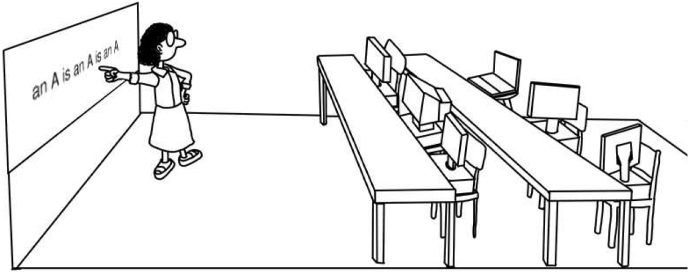
Machine Learning

Krystian Mikolajczyk & Deniz Gunduz

Department of Electrical and Electronic Engineering
Imperial College London



Part 4 summary

Department of Electrical and Electronic Engineering

Imperial College London

- Logistic Regression
- Gradient descent
- Neural Networks

EE2 Mathematics: Maximum likelihood predictor

Statistical model:

- Class: $y = \pm 1$, distribution $P(y)$ (predicting heart attack)
- Independent $0/1$ observations x_i (binary):
  $P(x_1, \dots, x_d|y) = P(x_1|y) \cdot \dots \cdot P(x_d|y)$

Posterior probability

$$\begin{aligned}
P(y|x_1,\dots,x_d) &= \frac{P(y, x_1, \dots, x_d)}{P(x_1, \dots, x_d)} \\
&= \frac{P(x_1, \dots, x_d|y)P(y)}{P(x_1, \dots, x_d)} && \text{(Bayes rule)} \\
&= \frac{P(x_1|y) \cdot \dots \cdot P(x_d|y)P(y)}{P(x_1, \dots, x_d)} && \text{(independence)}
\end{aligned}$$

Odds:

$$\frac{P(y=1|x_1,\dots,x_d)}{P(y=-1|x_1,\dots,x_d)} = \frac{P(x_1|y=1)}{P(x_1|y=-1)} \cdot \dots \cdot \frac{P(x_d|y=1)}{P(x_d|y=-1)} \cdot \frac{P(y=1)}{P(y=-1)}$$

EE2 Mathematics: Maximum likelihood predictor

Log odds:

$$\log \frac{P(y=1|x_1,\dots,x_d)}{P(y=-1|x_1,\dots,x_d)} = \log \frac{P(y=1)}{P(y=-1)} + \sum_{i=1}^{d} \log \frac{P(x_i|y=1)}{P(x_i|y=-1)}$$

$$= \sum_{i=1}^{d} w_i^* \mathbf{x}_i + w_0 = \mathbf{w}_*^\top \mathbf{x} \quad \text{linear model}$$

$$\log P(x_i|y) = x_i \log P(x_i=1|y) + (1-x_i)\log P(x_i=0|y)$$

$$= x_i \Big(\underbrace{\log P(x_i=1|y) - \log P(x_i=0|y)}_{\beta_{i,y}} \Big) + \underbrace{\log P(x_i=0|y)}_{\gamma_{i,y}}$$

$$\log \frac{P(x_i|y=1)}{P(x_i|y=-1)} = \log P(x_i|y=1) - \log P(x_i|y=-1)$$

$$= x_i(\underbrace{\beta_{i,1} - \beta_{i,-1}}_{w_i^*}) + (\gamma_{i,1} - \gamma_{i,-1})$$

Logistic regression

- Relax binary and independence assumptions on x_i
  - e.g., $\mathbf{x}$: cholesterol level, triglyceride level, age, weight, etc.
- With $\mathbf{x} = (1, x_1, \ldots, x_d)$ and $y \in \{-1, +1\}$, assume

$$\log \frac{P(y = +1|x_1, \ldots, x_d)}{P(y = -1|x_1, \ldots, x_d)} = \mathbf{w}_*^\top \mathbf{x} = h(\mathbf{x}, \mathbf{w})$$

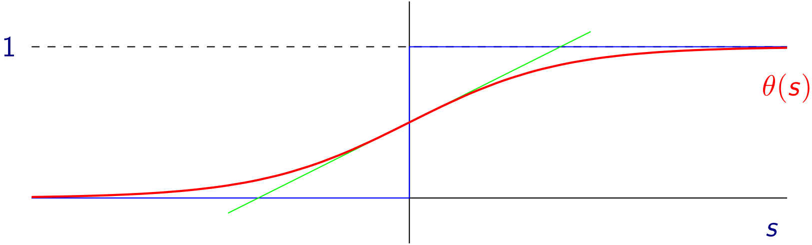
Using $P(y = -1|\mathbf{x}) = P(y = -1|x_1, \ldots, x_d) = 1 - P(y = +1|x_1, \ldots, x_d)$,

$$P(y = +1|\mathbf{x}) = \theta(\mathbf{w}_*^\top \mathbf{x}) \qquad P(y = -1|\mathbf{x}) = 1 - \theta(\mathbf{w}_*^\top \mathbf{x})$$

$$P(y = 1|\mathbf{x}) = \frac{e^{\mathbf{w}_*^\top \mathbf{x}}}{1 + e^{\mathbf{w}_*^\top \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}_*^\top \mathbf{x}}} = \theta(\mathbf{w}_*^\top \mathbf{x}) \qquad \text{(logistic function)}$$

