Introduction to Neural Networks and Deep Learning

Loss Functions

Andres Mendez-Vazquez

October 28, 2020

Outline

Outline

Why Loss Functions?

Long ago the Perceptron showed many shortcomings

- The XOR problem could not be solved by the Perceptron
- The loss function was simple

$$y(i) = \sum_{i=1}^{m} w_k(i) x_k(i)$$

We want a better function for classification

- The classification case is harder because it is not obvious what loss function to use!!!

Why Loss Functions?

Long ago the Perceptron showed many shortcomings

- The XOR problem could not be solved by the Perceptron
- The loss function was simple

$$y(i) = \sum_{i=1}^{m} w_k(i) x_k(i)$$

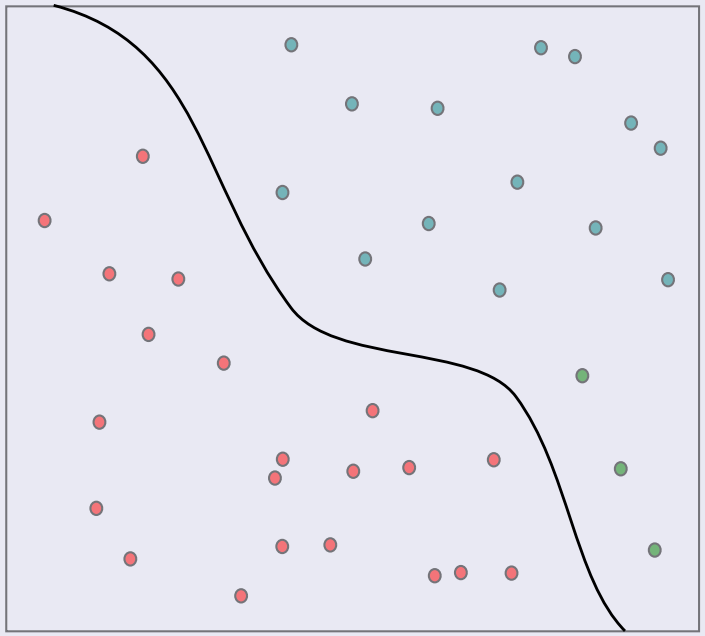
We want a better function for classification

- The classification case is harder because it is not obvious what loss function to use!!!

As we have found

Classification task started tweaking the Regression Method, $\sum_{i=1}^{N} L^2(x_i, y_i)$

- Which has serious disadvantages given that you are approximating a function where points do not exist...



Serious Disadvantages

You need to have dense classes with similar number of elements

- Basically, you are required to collect data under those two characteristics.

Thus, we have a need to find better loss functions

- That reflect better the task of classification

Way more explainable and adaptive

- Given the structures at the Deep Learners

Serious Disadvantages

You need to have dense classes with similar number of elements

- Basically, you are required to collect data under those two characteristics.

Thus, we have a need to find better loss functions

- That reflect better the task of classification

Way more explainable and adaptive

- Given the structures at the Deep Learners

Serious Disadvantages

You need to have dense classes with similar number of elements

- Basically, you are required to collect data under those two characteristics.

Thus, we have a need to find better loss functions

- That reflect better the task of classification

Way more explainable and adaptive

- Given the structures at the Deep Learners

Outline

Expected Risk for a function [1, 2]

We have

$$E[f] = \int_{\mathcal{X}\times\mathcal{Y}} L(f(\boldsymbol{x}), y)\, p(\boldsymbol{x}, y)\, d\boldsymbol{x}dy$$

- Where L is a non-negative function named loss function.

Thus, the ideal estimator or target function

$$f_0 = \min_{f\in\mathcal{F}} E[f]$$

- Where $\mathcal{F}$ is the space of measurable functions for which $E[f]$ is well-defined.

Expected Risk for a function [1, 2]

We have

$$E[f] = \int_{\mathcal{X}\times\mathcal{Y}} L(f(\boldsymbol{x}), y)\, p(\boldsymbol{x}, y)\, d\boldsymbol{x} dy$$

- Where L is a non-negative function named loss function.

Thus, the ideal estimator or target function

$$f_0 = \min_{f\in\mathcal{F}} E[f]$$

- Where $\mathcal{F}$ is the space of measurable functions for which $E[f]$ is well-defined.

However

In practice f_0 cannot be found

- Since the probability distribution $p(\boldsymbol{x}, y)$ is unknown.

That is the reason we use the empirical risk

$$f_D = \min_{f \in \mathcal{F}} E_{emp}[f] = \min_{f \in \mathcal{F}} \frac{1}{\ell} \sum_{i=1}^{\ell} L(f(\boldsymbol{x}), y)$$

However

In practice f_0 cannot be found

- Since the probability distribution $p(\boldsymbol{x}, y)$ is unknown.

That is the reason we use the empirical risk

$$f_D = \min_{f \in \mathcal{F}} E_{emp}[f] = \min_{f \in \mathcal{F}} \frac{1}{\ell} \sum_{i=1}^{\ell} L(f(\boldsymbol{x}), y)$$

Thus

This allows to restict the space to a limited hypothesis space $\mathcal{H}$

- This allows for a possible computation of the solution.

Therefore, we have

- A central problem of statistical learning theory is to find conditions under which f_D mimics the behavior of f_0.

Thus

This allows to restict the space to a limited hypothesis space $\mathcal{H}$

- This allows for a possible computation of the solution.

Therefore, we have

- A central problem of statistical learning theory is to find conditions under which f_D mimics the behavior of f_0.

Small Problem

The approximation of f_0

- From a finite set of data is an ill-posed problem [3].

However, we can use regularization on Hilbert Spaces

- To solve the ill-posed problem of finding f_D.

Small Problem

The approximation of f_0

- From a finite set of data is an ill-posed problem [3].

However, we can use regularization on Hilbert Spaces

- To solve the ill-posed problem of finding f_D.

Outline

Hilbert Space

Definition

- A Hilbert Space $\mathcal{H}$ is a complete inner product space.

Inner Product in Hilbert Spaces

The inner product satistasfies the following properties for $f, g \in \mathcal{H}$ and $\alpha_1, \alpha_2 \in \mathbb{R}$:

1. (Symmetry) $\langle f, g \rangle = \langle g, f \rangle$
2. (Linearity) $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle = \alpha_1 \langle f_1, g \rangle + \alpha_2 \langle f_2, g \rangle$
3. (Positive definiteness) $\langle f, f \rangle \geq 0$ with equality only if $f = 0$

Cauchy Sequences

Definition

- A metric space M is called complete (or a Cauchy space) if every Cauchy sequence of points in M has a limit that is also in M.

A Cauchy Sequence

- A metric space $(\mathcal{X}, d)$, a sequence $x_1, x_2, \ldots$ is Cauchy for every positive real number $\epsilon > 0$ there is a positive integer N such that $n, m > N$:

$$d(x_m, x_n) < \epsilon$$

Cauchy Sequences

Definition

- A metric space M is called complete (or a Cauchy space) if every Cauchy sequence of points in M has a limit that is also in M.

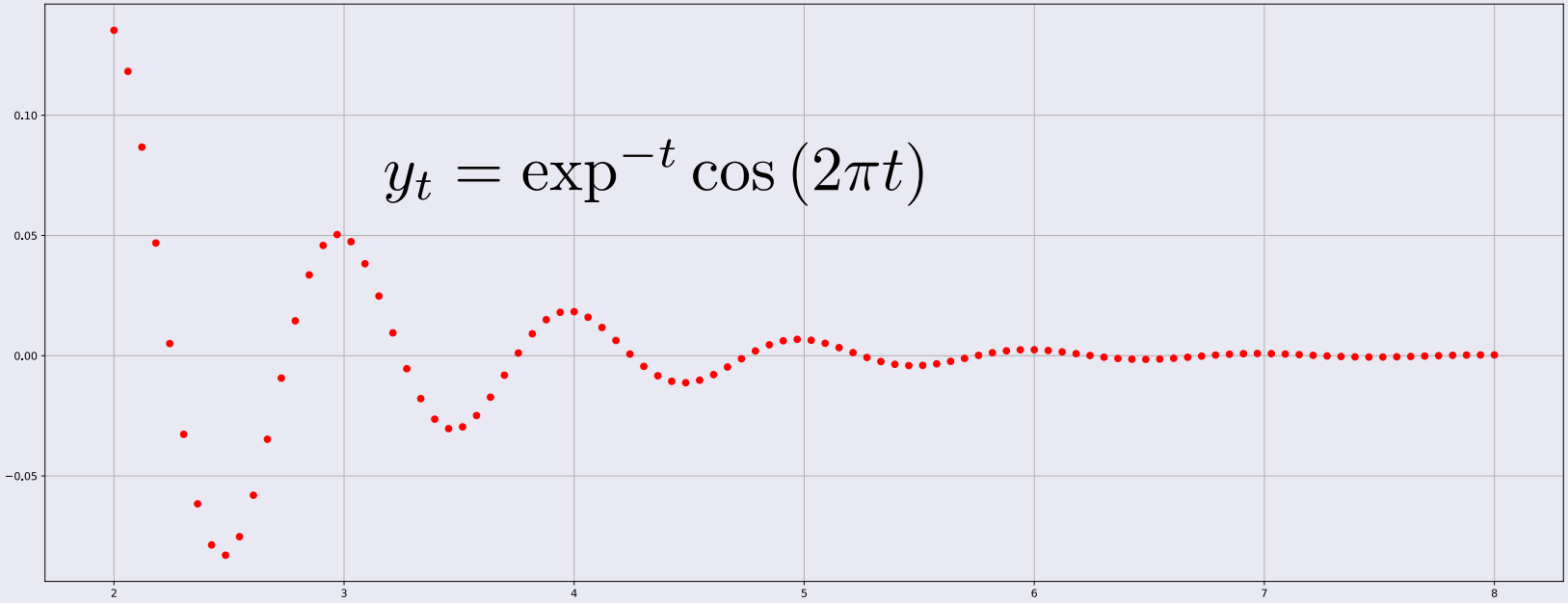
A Cauchy Sequence

- A metric space $(\mathcal{X}, d)$, a sequence $x_1, x_2, ...$ is Cauchy for every positive real number $\epsilon > 0$ there is a positive integer N such that $n, m > N$:

$$d\left(x_m, x_n\right) < \epsilon$$

Example of Cauchy Sequence

We have the following cauchy sequence in y_t for interval $[2, +\infty)$



Example of Hilbert Space

We have

- Let $C^k[a,b]$ the space of functions with k derivatives on $[a,b]$. We define an inner product as

$$\langle f, g \rangle = \sum_{j=0}^{k} \int_a^b \overline{f^{(j)}(t)} g^{(j)}(t)\, dt$$

Allowing to define a norm $\|\cdot\|_{\mathcal{H}}$

We have the following

$$\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle}$$

This allows to define what is called a feature map

- Given a Hilbert space $\mathcal{H}$, a feature map $\varphi : X \to \mathcal{H}$ takes inputs $x \in X$ to infinite feature vectors $\varphi(x) \in \mathcal{H}$.

Allowing to define a norm $\|\cdot\|_{\mathcal{H}}$

We have the following

$$\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle}$$

This allows to define what is called a feature map

- Given a Hilbert space $\mathcal{H}$, a feature map $\varphi : X \to \mathcal{H}$ takes inputs $x \in X$ to infinite feature vectors $\varphi(x) \in \mathcal{H}$.

Outline

Kernels

Definition

- Let $\mathcal{X}$ be a nonempty set, sometimes referred to as the index set. A symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive-definite kernel on $\mathcal{X}$ if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \geq 0$$

  - holds for any finite set of points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathcal{X}$ and $c_1, ..., c_n \in \mathbb{R}$ (i.e. positive semidefinite).

Actually, Symmetry

This can be seen as a matrix product with K

$$\begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix} \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & K(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \boldsymbol{c}^T K \boldsymbol{c} \geq 0$$

- Which is the property of positive semidefinite.

Actually, it is easy to see that this comes from the study of convex functions

- What?!!!

Actually, Symmetry

This can be seen as a matrix product with K

$$\begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix} \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & K(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \boldsymbol{c}^T K \boldsymbol{c} \geq 0$$

- Which is the property of positive semidefinite.

Actually, it is easy to see that this comes from the study of convex functions

- What?!!!

Convex Functions

We have that when deriving convex functions as

$$f(x,y) = x^2 + y^2$$

We have that the Hessian function

$$Hf(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Thus, we have

- A positive definite matrix...

Convex Functions

We have that when deriving convex functions as

$$f(x,y) = x^2 + y^2$$

We have that the Hessian function

$$Hf(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Thus, we have

- A positive definite matrix...

Convex Functions

We have that when deriving convex functions as

$$f(x,y) = x^2 + y^2$$

We have that the Hessian function

$$Hf(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Thus, we have

- A positive definite matrix...

In addition to the study of linear maps

Yes, in linear algebra for each matrix

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

There is a linear function associated to it

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

$$f(x) = Ax$$

In addition to the study of linear maps

Yes, in linear algebra for each matrix

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

There is a linear function associated to it

$$f : \mathbb{R}^n \to \mathbb{R}^m$$
$$f(x) = Ax$$

Examples of Kernels

Linear kernels

$$k(x, x') = \langle x, x' \rangle$$

Polynomial kernels

$$k(x, x') = (1 + \langle x, x' \rangle)^p$$

And clearly the Gaussian Ones

$$k(x, x') = \exp\left\{-\frac{\langle x - x', x - x' \rangle}{2\sigma^2}\right\}$$

Examples of Kernels

Linear kernels

$$k\left(x, x'\right) = \left\langle x, x' \right\rangle$$

Polynomial kernels

$$k\left(x, x'\right) = \left(1 + \left\langle x, x' \right\rangle\right)^p$$

And clearly the Gaussian Ones

$$k\left(x, x'\right) = \exp\left\{-\frac{\left\langle x - x', x - x' \right\rangle}{2\sigma^2}\right\}$$

Examples of Kernels

Linear kernels

$$k\left(x, x'\right) = \left\langle x, x' \right\rangle$$

Polynomial kernels

$$k\left(x, x'\right) = \left(1 + \left\langle x, x' \right\rangle\right)^p$$

And clearly the Gaussian Ones

$$k\left(x, x'\right) = \exp\left\{-\frac{\left\langle x - x', x - x' \right\rangle}{2\sigma^2}\right\}$$

Outline

General Case

Base case

- For any function $f : \mathcal{X} \to \mathbb{R}$, $K(x, x') = f(x) f(x')$ is positive semidefinite.

Proof

- Imagine the kernel matrix written as

$$K = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} \begin{pmatrix} f(x_1), & \cdots & , f(x_n) \end{pmatrix} = \begin{pmatrix} f(x_1) f(x_1) & \cdots & f(x_1) f(x_n) \\ \vdots & \ddots & \vdots \\ f(x_n) f(x_1) & \cdots & f(x_n) f(x_n) \end{pmatrix}$$

A symmetric matrix

- Which is positive semidefinite...

General Case

Base case

- For any function $f : \mathcal{X} \to \mathbb{R}$, $K(x, x') = f(x) f(x')$ is positive semidefinite.

