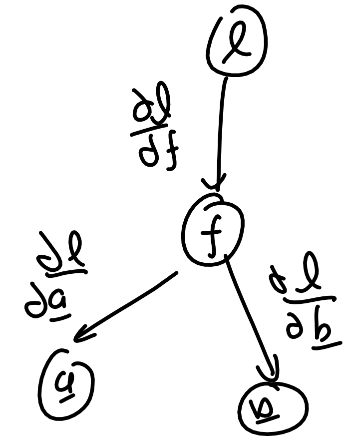
$$f(\underline{a}, \underline{b}) = \frac{1}{1+\exp(-\underline{a}^T \underline{b})} \qquad \underline{a} \in \mathbb{R}^n \quad \underline{b} \in \mathbb{R}^n \qquad f \in \mathbb{R}$$

Vector Jacobian Product.



$$\frac{\partial l}{\partial \underline{a}} =$$

$$\frac{\partial l}{\partial \underline{b}} =$$

$$\frac{d}{dx}\frac{1}{x} = ? \quad \Rightarrow \frac{d}{dx} x^{-1} = -1(x)^{-1-1} = -x^{-2} = -\frac{1}{x^2}$$

$$\frac{d}{dx}\exp(x) = ? = \exp(x)$$

$$\frac{\partial}{\partial \underline{a}} \underline{a}^T \underline{b} = \underline{b}^T \quad \text{and} \quad \frac{\partial}{\partial \underline{b}} \underline{a}^T \underline{b} = \underline{a}^T$$

$$\frac{\partial l}{\partial \underline{a}} = \frac{\partial l}{\partial f}\left(\frac{-1}{(1+\exp(-\underline{a}^T \underline{b}))^2}\right)\left(\exp(-\underline{a}^T \underline{b})\right)\frac{\partial}{\partial \underline{a}}(-\underline{a}^T \underline{b})$$

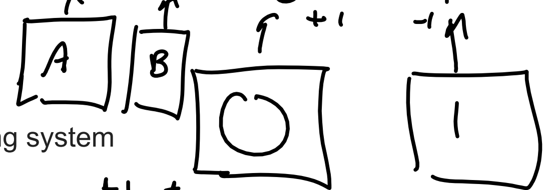
$$= \frac{\partial l}{\partial f} \quad \frac{\underline{b}^T \exp(-\underline{a}^T \underline{b})}{(1+\exp(-\underline{a}^T \underline{b}))^2}$$

$$\frac{\partial l}{\partial \underline{b}}$$

ML → Supervised → Classification = Output labels are discrete
Regression = Output labels are continuous
Unsupervised

Data, Models and Learning

"A", "B", "0" (+1), "1" (-1)



Ref: MML Book (Chapter 8) https://mml-book.github.io/

There are three major components of a machine learning system

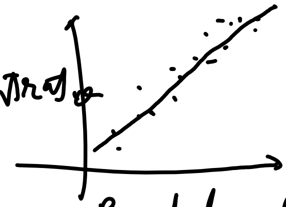
1. Data

   Input data → output labels

   SUPERVISED Learning

   A. Example: Handwritten digit images and corresponding labels
   B. Example: Road density and Salt concentration
   C. Example: X-Y coordinate of point cloud and corresponding Z coordinate

Salt concentration



Road density

Predicted label, Input

$\hat{y} = mx + c$

2. Models

   Input data → function → Predicted label

   Model

   A. Example: Linear model: Equation of line or plane
   B. Example: Multi Layer perceptron: Two Linear models sandwiching a non-linear activation function.

Inputs

$\hat{l} = \mathrm{sign}(y - (mx + c))$

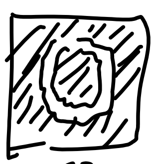
Predicted label

3. Learning

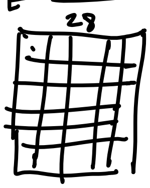
   A. Example: Least square solution. → $\frac{\partial}{\partial x} l(x) \Big|_{x^*} = 0$
   B. Example: Gradient descent. → $x_{t+1} = x_t - \alpha \frac{\partial}{\partial x} l(x)$

Data as Vectors

$x_i =$



$x_i =$ 28 × 28



$x_i =$ 28×28

Raw input

Let us consider the problem of identifying the digit from handwritten images based on data. This is called a supervised learning problem, where have a label y_i (the digit) associated with each example $\mathbf{x}_i$ (the handwritten image). The label y_i has various other names, including target, response variable and annotation. A dataset is written as a set of example- label pairs $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\}$. The features $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ are often concatenated and written as a big matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and the labels $\mathbf{y} \in \mathbb{R}^n$

$\in \mathbb{R}^d$

Input, output label, Training data

$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n\times d}$

Models as functions

Once we have data in an appropriate vector representation, we can get to the business of constructing a predictive function (known as a predictor).

