ETH zürich

Background Review

Reliable and Trustworthy Artificial Intelligence

Assumed Background

Logic

- $\land$, $\lor$, $\rightarrow$, $\implies$
- $\forall$, $\exists$
- Predicates

Linear Algebra

- Vectors
- Matrices

ETH zürich

Probability Theory

Probability Distribution

Definition

A random variable is a variable whose values depend on outcomes of a random experiment.

Definition

A probability distribution is a function that gives the probabilities of occurrence of different possible outcomes for an experiment.

Example (Rolling a Dice)

If random variable D is the result of a rolling a fair dice, then its probability distribution is

$$\Pr[D = k] = \begin{cases} \frac{1}{6} & \text{if } 1 \leq k \leq 6, k \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}$$

Indicator variables

Definition

An indicator variable is a random variable with possible outcomes in $\{0, 1\}$.

Example (Indicator Variable)

If I indicates whether $D = 6$, then

$$\Pr[I = 1] = \frac{1}{6}$$
$$\Pr[I = 0] = \frac{5}{6}$$

Continuous Distributions

Definition

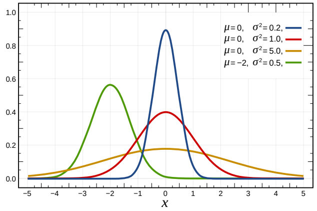
A random variable X has a continuous distribution if there is a *probability density function* $f : \mathbb{R} \to [0, \infty]$ such that its *cumulative distribution function* is

$$\Pr[X \leq b] = \int_{-\infty}^{b} f(x)dx$$

Example (Normal Distribution)

The probability density function for random variable $X \sim \mathcal{N}(\mu, \sigma)$ is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



Expectation

Definition

The expected value of a random variable X, denoted $\mathbb{E}[X]$, is $\sum_x x \cdot \Pr[X = x]$ if X is discrete and $\int_c x \cdot f(x)dx$ if x is continuous.

Example (Fair dice)

The expected value of the result D of a fair dice roll is

$$\mathbb{E}[D] = \sum_{d=1}^{6} d \cdot \tfrac{1}{6} = 3.5$$

Linear Algebra

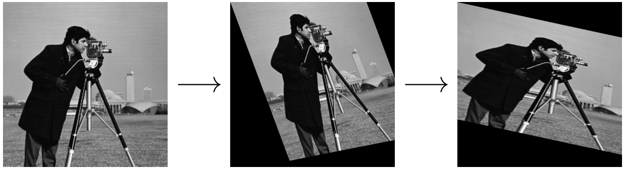
Linear transformations

Definition

A linear transformation is a map $f : \mathbb{R}^m \to \mathbb{R}^n$ that preserves linear combinations:

$$f\left(\sum_{i=1}^{k} \lambda_i \vec{u}_i\right) = \sum_{i=1}^{k} \lambda_i f\left(\vec{u}_i\right).$$

Example (in $\mathbb{R}^2$)



Linear transformations as matrices

Proposition

Every linear transformation $f : \mathbb{R}^m \to \mathbb{R}^n$ can be expressed as a matrix $M \in \mathbb{R}^{n \times m}$:

$$\forall \vec{u} \in \mathbb{R}^m \colon f(\vec{u}) = M\vec{u}.$$

Example (in $\mathbb{R}^2$)

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Linear transformations as matrices

Proposition

Every linear transformation $f : \mathbb{R}^m \to \mathbb{R}^n$ can be expressed as a matrix $M \in \mathbb{R}^{n \times m}$:

$$\forall \vec{u} \in \mathbb{R}^m\colon f(\vec{u}) = M\vec{u}.$$

Example (in $\mathbb{R}^2$)

$$\underset{\text{rotation}}{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}} \qquad \underset{\text{dilation}}{\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}} \qquad \underset{\text{projection}}{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}$$