Proof

- Imagine the kernel matrix written as

$$K = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} \begin{pmatrix} f(x_1), & \cdots & , f(x_n) \end{pmatrix} = \begin{pmatrix} f(x_1) f(x_1) & \cdots & f(x_1) f(x_n) \\ \vdots & \ddots & \vdots \\ f(x_n) f(x_1) & \cdots & f(x_n) f(x_n) \end{pmatrix}$$

A symmetric matrix

- Which is positive semidefinite...

General Case

Base case

- For any function $f : \mathcal{X} \to \mathbb{R}$, $K(x, x') = f(x) f(x')$ is positive semidefinite.

Proof

- Imagine the kernel matrix written as

$$K = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} \begin{pmatrix} f(x_1), & \cdots & , f(x_n) \end{pmatrix} = \begin{pmatrix} f(x_1) f(x_1) & \cdots & f(x_1) f(x_n) \\ \vdots & \ddots & \vdots \\ f(x_n) f(x_1) & \cdots & f(x_n) f(x_n) \end{pmatrix}$$

A symmetric matrix

- Which is positive semidefinite...

Recursive case

Given two kernels k_1, k_2 , we can create new kernels k

- By using sums and products

Sum case

- $k(x, x') = k_1(x, x') + k_2(x, x')$

Easy, we have

- Since positive semidefiniteness is closed under addition of matrices

$$K = K_1 + K_2 \succeq 0$$

Recursive case

Given two kernels k_1, k_2 , we can create new kernels k

- By using sums and products

Sum case

- $k(x, x') = k_1(x, x') + k_2(x, x')$

Easy, we have

- Since positive semidefiniteness is closed under addition of matrices

$$K = K_1 + K_2 \succeq 0$$

Recursive case

Given two kernels k_1, k_2 , we can create new kernels k

- By using sums and products

Sum case

- $k(x, x') = k_1(x, x') + k_2(x, x')$

Easy, we have

- Since positive semidefiniteness is closed under addition of matrices

$$K = K_1 + K_2 \succeq 0$$

Product Case

We have

$$k(x, x') = k_1(x, x') k_2(x, x')$$

Here, we have $K = K_1 \circ K_2$ pointwise product

- Since K_1, K_2 are positive semidefinite

We have their decomposition

- $K_1 = \sum_{i=1}^{n} \lambda_i u_i u_i^T$ and $K_1 = \sum_{j=1}^{n} \tau_i z_j z_j^T$

Product Case

We have

$$k(x, x') = k_1(x, x') k_2(x, x')$$

Here, we have $K = K_1 \circ K_2$ pointwise product

- Since K_1, K_2 are positive semidefinite

We have their decomposition

- $K_1 = \sum_{i=1}^{n} \lambda_i u_i u_i^T$ and $K_1 = \sum_{j=1}^{n} \tau_i z_j z_j^T$

Product Case

We have

$$k\left(x,x'\right)=k_1\left(x,x'\right)k_2\left(x,x'\right)$$

Here, we have $K=K_1\circ K_2$ pointwise product

- Since K_1,K_2 are positive semidefinite

We have their decomposition

- $K_1=\sum_{i=1}^n\lambda_iu_iu_i^T$ and $K_1=\sum_{j=1}^n\tau_iz_jz_j^T$

Therefore, we have

Taking the element-wise product yields the following eigendecomposition

$$K = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i t_j \left(u_i \circ z_j\right) \left(u_i \circ z_j\right)^T$$

Which is also positive semidefinite

- Using these three principles, we can show that the linear, polynomial, and Gaussian kernels are valid.

Therefore, we have

Taking the element-wise product yields the following eigendecomposition

$$K = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i t_j \left(u_i \circ z_j\right) \left(u_i \circ z_j\right)^T$$

Which is also positive semidefinite

- Using these three principles, we can show that the linear, polynomial, and Gaussian kernels are valid.

Outline

Feature Maps

Definition(Feature Maps)

- Given a Hilbert space $\mathcal{H}$, a feature map $\phi : \mathcal{X} \to \mathcal{H}$ takes inputs $x \in \mathcal{X}$ to infinite feature vectors $\phi(x) \in \mathcal{H}$.

Theorem(A Feature map defines a kernel)

- Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature mapping some input space $\mathcal{X}$ to a Hilbert space $\mathcal{H}$. Then, $k(x, x') = \langle \phi(x), \phi(x') \rangle$ is a kernel.

Proof (Using the finitness of the set of points)

- Let $x_1, ..., x_n$ be a set of points, and let K be the kernel matrix where $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$

Feature Maps

Definition(Feature Maps)

- Given a Hilbert space $\mathcal{H}$, a feature map $\phi : \mathcal{X} \to \mathcal{H}$ takes inputs $x \in \mathcal{X}$ to infinite feature vectors $\phi(x) \in \mathcal{H}$.

Theorem(A Feature map defines a kernel)

- Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature mapping some input space $\mathcal{X}$ to a Hilbert space $\mathcal{H}$. Then, $k(x, x') = \langle \phi(x), \phi(x') \rangle$ is a kernel.

Proof (Using the finitness of the set of points)

- Let $x_1, ..., x_n$ be a set of points, and let K be the kernel matrix where $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$

Feature Maps

Definition(Feature Maps)

- Given a Hilbert space $\mathcal{H}$, a feature map $\phi : \mathcal{X} \to \mathcal{H}$ takes inputs $x \in \mathcal{X}$ to infinite feature vectors $\phi(x) \in \mathcal{H}$.

Theorem(A Feature map defines a kernel)

- Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature mapping some input space $\mathcal{X}$ to a Hilbert space $\mathcal{H}$. Then, $k(x, x') = \langle \phi(x), \phi(x') \rangle$ is a kernel.

Proof (Using the finitness of the set of points)

- Let $x_1, ..., x_n$ be a set of points, and let K be the kernel matrix where $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$

Therefore

To show that K is positive semidefinite, take any $\alpha \in \mathbb{R}^n$

$$\alpha^T K \alpha = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left\langle \phi\left(x_i\right), \phi\left(x_j\right) \right\rangle$$

Therefore by the linearity of

$$\alpha^T K \alpha = \left\langle \sum_{i=1}^{n} \alpha_i \phi\left(x_i\right), \sum_{j=1}^{n} \alpha_j \phi\left(x_j\right) \right\rangle \geq 0$$

Therefore

To show that K is positive semidefinite, take any $\alpha \in \mathbb{R}^n$

$$\alpha^T K \alpha = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \left\langle \phi\left(x_i\right), \phi\left(x_j\right) \right\rangle$$

Therefore by the linearity of

$$\alpha^T K \alpha = \left\langle \sum_{i=1}^{n} \alpha_i \phi\left(x_i\right), \sum_{j=1}^{n} \alpha_j \phi\left(x_j\right) \right\rangle \geq 0$$

Thus, we also have

Theorem

- For every kernel k positive definite, there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$.

The Proof can be done by using the spectral decomposition

- Given k positive definite, then $K = UDU^*$ by spectral decomposition of K and D is positive definite
  - Then, we can define $\varphi(x) = D^{\frac{1}{2}} D^* 1_x$ ... you can figure out the rest

Thus, we also have

Theorem

- For every kernel k positive definite, there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$.

The Proof can be done by using the spectral decomposition

- Given k positive definite, then $K = UDU^*$ by spectral decomposition of K and D is positive definite
  - Then, we can define $\varphi(x) = D^{\frac{1}{2}} D^* 1_x$ ... you can figure out the rest

Outline

Bounding the Hypothesis Space for Learning

Here, we need to bound the $\mathcal{H}$

- For this, we can use Reproducing Kernel Hilbert Spaces.

They are Hilbert Spaces

- They are characterized by a symmetric positive definite function, named Mercer kernel:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

We can define a map from $\mathcal{X}$ to the space of functions $f : \mathcal{X} \to \mathbb{R}$ [4, 5]

$$\begin{aligned} \phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(\cdot, x) \end{aligned}$$

Bounding the Hypothesis Space for Learning

Here, we need to bound the $\mathcal{H}$

- For this, we can use Reproducing Kernel Hilbert Spaces.

They are Hilbert Spaces

- They are characterized by a symmetric positive definite function, named Mercer kernel:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

We can define a map from $\mathcal{X}$ to the space of functions $f : \mathcal{X} \to \mathbb{R}$ [4, 5]

$$\begin{aligned} \phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(\cdot, x) \end{aligned}$$

Bounding the Hypothesis Space for Learning

Here, we need to bound the $\mathcal{H}$

- For this, we can use Reproducing Kernel Hilbert Spaces.

They are Hilbert Spaces

- They are characterized by a symmetric positive definite function, named Mercer kernel:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

We can define a map from $\mathcal{X}$ to the space of functions $f : \mathcal{X} \to \mathbb{R}$ [4, 5]

$$\begin{aligned} \phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(\cdot, x) \end{aligned}$$

Thus, defining the dot product

We have a set of functions that can be defined by the use of kernels

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$$

- with $\alpha_i \in \mathbb{R}$ (Note, we simplify this presentation but actually you need to consider the complex field number)

Therefore, we can define the following dot product for such functions

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, x_j)$$

Thus, defining the dot product

We have a set of functions that can be defined by the use of kernels

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$$

- with $\alpha_i \in \mathbb{R}$ (Note, we simplify this presentation but actually you need to consider the complex field number)

Therefore, we can define the following dot product for such functions

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, x_j)$$

Therefore, we have

An interesting property

$$\begin{aligned}\langle k(\cdot, x), f\rangle &= \left\langle k(\cdot, x), \sum_{i=1}^{m} \alpha_i k(\cdot, x_i) \right\rangle \\ &= \sum_{i=1}^{m} \alpha_i k(x, x_i) = f(x)\end{aligned}$$

Basically, we can reproduce the functions by using the kernels

- One of the most important results for applications in the XX century

Therefore, we have

An interesting property

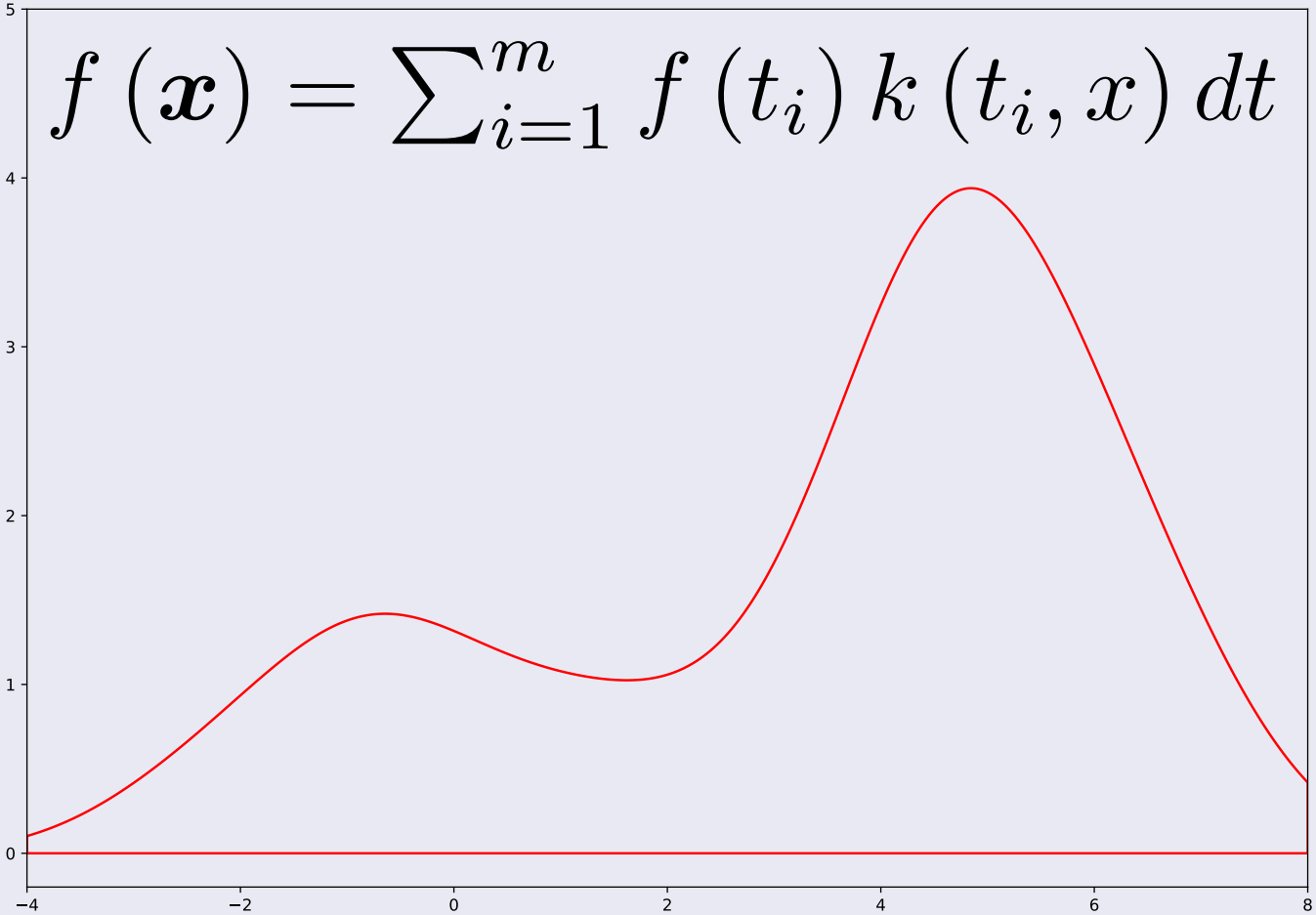
$$\begin{aligned}\langle k(\cdot, x), f\rangle &= \left\langle k(\cdot, x), \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)\right\rangle \\ &= \sum_{i=1}^{m} \alpha_i k(x, x_i) = f(x)\end{aligned}$$

Basically, we can reproduce the functions by using the kernels

- One of the most important results for applications in the XX century

Example

Gaussian Kernels (Remember Expectation Maximization)



Outline

Now imagine the following spaces

Let X be a measure space

- Let $L^2(\mathcal{X})$ the Hilbert space of square-integrable functions $f : \mathcal{X} \to \mathbb{R}$ with inner product

$$\langle f, g \rangle = \int \overline{f(x)} g(x)\, dx$$

Let $k \in L^2(\mathcal{X} \times \mathcal{X})$

- In addition, the corresponding Hilbert-Schmidt operator $K : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$Kf(x) = \int_{\mathcal{X}} k(x, x') f(x')\, dx$$

Given that we are asking symmetric kernels a.k.a positive semidefinite

- We have the Fubini's Theorem

Now imagine the following spaces

Let X be a measure space

- Let $L^2(\mathcal{X})$ the Hilbert space of square-integrable functions $f : \mathcal{X} \to \mathbb{R}$ with inner product

$$\langle f, g \rangle = \int \overline{f(x)} g(x)\, dx$$

Let $k \in L^2(\mathcal{X} \times \mathcal{X})$

- In addition, the corresponding Hilbert-Schmidt operator $K : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$Kf(x) = \int_{\mathcal{X}} k(x, x') f(x')\, dx$$

Given that we are asking symmetric kernels a.k.a positive semidefinite

- We have the Fubini's Theorem

Now imagine the following spaces

Let X be a measure space

- Let $L^2(\mathcal{X})$ the Hilbert space of square-integrable functions $f : \mathcal{X} \to \mathbb{R}$ with inner product

$$\langle f, g \rangle = \int \overline{f(x)} g(x)\, dx$$

Let $k \in L^2(\mathcal{X} \times \mathcal{X})$

- In addition, the corresponding Hilbert-Schmidt operator $K : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$Kf(x) = \int_{\mathcal{X}} k(x, x') f(x')\, dx$$

Given that we are asking symmetric kernels a.k.a positive semidefinite

- We have the Fubini's Theorem

Fubini's Theorem

Statement

- Suppose A and B are complete measure spaces. Suppose $f(x,y)$ is $A \times B$ measurable (Under inverse the image is in the σ-algebra of $A \times B$).

