What is Machine Learning?

- Machine Learning (ML): branch of artificial intelligence (AI) with goals to extract structures from large data sets
- Applications: $\underline{\text{identifying patterns}}$ and $\underline{\text{making decisions}}$, image recognition, fraud detection, extracting meaning from text, ...
- Specifically, Face ID, Google Translate, self-driving cars, cancer diagnosis, product recommendations, ChatGPT, ...

- Recently, Large Language Models and Artificial General Intelligence have become hot issues.
- Clearly, AI and Machine Learning are making revolutions in many areas.
- Machine Learning brings together ideas from

  Computer Science, Statistics, and Mathematics.
- In this course, we will discuss some of the essential mathematical ideas in Machine Learning and practice computational techniques using Python.

Types of Machine Learning

- Regression:

  predict numerical values

  example: the value of a house

- Classification:

  predict categorical labels

  example: hand-written digits

Types of Machine Learning

- Supervised learning:

  train a machine with known classification labels

- Unsupervised learning:

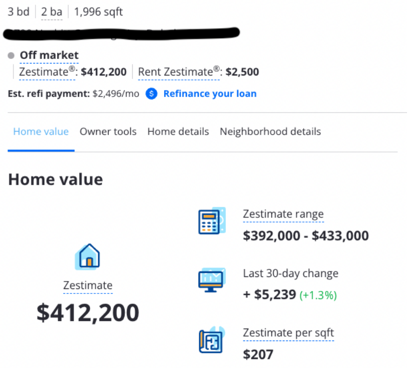
  recognize patterns or clusters from an unlabelled dataset

- Reinforcement learning:

  enable an agent to learn in an environment by trial and error

Typical ML problems

- Estimate the value of a house


3 bd | 2 ba | 1,996 sqft

[redacted]

Off market

Zestimate®: $412,200 | Rent Zestimate®: $2,500

Est. refi payment: $2,496/mo Refinance your loan

Home value

Zestimate
$412,200

Zestimate range
$392,000 - $433,000

Last 30-day change
+ $5,239 (+1.3%)

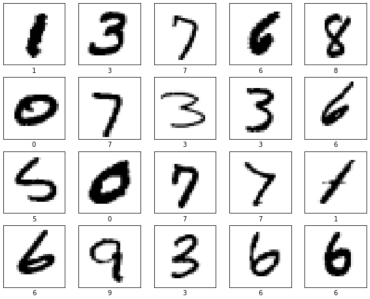
Zestimate per sqft
$207

$$\mathcal{F}: \text{(\# of bedrooms, square footage, year built, location, ...)}$$

$$\longrightarrow \text{(the value of a house)}$$

Typical ML problems

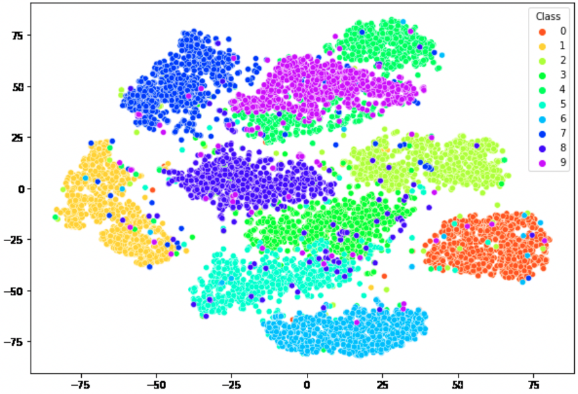
- Recognizing hand-written digits



$$\mathcal{C} : \{ \text{ pictures } \} \longrightarrow \{0, 1, \ldots, 9\}$$

Typical ML problems

- Clustering the data points of hand-written digits



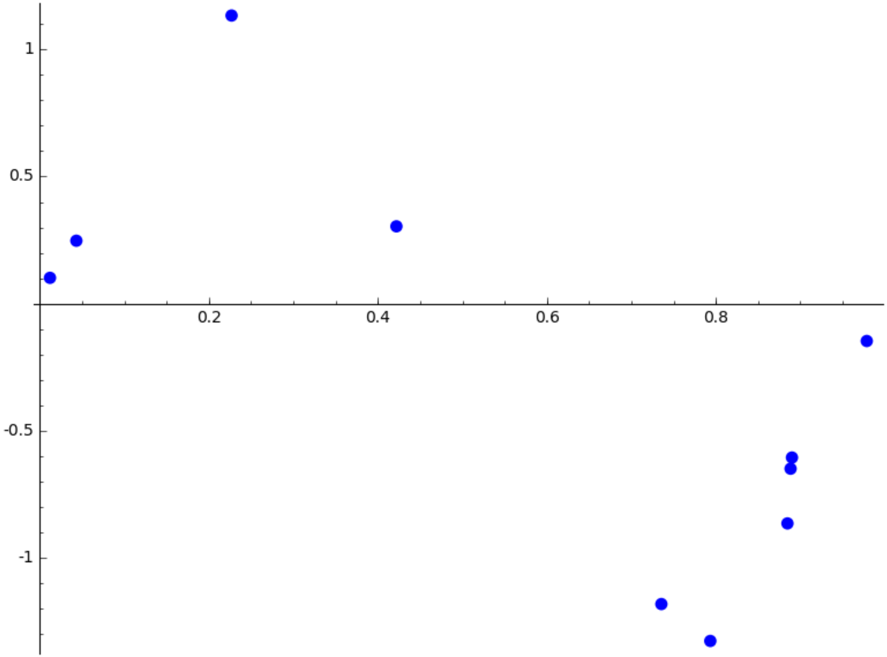
TSNE visualization of MNIST data set encodings (VAE encoder). 10000 examples are shown.

Linear Regression – part (1)

- Input: $x \in \mathbb{R}$ Output: $t \in \mathbb{R}$
  Observations: $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$
- Use these observations as training examples.
  Task: $\boxed{\text{Given a new input } \tilde{x}\text{, predict the output } \tilde{t}.}$
- If t is a continuous variable, this task is called regression.
  It is an example of supervised learning.

- For example, $x \in [0,1)$, $N = 10$

x	t
0.884644066199	−0.864791215635069
0.793349886821	−1.32738612014193
0.735440841558	−1.18222466237236
0.421871764847	0.304255805886633
0.0118832729931	0.101594120287724
0.226770188973	1.13377458999431
0.978530671629	−0.147028527196347
0.0431076970157	0.247622971933151
0.890003286931	−0.605625802202937
0.888362799625	−0.649537521948140


1
0.5
-0.5
-1
0.2
0.4
0.6
0.8

- Fit the data using a polynomial

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K,$$

where $\boldsymbol{w} = [w_0, w_1, \ldots, w_K]^\top$.