Logistic regression: Logistic function

$s = \mathbf{w}_*^\top \mathbf{x}$ (score)



$$\theta(s) = \frac{e^s}{1 + e^s} = \frac{1}{1 + e^{-s}}$$

soft threshold: uncertainty

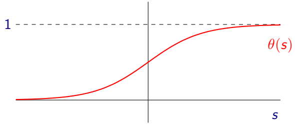
sigmoid: flattened out "S"

Logistic regression: Error measure

If

$$P(y|\mathbf{x}) = \begin{cases} \theta(\mathbf{w}_*^\top \mathbf{x}) & \text{if } y = +1; \\ 1 - \theta(\mathbf{w}_*^\top \mathbf{x}) & \text{if } y = -1, \end{cases}$$

how likely is it to get y from $\mathbf{x}$?



Note: $\theta(-s) = 1 - \theta(s)$.

Using $\theta(-s) = 1 - \theta(s)$ and binary label y

$$P(y|\mathbf{x}) = \theta(y\mathbf{w}^\top \mathbf{x}).$$

Likelihood of $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is

$$\prod_{i=1}^{n} P(y_n|\mathbf{x}_n) = \prod_i \theta(y_i \mathbf{w}^\top \mathbf{x}) \ .$$

Instead of maximizing likelihood ...

Logistic regression: : Error measure

Minimize negative log likelihood

$$
\begin{aligned}
-\log\left(\prod_{i=1}^{n} P(y_n|\mathbf{x}_n)\right) &= -\log\left(\prod_{i} \theta(y_i \mathbf{w}^\top \mathbf{x})\right) \\
&= \sum_{i=1}^{n} \log\left(\frac{1}{\theta(y_i \mathbf{w}^\top \mathbf{x})}\right) \\
&= \sum_{i=1}^{n} \underbrace{\log\left(1 + e^{-y_i \mathbf{w}^\top \mathbf{x}_i}\right)}_{\ell(\theta(\mathbf{w}_*^\top \mathbf{x}_i), y_i)}
\end{aligned}
$$

Empirical error: cross-entropy

$$
\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-y_i \mathbf{w}^\top \mathbf{x}_i}\right)
$$

Logistic regression: Cross-entropy error

Let $h(\mathbf{x}) = \theta(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}}}$

then $\ell(h(\mathbf{x}), y) = \log(1 + e^{-y\mathbf{w}^\top \mathbf{x}}) = \begin{cases} \log \frac{1}{h(\mathbf{x})} & \text{if } y = +1; \\ \log \frac{1}{1-h(\mathbf{x})} & \text{if } y = -1. \end{cases}$

Expression for $\widehat{R}_n$:

$$\frac{1}{n}\sum_{i=1}^{n}[\mathbb{I}\,(y_i = +1)\log\frac{1}{h(\mathbf{x}_i)} + \mathbb{I}\,(y_i = -1)\log\frac{1}{1-h(\mathbf{x}_i)}]$$

For two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$

- Entropy: $H(p) = p\log\frac{1}{p} + (1-p)\log\frac{1}{1-p}$
  - maximum compression: minimum expected code length when coding symbols from p (achieved with coding distribution p)
- Cross entropy: $p\log\frac{1}{q} + (1-p)\log\frac{1}{1-q}$
  - expected code length when compressing symbols from p using coding distribution q

Learning algorithm: ERM

How to minimize $\widehat{R}_n(\mathbf{w})$?

Linear regression

$$\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \Rightarrow \text{ closed-form solution}$$

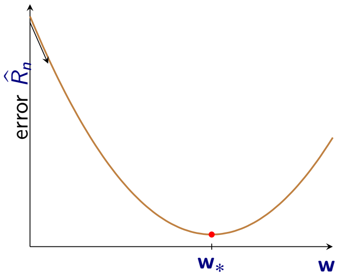
For logistic regression:

$$\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\log\left(1 + e^{-y_i \mathbf{w}^\top \mathbf{x}_i}\right) \Rightarrow \text{ no closed-form solution}$$

Use gradient descent!

Gradient descent: Direction

- General method for non-linear optimization: $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \mathbf{v}$
- Direction $\mathbf{v}$: starting from $\mathbf{w}_t$, step along the steepest slope
  - $\mathbf{v}$ is a unit vector.
- Step size η: how quickly find the minimum
  - η is a scalar.