If we have

$$\int_{A\times B} |f(x,y)| \, d(x,y) < \infty$$

Then

$$\int_{A\times B} f(x,y) \, d(x,y) = \int_A \left[\int_B f(x,y) \, dy \right] dx = \int_B \left[\int_A f(x,y) \, dx \right] dy$$

Fubini's Theorem

Statement

- Suppose A and B are complete measure spaces. Suppose $f(x,y)$ is $A \times B$ measurable (Under inverse the image is in the σ-algebra of $A \times B$).

If we have

$$\int_{A\times B} |f(x,y)|\, d(x,y) < \infty$$

Then

$$\int_{A\times B} f(x,y)\, d(x,y) = \int_A \left[\int_B f(x,y)\, dy\right] dx = \int_B \left[\int_A f(x,y)\, dx\right] dy$$

Fubini's Theorem

Statement

- Suppose A and B are complete measure spaces. Suppose $f(x,y)$ is $A \times B$ measurable (Under inverse the image is in the σ-algebra of $A \times B$).

If we have

$$\int_{A\times B} |f(x,y)|\, d(x,y) < \infty$$

Then

$$\int_{A\times B} f(x,y)\, d(x,y) = \int_A \left[\int_B f(x,y)\, dy\right] dx = \int_B \left[\int_A f(x,y)\, dx\right] dy$$

Therefore

We have that (Self-Adjoint Operator)

$$\begin{aligned}\langle f, Kg\rangle &= \int_{\mathcal{X}} f(x)\left[\int_{\mathcal{X}} k(x,x')\, g(x')\, dx'\right] dx \\ &= \int_{\mathcal{X}} \left[\int_{\mathcal{X}} k(x,x')\, f(x)\, dx\right] g(x')\, dx' \\ &= \langle Kf, g\rangle\end{aligned}$$

Additionally

We say that k satisfies Mercer's condition if and only if

$$\int_{\mathcal{X}\times\mathcal{X}} k\left(x,x'\right) f\left(x\right) f\left(x'\right) dxdx' \geq 0 \text{ for all } f \in L^2\left(\mathcal{X}\right).$$

Finally, the Mercer's Theorem

Mercer's Theorem

- A symmetric $k \in L^2(\mathcal{X} \times \mathcal{X})$ satisfies the Mercer's condition if and only if k is a kernel.

Proof

First, a symmetric $k \in L^2(\mathcal{X} \times \mathcal{X})$ is Mercer

- Let K be the self-adjoint Hilbert-Schmidt operator corresponding to k.

Theorem

- Let $T : \mathcal{H} \to \mathcal{H}$ be a compact and self-adjoint operator on a Hilbert Space $\mathcal{H}$. Then, there is a finite or infinite sequence $\{\lambda_n\}_{n=1}^{N}$ of real eigenvalues $\lambda_n \neq 0$, and a corresponding orthonormal sequence $\{e_n\}_{n=1}^{N}$ in $\mathcal{H}$ such that:
  1. $Te_n = \lambda_n e_n$ for all n
  2. $Null(T) = Span\left[\{e_n\}_{n=1}^{N}\right]^{\perp}$
  3. If $N = \infty$ then $\lim_{n\to\infty} \lambda_n = 0$

Proof

First, a symmetric $k \in L^2(\mathcal{X} \times \mathcal{X})$ is Mercer

- Let K be the self-adjoint Hilbert-Schmidt operator corresponding to k.

Theorem

- Let $T : \mathcal{H} \to \mathcal{H}$ be a compact and self-adjoint operator on a Hilbert Space $\mathcal{H}$. Then, there is a finite or infinite sequence $\{\lambda_n\}_{n=1}^N$ of real eigenvalues $\lambda_n \neq 0$, and a corresponding orthonormal sequence $\{e_n\}_{n=1}^N$ in $\mathcal{H}$ such that:
  1. $Te_n = \lambda_n e_n$ for all n
  2. $Null(T) = Span\left[\{e_n\}_{n=1}^N\right]^\perp$
  3. If $N = \infty$ then $\lim_{n\to\infty} \lambda_n = 0$

Therefore

K has a countable collection of orthonormal eigenvectors (We will assume for simplicity real valued functions)

- It has a countable collection of orthonormal eigenfunctions $U_i \in L^2(\mathcal{X})$ with eigenvalues λ_i such that for all $f \in L^2(\mathcal{X})$

$$f = \sum_{i=1}^{\infty} \alpha_i U_i$$

Then applying the operator K

$$Kf = \sum_{i=1}^{n} \alpha_i K U_i = \sum_{i=1}^{n} \alpha_i \lambda_i U_i$$

Using the inner product of the space

$$\langle f, U_i \rangle = \left\langle \sum_{j=1}^{\infty} \alpha_j U_j, U_i \right\rangle = \alpha_j \langle U_i, U_i \rangle = \alpha_j$$

Therefore

K has a countable collection of orthonormal eigenvectors (We will assume for simplicity real valued functions)

- It has a countable collection of orthonormal eigenfunctions $U_i \in L^2(\mathcal{X})$ with eigenvalues λ_i such that for all $f \in L^2(\mathcal{X})$

$$f = \sum_{i=1}^{\infty} \alpha_i U_i$$

Then applying the operator K

$$Kf = \sum_{i=1}^{n} \alpha_i K U_i = \sum_{i=1}^{n} \alpha_i \lambda_i U_i$$

Using the inner product of the space

$$\langle f, U_i \rangle = \left\langle \sum_{j=1}^{\infty} \alpha_j U_j, U_i \right\rangle = \alpha_j \langle U_i, U_i \rangle = \alpha_j$$

Therefore

K has a countable collection of orthonormal eigenvectors (We will assume for simplicity real valued functions)

- It has a countable collection of orthonormal eigenfunctions $U_i \in L^2(\mathcal{X})$ with eigenvalues λ_i such that for all $f \in L^2(\mathcal{X})$

$$f = \sum_{i=1}^{\infty} \alpha_i U_i$$

Then applying the operator K

$$Kf = \sum_{i=1}^{n} \alpha_i K U_i = \sum_{i=1}^{n} \alpha_i \lambda_i U_i$$

Using the inner product of the space

$$\langle f, U_i \rangle = \left\langle \sum_{j=1}^{\infty} \alpha_j U_j, U_i \right\rangle = \alpha_j \langle U_i, U_i \rangle = \alpha_j$$

Therefore

By Mercer's Condition and Fubini's

$$\lambda_i \langle U_i, U_i \rangle = \langle \varphi_i, K\varphi_i \rangle = \int_{\mathcal{X}} U_i(x') \left[\int k(x, x') U_i(x') \, dx' \right] dx \geq 0$$

Because $\langle \cdot, \cdot \rangle$ is positive

- We have that $\lambda_i \geq 0$

Therefore

By Mercer's Condition and Fubini's

$$\lambda_i \langle U_i, U_i \rangle = \langle \varphi_i, K\varphi_i \rangle = \int_{\mathcal{X}} U_i(x') \left[\int k(x, x') U_i(x') dx' \right] dx \geq 0$$

Because $\langle \cdot, \cdot \rangle$ is positive

- We have that $\lambda_i \geq 0$

Now, we have

We can define $\varphi_i \in L^2(\mathcal{X})$

$$x \to \sqrt{\lambda_i} U_i(x)$$
$$\varphi_i(x) = \sqrt{\lambda_i} U_i(x)$$

Therefore, we have the following function in $L^2(\mathcal{X})$

$$\varphi(x) = \sum_{i=1}^{\infty} \sqrt{\lambda_i} U_i(x)$$

Now, we have

We can define $\varphi_i \in L^2(\mathcal{X})$

$$x \to \sqrt{\lambda_i} U_i(x)$$
$$\varphi_i(x) = \sqrt{\lambda_i} U_i(x)$$

Therefore, we have the following function in $L^2(\mathcal{X})$

$$\varphi(x) = \sum_{i=1}^{\infty} \sqrt{\lambda_i} U_i(x)$$

Therefore

We have that

$$\int_{\mathcal{X}} \langle \varphi(x), \varphi(x') \rangle f(x')\, dx' = \int_{\mathcal{X}} \left\langle \sum_{j=1}^{\infty} \sqrt{\lambda_j} U_j, \sum_{i=1}^{\infty} \sqrt{\lambda_i} U_i \right\rangle f(x')\, dx'$$

Therefore

$$\int_{\mathcal{X}} \langle \varphi(x), \varphi(x') \rangle f(x')\, dx' = \int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i \langle U_i(x), U_i(x') \rangle f(x')\, dx'$$

Here, we have that with fixed x, x'

- $U_i(x), U_i(x')$ are real valued

Therefore

We have that

$$\int_{\mathcal{X}} \langle \varphi(x), \varphi(x') \rangle f(x') \, dx' = \int_{\mathcal{X}} \left\langle \sum_{j=1}^{\infty} \sqrt{\lambda_j} U_j, \sum_{i=1}^{\infty} \sqrt{\lambda_i} U_i \right\rangle f(x') \, dx'$$

Therefore

$$\int_{\mathcal{X}} \langle \varphi(x), \varphi(x') \rangle f(x') \, dx' = \int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i \langle U_i(x), U_i(x') \rangle f(x') \, dx'$$

Here, we have that with fixed x, x'

- $U_i(x), U_i(x')$ are real valued

Therefore

We have that

$$\int_{\mathcal{X}} \langle \varphi(x), \varphi(x') \rangle f(x') \, dx' = \int_{\mathcal{X}} \left\langle \sum_{j=1}^{\infty} \sqrt{\lambda_j} U_j, \sum_{i=1}^{\infty} \sqrt{\lambda_i} U_i \right\rangle f(x') \, dx'$$

Therefore

$$\int_{\mathcal{X}} \langle \varphi(x), \varphi(x') \rangle f(x') \, dx' = \int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i \langle U_i(x), U_i(x') \rangle f(x') \, dx'$$

Here, we have that with fixed x, x'

- $U_i(x), U_i(x')$ are real valued

Then, we have that

An interesting fact

$$\left\langle U_i\left(x\right), U_i\left(x'\right)\right\rangle = U_i\left(x\right) U_i\left(x'\right)$$

Therefore

$$\int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i \left\langle U_i\left(x\right), U_i\left(x'\right)\right\rangle f\left(x'\right) dx' = \int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i U_i\left(x\right) U_i\left(x'\right) f\left(x'\right) dx'$$

Then, we have that

An interesting fact

$$\left\langle U_i\left(x\right), U_i\left(x'\right)\right\rangle = U_i\left(x\right) U_i\left(x'\right)$$

Therefore

$$\int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i \left\langle U_i\left(x\right), U_i\left(x'\right)\right\rangle f\left(x'\right) dx' = \int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i U_i\left(x\right) U_i\left(x'\right) f\left(x'\right) dx'$$

Therefore

We have that

$$\int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i U_i(x) U_i(x') f(x') dx' = \sum_{i=1}^{\infty} \lambda_i U_i(x) \int_{\mathcal{X}} U_i(x') f(x') dx'$$
$$= \sum_{i=1}^{\infty} \lambda_i \langle f, U_i \rangle U_i(x) = Kf(x)$$

This holds for all f and therefore

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

Therefore

We have that

$$\int_{\mathcal{X}} \sum_{i=1}^{\infty} \lambda_i U_i(x) U_i(x') f(x') dx' = \sum_{i=1}^{\infty} \lambda_i U_i(x) \int_{\mathcal{X}} U_i(x') f(x') dx'$$
$$= \sum_{i=1}^{\infty} \lambda_i \langle f, U_i \rangle U_i(x) = Kf(x)$$

This holds for all f and therefore

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

Conversely

if k is a kernel then you have symmetry and

$$\int_{\mathcal{X}\times\mathcal{X}} k\left(x,x'\right) f\left(x\right) f\left(x'\right) dxdx' = \int_{\mathcal{X}\times\mathcal{X}} \left\langle \varphi\left(x\right) f\left(x\right), \varphi\left(x'\right) f\left(x'\right)\right\rangle_{\mathcal{H}} dxdx'$$
$$= \left\langle \int_{\mathcal{X}} \varphi\left(x\right) f\left(x\right) dx, \int_{\mathcal{X}} \varphi\left(x'\right) f\left(x'\right) dx' \right\rangle_{\mathcal{H}} \geq 0$$

Outline

The Final Connection

When Looking at Empirical Risk, we have

$$f_D = \min_{f \in \mathcal{F}} E_{emp}[f] = \min_{f \in \mathcal{F}} \frac{1}{\ell} \sum_{i=1}^{\ell} L(f(\boldsymbol{x}), y)$$

Thus, the loss function can be seen as

$$L : \mathbb{R} \times \mathcal{Y} \to [0, +\infty]$$

- representing the price to pay by predicting $f(x)$ in place of y.

The Final Connection

When Looking at Empirical Risk, we have

$$f_D = \min_{f \in \mathcal{F}} E_{emp}[f] = \min_{f \in \mathcal{F}} \frac{1}{\ell} \sum_{i=1}^{\ell} L(f(\boldsymbol{x}), y)$$

Thus, the loss function can be seen as

$$L : \mathbb{R} \times \mathcal{Y} \to [0, +\infty]$$

- representing the price to pay by predicting $f(x)$ in place of y.

The Representer Theorem

Theorem (Nonparametric Representer Theorem) [5]

- Suppose we are given a nonempty set $\mathcal{X}$, a positive definite real-valued kernel k on $\mathcal{X} \times \mathcal{X}$, a training sample $\{(x_i, y_i)\}_{i=1}^m$ on $\mathcal{X} \times \mathcal{X}$, a strictly monotonically increasing real-valued function g on $[0, \infty]$, an arbitrary cost function $c : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$, and a class of functions:

$$\mathcal{F} = \left\{ f \in \mathbb{R}^{\mathcal{X}} | f(\cdot) = \sum_{i=1}^{\infty} \beta_i k(\cdot, z_i), \beta_i \in \mathbb{R}, z_i \in \mathcal{X}, \|f\|_{H_k} < \infty \right\}$$

- thus

$$\left\| \sum_{i=1}^{\infty} \beta_i k(\cdot, z_i) \right\|^2 = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \beta_i \beta_j k(z_i, z_j)$$

Then

We have that any $f \in \mathcal{F}$ minimizing the regularized risk functional

$$c\left(\left\{x_i, y_i, f\left(x_i\right)\right\}_{i=1}^{m}\right) + g\left(\|f\|\right)$$

- It admits a representation of the form

$$f\left(\cdot\right) = \sum_{i=1}^{m} \alpha_i k\left(\cdot, x_i\right)$$

Basically

The proof of representing the real world with a finite number of elements from it

- We can learn under certain circumstances... actually regularization...