- How to obatain the best approximation?

- Want: Minimize the distance between

$$(t_1, t_2, \ldots, t_N) \qquad \text{and} \qquad (y_1, y_2, \ldots, y_N),$$

where $y_n = y(x_n, \boldsymbol{w})$.

- Minimize the error function

$$
\begin{aligned}
E(\boldsymbol{w}) &= \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2 \\
&= \sum_{n=1}^{N} \{w_0 + w_1 x_n + w_2 x_n^2 + \cdots + w_K x_n^K - t_n\}^2.
\end{aligned}
$$

- Introduce the following matrices

$$X = \begin{bmatrix} 1 & x_1 & \cdots & x_1^K \\ 1 & x_2 & \cdots & x_2^K \\ 1 & x_3 & \cdots & x_3^K \\ \vdots & \vdots & & \vdots \\ 1 & x_N & \cdots & x_N^K \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix}, \quad \text{and } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}.$$

  Then

$$E(\boldsymbol{w}) = \|X\boldsymbol{w} - \mathbf{t}\|^2.$$

- The matrix X can be considered as a data matrix.
  Tidy Convention:
  - Each row is an observation.
  - Each column is a feature.

- How can we find a minimum of a function?
- We take the “derivative”.
- $E(\boldsymbol{w})$ is a multi-variable function.

K features, and let $M = K + 1$.

- Let $f : \mathbb{R}^M \to \mathbb{R}$ be a differentiable function. The gradient of f at $\boldsymbol{x} \in \mathbb{R}^M$ is defined by

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{x}) \\ \frac{\partial f}{\partial x_2}(\boldsymbol{x}) \\ \vdots \\ \frac{\partial f}{\partial x_M}(\boldsymbol{x}) \end{bmatrix}.$$

Lemma

Assume that A is an $N \times M$ matrix and $\boldsymbol{x} \in \mathbb{R}^M$, $\boldsymbol{b} \in \mathbb{R}^N$. Then we have

$$\nabla \|A\boldsymbol{x} + \boldsymbol{b}\|^2 = 2A^\top (A\boldsymbol{x} + \boldsymbol{b}).$$

Proof. $A = [a_{ij}], \boldsymbol{x} = [x_j]_{j=1,\dots,M}, \boldsymbol{b} = [b_i]_{i=1,\dots,N}$

$$\|A\boldsymbol{x} + \boldsymbol{b}\|^2 = \|[\sum_j a_{ij}x_j + b_i]_{i=1,\dots,N}\|^2 = \sum_i (\sum_j a_{ij}x_j + b_i)^2$$

$$\left[\frac{\partial \|A\boldsymbol{x} + \boldsymbol{b}\|^2}{\partial x_k}\right]_{k=1,\dots,N} = \left[\sum_i 2(\sum_j a_{ij}x_j + b_i)a_{ik}\right]_{k=1,\dots,N}$$

$$\begin{aligned} 2A^\top(A\boldsymbol{x} + \boldsymbol{b}) &= 2[a_{ij}]^\top[\sum_j a_{ij}x_j + b_i]_{i=1,\dots,N} \\ &= 2[\sum_i a_{ik}(\sum_j a_{ij}x_j + b_i)]_{k=1,\dots,N} \end{aligned}$$

$\square$

Theorem (Fermat's Theorem)

Let $f : \mathbb{R}^M \to \mathbb{R}$ be differentiable. If f has a local extremum at $\boldsymbol{x}$, then $\nabla f(\boldsymbol{x}) = 0$.

- Is the converse of Fermat's Theorem true?
- No, in general.

- A set $\mathcal{X} \in \mathbb{R}^M$ is said to be convex if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,

$$\{\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y} : 0 \le \alpha \le 1\} \subseteq \mathcal{X}.$$

- Let $\mathcal{X}$ be a convex set. A function $f : \mathcal{X} \to \mathbb{R}$ is said to be convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and $\alpha \in [0, 1]$,

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \le \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y}).$$

Theorem

Assume $\mathcal{X}$ is convex. Then a differentiable function $f : \mathcal{X} \to \mathbb{R}$ is convex if and only if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \nabla f(\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x}) = \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}).$$

Corollary

Assume $\mathcal{X}$ is convex. Then a convex differentiable function $f : \mathcal{X} \to \mathbb{R}$ has a global minimum at $\boldsymbol{x}$ if and only if $\nabla f(\boldsymbol{x}) = 0$.

Proof.

($\Rightarrow$) Fermat's Theorem

($\Leftarrow$) Theorem in the above $\square$

X: $N \times M$ data matrix $\qquad \mathbf{t} \in \mathbb{R}^N$

Proposition

The function $E(\boldsymbol{w}) = \|X\boldsymbol{w} - \mathbf{t}\|^2$ is convex and differentiable.

Proof.

- $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (triangle inequality)
- x^2 is convex.

Now we have

$$\begin{aligned} E(\alpha\boldsymbol{w} + (1-\alpha)\boldsymbol{v}) &= \|X(\alpha\boldsymbol{w} + (1-\alpha)\boldsymbol{v}) - \mathbf{t}\|^2 \\ &= \|\alpha(X\boldsymbol{w} - \mathbf{t}) + (1-\alpha)(X\boldsymbol{v} - \mathbf{t})\|^2 \\ &\leq (\alpha\|X\boldsymbol{w} - \mathbf{t}\| + (1-\alpha)\|X\boldsymbol{v} - \mathbf{t}\|)^2 \\ &\leq \alpha E(\boldsymbol{w}) + (1-\alpha)E(\boldsymbol{v}). \end{aligned}$$

□

- Since $\nabla E(\boldsymbol{w}) = 2X^\top(X\boldsymbol{w} - \mathbf{t})$, the convex function $E(\boldsymbol{w})$ has a global minimum if and only if