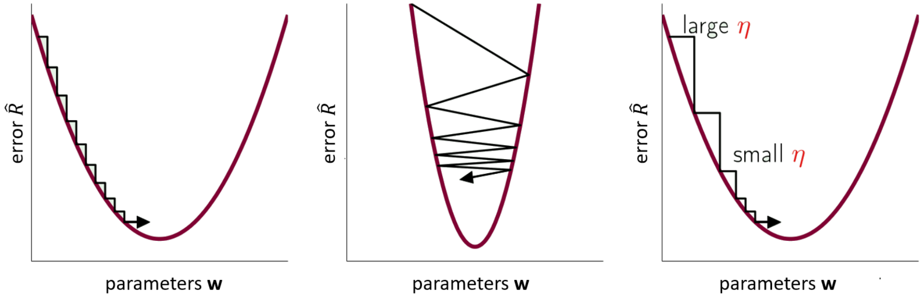


$$Direction: \Delta \widehat{R}_n = \widehat{R}_n(\mathbf{w}_{t+1}) - \widehat{R}_n(\mathbf{w}_t) = \widehat{R}_n(\mathbf{w}_t + \eta \mathbf{v}) - \widehat{R}_n(\mathbf{w}_t)$$

$$= \eta \nabla \widehat{R}_n(\mathbf{w}_t)^\top \mathbf{v} + O(\eta^2) \geqslant -\eta \|\nabla \widehat{R}_n(\mathbf{w}_t)\| + O(\eta^2)$$

$$\text{with “equality” if } \mathbf{v} = -\frac{\nabla \widehat{R}_n(\mathbf{w}_t)}{\|\nabla \widehat{R}_n(\mathbf{w}_t)\|}$$

Gradient descent: Step size



η is small $\qquad$ η too large $\qquad$ variable η

Heuristic: step size should increase with the slope

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta\mathbf{w}$$
$$\Delta\mathbf{w} = -\eta\nabla\widehat{R}_n(\mathbf{w}_t) \text{ (with } \eta \text{ redefined)}$$

Gradient descent: ERM

ERM: $\mathbf{w}_* = \text{argmin}_{\mathbf{w}} \widehat{R}_n(\mathbf{w})$ with gradient descent

- Initialize the weights $\mathbf{w}_0$.
- For $t = 0, 1, 2, \ldots$
  - Compute the gradient of $\widehat{R}_n$ or $\mathcal{L}_n$ (e.g. logistic regression)

$$\nabla \widehat{R}_n(\mathbf{w}_t) = -\frac{1}{n} \sum_{i=1}^{n} \frac{y_i \mathbf{x}_i}{1 + e^{y_i \mathbf{w}_t^\top \mathbf{x}_i}}$$

  - Update the weights $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \widehat{R}_n(\mathbf{w}_t)$.
  - Iterate until some stopping condition is met (small change, validation error, ...).
- Return the final (current) weights $\mathbf{w}_* = \mathbf{w}_t$.

Gradient descent: Loss

- Each loss function ℓ can be used in ERM

  Linear regressions: $\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2$.

  Logistic regression: $\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\log\left(1 + e^{-y_i \mathbf{w}^\top \mathbf{x}_i}\right)$.

  Classification: $\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left(y_i \neq \mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i)\right)$.

  Has to be differentiable!

- General error: $\widehat{R}_n(h) = \frac{1}{n}\sum_{i=1}^{n}\ell(h(\mathbf{x}_i), y_i)$.
- Problems: possibly non-convex, computational issues.
- Can be solved approximately using stochastic gradient descent

Stochastic Gradient Descent (SGD)

- Computing full gradient $\widehat{R}_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}_t, \mathbf{x}_i, y_i)$ may be very expensive: e.g., dimension d large, number of data points n large.

ERM: $\mathbf{w}_* = \operatorname{argmin}_{\mathbf{w}} \widehat{R}_n(\mathbf{w})$ with stochastic gradient descent

- Initialize the weights $\mathbf{w}_0$ and iterate for $i = 1, \ldots, n$ and $t = 0, 1, 2, \ldots$
  - Compute gradient at one point (an estimate) :

$$\nabla \ell_{t,i} = \frac{\partial \ell(\mathbf{w}_t, \mathbf{x}_i, y_i)}{\partial \mathbf{w}_t}.$$

    where i is selected uniformly at random from $\{1, \ldots, n\}$.
  - Update the weights $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \ell_{t,i}$.
  - Iterate until some stopping condition is met (small change, validation error, ...).
- Return the final (current) weights $\mathbf{w}_* = \mathbf{w}_t$.

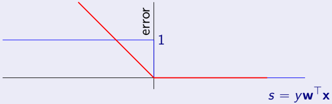
$$\mathbb{E}_i\left[\nabla \ell_{t,i}\right] = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial \ell(\mathbf{w}_t, \mathbf{x}_i, y_i)}{\partial \mathbf{w}_t} = \nabla \widehat{R}_n(\mathbf{w}_t)$$

Stochastic Gradient Descent

Hinge Loss

- Perceptron Learning Algorithm computes SG estimate for the shifted hinge loss

$$\ell(\mathbf{w}, \mathbf{x}_i, y_i) = \max(0, -y_i \mathbf{w}^\top \mathbf{x}_i)$$
$$\partial\ell/\partial w = -y_i \mathbf{x}_i \mathbb{I}\left(\mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i\right)$$



- simple and efficient - based on one example
- variants of gradient descent (SGD adds randomness)
- improved gradient estimates
- minibatches
- non-uniform sampling
- different step sizes for different coordinates
- distributed
- ...