Proof

As we have assumed that k maps into $\mathbb{R}$, we will use

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}, \phi(x) = k(\cdot, x)$$

Since k is a reproducing kernel

$$[\phi(\boldsymbol{x})](x') = k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

for all $x, x' \in \mathcal{X}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the dot product of $\mathcal{H}$

Proof

As we have assumed that k maps into $\mathbb{R}$, we will use

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}, \phi(x) = k(\cdot, x)$$

Since k is a reproducing kernel

$$[\phi(\boldsymbol{x})](x') = k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

for all $x, x' \in \mathcal{X}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the dot product of $\mathcal{H}$

Proof

As we have assumed that k maps into $\mathbb{R}$, we will use

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}, \phi(x) = k(\cdot, x)$$

Since k is a reproducing kernel

$$[\phi(\boldsymbol{x})](x') = k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

for all $x, x' \in \mathcal{X}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the dot product of $\mathcal{H}$

Given that $\mathcal{F}$ is a vectorial space

Given $x_1, ..., x_m$, we have a subspace generated by

$$X_\phi = span\left\{\phi\left(x_1\right), ..., \phi\left(x_m\right)\right\}$$

Then, we have a $v \in X_\phi^\perp$ such that

$$f = \sum_{i=1}^{m} \alpha_i \phi\left(x_i\right) + v$$

Satisfying for all x_j

$$\left\langle v, \phi\left(x_j\right)\right\rangle_{\mathcal{H}} = 0$$

Given that $\mathcal{F}$ is a vectorial space

Given $x_1, ..., x_m$, we have a subspace generated by

$$X_\phi = span\left\{\phi\left(x_1\right), ..., \phi\left(x_m\right)\right\}$$

Then, we have a $v \in X_\phi^\perp$ such that

$$f = \sum_{i=1}^{m} \alpha_i \phi\left(x_i\right) + v$$

Satisfying for all x_j

$$\left\langle v, \phi\left(x_j\right)\right\rangle_{\mathcal{H}} = 0$$

Given that $\mathcal{F}$ is a vectorial space

Given $x_1, ..., x_m$, we have a subspace generated by

$$X_\phi = span\left\{\phi\left(x_1\right), ..., \phi\left(x_m\right)\right\}$$

Then, we have a $v \in X_\phi^\perp$ such that

$$f = \sum_{i=1}^{m} \alpha_i \phi\left(x_i\right) + v$$

Satisfying for all x_j

$$\left\langle v, \phi\left(x_j\right)\right\rangle_{\mathcal{H}} = 0$$

Therefore, we have that

Using this fact and the reproducibility of k

$$
\begin{aligned}
f(x_j) &= \langle f, k(\cdot, x_j) \rangle \\
&= \left\langle \sum_{i=1}^{m} \alpha_i \phi(x_i) + v, \phi(x_j) \right\rangle \\
&= \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x_j) \rangle
\end{aligned}
$$

Consequently, the first term is independent of v

$$
c\left(\{x_i, y_i, f(x_i)\}_{i=1}^{m}\right) + g(\|f\|)
$$

What about g? Take in account that v is orthogonal to $\sum_{i=1}^{m} \alpha_i \phi(x_i)$

$$
g(\|f\|) = g\left[\sqrt{\left\|\sum \alpha_i \phi(x_i)\right\|^2 + \|v\|^2}\right] \geq g\left[\left\|\sum \alpha_i \phi(x_i)\right\|\right]
$$

Therefore, we have that

Using this fact and the reproducibility of k

$$
\begin{aligned}
f(x_j) &= \langle f, k(\cdot, x_j) \rangle \\
&= \left\langle \sum_{i=1}^{m} \alpha_i \phi(x_i) + v, \phi(x_j) \right\rangle \\
&= \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x_j) \rangle
\end{aligned}
$$

Consequently, the first term is independent of v

$$\boldsymbol{c\left(\{x_i, y_i, f(x_i)\}_{i=1}^{m}\right) + g(\|f\|)}$$

What about g? Take in account that v is orthogonal to $\sum_{i=1}^{m} \alpha_i \phi(x_i)$

$$g(\|f\|) = g\left[\sqrt{\left\|\sum \alpha_i \phi(x_i)\right\|^2 + \|v\|^2}\right] \geq g\left[\left\|\sum \alpha_i \phi(x_i)\right\|\right]$$

Therefore, we have that

Using this fact and the reproducibility of k

$$\begin{aligned} f(x_j) &= \langle f, k(\cdot, x_j) \rangle \\ &= \left\langle \sum_{i=1}^{m} \alpha_i \phi(x_i) + v, \phi(x_j) \right\rangle \\ &= \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x_j) \rangle \end{aligned}$$

Consequently, the first term is independent of v

$$\boldsymbol{c}\left(\{\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{f}(\boldsymbol{x}_i)\}_{i=1}^{m}\right) + g(\|f\|)$$

What about g? Take in account that v is orthogonal to $\sum_{i=1}^{m} \alpha_i \phi(x_i)$

$$g(\|f\|) = g\left[\sqrt{\left\|\sum \alpha_i \phi(x_i)\right\|^2 + \|v\|^2}\right] \geq g\left[\left\|\sum \alpha_i \phi(x_i)\right\|\right]$$

Therefore

We have

- with equality occurring if and only if $v = 0$.

Thus, setting $v = 0$ thus does not affect the first term of

$$c\left(\left\{x_i, y_i, f\left(x_i\right)\right\}_{i=1}^{m}\right) + g\left(\|f\|\right)$$

- while strictly reducing the second term.

Any minimizer must have $v = 0$

- Any solution takes the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k\left(\cdot, x_i\right)$$

Therefore

We have

- with equality occurring if and only if $v = 0$.

Thus, setting $v = 0$ thus does not affect the first term of

$$c\left(\left\{x_i, y_i, f\left(x_i\right)\right\}_{i=1}^{m}\right) + g\left(\|f\|\right)$$

- while strictly reducing the second term.

Any minimizer must have $v = 0$

- Any solution takes the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k\left(\cdot, x_i\right)$$

Therefore

We have

- with equality occurring if and only if $v = 0$.

Thus, setting $v = 0$ thus does not affect the first term of

$$\boldsymbol{c}\left(\left\{\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{f}\left(\boldsymbol{x}_i\right)\right\}_{i=1}^{m}\right) + g\left(\|f\|\right)$$

- while strictly reducing the second term.

Any minimizer must have $v = 0$

- Any solution takes the form

$$f\left(\cdot\right) = \sum_{i=1}^{m} \alpha_i k\left(\cdot, x_i\right)$$

However

Take a look at

- "A Generalized Representer Theorem" by Bernhard Scholkopf, Ralf Herbrich and Alex J. Smola

To look at the Semiparametric Representer Theorem

- Where the minimizer takes a form of $\widetilde{f} = f + h$ based in a given set of real-valued functions $\{\psi_p\}_{p=1}^{M}$ such that
  - $f \in \mathcal{F}$
  - $h \in span\{\psi_p\}$

However

Take a look at

- "A Generalized Representer Theorem" by Bernhard Scholkopf, Ralf Herbrich and Alex J. Smola

To look at the Semiparametric Representer Theorem

- Where the minimizer takes a form of $\widetilde{f} = f + h$ based in a given set of real-valued functions $\{\psi_p\}_{p=1}^{M}$ such that
  - $f \in \mathcal{F}$
  - $h \in span\{\psi_p\}$

Outline

We have

The following cost function

$$E_{emp}[f] = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2 + \lambda \|f\|_{\mathcal{F}}^2$$

The representer theorem applies with $C = \sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2$ and $g(\|f\|) = \lambda \|f\|_{\mathcal{F}}^2$

- Thus, if we assume that $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$

Thus, we need to solve

$$\min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N}\left[y_i - \sum_{j=1}^{N}\alpha_j k(x_j, x_i)\right]^2 + \lambda \left\|\sum_{j=1}^{N}\alpha k(\cdot, x_j)\right\|^2$$

We have

The following cost function

$$E_{emp}[f] = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2 + \lambda \|f\|_{\mathcal{F}}^2$$

The representer theorem applies with $C = \sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2$ and $g(\|f\|) = \lambda \|f\|_{\mathcal{F}}^2$

- Thus, if we assume that $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$

Thus, we need to solve

$$\min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N}\left[y_i - \sum_{j=1}^{N}\alpha_j k(x_j, x_i)\right]^2 + \lambda \left\|\sum_{j=1}^{N}\alpha k(\cdot, x_j)\right\|^2$$

We have

The following cost function

$$E_{emp}[f] = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2 + \lambda \|f\|_{\mathcal{F}}^2$$

The representer theorem applies with $C = \sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2$ and $g(\|f\|) = \lambda \|f\|_{\mathcal{F}}^2$

- Thus, if we assume that $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$

Thus, we need to solve

$$\min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N}\left[y_i - \sum_{j=1}^{N}\alpha_j k(x_j, x_i)\right]^2 + \lambda \left\|\sum_{j=1}^{N}\alpha k(\cdot, x_j)\right\|^2$$

Now

Denote $K = \left[k\left(x_i, x_j\right)\right]_{i,j=1}^{N}$ and $y = \left(y_1, y_2, ..., y_N\right)^T$

$$J\left(\alpha\right) = \alpha^T K \alpha - 2y^T K \alpha + y^T y + \lambda \alpha^T K \alpha$$

This objective function is strongly convex... what?

- Then, we have that for a K invertible

$$\frac{\partial J}{\partial \alpha} = 0$$

Now

Denote $K = \left[k\left(x_i, x_j\right)\right]_{i,j=1}^{N}$ and $y = \left(y_1, y_2, ..., y_N\right)^T$

$$J\left(\alpha\right) = \alpha^T K \alpha - 2y^T K \alpha + y^T y + \lambda \alpha^T K \alpha$$

This objective function is strongly convex... what?

- Then, we have that for a K invertible

$$\frac{\partial J}{\partial \alpha} = 0$$

Thus, we have

The following equalities

$$\alpha = (K + \lambda)^{-1} y$$

$$\widehat{f}(x) = \alpha^T \begin{pmatrix} k(x, x_1) \\ k(x, x_2) \\ \vdots \\ k(x, x_N) \end{pmatrix}$$

This similarly can be derived from ridge regression

- By kernelization of the inner product...

Thus, we have

The following equalities

$$\alpha = (K + \lambda)^{-1} y$$

$$\widehat{f}(x) = \alpha^T \begin{pmatrix} k(x, x_1) \\ k(x, x_2) \\ \vdots \\ k(x, x_N) \end{pmatrix}$$

This similarly can be derived from ridge regression

- By kernelization of the inner product...

Finally

We have that

- The use of kernels allows to have a better generalizations...

However

- There is an important property on the use of Loss functions, the convexity of them...

Finally

We have that

- The use of kernels allows to have a better generalizations...

However

- There is an important property on the use of Loss functions, the convexity of them...

Outline

Convexity Assumption

Something Notable

- We first notice that the loss function is always a function of only one variable t, if

$$t = w - y \text{ (Regression) and } t = wy \text{ (Classification)}$$

Thus, a classic assumption for the mapping

$$t \longrightarrow J(t)$$

- to be convex

Convexity Assumption

Something Notable

- We first notice that the loss function is always a function of only one variable t, if