$$X^\top X\boldsymbol{w} = X^\top\mathbf{t}.$$

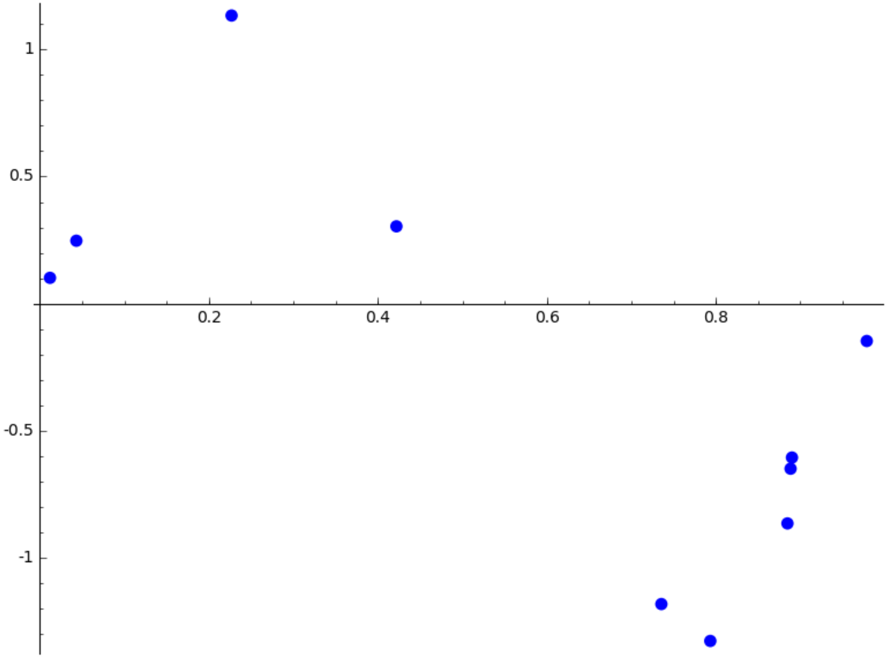
- When $S := X^\top X$ is invertible, there is a unique solution

$$\boxed{\boldsymbol{w} = S^{-1}X^\top\mathbf{t}}.$$

- S is not invertible if and only if $\det S = 0$.
  When $\det S \approx 0$, the formula for $\boldsymbol{w}$ is unstable.
- If $\det S = 0$, what can we say about the features?

- Back to the example, $x \in [0, 1)$, $N = 10$

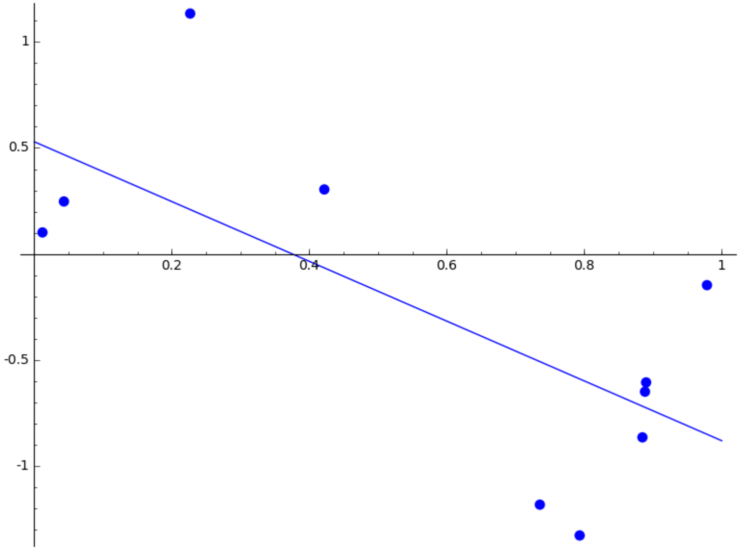
x	t
0.884644066199	−0.864791215635069
0.793349886821	−1.32738612014193
0.735440841558	−1.18222466237236
0.421871764847	0.304255805886633
0.0118832729931	0.101594120287724
0.226770188973	1.13377458999431
0.978530671629	−0.147028527196347
0.0431076970157	0.247622971933151
0.890003286931	−0.605625802202937
0.888362799625	−0.649537521948140


1
0.5
-0.5
-1
0.2
0.4
0.6
0.8

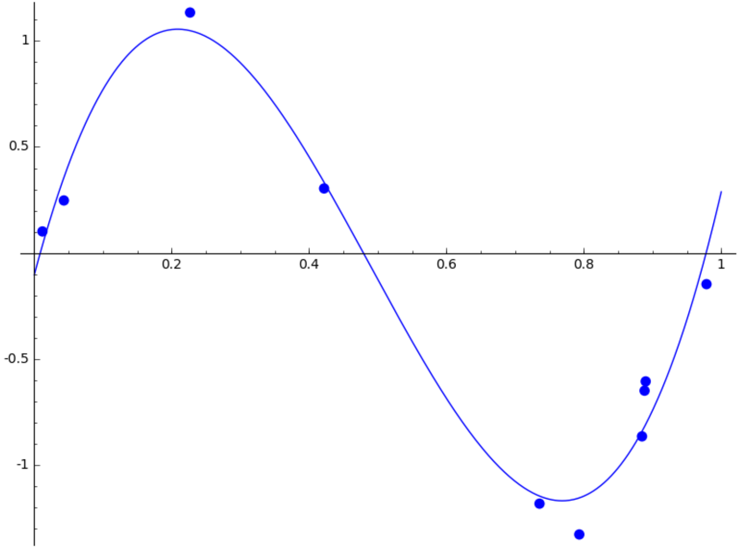
Using the formula, we obtain the following.

$\boldsymbol{w}$	$K = 1$	$K = 3$	$K = 6$	$K = 9$
w_0	0.53	-0.10	0.03	-1.61
w_1	-1.41	12.21	4.49	200.18
w_2		-37.18	34.15	-5191.34
w_3		25.37	-211.33	41628.04
w_4			339.78	-141666.70
w_5			-210.56	210688.85
w_6			43.45	-61808.88
w_7				-189181.01
w_8				214844.27
w_9				-69519.92

