Big Data Analytics

The Linear Model

Souhaib Ben Taieb

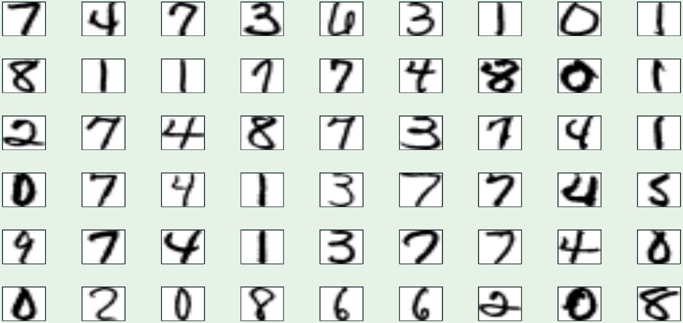
March 25, 2021

University of Mons

Table of contents

Table of contents

A real data set



Input representation

'raw' input $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_{256})$

linear model: $(w_0, w_1, w_2, \cdots, w_{256})$

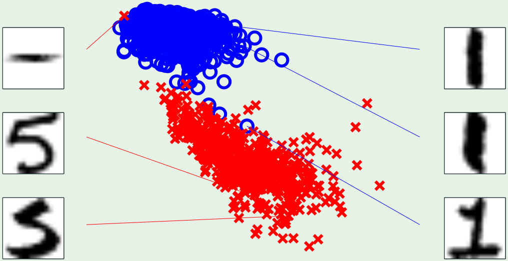
Features: Extract useful information, e.g.,

intensity and symmetry $\mathbf{x} = (x_0, x_1, x_2)$

linear model: (w_0, w_1, w_2)

Illustration of features

$\mathbf{x} = (x_0, x_1, x_2)$ $\quad x_1$: intensity $\quad x_2$: symmetry



A simple learning algorithm - PLA

The perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\intercal}\mathbf{x})$$

Given the training set:

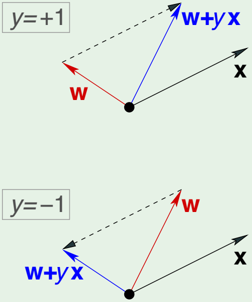
$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

pick a misclassified point:

$$\text{sign}(\mathbf{w}^{\intercal}\mathbf{x}_n) \neq y_n$$

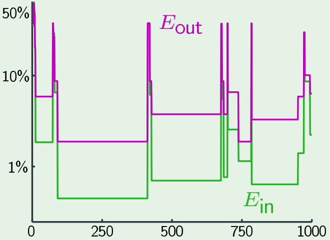
and update the weight vector:

$$\boxed{\mathbf{w} \leftarrow \mathbf{w} + y_n\mathbf{x}_n}$$

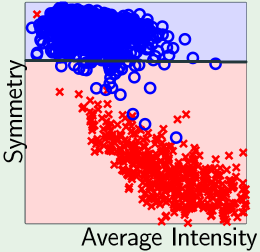


What PLA does

Evolution of E_{in} and E_{out}

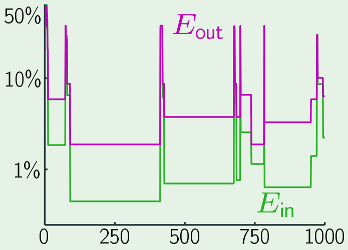


Final perceptron boundary

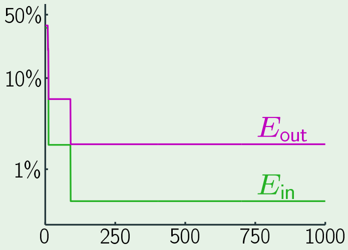


The ‘pocket’ algorithm

PLA:

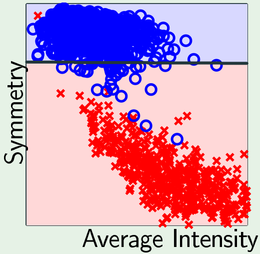


Pocket:



Classification boundary - PLA versus Pocket

PLA:



Pocket:

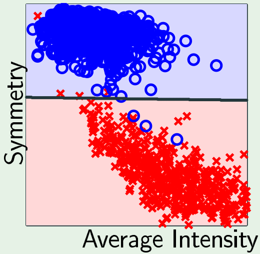


Table of contents

Credit again

Classification: Credit approval (yes/no)

Regression: Credit line (dollar amount)

Input: $\mathbf{x}$ =

age	23 years
annual salary	\$30,000
years in residence	1 year
years in job	1 year
current debt	\$15,000
$\cdots$	$\cdots$

Linear regression output: $h(\mathbf{x}) = \sum_{i=0}^{d} w_i \, x_i = \mathbf{w}^{\mathsf{T}}\mathbf{x}$

The data set

Credit officers decide on credit lines:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

$y_n \in \mathbb{R}$ is the credit line for customer $\mathbf{x}_n$.

Linear regression tries to replicate that.

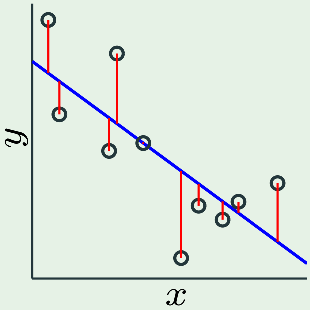
How to measure the error

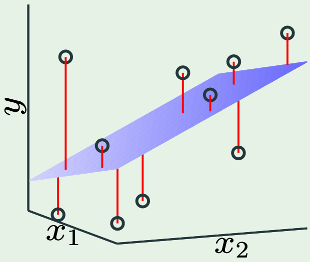
How well does $h(\mathbf{x}) = \mathbf{w}^{\intercal}\mathbf{x}$ approximate $f(\mathbf{x})$?

In linear regression, we use squared error $(h(\mathbf{x}) - f(\mathbf{x}))^2$

in-sample error: $E_{\text{in}}(h) = \dfrac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$

Illustration of linear regression





The expression for E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^{\intercal}\mathbf{x}_n - y_n)^2$$

$$= \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

where $\mathrm{X} = \begin{bmatrix} \text{—}\mathbf{x}_1{}^{\intercal}\text{—} \\ \text{—}\mathbf{x}_2{}^{\intercal}\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N{}^{\intercal}\text{—} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

The expression for E_{in}

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$$

$$E_{\text{in}}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 \tag{1}$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N \quad \text{and} \quad \boldsymbol{X} = \begin{bmatrix} — \boldsymbol{x}_1^T — \\ — \boldsymbol{x}_2^T — \\ \vdots \\ — \boldsymbol{x}_N^T — \end{bmatrix} \in \mathbb{R}^{N \times (d+1)}$$

$$E_{\text{in}}(\boldsymbol{w}) = \frac{1}{N} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 , \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm of a vector.

Minimizing E_{in}

$$E_{\text{in}}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2 \tag{3}$$

$$= \frac{1}{N}\|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 \tag{4}$$

$$= \frac{1}{N}(\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w}) \qquad (\|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^T\boldsymbol{v}) \tag{5}$$

$$= \frac{1}{N}(\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{w}^T X^T\boldsymbol{y} + \boldsymbol{w}^T X^T X\boldsymbol{w}). \tag{6}$$

We need to solve the following optimization problem

$$\boldsymbol{w}_{\text{lin}} = \underset{\boldsymbol{w}\in\mathbb{R}^{d+1}}{\operatorname{argmin}}\ E_{\text{in}}(\boldsymbol{w}),$$

where $E_{\text{in}}(\boldsymbol{w})$ is a single-valued **multivariable** function.

Table of contents

Multivariate calculus - Gradient

Consider a function $r(\boldsymbol{z})$ where $\boldsymbol{z}$ is a d-vector $\boldsymbol{z} = [z_1, z_2, \ldots, z_d]^T$. The **gradient vector** of this function is given by the **partial derivatives** with respect to each of the independent variables,

$$\nabla r(\boldsymbol{z}) \equiv g(\boldsymbol{z}) \equiv \begin{bmatrix} \frac{\partial r}{\partial z_1}(\boldsymbol{z}) \\ \frac{\partial r}{\partial z_2}(\boldsymbol{z}) \\ \vdots \\ \frac{\partial r}{\partial z_d}(\boldsymbol{z}) \end{bmatrix}.$$

Multivariate calculus - Gradient

The gradient is the generalization of the concept of derivative, which captures the local rate of change in the value of a function, in multiple directions.

