(y^i \log g^i + (1 - y^i) \log(1 - g^i) \right)$$

$$g = P(y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-(\beta^\top \boldsymbol{x})}}$$

$$\begin{aligned} \boldsymbol{x} &\leftarrow [1, x_1, \ldots, x_p] \\ \beta &\leftarrow [\beta_0, \beta_1, \ldots, \beta_p] \end{aligned}$$

- **Gradient of the log likelihood** $\nabla_\beta \mathcal{L}$

$$\nabla_\beta g^i = \boldsymbol{x}^i g^i (1 - g^i) \qquad \nabla_\beta \mathcal{L} = \sum_{i=1}^{n} (y^i - g^i) \boldsymbol{x}^i$$

- **To maximize the likelihood:**
  - set the gradient to 0 $\sum_{i=1}^{n} \left(y^i - \frac{1}{1 + e^{-\beta^\top \boldsymbol{x}^i}} \right) \boldsymbol{x}^i = 0$
  - cannot be solved analytically
  - -L **convex** so we can use **gradient descent** (no local minima)

Proof that $\mathcal{L}$ is concave:

$$\mathcal{L}(\beta|X) = \sum_{i=1}^{n} y^i \log\left(\frac{e^{\beta^T x^i}}{1+e^{\beta^T x^i}}\right) + (1-y^i)\log\left(\frac{1}{1+e^{\beta^T x^i}}\right)$$

$$= \sum_{i=1}^{n} -y^i \log(1+e^{\beta^T x^i}) + y \log e^{\beta^T x^i} + (1-y^i)(-\log(1+e^{\beta^T x^i}$$

$$= \sum_{i=1}^{n} \underbrace{-\log(1+e^{\beta^T x^i})}_{\text{negative log sum exp}} + \underbrace{y\,\beta^T x^i}_{\text{affine}}$$

negative log sum exp
$\Rightarrow$ concave

affine

Sum of concave + affine = concave

Sum of concave = concave

Summary

- **MAP estimate:**

$$\theta_{\text{MAP}} = \arg\max_{\theta} p(\theta|\mathcal{X})$$

- **MLE:**

$$\theta_{\text{MLE}} = \arg\max_{\theta} p(\mathcal{X}|\theta)$$

- **Bayes estimate:**

$$\theta_{\text{Bayes}} = \mathbb{E}[\theta|\mathcal{X}] = \int \theta p(\theta|\mathcal{X}) d\theta$$

- Assuming Gaussian error, maximizing the likelihood is equivalent to minimizing the RSS.
- **Linear regression MLE:**

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

- **Logistic regression MLE:** solve with gradient descent.

References

- *A Course in Machine Learning.* `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`
  - **Least-squares regression:** Chap 7.6
- *The Elements of Statistical Learning.* `http://web.stanford.edu/~hastie/ElemStatLearn/`
  - **Least-squares regression:** Chap 2.2.1, 3.1, 3.2.1
  - **Gauss-Markov theorem:** Chap 3.2.3

class GradientDescentOptimizer():

```
def compute_fprime(self):
    self.fgx    = self.fprime(self.beta)

def step(self):
    self.compute_fprime()
    d_beta    = -1*self.lr*self.fgx
    self.beta = self.beta + d_beta

def optimize(self, max_iter=100):
    it = 0
    while it < max_iter:+
        self.step()
        it = it + 1
        self.beta_history += [self.beta]
        # You may have to do a deep copy of self.beta instead:
        # b = copy.deepcopy(self.beta)
        # self.beta_history += [b]
```

class LeastSquaresRegr()

```
class LeastSquaresRegr():
    def __init__(self,):
        self.coef_ = None

    def fit(self, X, y):
        (n, p) = X.shape
        X_aug = np.ones((n, p + 1))
        X_aug[:, 1:] = X
        self.coef_ = (np.dot(np.linalg.inv(X_aug.T.dot(X_aug)), X_aug.T)).dot(y)

    def predict(self, X):
        (n, p) = X.shape
        X_aug = np.ones(((n, p + 1))
        X_aug[:, 1:] = X
        return self.coef_.dot(X_aug.T)
```