$x_i = \begin{bmatrix} \text{\# of white pixels} \\ \text{spread of white pixels on y-axis} \end{bmatrix} \in \mathbb{R}^2$

feature vector

There are two major approaches:

Input dimension

1. a predictor as a function, $f : \mathbb{R}^d \mapsto \mathbb{R}$

   $f(x) = y \in \mathbb{R}$, $x \in \mathbb{R}^d$

   A. Example: Linear Model: $f(\mathbf{x}) = \cancel{\backslash\mathrm{bfw}}^\top \mathbf{x} + w_0$

   w

   $= w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_0$

   $f(x) = mx + c$

activation function

$f(x) = \underline{W}_2^T \sigma(\underline{W}_1 \underline{x} + \underline{w}_0) + w_{20}$ $\quad W_2 \in \mathbb{R}^m \quad W_1 \in \mathbb{R}^{m \times d} \quad w_0 \in \mathbb{R}^m$

$\underline{x} \in \mathbb{R}^d$

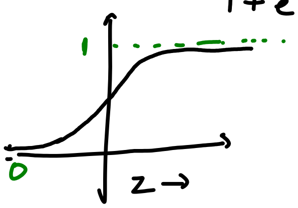
$\mathrm{ReLU}(z) = \begin{cases} z & \text{if } z>0 \\ 0 & \text{if } z<0 \end{cases}$

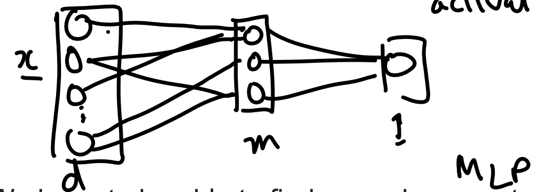
B. Example: MLP: $f(\mathbf{x}) =$ $\cancel{\backslash\mathrm{bfw}_2^\top\sigma(\backslash\mathrm{bfW}_1\mathbf{x} + \backslash\mathrm{bfw}_0)}$, where $\sigma : \mathbb{R}^m \mapsto \mathbb{R}^m$ is some non-linear activation function like ReLU, $\mathrm{sigmoid}$ or $\tanh$.

σ is any activation functo

2. a predictor as a probabilistic model. Later

$\mathrm{sigmoid}(z) = \dfrac{1}{1+\exp(-\alpha z)}$





$f(x)$

Hypothesis class of functions

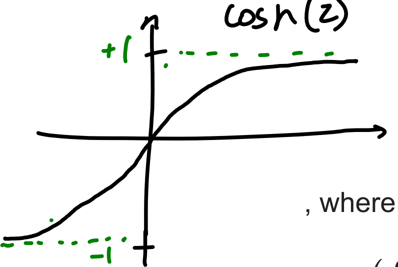
A predictor $f : \mathbb{R}^d \mapsto \mathbb{R}$, parametrized by θ . We hope to be able to find a good parameter θ^* such that we fit the data well, that is, $f(\mathbf{x}_i, \theta^*) \approxeq y_i$ for all $i = 1, \ldots, n$

We use the notation $\hat{y}_i = f(\mathbf{x}_i, \theta^*)$ to represent the output of the predictor.

$f(\underline{x}) = \underline{W}^T \underline{x} + w_0$

$\tanh(z)$ = Hyperbolic tangent

$\tanh(z) = \dfrac{\sinh(z)}{\cosh(z)}$



$f(x) = mx + c$

$f_1(x) = 2x + 1$

$f_2(x) = 3x + 0$

$f(x; m, c)$ ← Parameters

Loss Function for Training

$$r_{\text{emp}}(f, \mathbf{X}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

, where $\hat{y}_i = f(\mathbf{x}_i, \theta^*)$.

$r_{\text{emp}}(f, \mathbf{X}, \mathbf{y})$ is called the empirical risk.

Homework 5

MLP

=

activation function

$$f(\underline{x}) = \underline{w}_2^T a(\underline{W}_1 \underline{x} + \underline{w}_0) + w_{20}$$

Parameters in MLP $= \{\underline{w}_2, w_{20}, \underline{W}_1, \underline{w}_0\}$

Linear model

$$f(x) = \underline{w}^T x + w_0$$

Parameter $= \{\underline{w}, w_0\}$

3-Layer Perceptron or 3-Layer N.

$$f(\underline{x}) = \underline{w}_3^T a_2\left(W_2\, a_1(W_1 \underline{x} + \underline{w}_1) + \underline{w}_2\right) + w_{30}$$

Parameters $= \{\underline{w}_3, W_2, W_1, \underline{w}_1, \underline{w}_2, w_{30}\}$

Learning:

while (not coverged):

$$\text{Parameters}_{t+1} = \text{Parameters}_t - \alpha \frac{\partial l}{\partial \text{Parameter}}$$

$$f(\underline{x}; \underline{\theta}) \qquad f_{\underline{\theta}}(x)$$

input

Parameters (weights and biases)

① Data $\{(\underline{x}_i, y_i) \dots (\underline{x}_n, y_n)\}$

② Model -

③ Learning

↳ Gradient Descent

↳ Loss function

Model: $f(\underline{x}; \underline{\theta})$

common notation for predicted label

$$\hat{y}_i = f(\underline{x}_i; \theta)$$

$$\hat{y}_i = f(x_i; m, c)$$

$$\hat{y}_i = m x_i + c$$

$$\hat{y}_i \approx y_{\underline{i}}$$

Least square loss function

Loss function

$$\ell(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 = (y_i - \underbrace{(m x_i + c)}_{\hat{y}_i})^2$$

Thresholded L1 loss

$$\hat{y}_i = f(\underline{x}_i; \underline{w}) = \text{sign}(\underline{w}^T \underline{x}_i + w_0)$$

$$\hat{y}_i \in \{-1, +1\}$$

$$\ell(y_i, \hat{y}_i) = \begin{cases} 0 & \text{if } y_i = \hat{y}_i \\ \underbrace{|\underline{w}^T x_i + w_0|}_{\text{abs}} & \text{if } y_i \neq \hat{y}_i \end{cases}$$

The learning problem in general is formulated as an optimization problem

$$\theta^* = \arg\min_{\theta} r_{emp}\left(f, \{(\underline{x}_1, y_1), \dots (\underline{x}_n, y_n)\}\right)$$

Data $(\underline{x}_i, y_i)$ being identically independently distributed. (i.i.d.)

$$r_{emp}(f \dots) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, \hat{y}_i)$$