Affine transformations

Definition

An affine transformation is a map $f : \mathbb{R}^m \to \mathbb{R}^n$ that preserves affine combinations:

$$\sum_{i=1}^{k} \lambda_i = 1 \implies f\left(\sum_{i=1}^{k} \lambda_i \vec{u}_i\right) = \sum_{i=1}^{k} \lambda_i f\left(\vec{u}_i\right).$$

Proposition

Every affine $f : \mathbb{R}^m \to \mathbb{R}^n$ decomposes into a translation after a linear transformation:

$$f(\vec{u}) = \vec{t} + g(\vec{u}).$$

Norms and distances

Euclidean (l_2)

$$\|\vec{u}\|_2 = \sqrt[2]{\vec{u} \cdot \vec{u}} \qquad d_2(\vec{u}, \vec{v}) = \|\vec{v} - \vec{u}\|_2.$$

l_p $(1 \leq p \leq \infty)$

$$\|\vec{u}\|_p = \sqrt[p]{\sum |u_i|^p} \qquad d_p(\vec{u}, \vec{v}) = \|\vec{v} - \vec{u}\|_p.$$

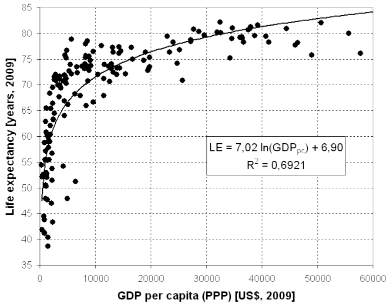
l_∞

$$\|\vec{u}\|_\infty = \lim_{p \to \infty} \|\vec{u}\|_p = \sup |u_i|.$$

Prediction

Prediction problems

Predict lifespan given GDP per capita.



Predict quantity $\Longrightarrow$ regression.

Predict digit given a pixelated scan.

MNIST `http://yann.lecun.com/exdb/mnist/`

Predict label $\Longrightarrow$ classification.

Mathematical formulation

The simple version

Predict target variables $\vec{Y}$ from input variables $\vec{X}$, i.e., select a model f such that

$$f(\vec{X}) \approx \vec{Y}.$$

Consistency

1. In practice, we need to sample $(\vec{X}, \vec{Y})$ in order to select f.
2. This means that $f = f_n$ depends on the sample size n.
3. The selection method is consistent if $f_n(\vec{X}) \to \vec{Y}$ as $n \to \infty$.

The not so simple version

What is $f(\vec{X}) \approx \vec{Y}$?

1. No universal answer.
2. A choice depending on the task.
3. Usually defined by a loss function:

$$\mathcal{L}_f(\vec{x}, \vec{y}) \in \mathbb{R}$$

Where does f come from?

1. Comes from a model space F.
2. That space is, again, a choice.
3. Usually parameterized by a vector:

$$F = \{f_\theta \mid \theta \in \mathbb{R}^d\}.$$

Model fitting

Ideally, we want a model $f_{\theta^*} \in F$ minimizing the risk $R(\theta) = \mathbb{E}_{\vec{X},\vec{Y}}\left[\mathcal{L}_{f_\theta}\left(\vec{X}, \vec{Y}\right)\right]$.

Empirical risk minimization

Risk estimation

In practice, we can only estimate the risk $R(\theta)$ from a sample of $(\vec{X}, \vec{Y})$.

Definition

The empirical risk for a sample $s_n = \{(\vec{x}_1, \vec{y}_1), \dots, (\vec{x}_n, \vec{y}_n)\}$ is the average loss over s_n:

$$\widehat{R}_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{f_\theta}(\vec{x}_i, \vec{y}_i).$$

Selection rule

Given a sample s_n of $(\vec{X}, \vec{Y})$, select a model f_θ minimizing the empirical risk $\widehat{R}_n$.