It is very important to remember that the gradient of a function is only defined if the function is real-valued, that is, if it returns a scalar value.

If the gradient exists at every point, the function is said to be **differentiable**. If each entry of the gradient is continuous, we say the function is **once continuously differentiable**.

Note on notations: $\nabla_{\boldsymbol{z}} r$ means the gradient of r where the ith partial derivative is taken with respect to z_i. The same notation is used for the directional derivatives defined as $\nabla_{\boldsymbol{z}} r = \nabla r \cdot \boldsymbol{z}$.

Multivariate calculus - Hessian

While the gradient of a function of d variables (i.e. the "first derivative") is an d-vector, the "second derivative" of an d-variable function is defined by d^2 partial derivatives (the derivatives of the d first partial derivatives with respect to the d variables):

$$\frac{\partial r}{\partial z_i}\left(\frac{\partial r}{\partial z_j}\right) = \frac{\partial^2 r}{\partial z_i \partial z_j}, i \neq j \quad \text{and} \quad \frac{\partial r}{\partial z_i}\left(\frac{\partial r}{\partial z_i}\right) = \frac{\partial^2 r}{\partial^2 z_i}, i = j$$

where $i, j = 1, \ldots, d$.

Multivariate calculus - Hessian

If r is single-valued and the partial derivatives $\frac{\partial r}{\partial z_i}$, $\frac{\partial r}{\partial z_j}$ and $\frac{\partial^2 r}{\partial z_i \partial z_j}$ are *continuous*, then $\frac{\partial^2 r}{\partial z_i \partial z_j}$ exists and $\frac{\partial^2 r}{\partial z_i \partial z_j} = \frac{\partial^2 r}{\partial z_j \partial z_i}$.

Therefore the second order partial derivatives can be represented by a square *symmetric* matrix called the **Hessian** matrix:

$$\nabla^2 r(\boldsymbol{z}) \equiv H(\boldsymbol{z}) \equiv \begin{pmatrix} \frac{\partial^2 r}{\partial^2 z_1}(\boldsymbol{z}) & \cdots & \frac{\partial^2 r}{\partial z_1 \partial z_d}(\boldsymbol{z}) \\ \vdots & & \vdots \\ \frac{\partial^2 r}{\partial z_d \partial z_1}(\boldsymbol{z}) & \cdots & \frac{\partial^2 r}{\partial^2 z_d}(\boldsymbol{z}) \end{pmatrix},$$

which contains $d(d+1)/2$ independent elements.

If a function has a Hessian matrix at every point, we say that the function is **twice differentiable**. If each entry of the Hessian is continuous, we say the function is **twice continuously differentiable**.

Multivariate calculus - Hessian

For functions of a vector, the gradient is a vector, and **we cannot take the gradient of a vector**. *Therefore, it is* not *the case that the Hessian is the gradient of the gradient*. However, this is **almost** true, in the following sense: If we look at the ith entry of the gradient $(\nabla_{\boldsymbol{z}} r(\boldsymbol{z}))_i = \frac{\partial r(\boldsymbol{z})}{\partial z_i}$, and take the gradient with respect to $\boldsymbol{z}$, we get

$$\nabla_{\boldsymbol{z}} \frac{\partial r(\boldsymbol{z})}{\partial z_i} \equiv \begin{bmatrix} \frac{\partial r}{\partial z_i \partial z_1}(\boldsymbol{z}) \\ \frac{\partial r}{\partial z_i \partial z_2}(\boldsymbol{z}) \\ \vdots \\ \frac{\partial r}{\partial z_i \partial z_d}(\boldsymbol{z}) \end{bmatrix},$$

which is the ith column (or row) of the Hessian. Therefore,

$$\nabla^2_{\boldsymbol{z}} = [\nabla_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} r(\boldsymbol{z}))_1 \ \nabla_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} r(\boldsymbol{z}))_2 \ \cdots \ \nabla_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} r(\boldsymbol{z}))_d].$$