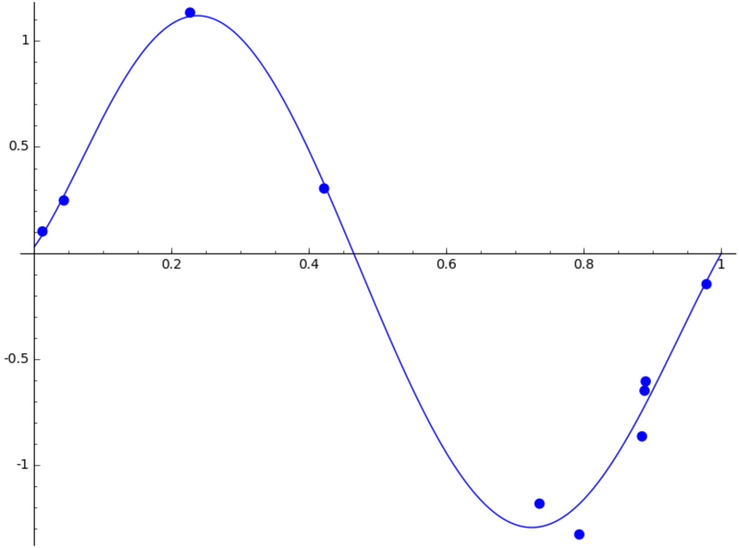
$K = 1$



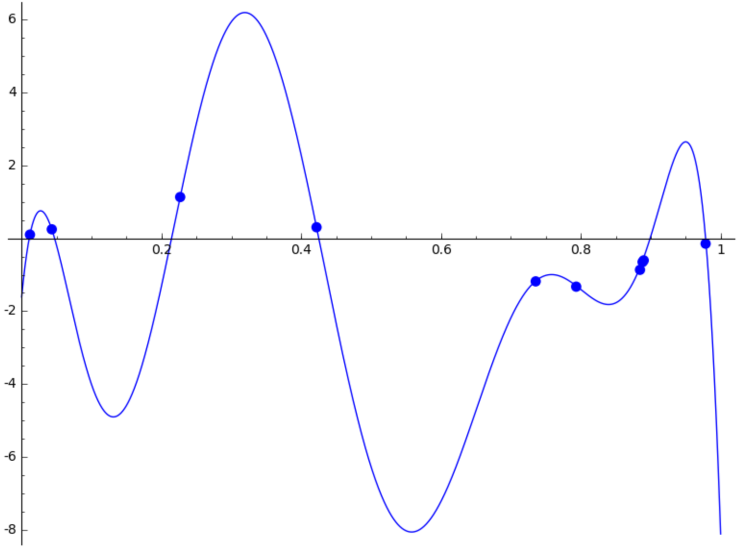
$K = 3$



$K = 6$



$K = 9$



- If we choose $K = 6$ as our model, then we get

$$y(x) = 0.03 + 4.49x + 34.15x^2 - 211.33x^3 \\ + 339.78x^4 - 201.78x^5 + 43.45x^6.$$

- For a new input $x = 0.741234$, the prediction is

$$y(0.741234) = -1.28212.$$

- This is how machine learning works in this example.

- The case $K = 9$ is over-fitting.

  It is a perfect fit. However, it is useless in machine learning.

- In order to avoid over-fitting, we can use regularization.

  Observe that when $K = 9$ the coordinates of $\boldsymbol{w}$ tend to be large.

- Ridge regression

$$\widetilde{E}(\boldsymbol{w}) = \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2 + \lambda \|\boldsymbol{w}\|^2.$$

  This can be considered as a result of Bayesian Learning.

- Lasso regression

$$\widehat{E}(\boldsymbol{w}) = \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2 + \lambda \sum_{n=0}^{K} |w_n|.$$

- For other situations, we can take different basis functions $\phi_n(x)$ instead of power functions x^n, and consider

$$y(x, \boldsymbol{w}) = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_k\phi_K(x).$$

- For example,

$$\phi_n(x) = \exp\left(-\frac{(x-\mu_n)^2}{2\sigma^2}\right).$$

- Minimize the error function

$$E(\boldsymbol{w}) = \sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2.$$

- We have

$$X = \begin{bmatrix} 1 & \phi_1(x_1) & \cdots & \phi_K(x_1) \\ 1 & \phi_1(x_2) & \cdots & \phi_K(x_2) \\ 1 & \phi_1(x_3) & \cdots & \phi_K(x_3) \\ \vdots & \vdots & & \vdots \\ 1 & \phi_1(x_N) & \cdots & \phi_K(x_N) \end{bmatrix}.$$

- We obtain the solution

$$\boxed{\boldsymbol{w} = S^{-1} X^\top \mathbf{t}},$$

where $S = X^\top X$.

Linear Regression – part (2)

- Input: $(x_1, x_2 \ldots, x_K) \in \mathbb{R}^K$ Output: $t \in \mathbb{R}$

  Observations: $(x_{11}, x_{12}, \ldots, x_{1K}; t_1), \ldots, (x_{N1}, x_{N2}, \ldots, x_{NK}; t_N)$

- Use these observations as training examples.

  Task: $\boxed{\text{Given a new input } (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_K)\text{, predict the output } \tilde{t}.}$

- Fit the data using a linear function

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K,$$

  where $\boldsymbol{w} = [w_0, w_1, \dots, w_K]^\top$.
- w_0 is called a bias.

- Want: Minimize the distance between

$$(t_1, t_2, \ldots, t_N) \qquad \text{and} \qquad (y_1, y_2, \ldots, y_N),$$

where $y_n = y(\boldsymbol{x}_n, \boldsymbol{w})$, $n = 1, 2, \ldots, N$.

- Minimize the error function

$$\begin{aligned} E(\boldsymbol{w}) &= \sum_{n=1}^{N} \{y_n - t_n\}^2 = \sum_{n=1}^{N} \{y(\boldsymbol{x}_n, \boldsymbol{w}) - t_n\}^2 \\ &= \sum_{n=1}^{N} \{w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK} - t_n\}^2. \end{aligned}$$

- Introduce the following matrices

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1K} \\ 1 & x_{21} & \cdots & x_{2K} \\ 1 & x_{31} & \cdots & x_{3K} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N1} & \cdots & x_{NK} \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix}, \quad \text{and } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}.$$

  Then

$$E(\boldsymbol{w}) = \|X\boldsymbol{w} - \mathbf{t}\|^2.$$

- The matrix X is the data matrix.

- Since $\nabla E(\boldsymbol{w}) = 2X^\top(X\boldsymbol{w} - \mathbf{t})$, the convex function $E(\boldsymbol{w})$ has a global minimum if and only if

$$X^\top X\boldsymbol{w} = X^\top \mathbf{t}.$$

- When $S := X^\top X$ is invertible, there is a unique solution

$$\boxed{\boldsymbol{w} = S^{-1}X^\top \mathbf{t}}.$$

- When $(\tilde{x}_1, \dots, \tilde{x}_K)$ is a new input, the prediction is given by

$$\tilde{t} = w_0 + w_1\tilde{x}_1 + \cdots + w_K\tilde{x}_K = [1, \tilde{x}_1, \dots, \tilde{x}_K]\boldsymbol{w}.$$

Geometry of Linear Regression

- $\tilde{\mathbf{t}} := X\boldsymbol{w}$ is the best approximation (or the closest point) to $\mathbf{t} \in \mathbb{R}^N$.
- $\tilde{\mathbf{t}}$ is in the column space of X.
- $\tilde{\mathbf{t}} - \mathbf{t}$ is orthogonal to the column space of X.

  ($\because \nabla E(\boldsymbol{w}) = 2X^\top(X\boldsymbol{w} - \mathbf{t}) = 0$ if and only if $X^\top(\tilde{\mathbf{t}} - \mathbf{t}) = 0$

  $$\text{if and only if} \quad \mathbf{c}_n \cdot (\tilde{\mathbf{t}} - \mathbf{t}) = 0 \text{ for all } n = 0, 1, \ldots, K,$$

  where $\mathbf{c}_n$ are the column vectors of X.)
- $\tilde{\mathbf{t}}$ is the orthogonal projection of $\mathbf{t}$ on to the column space of X.
- colunm space = feature space

Proposition

The matrix $S := X^\top X$ is not invertible if and only if the columns of X are linearly dependent.