$$t = w - y \text{ (Regression) and } t = wy \text{ (Classification)}$$

Thus, a classic assumption for the mapping

$$t \longmapsto J(t)$$

- to be convex

Implications on Convex Assumptions

A loss function is a Lipschitz function

- For every $M > 0$ there exists a constant $L_M > 0$ such that

$$|J(w_1, y) - J(w_2, y)| \leq L_M |w_1 - w_2|$$

  - For all $w_1, w_2 \in [-M, M]$ for all $y \in Y$

There exists a constant C_0 such that, for all $y \in Y$

$$J(0, y) \leq C_0$$

Implications on Convex Assumptions

A loss function is a Lipschitz function

- For every $M > 0$ there exists a constant $L_M > 0$ such that

$$|J(w_1, y) - J(w_2, y)| \leq L_M |w_1 - w_2|$$

  - For all $w_1, w_2 \in [-M, M]$ for all $y \in Y$

There exists a constant C_0 such that, for all $y \in Y$

$$J(0, y) \leq C_0$$

Which Loss Functions

Regression

- The square loss

$$J(w, y) = (w - y)^2$$

- The absolute value loss

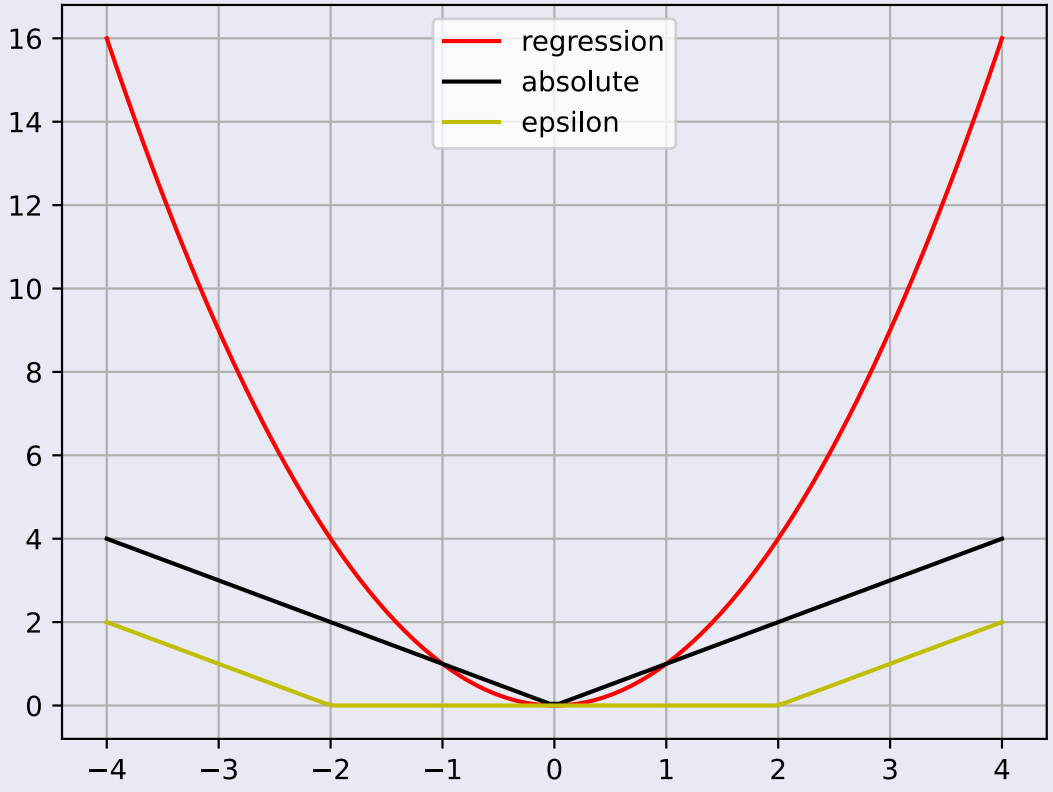
$$J(w, y) = |w - y|$$

- The ϵ-insensitive loss

$$J(w, y) = \max\{|w - y| - \epsilon, 0\}$$

Example

Regression



Classification

We have

- The Square Loss

$$J(w,y) = (w-y)^2 = (1-wy)^2$$

- The Hinge Loss

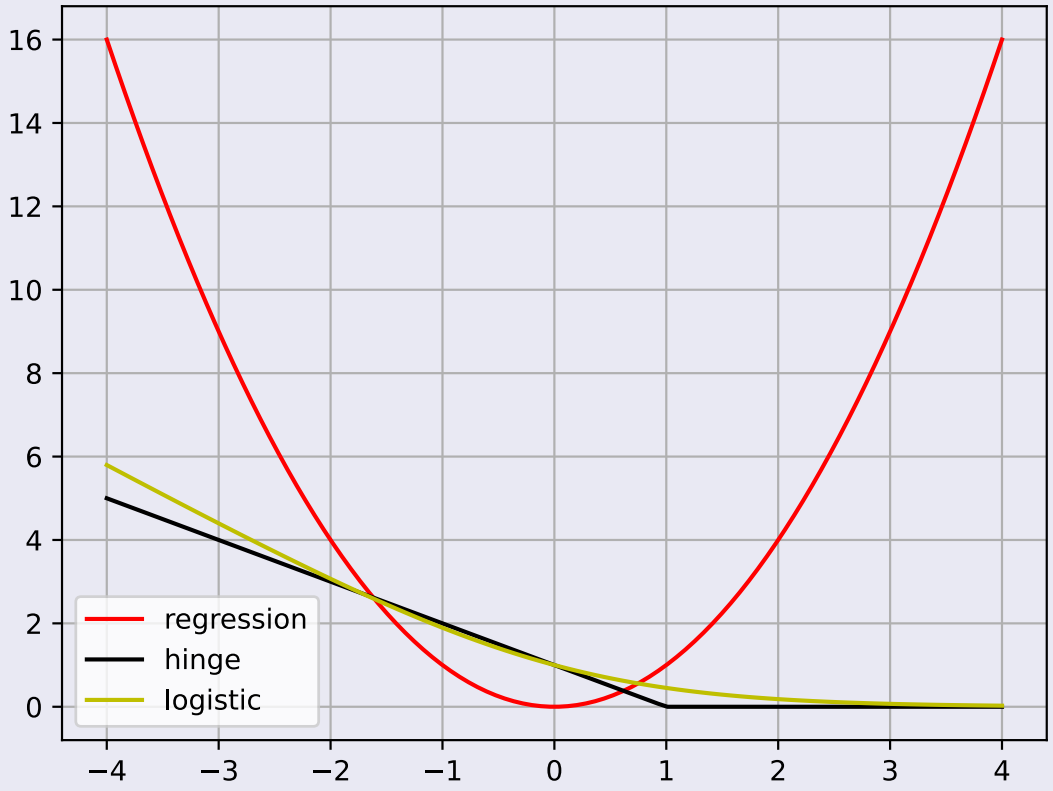
$$J(w,y) = \max\{1-wy, 0\}$$

- The Logistic Loss

$$J(w,y) = \frac{1+\exp^{-wy}}{\ln 2}$$

Example

Classification



Outline

At [6]

Several Loss Functions for Neural Networks are studied

- Here, o is the output of the last layer in the deep learner and σ is the probability estimate

Name	Equation
L_1 Loss	$\mathcal{L}_1 = \|y - o\|_1$
L_2 Loss	$\mathcal{L}_2 = \|y - o\|_2^2$
Expectation Loss	$\|y - \sigma(o)\|_1$
Regularized expectation Loss	$\|y - \sigma(o)\|_1$
Chebyshev Loss	$\max_j \left\|\sigma(o)^{(j)} - y^{(j)}\right\|$
Hinge Loss	$\sum_j \max\left\{0, \frac{1}{2} - \hat{y}^{(j)} o^{(j)}\right\}$
Log Loss (Cross Entropy)	$-\sum_j y^{(j)} \log \sigma(o)^{(j)}$
Squared Log Loss	$-\sum_j \left[y^{(j)} \log \sigma(o)^{(j)}\right]^2$

For example, we have the following property

We have that for $\boldsymbol{y}_i \in \{0,1\}^K$ with $L_j(y_i) = 1$ and $p_i = \widehat{p}(y_i|x_i)$

$$\mathcal{L}_1 = \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} - y_i^{(j)}\right|$$

$$= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} + y_i^{(j)}p_i^{(j)} - y_i^{(j)}p_i^{(j)} - y_i^{(j)}\right|$$

$$= \frac{1}{N}\sum_i\sum_j \left|y_i^{(j)}\left(p_i^{(j)} - 1\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right|$$

$$= \frac{1}{N}\sum_i\sum_j \left[y_i^{(j)}\left(1 - p_i^{(j)}\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right]$$

$$= \frac{1}{N}\sum_i\left[\sum_j y_i^{(j)} - 2\sum_j y_i^{(j)}p_i^{(j)} + \sum_j p_i^{(j)}\right]$$

$$= \frac{1}{N}\sum_i\sum_j y_i^{(j)} - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} + \frac{1}{N}\sum_i\sum_j p_i^{(j)}$$

$$= 2 - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} \approx -2E_{P(x,y)}\left[P\left(\widehat{l} = l|\widehat{l} \sim p_i, l \sim y_i\right)\right]$$

For example, we have the following property

We have that for $\boldsymbol{y}_i \in \{0,1\}^K$ with $L_j(y_i) = 1$ and $p_i = \widehat{p}(y_i|x_i)$

$$
\begin{aligned}
\mathcal{L}_1 &= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} + y_i^{(j)}p_i^{(j)} - y_i^{(j)}p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|y_i^{(j)}\left(p_i^{(j)} - 1\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right| \\
&= \frac{1}{N}\sum_i\sum_j \left[y_i^{(j)}\left(1 - p_i^{(j)}\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right] \\
&= \frac{1}{N}\sum_i\left[\sum_j y_i^{(j)} - 2\sum_j y_i^{(j)}p_i^{(j)} + \sum_j p_i^{(j)}\right] \\
&= \frac{1}{N}\sum_i\sum_j y_i^{(j)} - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} + \frac{1}{N}\sum_i\sum_j p_i^{(j)} \\
&= 2 - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} \approx -2E_{P(x,y)}\left[P\left(\widehat{l} = l|\widehat{l} \sim p_i, l \sim y_i\right)\right]
\end{aligned}
$$

For example, we have the following property

We have that for $\boldsymbol{y}_i \in \{0,1\}^K$ with $L_j(y_i) = 1$ and $p_i = \widehat{p}(y_i|x_i)$

$$
\begin{aligned}
\mathcal{L}_1 &= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} + y_i^{(j)}p_i^{(j)} - y_i^{(j)}p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|y_i^{(j)}\left(p_i^{(j)} - 1\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right| \\
&= \frac{1}{N}\sum_i\sum_j \left[y_i^{(j)}\left(1 - p_i^{(j)}\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right] \\
&= \frac{1}{N}\sum_i \left[\sum_j y_i^{(j)} - 2\sum_j y_i^{(j)}p_i^{(j)} + \sum_j p_i^{(j)}\right] \\
&= \frac{1}{N}\sum_i\sum_j y_i^{(j)} - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} + \frac{1}{N}\sum_i\sum_j p_i^{(j)} \\
&= 2 - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} \approx -2E_{P(x,y)}\left[P\left(\widehat{l} = l|\widehat{l} \sim p_i, l \sim y_i\right)\right]
\end{aligned}
$$

For example, we have the following property

We have that for $\boldsymbol{y}_i \in \{0,1\}^K$ with $L_j(y_i) = 1$ and $p_i = \widehat{p}(y_i|x_i)$

$$
\begin{aligned}
\mathcal{L}_1 &= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} + y_i^{(j)}p_i^{(j)} - y_i^{(j)}p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|y_i^{(j)}\left(p_i^{(j)} - 1\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right| \\
&= \frac{1}{N}\sum_i\sum_j \left[y_i^{(j)}\left(1 - p_i^{(j)}\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right] \\
&= \frac{1}{N}\sum_i\left[\sum_j y_i^{(j)} - 2\sum_j y_i^{(j)}p_i^{(j)} + \sum_j p_i^{(j)}\right] \\
&= \frac{1}{N}\sum_i\sum_j y_i^{(j)} - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} + \frac{1}{N}\sum_i\sum_j p_i^{(j)} \\
&= 2 - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} \approx -2E_{P(x,y)}\left[P\left(\widehat{l} = l|\widehat{l} \sim p_i, l \sim y_i\right)\right]
\end{aligned}
$$

For example, we have the following property

We have that for $\boldsymbol{y}_i \in \{0,1\}^K$ with $L_j(y_i) = 1$ and $p_i = \widehat{p}(y_i|x_i)$

$$
\begin{aligned}
\mathcal{L}_1 &= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} + y_i^{(j)}p_i^{(j)} - y_i^{(j)}p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|y_i^{(j)}\left(p_i^{(j)} - 1\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right| \\
&= \frac{1}{N}\sum_i\sum_j \left[y_i^{(j)}\left(1 - p_i^{(j)}\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right] \\
&= \frac{1}{N}\sum_i \left[\sum_j y_i^{(j)} - 2\sum_j y_i^{(j)}p_i^{(j)} + \sum_j p_i^{(j)}\right] \\
&= \frac{1}{N}\sum_i\sum_j y_i^{(j)} - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} + \frac{1}{N}\sum_i\sum_j p_i^{(j)} \\
&= 2 - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} \approx -2E_{P(x,y)}\left[P\left(\widehat{l} = l|\widehat{l} \sim p_i, l \sim y_i\right)\right]
\end{aligned}
$$

For example, we have the following property

We have that for $\boldsymbol{y}_i \in \{0,1\}^K$ with $L_j(y_i) = 1$ and $p_i = \widehat{p}(y_i|x_i)$

$$
\begin{aligned}
\mathcal{L}_1 &= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|p_i^{(j)} + y_i^{(j)}p_i^{(j)} - y_i^{(j)}p_i^{(j)} - y_i^{(j)}\right| \\
&= \frac{1}{N}\sum_i\sum_j \left|y_i^{(j)}\left(p_i^{(j)} - 1\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right| \\
&= \frac{1}{N}\sum_i\sum_j \left[y_i^{(j)}\left(1 - p_i^{(j)}\right) + p_i^{(j)}\left(1 - y_i^{(j)}\right)\right] \\
&= \frac{1}{N}\sum_i \left[\sum_j y_i^{(j)} - 2\sum_j y_i^{(j)}p_i^{(j)} + \sum_j p_i^{(j)}\right] \\
&= \frac{1}{N}\sum_i\sum_j y_i^{(j)} - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} + \frac{1}{N}\sum_i\sum_j p_i^{(j)} \\
&= 2 - 2\frac{1}{N}\sum_i\sum_j y_i^{(j)}p_i^{(j)} \approx -2E_{P(x,y)}\left[P\left(\widehat{l} = l|\widehat{l} \sim p_i, l \sim y_i\right)\right]
\end{aligned}
$$

Therefore

We have

- For this reason we refer to this loss as expectation loss

However, Why is this loss not being used?

- Maybe the following proposition will answer the question

Therefore

We have

- For this reason we refer to this loss as expectation loss

However, Why is this loss not being used?

- Maybe the following proposition will answer the question

We have

Proposition

- $\mathcal{L}_1$ and $\mathcal{L}_2$ losses applied to probabilities estimates coming from sigmoid (or softmax) have non-monotonic partial derivatives w.r.t. to the output of the final layer (and the loss is not convex nor concave w.r.t. to last layer weights). Furthermore, they vanish in both infinities, which slows down learning of heavily misclassified examples.

Proof

- Let us denote sigmoid activation as

$$\sigma(x) = \frac{1}{1 + \exp\{-x\}}$$

We have

Proposition

- $\mathcal{L}_1$ and $\mathcal{L}_2$ losses applied to probabilities estimates coming from sigmoid (or softmax) have non-monotonic partial derivatives w.r.t. to the output of the final layer (and the loss is not convex nor concave w.r.t. to last layer weights). Furthermore, they vanish in both infinities, which slows down learning of heavily misclassified examples.

Proof

- Let us denote sigmoid activation as

$$\sigma(x) = \frac{1}{1 + \exp\{-x\}}$$

Thus, we have

Using Chain Rule

$$\frac{\partial \mathcal{L}_1 \circ \sigma}{\partial o}(o_p) = \frac{\partial \left[\left|1 - \frac{1}{1+\exp\{-o\}}\right|\right] o_p}{\partial o}$$
$$= -\frac{\exp\{-o_p\}}{1+\exp\{-o_p\}}$$

In addition, we have that

$$\lim_{o_p \to \infty} -\frac{\exp\{-o_p\}}{1+\exp\{-o_p\}} = \lim_{o_p \to -\infty} -\frac{\exp\{-o_p\}}{1+\exp\{-o_p\}} = 0$$

Thus, we have

Using Chain Rule

$$\frac{\partial \mathcal{L}_1 \circ \sigma}{\partial o}(o_p) = \frac{\partial \left[\left|1 - \frac{1}{1+\exp\{-o\}}\right|\right] o_p}{\partial o}$$
$$= -\frac{\exp\{-o_p\}}{1+\exp\{-o_p\}}$$

In addition, we have that

$$\lim_{o_p \to \infty} -\frac{\exp\{-o_p\}}{1+\exp\{-o_p\}} = \lim_{o_p \to -\infty} -\frac{\exp\{-o_p\}}{1+\exp\{-o_p\}} = 0$$

Additionally

We have that

$$\frac{\partial \mathcal{L}_1 \circ \sigma}{\partial o}(0) - \frac{\exp\{0\}}{1+\exp\{0\}} = -\frac{1}{4} < 0$$

Additionally

- Lack of convexity comes from the same argument since second derivative w.r.t. to any weight in the final layer of the model changes sign

Additionally

We have that

$$\frac{\partial \mathcal{L}_1 \circ \sigma}{\partial o}(0) - \frac{\exp\{0\}}{1+\exp\{0\}} = -\frac{1}{4} < 0$$

Additionally

- Lack of convexity comes from the same argument since second derivative w.r.t. to any weight in the final layer of the model changes sign

Outline

Therefore

We have a problem with the use of these functions

- For the use on Neural Networks...