Definiteness of a matrix

- $A \in \mathbb{R}^{n \times n}$ is **positive semi-definite**, denoted $A \succeq 0$, if $\boldsymbol{v}^T A \boldsymbol{v} \geq 0, \forall \boldsymbol{v} \in \mathbb{R}^n_{>0}$. If $A = A^T$, all the eigenvalues of A are larger or equal to zero.
- $A \in \mathbb{R}^{n \times n}$ is **positive definite**, denoted $A \succ 0$, if $\boldsymbol{v}^T A \boldsymbol{v} > 0, \forall \boldsymbol{v} \in \mathbb{R}^n_{>0}$. If $A = A^T$, all the eigenvalues of A are strictly positive.
- $A \in \mathbb{R}^{n \times n}$ is **negative definite**, denoted $A \prec 0$, if $\boldsymbol{v}^T A \boldsymbol{v} < 0, \forall \boldsymbol{v} \in \mathbb{R}^n_{>0}$. If $A = A^T$, all the eigenvalues of A are strictly negative.
- $A \in \mathbb{R}^{n \times n}$ is **indefinite** if there exists $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^n_{>0}$ such that $\boldsymbol{v}_1^T A \boldsymbol{v}_1 > 0$ and $\boldsymbol{v}_2^T A \boldsymbol{v}_2 < 0$. If $A = A^T$, A has eigenvalues of mixed sign.

Convex functions

A function $r : \mathbb{R}^d \to \mathbb{R}$ is called **convex** if for any $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{R}^d$ and every $\lambda \in [0, 1]$, we have

$$f(\lambda \boldsymbol{z}_1 + (1-\lambda)\boldsymbol{z}_2) \leq \lambda f(\boldsymbol{z}_1) + (1-\lambda) f(\boldsymbol{z}_2).$$

- r is **convex** <u>if and only if</u> its Hessian is **positive semidefinite**.
- *For a convex function, any local minimum (optimum) is also a global minimum (optimum)*.

Optimality conditions for unconstrained optimization

- Necessary Optimality Conditions
  - *First-Order Necessary Conditions*
    If $\boldsymbol{z}^*$ is a local minimum of r and r is once continously differentiable, **then** $\nabla r(\boldsymbol{z}^*) = 0$.
  - *Second-Order Necessary Conditions*
    If $\boldsymbol{z}^*$ is a local minimum of r and r is twice continously differentiable, **then** $\nabla^2 r(\boldsymbol{z}^*) \succeq 0$.
  - There may exist points that satisfy these conditions but are not local minima, e.g. $z = 0$ for $r(z) = z^3$, $r(z) = |z|^3$ or $r(z) = -|z|^3$.
- Sufficient Optimality Conditions
  - *First-Order Sufficient Conditions*
    If r is once continously differentiable and convex, and $\nabla r(\boldsymbol{z}^*) = 0$ **then** $\boldsymbol{z}^*$ is a global minimum of r.
  - *Second-Order Sufficient Conditions*
    If r is once continously differentiable, $\nabla r(\boldsymbol{z}^*) = 0$ and $\nabla^2 r(\boldsymbol{z}^*) \succ 0$., **then** $\boldsymbol{z}^*$ is a (strict) local minimum of r.

Table of contents

Minimizing E_{in}

Since $E_{\text{in}}(\boldsymbol{w})$ is differentiable and convex, we can find the global minimum of $E_{\text{in}}(\boldsymbol{w})$ by requiring $\nabla E_{\text{in}}(\boldsymbol{w}) = 0$. We have

$$
\begin{aligned}
&\nabla_{\boldsymbol{w}} \left(\frac{1}{N} (\boldsymbol{y}^T \boldsymbol{y} - 2\boldsymbol{w}^T X^T \boldsymbol{y} + \boldsymbol{w}^T X^T X \boldsymbol{w}) \right) = 0 \\
&\iff \frac{2}{N} \left(X^T X \boldsymbol{w} - X^T \boldsymbol{y} \right) = 0 \iff X^T X \boldsymbol{w} = X^T \boldsymbol{y} \\
&\iff \boldsymbol{w} = (X^T X)^{-1} X^T \boldsymbol{y} \qquad \text{(assuming } X^T X \text{ is invertible)}
\end{aligned}
$$

where we used the following gradient identities

$$
\nabla_{\boldsymbol{w}} (\boldsymbol{w}^T A \boldsymbol{w}) = (A + A^T)\boldsymbol{w}, \qquad \nabla_{\boldsymbol{w}} (\boldsymbol{w}^T \boldsymbol{b}) = \boldsymbol{b}.
$$