Proof.

- $\Rightarrow$) There exists $\boldsymbol{w} \neq 0$ such that $S\boldsymbol{w} = X^\top X\boldsymbol{w} = 0$. Then $X\boldsymbol{w}$ is orthogonal to the column space of X. However, $X\boldsymbol{w}$ is actually in the column space of X. Therefore $X\boldsymbol{w} = 0$, and the columns of X are linearly dependent.
- $\Leftarrow$) There exists $\boldsymbol{w} \neq 0$ such that $X\boldsymbol{w} = 0$. Then $S\boldsymbol{w} = X^\top X\boldsymbol{w} = 0$, implying that S is not invertible.

Binary Classification

- Given a dataset $\mathcal{D}$, we like to construct a function
$$f : \mathcal{D} \to \{0, 1\},$$
where 0 = (category 0) and 1 = (category 1).

- Examples
  1. $\mathcal{D} = \{\text{SAT scores}\}$, 0 = rejection, 1 = admission
  2. $\mathcal{D} = \{\text{emails}\}$, 0 = non-spam, 1 = spam

- More precisely, we will construct a function

$$f : \mathcal{D} \to [0, 1]$$

so that $f(d)$, $d \in \mathcal{D}$, represents the **probability** that d belongs to (category 1).

- In the SAT scores example,

$$f(1350) = 0.732$$

will mean "A student with SAT score 1350 is accepted with probability 0.732."

Q: How can we construct such a function? Logistic Regression

- In linear regression, we use a linear model and minimize the mean square error (MSE).
- In logistic regression, our strategy will be similar to that of linear regression, and the method is called **maximum likelihood estimation** (MLE).

A dataset $\mathcal{D}$ is given with known classification.

- Step 1: Using a probabilistic model, write a function out of $\mathcal{T}$ with unknown *parameters* or *weights*.
- Step 2: Determine the parameters so that the known classification may have the maximum likelihood.

It is customary to take the negative log of the likelihood function, and the resulting function is called the cross-entropy. Then we need to *minimize* the cross-entropy.

$$\begin{aligned} \text{Learning} &\iff \text{Maximizing Certainty} \\ &\iff \text{Minimizing Randomness (Entropy)} \end{aligned}$$

- $\mathcal{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ with known classification $\{t_1, \dots, t_N\}$
- $y_n = f(\boldsymbol{x}_n, \boldsymbol{w})$: probability that $\boldsymbol{x}_n$ belongs to class 1

  Then $1 - y_n$ is the probability that $\boldsymbol{x}_n$ belongs to class 0.
- Apply this to $\{t_1, \dots, t_N\}$ to compute the total probability.
- For example,

t_n	1	0	1	1	0
p_n	y_1	$1 - y_2$	y_3	y_4	$1 - y_5$

  Assuming that data points are independent, we obtain

$$\text{likelihood} = y_1(1 - y_2)y_3y_4(1 - y_5).$$

- We can write
$$\text{likelihood} = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n},$$
where $y_n = f(\boldsymbol{x}_n, \boldsymbol{w})$.

- Want: Determine $\boldsymbol{w}$ that maximizes the likelihood.
- Equivalently, minimize the cross-entropy
$$E(\boldsymbol{w}) := -\log(\text{likelihood}).$$

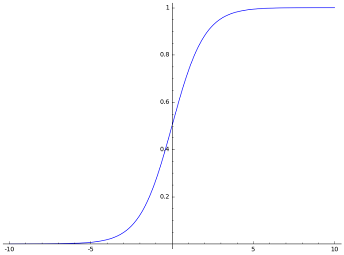
- In logistic regression, we take

$$y_n = f(\boldsymbol{x}_n, \boldsymbol{w}) = \sigma(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK}),$$

where $\boldsymbol{x}_n = (x_{n1}, \dots, x_{nK})$, $\boldsymbol{w} = (w_0, w_1, \dots, w_K)$ and

$$\sigma(x) := \frac{1}{1+e^{-x}}.$$

- The function $\sigma(x)$ is called the sigmoid function.



The cross-entropy function is given by

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\},$$

where

$$y_n = \sigma(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK})$$

and

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

<u>Goal</u>: Determine $\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_K)$ which minimizes $E(\boldsymbol{w})$.

- Compute the gradient $\nabla E(\boldsymbol{w})$.
- Crucial identity:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Thus it is a solution to

$$y' = y(1 - y),$$

which is called a logistic equation.

$$(\ln y)' = \frac{1}{y} y' = 1 - y.$$

$$(\ln(1 - y))' = \frac{1}{1 - y}(-y') = -y.$$

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\},$$
$$y_n = \sigma(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK}).$$

Write $x_n = w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK}$ and $y_n = \sigma(x_n)$.

$$\begin{aligned}
\frac{\partial E}{\partial w_j} &= -\sum \frac{\partial}{\partial w_j}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \\
&= -\sum \{t_n \frac{\partial}{\partial x_n} \ln y_n \cdot \frac{\partial x_n}{\partial w_j} + (1-t_n)\frac{\partial}{\partial x_n}\ln(1-y_n) \cdot \frac{\partial x_n}{\partial w_j}\} \\
&= -\sum\{t_n(1-y_n)x_{nj} + (1-t_n)(-y_n)x_{nj}\} \\
&= \sum (y_n - t_n)x_{nj}
\end{aligned}$$

$$\boxed{\frac{\partial E}{\partial w_j} = \sum_{n=1}^{N}(y_n - t_n)x_{nj}}.$$

- We have

$$\nabla E(\boldsymbol{w}) = \left[\sum_{n=1}^{N}(y_n - t_n)x_{nj}\right] = X^\top(\boldsymbol{y} - \mathbf{t}),$$

where

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1K} \\ 1 & x_{21} & \cdots & x_{2K} \\ \vdots & & \vdots & \vdots \\ 1 & x_{N1} & \cdots & x_{NK} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \text{and } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}.$$

$$\boxed{\nabla E(\boldsymbol{w}) = X^\top(\boldsymbol{y} - \mathbf{t})}$$

- Assume $\nabla E = 0$.

  Can we determine $\boldsymbol{w}$ explicitly?