Risk minimization flow

1. Select loss
2. Select model space
3. Minimize empirical risk

Loss: Prediction form

Form of the loss

For prediction it is standard to derive the loss $\mathcal{L}_{f_\theta}$ from a distance d:

$$\mathcal{L}_{f_\theta}(\vec{x}, \vec{y}) = d(\vec{y}, f_\theta(\vec{x})).$$

Shape of d

The shape of the distance d determines how difficult is to find a minimizer of $\widehat{R}_n$:

- The simple 0–1 *distance* $d(\vec{y}, \vec{y}') = (0 \text{ if } \vec{y} = \vec{y}' \text{ else } 1)$ is too difficult in practice.

Loss: One-Hot Encodings

Predict k classes $\{1, \dots, k\}$

1. The 0–1 distance most natural but difficult.
2. The Euclidean distance d_2 easier but biased: $y = 1 \implies d_2(y, 2) < d_2(y, 8)$.

Encode $\{1, \dots, k\}$ into $\mathbb{R}^k$

$$\begin{array}{ccc} 1 & \mapsto & (1, 0, \dots, 0) \\ 2 & \mapsto & (0, 1, \dots, 0) \\ & \dots & \\ k & \mapsto & (0, 0, \dots, 1) \end{array}$$

Predict probability vectors in $\mathbb{R}^k$

$$\mathcal{L}_{f_\theta}(\vec{x}, y) = d(\text{encode}(y), f_\theta(\vec{x}))$$

Use smooth distance d such as cross-entropy.

Loss: Classification details

1. Pass prediction through softmax, $f_\theta = \text{softmax} \circ g_\theta$, to create probability vectors.
2. The cross-entropy $H(\vec{p}, \vec{q})$ measures a distance from a true $\vec{p}$ to an estimate $\vec{q}$.

Softmax : $\mathbb{R}^k \to \mathbb{R}^k$

Maps any vector to a probability vector.

$$\text{softmax}(\vec{x}) = \frac{1}{e^{x_1} + \cdots + e^{x_k}}(e^{x_1}, \ldots, e^{x_k}).$$

Cross-entropy : $\mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$

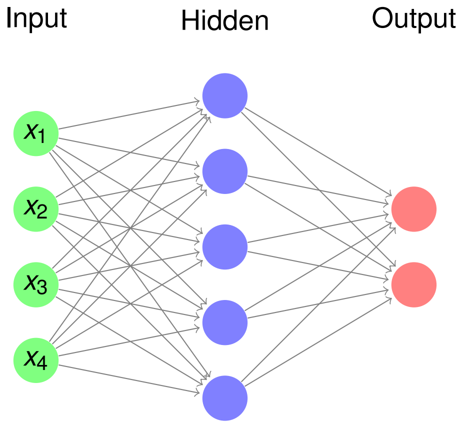
Inputs must be probability vectors.

$$H(\vec{p}, \vec{q}) = -\sum_{i=1}^{k} p_i \log(q_i).$$

Full loss

$$\mathcal{L}_{f_\theta}(\vec{x}, \vec{y}) = H(\text{encode}(\vec{y}), \text{softmax} \circ g_\theta(\vec{x})) \qquad f_\theta = \text{softmax} \circ g_\theta.$$

Model: Feedforward networks



Neuron: $\eta_{\vec{w},b}(\vec{x}) = \sigma(b + \vec{w} \cdot \vec{x})$

- $\vec{w}$: weights, b: bias
- σ: non-linearity

Layer: $\ell_{W,\vec{b}}(\vec{x}) = \sigma(\vec{b} + W \cdot \vec{x})$

- $\vec{b}$: bias vector, W: weights matrix

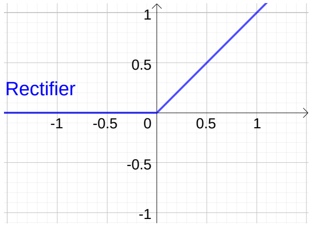
Network: $g_\theta(\vec{x}) = \ell_{\theta_m} \circ \cdots \circ \ell_{\theta_1}(\vec{x})$