The Hessian of $E_{\text{in}}(\boldsymbol{w})$ is given by

$$
H(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \left(\frac{2}{N} \left(X^T X \boldsymbol{w} - X^T \boldsymbol{y} \right) \right) = \frac{2}{N} X^T X, \tag{7}
$$

where we used $\nabla_{\boldsymbol{w}} (A\boldsymbol{w}) = A$. $H(\boldsymbol{w})$ is positive semi-definite.

Minimizing E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\mathrm{X}^{\intercal}(\mathrm{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$$

$$\mathrm{X}^{\intercal}\mathrm{X}\mathbf{w} = \mathrm{X}^{\intercal}\mathbf{y}$$

$$\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y} \quad \text{where} \quad \mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$$

$\mathrm{X}^{\dagger}$ is the '**pseudo-inverse**' of X

The linear regression algorithm

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

$$\underbrace{\mathrm{X} = \begin{bmatrix} —\mathbf{x}_1^\intercal— \\ —\mathbf{x}_2^\intercal— \\ \vdots \\ —\mathbf{x}_N^\intercal— \end{bmatrix}}_{\text{input data matrix}}, \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2: Compute the pseudo-inverse $\mathrm{X}^\dagger = (\mathrm{X}^\intercal \mathrm{X})^{-1} \mathrm{X}^\intercal$.

3: Return $\mathbf{w} = \mathrm{X}^\dagger \mathbf{y}$.

Learning curves in linear regression

Expected E_{out} and E_{in}

Data set $\mathcal{D}$ of size N

Expected out-of-sample error $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}(g^{(\mathcal{D})})]$

Expected in-sample error $\mathbb{E}_{\mathcal{D}}[E_{\text{in}}(g^{(\mathcal{D})})]$

How do they vary with N?

Learning curves in linear regression

Linear regression case

Noisy target $y = \mathbf{w}^{*\intercal}\mathbf{x}$ + noise

Data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

Linear regression solution: $\mathbf{w} = (\mathrm{X}^\intercal \mathrm{X})^{-1}\mathrm{X}^\intercal \mathbf{y}$

In-sample error vector $= \mathrm{X}\mathbf{w} - \mathbf{y}$

'Out-of-sample' error vector $= \mathrm{X}\mathbf{w} - \mathbf{y}'$

Learning curves in linear regression

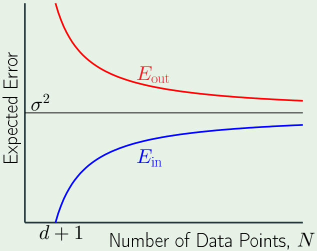
Learning curves for linear regression

Best approximation error $= \sigma^2$

Expected in-sample error $= \sigma^2 \left(1 - \frac{d+1}{N}\right)$

Expected out-of-sample error $= \sigma^2 \left(1 + \frac{d+1}{N}\right)$

Expected generalization error $= 2\sigma^2 \left(\frac{d+1}{N}\right)$



Linear regression for classification

Linear regression learns a real-valued function $y = f(\mathbf{x}) \in \mathbb{R}$

Binary-valued functions are also real-valued! $\pm 1 \in \mathbb{R}$

Use linear regression to get $\mathbf{w}$ where $\mathbf{w}^{\intercal}\mathbf{x}_n \approx y_n = \pm 1$

In this case, $\text{sign}(\mathbf{w}^{\intercal}\mathbf{x}_n)$ is likely to agree with $y_n = \pm 1$