  When $j = 0$, we have $x_{n,0} = 1$ for all n, and thus

$$\sum_{n=1}^{N} y_n = \sum_{n=1}^{N} t_n.$$

Therefore, we have

$$\begin{aligned}\sum_{n=1}^{N} y_n &= \sum_{n=1}^{N} \frac{1}{1+\exp(-(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK}))} \\ &= \sum_{n=1}^{N} t_n.\end{aligned}$$

We have more complicated equations for $j \neq 0$. Clearly, it is not feasible to write $w_0, w_1, \ldots, w_K$ explicitly.

- The cross-entropy $E(\boldsymbol{w})$ is not linear, and it is not possible to calculate a closed-form formula for $\boldsymbol{w}_*$ which minimizes $E(\boldsymbol{w})$.
- On the other hand, it can be shown that $E(\boldsymbol{w})$ is convex, and a global minimum exists.
- We will find an approximate value for $\boldsymbol{w}_*$ using gradient descent and Newton's method.
- The method of gradient descent is widely used in many other parts of machine learning.

Gradient Descent

Consider a function $E : \mathbb{R}^M \to \mathbb{R}$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_M) \mapsto E(\boldsymbol{w})$. The gradient ∇E of E is defined by

$$\nabla E := \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_M} \right).$$

Proposition

$E(\boldsymbol{w})$: differentiable in a nbhd of $\boldsymbol{w}$

The function $E(\boldsymbol{w})$ decreases fastest in the direction of $-\nabla E(\boldsymbol{w})$.

Proof: For a unit vector $\boldsymbol{u}$, the directional derivative $D_{\boldsymbol{u}}E$ is given by

$$D_{\boldsymbol{u}}E = \lim_{t\to 0} \frac{E(\boldsymbol{w}_0 + t\boldsymbol{u}) - E(\boldsymbol{w}_0)}{t} = g'(0),$$

where $g(t) = E(\boldsymbol{w}_0 + t\boldsymbol{u})$. Let $\boldsymbol{w} = \boldsymbol{w}_0 + t\boldsymbol{u}$. Using the chain rule,

$$g'(0) = \sum_{i=1}^{m} \frac{\partial E}{\partial w_i} \cdot \frac{dw_i}{dt} = \sum_{i=1}^{m} \frac{\partial E}{\partial w_i} \cdot \frac{du_i}{dt} = \nabla E \cdot \boldsymbol{u}.$$

Furthermore,

$$D_{\boldsymbol{u}}E = \nabla E \cdot \boldsymbol{u} = |\nabla E|\,|\boldsymbol{u}| \cos\theta = |\nabla E| \cos\theta,$$

where θ is the angle between ∇E and $\boldsymbol{u}$. The minimum value of $D_{\boldsymbol{u}}E$ occurs when $\cos\theta$ is -1. $\square$

- Choose an initial point $\boldsymbol{w}_0$.
- Set

$$\boxed{\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta_k \nabla E(\boldsymbol{w}_k)}$$

  where η_k is the step size or learning rate.

  Usually, for some $\eta > 0$, we set $\eta_k = \eta$ for all k.
- From Proposition, we have

$$E(\boldsymbol{w}_k) \geq E(\boldsymbol{w}_{k+1}).$$

- Under some moderate conditions,

$$E(\boldsymbol{w}_k) \to \text{local minimum} \qquad \text{as } \ k \to \infty.$$

  In particular, this is true when E is convex.

  - E is convex $\iff$

    for all $0 \le \alpha \le 1$ and all $x_1, x_2 \in \mathbb{R}^n$, we have

$$E(\alpha x_1 + (1-\alpha)x_2) \le \alpha\, E(x_1) + (1-\alpha)E(x_2).$$

- $\boldsymbol{w}_* = \lim_{k\to\infty} \boldsymbol{w}_k$

  We use $\boldsymbol{w}_k$ for a sufficiently large k as an approximation of $\boldsymbol{w}_*$.

  This method is called gradient descent.

  Caveat: Making a right choice of η is crucial.

Example

- Consider $E(\boldsymbol{w}) = E(w_1, w_2) = w_1^4 + w_2^4 - 16w_1w_2$.
  Then $\nabla E(\boldsymbol{w}) = [4w_1^3 - 16w_2, 4w_2^3 - 16w_1]$.
  Choose $\boldsymbol{w}_0 = (1, 1)$ and $\eta = 0.01$.
  $\boldsymbol{w}_{30} = (1.99995558586289, 1.99995558586289)$
  $E(\boldsymbol{w}_{30}) = -31.9999999368777$

- We see that $\boldsymbol{w}_k \to (2, 2)$ and $E(2, 2) = -32$.

- Indeed, when $\boldsymbol{w} = (2, 2)$, a local minimum of $E(\boldsymbol{w})$ is -32.

k	w1	w2	E(w1,w2)
1	1.12000000000000	1.12000000000000	-16.9233612800000
2	1.24300288000000	1.24300288000000	-19.9465014818312
3	1.36506297054983	1.36506297054983	-22.8698545020842
4	1.48172688079195	1.48172688079195	-25.4876645161458
5	1.58867706472624	1.58867706472624	-27.6422269714610
6	1.68247924276483	1.68247924276483	-29.2656452783487
7	1.76116971206054	1.76116971206054	-30.3861816086105
8	1.82445074094736	1.82445074094736	-31.0984991504577
9	1.87344669354831	1.87344669354831	-31.5194128485897
10	1.91018104795404	1.91018104795404	-31.7533053700606
11	1.93701591038872	1.93701591038872	-31.8770223901250
12	1.95622873443784	1.95622873443784	-31.9400248989010
13	1.96977907222858	1.96977907222858	-31.9712142030755
14	1.97923168007769	1.97923168007769	-31.9863406140263
15	1.98577438322011	1.98577438322011	-31.9935701975589
16	1.99027812738069	1.99027812738069	-31.9969902100773
17	1.99336647981957	1.99336647981957	-31.9985965516271
18	1.99547865709166	1.99547865709166	-31.9993473166738
19	1.99692058430943	1.99692058430943	-31.9996970174121
20	1.99790372262623	1.99790372262623	-31.9998595272283
21	1.99857347710339	1.99857347710339	-31.9999349274762
22	1.99902947615421	1.99902947615421	-31.9999698732955
23	1.99933981776146	1.99933981776146	-31.9999860577045
24	1.99955097148756	1.99955097148756	-31.9999935493971
25	1.99969461222478	1.99969461222478	-31.9999970160815
26	1.99979231393118	1.99979231393118	-31.9999986198712
27	1.99985876312152	1.99985876312152	-31.9999993617137
28	1.99990395413526	1.99990395413526	-31.9999997048203
29	1.99993468659806	1.99993468659806	-31.9999998634976
30	1.99995558586289	1.99995558586289	-31.9999999368777

Back to Binary Classification

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\},$$

$$y_n = \sigma(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK}).$$

$$\boxed{\nabla E(\boldsymbol{w}) = X^\top (\boldsymbol{y} - \mathbf{t})}$$

- We will perfom gradient descent using this.