We need something different

- Because even with the kernelized versions of them of the output at $\mathcal{L}_2$

$$\frac{\partial \mathcal{L}_2^{ker} \circ \sigma}{\partial o}\left(o_p\right) = \frac{\partial \left\| y - \sum_{i=1}^{m} \alpha_i k\left(\sigma\left(o\right), x_i\right) \right\|_2^2 \left(o_p\right)}{\partial o}$$

A small problem

- $k\left(\sigma\left(o\right), x_i\right)$ needs to be derivable by o

Therefore

We have a problem with the use of these functions

- For the use on Neural Networks...

We need something different

- Because even with the kernelized versions of them of the output at $\mathcal{L}_2$

$$\frac{\partial \mathcal{L}_2^{ker} \circ \sigma}{\partial o}(o_p) = \frac{\partial \left\| y - \sum_{i=1}^{m} \alpha_i k\left(\sigma\left(o\right), x_i\right) \right\|_2^2 (o_p)}{\partial o}$$

A small problem

- $k\left(\sigma\left(o\right), x_i\right)$ needs to be derivable by o

Therefore

We have a problem with the use of these functions

- For the use on Neural Networks...

We need something different

- Because even with the kernelized versions of them of the output at $\mathcal{L}_2$

$$\frac{\partial \mathcal{L}_2^{ker} \circ \sigma}{\partial o}\left(o_p\right) = \frac{\partial \left\| y - \sum_{i=1}^{m} \alpha_i k\left(\sigma\left(o\right), x_i\right)\right\|_2^2 \left(o_p\right)}{\partial o}$$

A small problem

- $k\left(\sigma\left(o\right), x_i\right)$ needs to be derivable by o

Not only that

This is applied to the exit of the neural network

- Actually, there is a layer that acts a kernel, the convolutional layer

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{ij}^{(l)} * Y_j^{(l-1)}$$

Thus, we can generalize this to a kernel layer

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} k\left(K_{ij}^{(l)}, Y_j^{(l-1)}\right)$$

And the problem

- Which One? A Research Topic...

Not only that

This is applied to the exit of the neural network

- Actually, there is a layer that acts a kernel, the convolutional layer

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{ij}^{(l)} * Y_j^{(l-1)}$$

Thus, we can generalize this to a kernel layer

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} k\left(K_{ij}^{(l)}, Y_j^{(l-1)}\right)$$

And the problem

- Which One? A Research Topic...

Not only that

This is applied to the exit of the neural network

- Actually, there is a layer that acts a kernel, the convolutional layer

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{ij}^{(l)} * Y_j^{(l-1)}$$

Thus, we can generalize this to a kernel layer

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} k\left(K_{ij}^{(l)}, Y_j^{(l-1)}\right)$$

And the problem

- Which One? A Research Topic...

Outline

Going back to our original cost function

Recall the binary linear classifiers with targets $y \in \{0, 1\}$

$$z = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

The Goal is to correctly classify every training example

- this might be impossible if the dataset is not linearly separable.

We want to avoid

- To do overfitting...

Going back to our original cost function

Recall the binary linear classifiers with targets $y \in \{0, 1\}$

$$z = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

The Goal is to correctly classify every training example

- this might be impossible if the dataset is not linearly separable.

We want to avoid

- To do overfitting...

Going back to our original cost function

Recall the binary linear classifiers with targets $y \in \{0, 1\}$

$$z = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

The Goal is to correctly classify every training example

- this might be impossible if the dataset is not linearly separable.

We want to avoid

- To do overfitting...

How to deal with this?

One natural criterion is to minimize the number of misclassified training examples

- We can try to solve by the using 0-1 loss:

$$\mathcal{L}_{0-1}(y,t) = \begin{cases} 0 & \text{if } y = t \\ 1 & otherwise \end{cases}$$

The cost function is just the loss averaged over the training examples

- We try to make it small

How to deal with this?

One natural criterion is to minimize the number of misclassified training examples

- We can try to solve by the using 0-1 loss:

$$\mathcal{L}_{0-1}(y,t) = \begin{cases} 0 & \text{if } y = t \\ 1 & otherwise \end{cases}$$

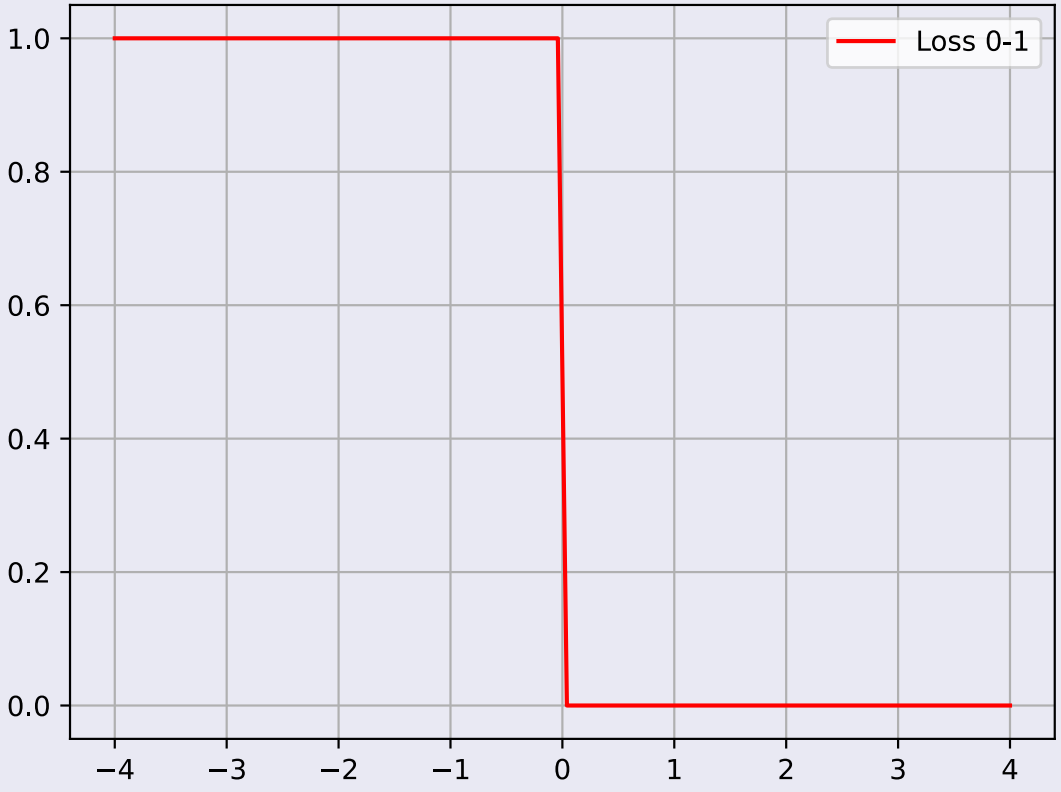
The cost function is just the loss averaged over the training examples

- We try to make it small

Outline

Attempt 0-1 Loss

We have something like this



Attempt 0-1 Loss

First Problem

- We need to compute the partial derivatives $\frac{\partial \mathcal{L}_{0-1}}{\partial w_j}$

Basically, we need to obtain

- How much does $\mathcal{L}_{0-1}$ change if you make a change to w_j?

We notice something

- As long we are not at the boundary, changes on w_j will not have no effect

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = 0$$

Attempt 0-1 Loss

First Problem

- We need to compute the partial derivatives $\frac{\partial \mathcal{L}_{0-1}}{\partial w_j}$

Basically, we need to obtain

- How much does $\mathcal{L}_{0-1}$ change if you make a change to w_j?

We notice something

- As long we are not at the boundary, changes on w_j will not have no effect

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = 0$$

Attempt 0-1 Loss

First Problem

- We need to compute the partial derivatives $\frac{\partial \mathcal{L}_{0-1}}{\partial w_j}$

Basically, we need to obtain

- How much does $\mathcal{L}_{0-1}$ change if you make a change to w_j?

We notice something

- As long we are not at the boundary, changes on w_j will not have no effect

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = 0$$

As in the original 0-1 Cortez and Vapnik problem

Yes... at the original problem you have a 0-1 problem (0-1 SVM with Soft Margins)

- Which falls into a combinatorial problem... forget also on using Gradient to optimize it...

Therefore, we need something different

- Ok... we need to look to another place...

As in the original 0-1 Cortez and Vapnik problem

Yes... at the original problem you have a 0-1 problem (0-1 SVM with Soft Margins)

- Which falls into a combinatorial problem... forget also on using Gradient to optimize it...

Therefore, we need something different

- Ok... we need to look to another place...

Attempt Linear Regression

We have the following situation

$$y = \boldsymbol{w}^T \boldsymbol{x} + b$$
$$\mathcal{L}_2 = \frac{1}{2}(y - t)^2$$

We have two solutions (Look at our slides on Machine Learning)

- Closed form
- Gradient Descent form

Does it make sense for classification?

- One obvious problem is that the predictions are real-valued rather than binary.

Attempt Linear Regression

We have the following situation

$$y = \boldsymbol{w}^T \boldsymbol{x} + b$$
$$\mathcal{L}_2 = \frac{1}{2}(y - t)^2$$

We have two solutions (Look at our slides on Machine Learning)

- Closed form
- Gradient Descent form

Does it make sense for classification?

- One obvious problem is that the predictions are real-valued rather than binary.

Attempt Linear Regression

We have the following situation

$$y = \boldsymbol{w}^T \boldsymbol{x} + b$$
$$\mathcal{L}_2 = \frac{1}{2}(y - t)^2$$

We have two solutions (Look at our slides on Machine Learning)

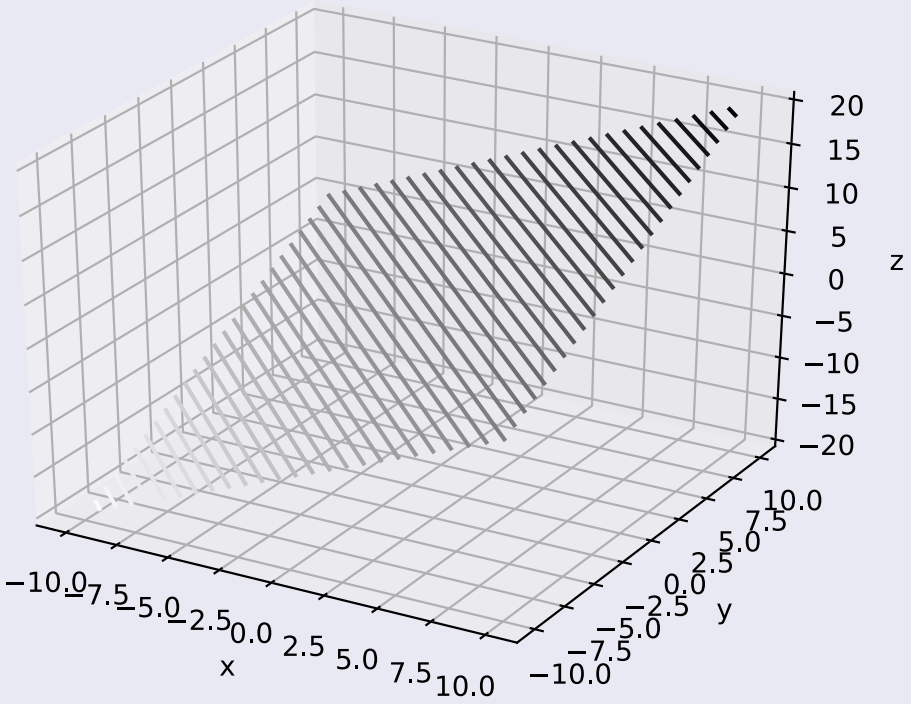
- Closed form
- Gradient Descent form

Does it make sense for classification?

- One obvious problem is that the predictions are real-valued rather than binary.

Example

WE have the loss function $y = \boldsymbol{w}^T \boldsymbol{x} + b$ with $t = 1.0$



It is possible to binarize this

By using a thrheshold

- At $y = \frac{1}{2}$

This type of relaxation

- It is called **surrogate loss function**.

It is possible to binarize this

By using a thrheshold

- At $y = \frac{1}{2}$

This type of relaxation

- It is called **surrogate loss function**.

There is still a problem

Suppose we have a positive example, $t = 1$

- If we predict $y = 1$, we get a cost of 0, whereas if we make the wrong prediction $y = 0$, we get a cost of $\frac{1}{2}$,

$$\mathcal{L}_2 = \frac{1}{2}(y - t)^2$$

However, we can trick our output

- We really confident you have a positive example and we predict $y = 9$,

$$\mathcal{L}_2 = \frac{1}{2}(9 - 1)^2 = 32$$

This is far higher than the cost for $y = 0$

- Therefore, the quadratic loss function sacrifices somethign when using it...

There is still a problem

Suppose we have a positive example, $t = 1$

- If we predict $y = 1$, we get a cost of 0, whereas if we make the wrong prediction $y = 0$, we get a cost of $\frac{1}{2}$,

$$\mathcal{L}_2 = \frac{1}{2}(y - t)^2$$

However, we can trick our output

- We really confident you have a positive example and we predict $y = 9$,

$$\mathcal{L}_2 = \frac{1}{2}(9 - 1)^2 = 32$$

This is far higher than the cost for $y = 0$

- Therefore, the quadratic loss function sacrifices somethign when using it...

There is still a problem

Suppose we have a positive example, $t = 1$

- If we predict $y = 1$, we get a cost of 0, whereas if we make the wrong prediction $y = 0$, we get a cost of $\frac{1}{2}$,

$$\mathcal{L}_2 = \frac{1}{2}(y - t)^2$$

However, we can trick our output

- We really confident you have a positive example and we predict $y = 9$,

$$\mathcal{L}_2 = \frac{1}{2}(9 - 1)^2 = 32$$

This is far higher than the cost for $y = 0$

- Therefore, the quadratic loss function sacrifices somethign when using it...

Outline

Attempt Logistic Nonlinearity

We can then filter the previous attempt by using a σ

$$
\begin{aligned}
z &= \boldsymbol{w}^T \boldsymbol{x} + b \\
y &= \sigma(z) \\
\mathcal{L}_2 &= \frac{1}{2}(y - t)^2 \\
\sigma(z) &= \frac{1}{1 + \exp\{-z\}}
\end{aligned}
$$

Something Notable

- Notice that this model solves the problem we observed with linear regression.
  - As the predictions get more and more confident on the correct answer, the loss continues to decrease.