- $\theta = [\theta_1, \dots, \theta_m]$: model parameters

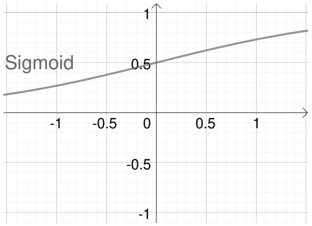
Model: Architecture

Choices

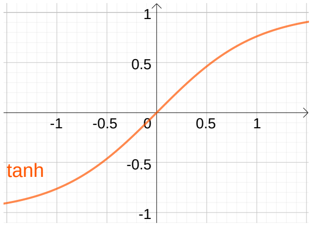
1. Connections:
   fully connected, convolutional, random
2. Non-linearities:
   rectifier, sigmoid, tanh, ...



steep, simple

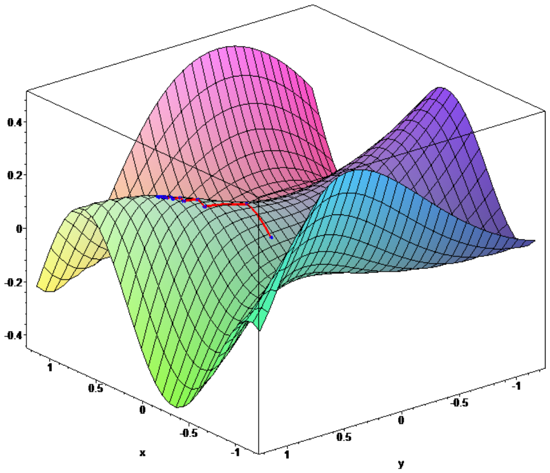


flat for small/large inputs



flat for small/large inputs

Minimization: Gradient descent



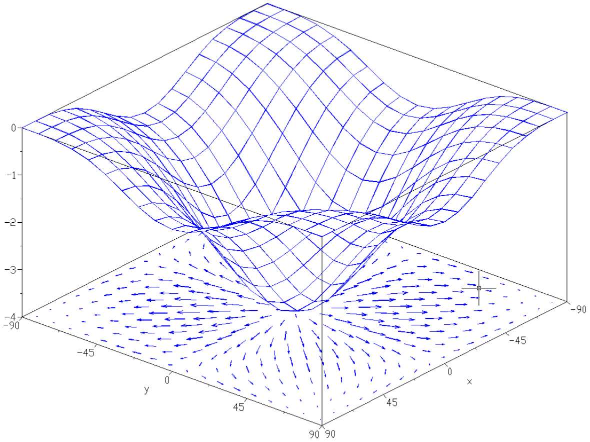
Objective

1. Input: a sample of $(\vec{X}, \vec{Y})$
2. Goal: minimize $\widehat{R} = \widehat{R}_n$

Gradient descent

1. Select: θ_0
2. Iterate: $\theta_{t+1} = \theta_t - \alpha_t \nabla \widehat{R}(\theta_t)$.

Minimization: Gradients



Gradient operator ∇

1. Input: $S : \mathbb{R}^d \to \mathbb{R}$
2. Output: $\nabla S : \mathbb{R}^d \to \mathbb{R}^d$

$$\nabla S(\vec{u}) = \left(\frac{\partial S}{\partial u_1}(\vec{u}), \dots, \frac{\partial S}{\partial u_d}(\vec{u})\right)$$

Evaluating $\nabla \widehat{R}(\vec{x})$

1. Backpropagation algorithm.
2. Fully automated in software.

Minimization: Stochastic gradient descent

Issues with gradient descent

- Gets stuck easily.
- Slow for large samples.

Variants and improvements

Momentum, Nesterov, Adagrad, AdaDetla, RMSProp, Adam, Natural gradient, ...

Stochastic gradient descent

- Subsample the sample into batches.
- Evaluate $\nabla \widehat{R}$ on one batch each step.