Newton's Method

- Second-order approximation

  Much faster in convergence, more expensive (and more subtle)

- $f(x)$: single-variable, convex, differentiable function

  Find a local minimum

  $$\iff \qquad \text{Find } x_* \text{ such that } f'(x_*) = 0$$

  Choose an intial point x_0 and set $x = x_0 + h$.

- Using Taylor's expansion,

$$
\begin{aligned}
f(x) &= f(x_0+h) \approx f(x_0) + f'(x_0)h + \tfrac{1}{2}f''(x_0)h^2 \\
f'(x) &\approx \frac{d}{dh}\left(f(x_0) + f'(x_0)h + \tfrac{1}{2}f''(x_0)h^2\right)\frac{dh}{dx} \\
&= f'(x_0) + f''(x_0)h
\end{aligned}
$$

From $f'(x) = 0$, we approximately obtain

$$0 = f'(x_0) + f''(x_0)h, \qquad h = -f'(x_0)/f''(x_0).$$

- We have shown that

$$x_1 = x_0 - f'(x_0)/f''(x_0)$$

is an approximation of x_*.

- Repeat the process to obtain

$$\boxed{x_{k+1} = x_k - f'(x_k)/f''(x_k)},$$

and $x_k \to x_*$ as $k \to \infty$.

- This is Newton's method for a single-variable function, and we generalize it to a multi-variable function.

- $F(\boldsymbol{x})$: multi-variable, convex, differentiable function

  The Hessian matrix is defined by

$$\mathbf{H}F = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_M} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_M \partial x_1} & \frac{\partial^2 F}{\partial x_M \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_M^2} \end{bmatrix}.$$

  In short, $\mathbf{H}F = [\frac{\partial^2 F}{\partial x_i \partial x_j}]$.

- Recall

$$x_{k+1} = x_k - f'(x_k)/f''(x_k).$$

- Generalizing the single-variable case,

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathbf{H}F(\boldsymbol{x}_k)^{-1}\nabla F(\boldsymbol{x}_k)}.$$

Proof: Using Taylor's expansion, we have for $\boldsymbol{h} \in \mathbb{R}^m$,

$$F(\boldsymbol{x}) = \boxed{F(\boldsymbol{x}_0 + \boldsymbol{h}) \approx F(\boldsymbol{x}_0) + \boldsymbol{h}^\top \nabla F(\boldsymbol{x}_0) + \tfrac{1}{2}\boldsymbol{h}^\top \mathbf{H}F(\boldsymbol{x}_0)\boldsymbol{h}}.$$

Write $\mathbf{H}F(\boldsymbol{x}_0) = [H_{k\ell}]$, and

$$\boldsymbol{h}^\top \nabla F(\boldsymbol{x}_0) = \sum_k \frac{\partial F}{\partial x_k} h_k,$$
$$\boldsymbol{h}^\top \mathbf{H}F(\boldsymbol{x}_0)\boldsymbol{h} = \sum_{k,\ell} H_{k\ell} h_k h_\ell.$$

We obtain

$$\frac{\partial F}{\partial x_i}(\boldsymbol{x}) \approx \frac{\partial}{\partial h_i}\left(F(\boldsymbol{x}_0) + \boldsymbol{h}^\top \nabla F(\boldsymbol{x}_0) + \tfrac{1}{2}\boldsymbol{h}^\top \mathbf{H}F(\boldsymbol{x}_0)\boldsymbol{h}\right)$$
$$= \frac{\partial F}{\partial x_i} + \sum_{k=1}^{M} H_{ik} h_k.$$

Thus

$$\nabla F(\boldsymbol{x}) \approx \nabla F(\boldsymbol{x}_0) + \mathbf{H}F(\boldsymbol{x}_0)\boldsymbol{h}.$$

From $\nabla F(\boldsymbol{x}) = 0$, we approximately obtain

$$\boldsymbol{h} = -\mathbf{H}F(\boldsymbol{x}_0)^{-1}\nabla F(\boldsymbol{x}_0).$$

$\square$

- Using a step size η_k, the formula may be modified to be

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k \mathbf{H}F(\boldsymbol{x}_k)^{-1}\nabla F(\boldsymbol{x}_k)}.$$

- Newton's method is much faster than gradient descent. However, it may be expensive to compute $\mathbf{H}F(\boldsymbol{x}_k)^{-1}$. Sometimes, $\mathbf{H}F(\boldsymbol{x}_k)$ is close to a singular matrix.

Example

- Consider $E(\boldsymbol{w}) = E(w_1, w_2) = w_1^4 + w_2^4 - 16w_1 w_2$.
  Then $\nabla E(\boldsymbol{w}) = [4w_1^3 - 16w_2, 4w_2^3 - 16w_1]^\top$.

$$\mathbf{H}E(\boldsymbol{w}) = \begin{bmatrix} 12w_1^2 & -16 \\ -16 & 12w_2^2 \end{bmatrix}$$

$$\mathbf{H}E(\boldsymbol{w})^{-1} = \frac{1}{9w_1^2 w_2^2 - 16} \begin{bmatrix} \frac{3}{4}w_2^2 & 1 \\ 1 & \frac{3}{4}w_1^2 \end{bmatrix}$$

$$\mathbf{H}E^{-1}\nabla E = \frac{1}{9w_1^2 w_2^2 - 16} \begin{bmatrix} 3w_1^3 w_2^2 - 8w_2^3 - 16w_1 \\ 3w_1^2 w_2^3 - 8w_1^3 - 16w_2 \end{bmatrix}$$

- Choose $\boldsymbol{w}_0 = (1, 1)$ and $\eta = 1$.

  Then $\boldsymbol{w}_1 = (2, 2)$ and $E(\boldsymbol{w}_1) = -32$.
- Choose $\boldsymbol{w}_0 = (1.2, 1.2)$ and $\eta = 1$.