Attempt Logistic Nonlinearity

We can then filter the previous attempt by using a σ

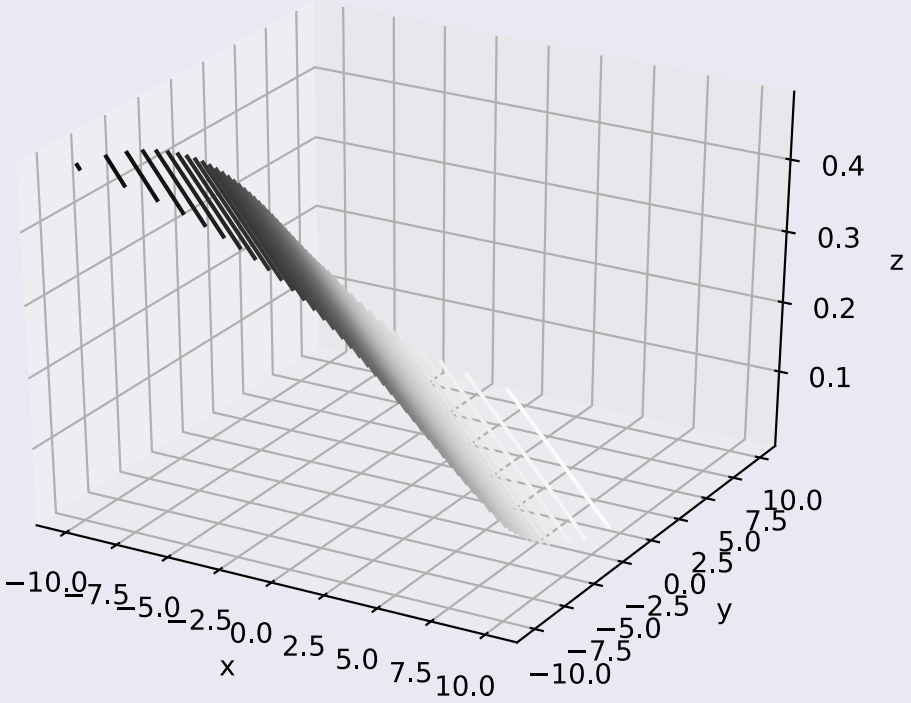
$$
\begin{aligned}
z &= \boldsymbol{w}^T \boldsymbol{x} + b \\
y &= \sigma(z) \\
\mathcal{L}_2 &= \frac{1}{2}(y - t)^2 \\
\sigma(z) &= \frac{1}{1 + \exp\{-z\}}
\end{aligned}
$$

Something Notable

- Notice that this model solves the problem we observed with linear regression.
  - As the predictions get more and more confident on the correct answer, the loss continues to decrease.

Example of this

We have the loss function $\mathcal{L}_2 = \frac{1}{2}\left(\sigma\left(z\right) - t\right)^2$



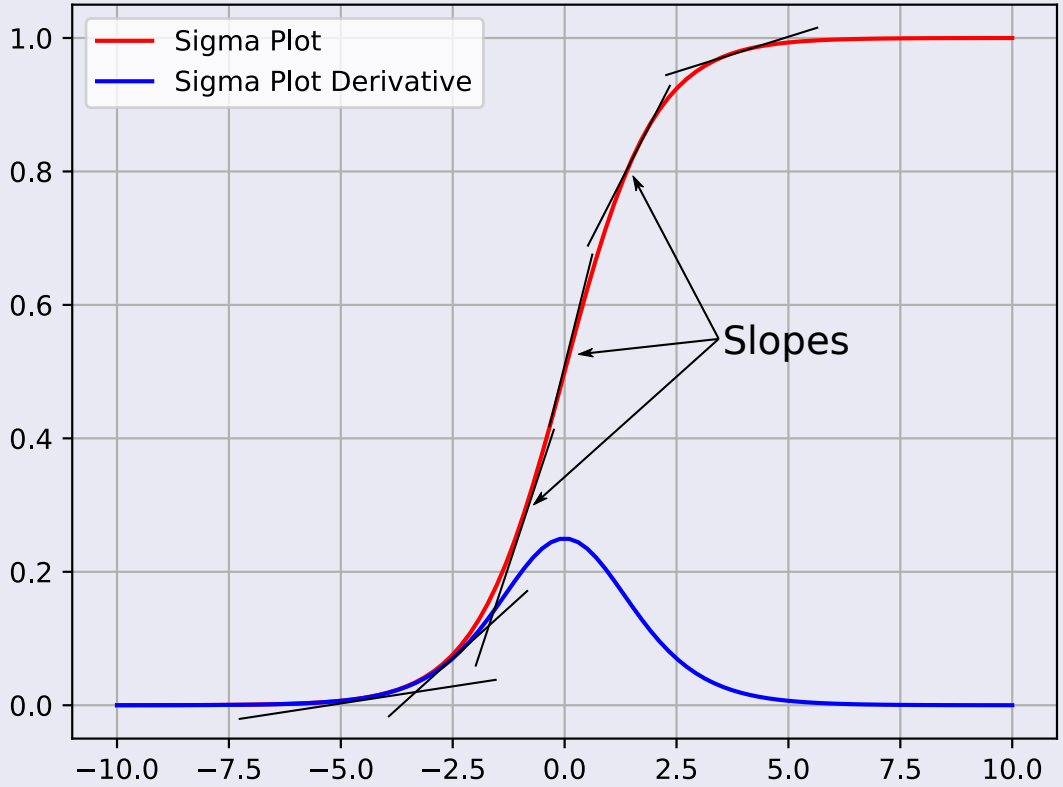
Therefore

The derivative is equal to

$$\frac{\partial \sigma(z)}{\partial z} = \frac{\exp\{-z\}}{[1+\exp\{-z\}]^2} = \sigma(z)[1-\sigma(z)]$$

Example

We have the following situation



The nice part of this function

Something Notable

- If your target is $t = 1$ and you are learning

You accelerate fast by the use of the Gradient Descent

- Once you get near to it you decelerate... in your learning

The nice part of this function

Something Notable

- If your target is $t = 1$ and you are learning

You accelerate fast by the use of the Gradient Descent

- Once you get near to it you decelerate... in your learning

How does this learning looks like?

By Chain Rule

$$\frac{d\mathcal{L}_2}{dz} = \frac{d\mathcal{L}_2}{dy} \times \frac{dy}{dz} = (y - t)\, y\, (1 - y)$$

Therefore, we have that

$$\frac{\partial \mathcal{L}_2}{\partial w_j} = \frac{d\mathcal{L}_2}{dz} \times \frac{\partial z}{\partial w_j} = \frac{d\mathcal{L}_2}{dz} \times x_j$$

How does this learning looks like?

By Chain Rule

$$\frac{d\mathcal{L}_2}{dz} = \frac{d\mathcal{L}_2}{dy} \times \frac{dy}{dz} = (y - t)\, y\, (1 - y)$$

Therefore, we have that

$$\frac{\partial \mathcal{L}_2}{\partial w_j} = \frac{d\mathcal{L}_2}{dz} \times \frac{\partial z}{\partial w_j} = \frac{d\mathcal{L}_2}{dz} \times x_j$$

Outline

Relation with Automatic Differentiation

This formula can be used re-used

- Actually there is a way to reuse the previous formula for the bias

We have that

$$\frac{\partial \mathcal{L}_2}{\partial b} = \frac{d\mathcal{L}_2}{dz} \times \frac{\partial z}{\partial b} = \frac{d\mathcal{L}_2}{dz}$$

This re-usability

- It is at the center of the Automatic Differentiation

Relation with Automatic Differentiation

This formula can be used re-used

- Actually there is a way to reuse the previous formula for the bias

We have that

$$\frac{\partial \mathcal{L}_2}{\partial b} = \frac{d\mathcal{L}_2}{dz} \times \frac{\partial z}{\partial b} = \frac{d\mathcal{L}_2}{dz}$$

This re-usability

- It is at the center of the Automatic Differentiation

Relation with Automatic Differentiation

This formula can be used re-used

- Actually there is a way to reuse the previous formula for the bias

We have that

$$\frac{\partial \mathcal{L}_2}{\partial b} = \frac{d\mathcal{L}_2}{dz} \times \frac{\partial z}{\partial b} = \frac{d\mathcal{L}_2}{dz}$$

This re-usability

- It is at the center of the Automatic Differentiation

However there is a glitch!!!

If you have an incorrect classification of a sample

- You can predict a negative label with $z = -5$ thus $y \approx 0.0067$ for a positive one.

We find that

$$\frac{d\mathcal{L}_2}{dz} = -0.0066$$

This is a pretty small value, considering how big the mistake was

- Therefore, we have that this gradient will not help this sample to get out of the error

However there is a glitch!!!

If you have an incorrect classification of a sample

- You can predict a negative label with $z = -5$ thus $y \approx 0.0067$ for a positive one.

We find that

$$\frac{d\mathcal{L}_2}{dz} = -0.0066$$

This is a pretty small value, considering how big the mistake was

- Therefore, we have that this gradient will not help this sample to get out of the error

However there is a glitch!!!

If you have an incorrect classification of a sample

- You can predict a negative label with $z = -5$ thus $y \approx 0.0067$ for a positive one.

We find that

$$\frac{d\mathcal{L}_2}{dz} = -0.0066$$

This is a pretty small value, considering how big the mistake was

- Therefore, we have that this gradient will not help this sample to get out of the error

The Problem

We have that

- The problem with squared error loss in the classification setting is that it does not distinguish bad predictions from extremely bad predictions.

We need something better for classification

- Question What?

The Problem

We have that

- The problem with squared error loss in the classification setting is that it does not distinguish bad predictions from extremely bad predictions.

We need something better for classification

- Question What?

Outline

Problem with Squared Error

It treats small values of different magnitudes equally

- $y = 0.01$ and $y = 0.00001$ as nearly equivalent (for a positive example)

What we want

- We want a loss function which makes these very different!!!

Problem with Squared Error

It treats small values of different magnitudes equally

- $y = 0.01$ and $y = 0.00001$ as nearly equivalent (for a positive example)

What we want

- We want a loss function which makes these very different!!!

Cross-Entropy(CE)

Defined as follow

$$\mathcal{L}_{\mathcal{CE}}(y,t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1-y) & \text{if } t = 0 \end{cases}$$

In our previous example

- $\mathcal{L}_{\mathcal{CE}}(0.01, 1) = 4.6$
- $\mathcal{L}_{\mathcal{CE}}(0.00001, 1) = 11.5$

Therefore

- cross-entropy treats the latter as much worse than the former.

Cross-Entropy(CE)

Defined as follow

$$\mathcal{L}_{\mathcal{CE}}(y,t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1-y) & \text{if } t = 0 \end{cases}$$

In our previous example

- $\mathcal{L}_{\mathcal{CE}}(0.01, 1) = 4.6$
- $\mathcal{L}_{\mathcal{CE}}(0.00001, 1) = 11.5$

Therefore

- cross-entropy treats the latter as much worse than the former.

Cross-Entropy(CE)

Defined as follow

$$\mathcal{L}_{\mathcal{CE}}(y,t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1-y) & \text{if } t = 0 \end{cases}$$

In our previous example

- $\mathcal{L}_{\mathcal{CE}}(0.01, 1) = 4.6$
- $\mathcal{L}_{\mathcal{CE}}(0.00001, 1) = 11.5$

Therefore

- cross-entropy treats the latter as much worse than the former.

A Better Loss Function

We can collapse the previous definition to

$$\mathcal{L}_{\mathcal{CE}}(y, t) = -t \log y - (1 - t) \log (1 - y)$$

We have the following example

- Split the real line in two classes positive side $t = 1$ and negative side $t = 0$

A Better Loss Function

We can collapse the previous definition to

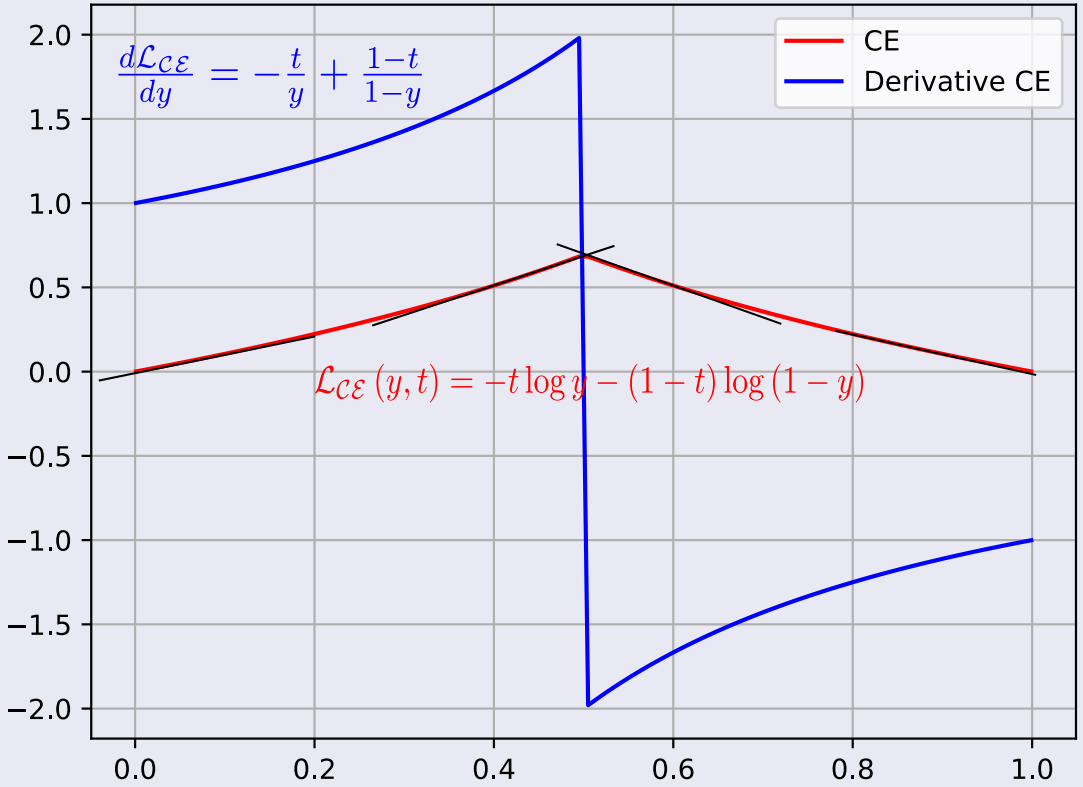
$$\mathcal{L}_{\mathcal{CE}}(y,t) = -t\log y - (1-t)\log(1-y)$$

We have the following example

- Split the real line in two classes positive side $t = 1$ and negative side $t = 0$

Example

We have the following



Therefore, we have

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to y

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dy} = -\frac{t}{y} + \frac{1-t}{1-y}$$

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to z

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dy} \times \frac{dy}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dy} \times y\left(1-y\right)$$

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to w_j

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dw_j} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} \times \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} \times x_j$$

Therefore, we have

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to y

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dy} = -\frac{t}{y} + \frac{1-t}{1-y}$$

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to z

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dy} \times \frac{dy}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dy} \times y\,(1-y)$$

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to w_j

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dw_j} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} \times \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} \times x_j$$

Therefore, we have

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to y

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dy} = -\frac{t}{y} + \frac{1-t}{1-y}$$

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to z

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dy} \times \frac{dy}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dy} \times y\,(1-y)$$

The derivative of $\mathcal{L}_{\mathcal{CE}}$ with respect to w_j

$$\frac{d\mathcal{L}_{\mathcal{CE}}}{dw_j} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} \times \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} = \frac{d\mathcal{L}_{\mathcal{CE}}}{dz} \times x_j$$

Outline

The final touch up

There is a big problem

- What happens if we have a positive example ($t = 1$)
  - And you get $y \approx 0$

This is likely to happen at the very beginning of training

- But if y is small enough, it could be smaller than the smallest floating point value
  - Basically 0 or near by to 0

Then when we compute the cross-entropy

- We have that $\frac{d\mathcal{L}_{\mathcal{CE}}}{dy}$ becomes extremely large in magnitud

The final touch up

There is a big problem

- What happens if we have a positive example ($t = 1$)
  - And you get $y \approx 0$

This is likely to happen at the very beginning of training

- But if y is small enough, it could be smaller than the smallest floating point value
  - Basically 0 or near by to 0

Then when we compute the cross-entropy

- We have that $\frac{d\mathcal{L}_{CE}}{dy}$ becomes extremely large in magnitud

The final touch up

There is a big problem

- What happens if we have a positive example ($t = 1$)
  - And you get $y \approx 0$

This is likely to happen at the very beginning of training

- But if y is small enough, it could be smaller than the smallest floating point value
  - Basically 0 or near by to 0

Then when we compute the cross-entropy

- We have that $\frac{d\mathcal{L}_{\mathcal{CE}}}{dy}$ becomes extremely large in magnitud

Better, we bound the output of the network

The so called Logistic-Cross Entropy

$$\mathcal{L}_{\mathcal{LCE}}(z,t) = \mathcal{L}_{\mathcal{CE}}(\sigma(z),t) = t\log(1+\exp\{-z\}) + (1-t)\log(1+\exp\{z\})$$

This is unstable given the term $\exp\{z\}$

- We need to deal with this... for example,
  - Python, numpy has np.logadddexp takes care of this

$$\mathbf{E} = \mathtt{t*np.logaddexp}(0,-\mathtt{z}) + (1-\mathtt{t})\mathtt{*np.logaddexp}(0,z)$$

Better, we bound the output of the network

The so called Logistic-Cross Entropy

$$\mathcal{L}_{\mathcal{LCE}}(z,t) = \mathcal{L}_{\mathcal{CE}}(\sigma(z),t) = t\log(1+\exp\{-z\}) + (1-t)\log(1+\exp\{z\})$$

This is unstable given the term $\exp\{z\}$

- We need to deal with this... for example,
  - Python, numpy has np.logadddexp takes care of this

$$\mathbf{E} = \mathtt{t}*\mathtt{np.logaddexp}(0,-\mathtt{z}) + (1-\mathtt{t})*\mathtt{np.logaddexp}(0,z)$$

What about the derivative?