Good initial weights for classification

Linear regression boundary

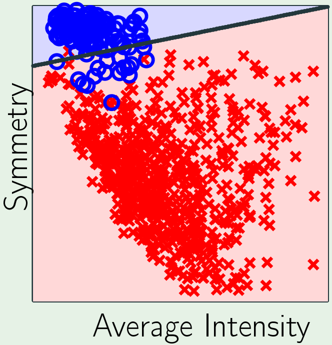
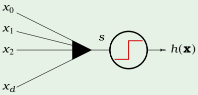


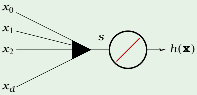
Table of contents

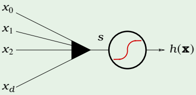
A third linear model

$$s = \sum_{i=0}^{d} w_i x_i$$

linear classification	linear regression	**logistic regression**
$h(\mathbf{x}) = \mathrm{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$



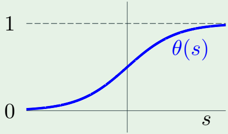




The logistic function θ

The formula:

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



soft threshold: uncertainty

sigmoid: flattened out 's'

Probability interpretation

$h(\mathbf{x}) = \theta(s)$ is interpreted as a probability

Example. Prediction of heart attacks

Input $\mathbf{x}$: cholesterol level, age, weight, etc.

$\theta(s)$: probability of a heart attack

The signal $s = \mathbf{w}^{\intercal}\mathbf{x}$ "risk score"

Genuine probability

Data $(\mathbf{x}, y)$ with binary y, generated by a noisy target:

$$P(y \mid \mathbf{x}) \; = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

The target $f : \mathbb{R}^d \to [0, 1]$ is the probability

Learn $g(\mathbf{x}) \; = \; \theta(\mathbf{w}^{\mathsf{T}} \mathbf{x}) \; \approx \; f(\mathbf{x})$

Genuine probability

Data $(\mathbf{x}, y)$ with binary y, generated by a noisy target:

$$P(y \mid \mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

The target $f : \mathbb{R}^d \to [0, 1]$ is the probability

Learn $g(\mathbf{x}) = \theta(\mathbf{w}^{\mathsf{T}} \mathbf{x}) \approx f(\mathbf{x})$

The data does not give us the value of f explicitly. It gives us samples generated by this probability. How do we learn from such data?

Error measure

For each $(\mathbf{x}, y)$, y is generated by probability $f(\mathbf{x})$

Plausible error measure based on **likelihood:**

If $h = f$, how likely to get y from $\mathbf{x}$?

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

Formula for likelihood

Since the data points are assumed to be (conditionally) independently generated, the probability of observing all the y_n's in the data set from the corresponding $\boldsymbol{x}_n$ is given by

$$\Pi_{n=1}^{N} P(y_n|\boldsymbol{x}_n).$$

where

$$P(y|\boldsymbol{x}) = \begin{cases} h(\boldsymbol{x}) & \text{for } y = +1; \\ 1 - h(\boldsymbol{x}) & \text{for } y = -1. \end{cases}$$

The method of *maximum likelihood* selects the hypothesis h which maximizes this probability.

From likelihood to E_{in}

$$\begin{aligned}
\text{Maximize } \Pi_{n=1}^{N} P(y_n|\boldsymbol{x}_n) &\equiv \text{Maximize } \ln\left(\Pi_{n=1}^{N} P(y_n|\boldsymbol{x}_n)\right) \\
&\equiv \text{Maximize } \frac{1}{N} \ln\left(\Pi_{n=1}^{N} P(y_n|\boldsymbol{x}_n)\right) \\
&\equiv \text{Minimize } -\frac{1}{N} \ln\left(\Pi_{n=1}^{N} P(y_n|\boldsymbol{x}_n)\right)
\end{aligned}$$

Furthermore, we can write

$$\begin{aligned}
& -\frac{1}{N} \ln\left(\Pi_{n=1}^{N} P(y_n|\boldsymbol{x}_n)\right) \\
&= \frac{1}{N} \sum_{n=1}^{N} \ln\left(\frac{1}{P(y_n|\boldsymbol{x}_n)}\right) \\
&= \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\{y_n = +1\} \ln\left(\frac{1}{h(\boldsymbol{x}_n)}\right) + \mathbb{1}\{y_n = -1\} \ln\left(\frac{1}{1 - h(\boldsymbol{x}_n)}\right)
\end{aligned}$$