class seq_LeastSquaresRegr()

```
def fit(self,X,y):
    X_aug      = np.ones((X.shape[0], X.shape[1]+1))
    X_aug[:, 1:] = X
    self.coef_ = np.random.randn(X_aug.shape[1],)

    def f_ls(beta):
        phi  = np.squeeze(np.dot(X_aug, beta.reshape([-1,1])))
        return np.mean(0.5*(y - phi)**2)

    def fprime_ls(beta):
        phi  = np.dot(X_aug, beta.reshape([-1,1]))
        diff = phi - y.reshape([-1, 1])
        diff = np.tile(diff, [1, X_aug.shape[1]])
        return np.mean(X_aug*diff, axis=0)

    self.gd = GradientDescentOptimizer(f_ls, fprime_ls, self.coef_, lr=1e-2)
    self.gd.optimize(max_iter=1000)
    self.coef_ = self.gd.beta
```

```
def predict(self,X):
    X_aug      = np.ones((X.shape[0], X.shape[1]+1))
    X_aug[:, 1:] = X
    return np.squeeze(np.dot(X_aug, self.coef_.reshape([-1, 1])))
```

class seq_LeastSquaresRegr()

```
def fit(self,X,y):
    X_aug      = np.ones((X.shape[0], X.shape[1]+1))
    X_aug[:, 1:] = X
    self.coef_ = np.random.randn(X_aug.shape[1],)

    def f_ls(beta):
        phi  = np.squeeze(np.dot(X_aug, beta.reshape([-1,1])))
        return np.mean(0.5*(y - phi)**2)

    def fprime_ls(bet
        phi  = np.dot
        diff = phi -
        diff = np.til
        return np.mea

    self.gd = Gradien                                    self.coef_, lr=1e-2)
    self.gd.optimize(max_iter=1000)
    self.coef_ = self.gd.beta
```

```
x = np.array([1, 2, 3, 4, 5])
print(x.shape, x)
x2 = x.reshape([-1, 1])
print(x2.shape, x2)
x3 = np.squeeze(x2)
print(x3.shape, x3)
```

class seq_LeastSquaresRegr()

```
def fit(self,X,y):
    X_aug       = np.ones((X.shape[0], X.shape[1]+1))
    X_aug[:, 1:] = X
    self.coef_ = np.random.randn(X_aug.shape[1],)

    def f_ls(beta):
        phi  = np.sq                                    1,1]))
        return np.me
                    M = np.tile(x2, [1, 7])
                    print M.shape
                    print M
    def fprime_ls(beta):
        phi  = np.dot(X_aug, beta.reshape([-1,1]))
        diff = phi - y.reshape([-1, 1])
        diff = np.tile(diff, [1, X_aug.shape[1]])
        return np.mean(X_aug*diff, axis=0)

    self.gd = GradientDescentOptimizer(f_ls, fprime_ls, self.coef_, lr=1e-2)
    self.gd.optimize(max_iter=1000)
    self.coef_ = self.gd.beta
```

class LogisticRegr()

```
def fit(self, X, y):
    (n, p) = X.shape
    X_aug = np.ones((n, p + 1))
    X_aug[:, 1:] = X
    coef_ = np.random.normal(size=(p+1,), loc=0.0, scale=0.1)

    c0_ids = np.where(y == 0)[0]
    c1_ids = np.where(y == 1)[0]

    def f_lr(beta):      # The loss (or cost) function.
        beta  = np.reshape(beta, [-1, 1])
        phi_0 = sigmoid(np.dot(X_aug[c0_ids,:], beta))
        phi_1 = sigmoid(np.dot(X_aug[c1_ids,:], beta))
        loss = -1*np.sum(np.log(phi_1))
        loss = loss - np.sum(np.log(1 - phi_0))
        return loss

    def fprime_lr(beta):   # The gradient of the loss function given beta.
        phi  = sigmoid(np.dot(X_aug, beta.reshape([-1,1])))
        diff = phi - y.reshape([-1, 1])
        grad = X_aug*diff
        return np.mean(grad, axis=0)

    self.gd = GradientDescentOptimizer(f_lr, fprime_lr, coef_, lr=2e-2)
    self.gd.optimize(max_iter=100)
    self.coef_ = self.gd.beta
```

class LogisticRegr()

```
def predict_proba(self, X):
    (n, p) = X.shape
    X_aug = np.ones((n, p+1))
    X_aug[:, 1:] = X
    r = sigmoid(np.dot(X_aug, self.coef_.reshape([-1,1]))).squeeze()
    pred  = np.zeros((n, 2))
    pred[:,1] = r
    pred[:,0] = 1 - r
    return pred

def predict(self, X):
    pred = self.predict_proba(X)
    return pred.argmax(axis=1)
```