We have

$$
\begin{aligned}
\frac{d\mathcal{L}_{\mathcal{LCE}}}{dz} &= \frac{d}{dz}\left[t\log\left(1+\exp\{-z\}\right)+(1-t)\log\left(1+\exp\{z\}\right)\right] \\
&= -t\times\frac{\exp\{-z\}}{1+\exp\{-z\}}+(1-t)\frac{\exp\{z\}}{1+\exp\{z\}} \\
&= -t\,(1-y)+(1-t)\,y \\
&= y-t
\end{aligned}
$$

Wow... quite simple derivative

- Observe that this is exactly the same formula $\frac{d\mathcal{L}_2}{dy}$ as for in the case of linear regression.

What about the derivative?

We have

$$\begin{aligned}\frac{d\mathcal{L}_{\mathcal{LCE}}}{dz} &= \frac{d}{dz}\left[t\log\left(1+\exp\{-z\}\right)+(1-t)\log\left(1+\exp\{z\}\right)\right]\\ &= -t\times\frac{\exp\{-z\}}{1+\exp\{-z\}}+(1-t)\frac{\exp\{z\}}{1+\exp\{z\}}\\ &= -t\,(1-y)+(1-t)\,y\\ &= y-t\end{aligned}$$

Wow... quite simple derivative

- Observe that this is exactly the same formula $\frac{d\mathcal{L}_2}{dy}$ as for in the case of linear regression.

Interpretation

if $y > t$, you made too positive a prediction

- You want to shift your prediction in the negative direction.

if $y < t$

- You want to shift your prediction in the positive direction.

Interpretation

if $y > t$, you made too positive a prediction

- You want to shift your prediction in the negative direction.

if $y < t$

- You want to shift your prediction in the positive direction.

Outline

Yann LeCunn "Who is afraid of non-convex loss functions?"[7]

Machine Learning theory has essentially never moved beyond convex models

- This is actually wrong

Given the previous development

- Accepting non-convexity allows elegant models

Not only that

- The price we pay for insisting on convexity is an unbearable increase in the size of the model
  - Actually fat shallow models vs something else...

Yann LeCunn "Who is afraid of non-convex loss functions?"[7]

Machine Learning theory has essentially never moved beyond convex models

- This is actually wrong

Given the previous development

- Accepting non-convexity allows elegant models

Not only that

- The price we pay for insisting on convexity is an unbearable increase in the size of the model
  - Actually fat shallow models vs something else...

Yann LeCunn "Who is afraid of non-convex loss functions?"[7]

Machine Learning theory has essentially never moved beyond convex models

- This is actually wrong

Given the previous development

- Accepting non-convexity allows elegant models

Not only that

- The price we pay for insisting on convexity is an unbearable increase in the size of the model
  - Actually fat shallow models vs something else...

Therefore

Based on this idea

- We need to look at different functions for loss

For example in [8]

- They proposed a more general loss function based in a parameter $\alpha \in (0, \infty]$

Therefore

Based on this idea

- We need to look at different functions for loss

For example in [8]

- They proposed a more general loss function based in a parameter $\alpha \in (0, \infty]$

Outline

We have

Definition [9, 8]

- Let $\mathcal{P}(\mathcal{Y})$ be the set of probability distributions over $\mathcal{Y}$. For $\alpha \in (0, \infty]$, we define α-loss for $\alpha \in (0,1) \cup (1, \infty)$, $l^{\alpha} : \mathcal{Y} \to \mathbb{R}^{+}$ as

$$l^{\alpha}(y, P_Y) = \frac{\alpha}{1-\alpha}\left[1 - P_Y(y)^{1-1/\alpha}\right]$$

and by continuous extension,

$$l^{1}(y, P_Y) = -\log P_Y(y) \text{ and}$$

$$l^{\infty}(y, P_Y) = 1 - \log P_Y(y)$$

Cases

For $\alpha = 1$

- Such a risk minimization involves minimizing the average log loss,
  - refining a posterior belief over all y for a given observation x.

Furthermore, as α increases from 1 to ∞

- The loss function increasingly limits the effect of the low probability outcomes

$$\lim_{\alpha \to \infty} l^{\alpha}(y, P_Y) = \lim_{\alpha \to \infty} \frac{\alpha}{1-\alpha} \times \lim_{\alpha \to \infty} \left[1 - P_Y(y)^{1-1/\alpha}\right] = P_Y(y) - 1$$

Cases

For $\alpha = 1$

- Such a risk minimization involves minimizing the average log loss,
  - refining a posterior belief over all y for a given observation x.

Furthermore, as α increases from 1 to ∞

- The loss function increasingly limits the effect of the low probability outcomes

$$\lim_{\alpha \to \infty} l^{\alpha}\left(y, P_Y\right) = \lim_{\alpha \to \infty} \frac{\alpha}{1-\alpha} \times \lim_{\alpha \to \infty} \left[1 - P_Y\left(y\right)^{1-1/\alpha}\right] = P_Y\left(y\right) - 1$$

Not only that

As α decreases from 1 towards 0

- The loss function places increasingly higher weights on the low probability outcomes

Until at $\alpha = 0$

$$\lim_{\alpha \to 0} \frac{\alpha}{1-\alpha}\left[1 - P_Y(y)^{1-1/\alpha}\right] = \lim_{\alpha \to 0} P_Y(y)^{1-1/\alpha} - 1 = \lim_{\alpha \to 0} \frac{P_Y(y)}{P_Y(y)^{1/\alpha}} - 1 = \infty$$

Not only that

As α decreases from 1 towards 0

- The loss function places increasingly higher weights on the low probability outcomes

Until at $\alpha = 0$

$$\lim_{\alpha \to 0} \frac{\alpha}{1-\alpha}\left[1 - P_Y(y)^{1-1/\alpha}\right] = \lim_{\alpha \to 0} P_Y(y)^{1-1/\alpha} - 1 = \lim_{\alpha \to 0} \frac{P_Y(y)}{P_Y(y)^{1/\alpha}} - 1 = \infty$$

Therefore

We have that

- The loss function pays an infinite cost by ignoring the training data distribution completely.

Note the following

- α quantifies the level of certainty placed on the posterior distribution

Therefore

- Larger α indicate increasing certainty over a smaller set of Y.
- Smaller α distributes the uncertainty over more (and eventually, all) possibles values of Y .

Therefore

We have that

- The loss function pays an infinite cost by ignoring the training data distribution completely.

Note the following

- α quantifies the level of certainty placed on the posterior distribution

Therefore

- Larger α indicate increasing certainty over a smaller set of Y.
- Smaller α distributes the uncertainty over more (and eventually, all) possibles values of Y .

Therefore

We have that

- The loss function pays an infinite cost by ignoring the training data distribution completely.

Note the following

- α quantifies the level of certainty placed on the posterior distribution

Therefore

- Larger α indicate increasing certainty over a smaller set of Y.
- Smaller α distributes the uncertainty over more (and eventually, all) possibles values of Y .

Actually

For $\alpha = \infty$

- The distribution becomes the hard-decoding Maximum A Posteriori rule.

Risk Minimization under this loss

Proposition

- For each $\alpha \in (0, \infty]$, the minimal α-risk is

$$\min_{P_{\widehat{Y}|X}} \mathbb{E}_{X,Y}\left[l^{\alpha}\left(Y, P_{\widehat{Y}|X}\right)\right] = \frac{\alpha}{\alpha - 1}\left[1 - \exp\left\{\frac{1-\alpha}{\alpha} H_{\alpha}^{A}\left(Y|X\right)\right\}\right]$$

where $H_{\alpha}^{A}\left(Y|X\right) = \frac{\alpha}{1-\alpha}\log\sum_{y}\left(\sum_{x} P_{X,Y}\left(x,y\right)^{\alpha}\right)^{1/\alpha}$ is the Arimoto conditional entropy of order α. The resultin minimizer is the α-tilted true posterior

$$P^{*}_{\widehat{Y}|X}\left(y|x\right) = \frac{P_{Y|X}\left(y|x\right)^{\alpha}}{\sum_{y} P_{Y|X}\left(y|x\right)^{\alpha}}$$

Take a look at [9]

- For the proof

Risk Minimization under this loss

Proposition

- For each $\alpha \in (0, \infty]$, the minimal α-risk is

$$\min_{P_{\widehat{Y}|X}} \mathbb{E}_{X,Y}\left[l^{\alpha}\left(Y, P_{\widehat{Y}|X}\right)\right] = \frac{\alpha}{\alpha - 1}\left[1 - \exp\left\{\frac{1-\alpha}{\alpha} H_{\alpha}^{A}\left(Y|X\right)\right\}\right]$$

where $H_{\alpha}^{A}\left(Y|X\right) = \frac{\alpha}{1-\alpha}\log\sum_{y}\left(\sum_{x} P_{X,Y}\left(x,y\right)^{\alpha}\right)^{1/\alpha}$ is the Arimoto conditional entropy of order α. The resultin minimizer is the α-tilted true posterior

$$P_{\widehat{Y}|X}^{*}\left(y|x\right) = \frac{P_{Y|X}\left(y|x\right)^{\alpha}}{\sum_{y} P_{Y|X}\left(y|x\right)^{\alpha}}$$

Take a look at [9]

- For the proof

Outline

Examples

Differentially Private Empirical Risk Minimization with Smooth Non-Convex Loss Functions: A Non-Stationary View [10]

- Here, the Differentially Private Empirical Risk Minimization is studied

From Convex to Nonconvex: a Loss Function Analysis for Binary Classification [11]

- A new smoothed version of the loss 0-1 function is proposed
  - Although, it seems to be that sigmoid cross entropy is better...
- An new method to compare different loss functions

Deep Neural Networks with Multi-Branch Architectures Are Intrinsically Less Non-Convex [12]

- Architectures using subnetworks as the transformers are non-convex in nature

Examples

Differentially Private Empirical Risk Minimization with Smooth Non-Convex Loss Functions: A Non-Stationary View [10]

- Here, the Differentially Private Empirical Risk Minimization is studied

From Convex to Nonconvex: a Loss Function Analysis for Binary Classification [11]

- A new smoothed version of the loss 0-1 function is proposed
  - Although, it seems to be that sigmoid cross entropy is better...
- An new method to compare different loss functions

Deep Neural Networks with Multi-Branch Architectures Are Intrinsically Less Non-Convex [12]

- Architectures using subnetworks as the transformers are non-convex in nature

Examples

Differentially Private Empirical Risk Minimization with Smooth Non-Convex Loss Functions: A Non-Stationary View [10]

- Here, the Differentially Private Empirical Risk Minimization is studied

From Convex to Nonconvex: a Loss Function Analysis for Binary Classification [11]

- A new smoothed version of the loss 0-1 function is proposed
  - Although, it seems to be that sigmoid cross entropy is better...
- An new method to compare different loss functions

Deep Neural Networks with Multi-Branch Architectures Are Intrinsically Less Non-Convex [12]

- Architectures using subnetworks as the transformers are non-convex in nature

Outline

It is clear that many connections need to be done

From the Reproducing Kernels

- As Layers on the Neuronal Networks
  - Still a Deeper study needs to be done to finish the connections on this regard...

To the need to explore novel non-convex loss functions

- Making possible to improve upon the traditional loss functions for Neural Networks

Therefore

- This is a new frontier in the study of neural networks...

It is clear that many connections need to be done

From the Reproducing Kernels

- As Layers on the Neuronal Networks
  - Still a Deeper study needs to be done to finish the connections on this regard...

To the need to explore novel non-convex loss functions

- Making possible to improve upon the traditional loss functions for Neural Networks

Therefore

- This is a new frontier in the study of neural networks...

It is clear that many connections need to be done

From the Reproducing Kernels

- As Layers on the Neuronal Networks
  - Still a Deeper study needs to be done to finish the connections on this regard...

To the need to explore novel non-convex loss functions

- Making possible to improve upon the traditional loss functions for Neural Networks

Therefore

- This is a new frontier in the study of neural networks...

L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?," *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, 1995.

N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International conference on computational learning theory*, pp. 416–426, Springer, 2001.

K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

Y. Lecun, "Who is afraid of nonconvex loss functions?."

T. Sypherd, M. Diaz, H. Laddha, L. Sankar, P. Kairouz, and G. Dasarathy, "A class of parameterized loss functions for classification: Optimization tradeoffs and robustness characteristics," *arXiv preprint arXiv:1906.02314*, 2019.

J. Liao, O. Kosut, L. Sankar, and F. P. Calmon, "A tunable measure for information leakage," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 701–705, IEEE, 2018.

D. Wang and J. Xu, "Differentially private empirical risk minimization with smooth non-convex loss functions: A non-stationary view," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1182–1189, 2019.

L. Zhao, M. Mammadov, and J. Yearwood, "From convex to nonconvex: a loss function analysis for binary classification," in *2010 IEEE International Conference on Data Mining Workshops*, pp. 1281–1288, IEEE, 2010.

H. Zhang, J. Shao, and R. Salakhutdinov, "Deep neural networks with multi-branch architectures are intrinsically less non-convex," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1099–1109, 2019.