From likelihood to E_{in}

We have $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x})$, where $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$ with $\theta(-s) = 1-\theta(s)$. So, we can write

$$
\begin{aligned}
&\frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\{y_n = +1\} \ln\left(\frac{1}{h(\boldsymbol{x}_n)}\right) + \mathbb{1}\{y_n = -1\} \ln\left(\frac{1}{1-h(\boldsymbol{x}_n)}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\{y_n = +1\} \ln\left(\frac{1}{\theta(\boldsymbol{w}^T\boldsymbol{x}_n)}\right) + \mathbb{1}\{y_n = -1\} \ln\left(\frac{1}{1-\theta(\boldsymbol{w}^T\boldsymbol{x}_n)}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\{y_n = +1\} \ln\left(\frac{1}{\theta(\boldsymbol{w}^T\boldsymbol{x}_n)}\right) + \mathbb{1}\{y_n = -1\} \ln\left(\frac{1}{\theta(-\boldsymbol{w}^T\boldsymbol{x}_n)}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \ln\left(\frac{1}{\theta(y_n\boldsymbol{w}^T\boldsymbol{x}_n)}\right) \quad \text{(i.e. } P(y_n|\boldsymbol{x}_n) = \theta(y_n\boldsymbol{w}^T\boldsymbol{x}_n)) \\
&= \underbrace{\frac{1}{N}\sum_{n=1}^{N} \ln\left(1+e^{-y_n\boldsymbol{w}^T\boldsymbol{x}_n}\right)}_{E_{\text{in}}(\boldsymbol{w})}
\end{aligned}
$$

Cross-entropy

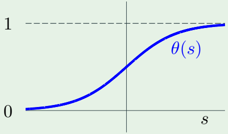
For two probability distributions with binary outcomes $\{p, 1-p\}$ and $\{q, 1-q\}$, the cross-entropy (from information theory) is

$$p \; log \; \frac{1}{q} + (1-p) \; log \; \frac{1}{1-q}.$$

The in-sample error above corresponds to a cross-entropy error measure on the data point $(\boldsymbol{x}_n, y_n)$, with $p = \mathbb{1}\{y_n = +1\}$ and $q = h(\boldsymbol{x}_n)$.

Formula for likelihood

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$



Substitute $h(\mathbf{x}) = \theta(\mathbf{w}^\intercal \mathbf{x})$, noting $\theta(-s) = 1 - \theta(s)$

$$P(y \mid \mathbf{x}) = \theta(y\ \mathbf{w}^\intercal \mathbf{x})$$

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

$$\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) = \prod_{n=1}^{N} \theta(y_n \mathbf{w}^\intercal \mathbf{x}_n)$$

Maximizing the likelihood

Minimize $$-\frac{1}{N}\ln\left(\prod_{n=1}^{N}\theta(y_n\ \mathbf{w}^{\mathsf{T}}\ \mathbf{x}_n)\right)$$

$$=\frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{\theta(y_n\ \mathbf{w}^{\mathsf{T}}\ \mathbf{x}_n)}\right)$$

$$\left[\theta(s)=\frac{1}{1\ +e^{-s}}\right]$$

$$E_{\text{in}}(\mathbf{w})\ =\frac{1}{N}\sum_{n=1}^{N}\underbrace{\ln\left(1+e^{-y_n\mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right)}_{\mathrm{e}\left(h(\mathbf{x}_n),y_n\right)}$$

"cross-entropy" error

How to minimize E_{in}

For logistic regression,

$$E_{\text{in}}(\mathbf{w}) \;=\; \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right) \qquad \longleftarrow \text{ iterative solution}$$

Compare to linear regression:

$$E_{\text{in}}(\mathbf{w}) \;=\; \frac{1}{N}\sum_{n=1}^{N} (\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n)^2 \qquad \longleftarrow \text{ closed-form solution}$$

Summary of Linear Models