  Then $\boldsymbol{w}_9 = (2.00000004189571, 2.00000004189571)$,

  $E(\boldsymbol{w}_9) = -31.9999999999999$.

k	w1	w2	E(w1,w2)
0	1.20000000000000	1.20000000000000	-18.8928000000000
1	10.8000000000000	10.8000000000000	25343.5392000001
2	7.28325624421832	7.28325624421832	4778.98521693644
3	4.98069646698406	4.98069646698406	833.890570717962
4	3.50906808575457	3.50906808575457	106.230520855080
5	2.62345045192591	2.62345045192591	-15.3824765840014
6	2.16920289601164	2.16920289601164	-31.0047054152139
7	2.01793795417254	2.01793795417254	-31.9896107961456
8	2.00023638179330	2.00023638179330	-31.9999982117454
9	2.00000004189571	2.00000004189571	-31.9999999999999
10	2.00000000000000	2.00000000000000	-32.0000000000000

- Apply Newton's method to our main example:

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\},$$

  where $y_n = \sigma(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK})$.
- Recall that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

- We have

$$\nabla E(\boldsymbol{w}) = \left[\sum_{n=1}^{N}(y_n - t_n)x_{nj}\right] = X^\top(\boldsymbol{y} - \mathbf{t}),$$

where

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1K} \\ 1 & x_{21} & \cdots & x_{2K} \\ \vdots & & \vdots & \vdots \\ 1 & x_{N1} & \cdots & x_{NK} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \text{and } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}.$$

- Calculate $\dfrac{\partial^2 E}{\partial w_i \partial w_j}$.

- Recall $\frac{\partial E}{\partial w_j} = \sum_{n=1}^{N} (y_n - t_n) x_{nj}$.
- Here $y_n = \sigma(w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK})$.

  Write $x_n = w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_K x_{nK}$ and $y_n = \sigma(x_n)$.

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_i} \sum_{n=1}^{N} (y_n - t_n) x_{nj} = \sum_{n=1}^{N} \frac{\partial (y_n - t_n) x_{nj}}{\partial x_n} \frac{\partial x_n}{\partial w_i}$$
$$= \sum_{n=1}^{N} y_n (1 - y_n) x_{nj} x_{ni}$$

Thus we have

$$\boxed{\frac{\partial^2 E}{\partial w_i \partial w_j} = \sum_{n=1}^{N} y_n (1 - y_n) x_{ni} x_{nj}}.$$

- $$\frac{\partial^2 E}{\partial w_i \partial w_j} = \sum_{n=1}^{N} y_n(1-y_n)x_{ni}x_{nj}$$
- We get
$$\mathbf{H}E = \left[\sum_{n=1}^{N} y_n(1-y_n)x_{ni}x_{nj}\right] = X^\top R X,$$
where $R = \operatorname{diag}(y_n(1-y_n))$.
- Then we have
$$\boxed{\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - (X^\top R X)^{-1} X^\top (\boldsymbol{y} - \mathbf{t})},$$
where R and $\boldsymbol{y}$ are determined by $\boldsymbol{w}_k$ in each step.

Stochastic Gradient Descent (SGD)

- Typically in Machine Learning, we minimize a function $E(\boldsymbol{w})$ given by a sum of the form

$$E(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} E_n(\boldsymbol{w}),$$

where N is the number of elements in the training set.

- When N is large, computation of the gradient ∇E may be expensive.

- The SGD selects a sample from the training set in each iteration step instead of using the whole batch of the training set, and use

$$\frac{1}{L}\sum_{i=1}^{L} \nabla E_{n_i}(\boldsymbol{w}),$$

where L is the size of the sample and

$$\{n_1, n_2, \dots, n_L\} \subset \{1, 2, \dots, N\}.$$

- The SGD is commonly used in implementations of many Machine Learning algorithms.

Probability

Suppose that we are performing an experiment.

- S: set of all outcomes, called the sample space
- An event is a subset of S.

Example: Roll a die. Then the sample space is given by

$$S = \{1, 2, 3, 4, 5, 6\}.$$

The subset $A = \{1, 3, 5\}$ is an event, which corresponds to the statement "The outcome is an odd number".

Assume that S is a sample space.

- A probability is a function $P : \{\text{events}\} \to [0,1]$ satisfying
  1. $P(S) = 1$;
  2. if $E_1, E_2, \ldots$ are events such that $E_i \cap E_j = \varnothing$ for $i \neq j$, then

$$P\left(\bigcup_i E_i\right) = \sum_i P(E_i).$$

Example: Roll a die. Then we have the usual probabilty defined by

$$P(\{1\}) = P(\{2\}) = P(\{3\}) = P(\{4\}) = P(\{5\}) = P(\{6\}) = \frac{1}{6}.$$

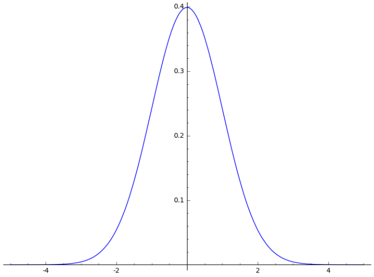
We obtain

$$P(\{1,3,5\}) = P(\{1\} \cup \{3\} \cup \{5\}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{3}{6} = \frac{1}{2}.$$

Example: Let $S = \mathbb{R}$. Define a function $f : S \to \mathbb{R}$ by

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

The graph of f is



For an event $A = [a, b] \subset S = \mathbb{R}$, define

$$P(A) = \int_A f(x)dx = \int_a^b f(x)dx.$$

Then P is a probability.

Proof: We have

$$P(S) = \int_S f(x)dx = \int_{-\infty}^{\infty} f(x)dx.$$

$$\begin{aligned} P(S)^2 &= \int_{-\infty}^{\infty} f(x)dx \int_{-\infty}^{\infty} f(y)dy = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)/2} dxdy \\ &= \frac{1}{2\pi} \int_0^{2\pi} \int_0^{\infty} e^{-r^2/2}\, r\, dr\, d\theta = \int_0^{\infty} e^{-u} du = 1 \end{aligned}$$

Thus $P(S) = 1$. The second condition is satisfied by the properties of an integral. $\square$

Conditional Probability

The conditional probability of B given that A has occurred is defined to be

$$\boxed{P(B|A) = \frac{P(A \cap B)}{P(A)}}.$$

<u>Example</u>: Roll a die.

$$A = \text{odd number}, \quad B = \text{prime number}$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{2/6}{3/6} = \frac{2}{3}$$

- From the definition $P(B|A) = \frac{P(A \cap B)}{P(A)}$, we obtain

$$\boxed{P(A \cap B) = P(A)P(B|A)}.$$

- Two events A and B are said to be independent if

$$P(A \cap B) = P(A)P(B).$$

  In this case, $P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A)P(B)}{P(A)} = P(B)$, and similarly, $P(A|B) = P(A)$.

Exmaples:

1. Draw a card from a deck of 52.

$$A = \text{the card is an ace}, \quad B = \text{the card is a spade}$$

$$P(A) = \frac{1}{13}, \quad P(B) = \frac{1}{4}, \quad P(A \cap B) = \frac{1}{52} = P(A)P(B)$$

2. Draw two cards from a deck of 52.

$$A = \text{the first card is a spade}, \quad B = \text{the second card is a spade}$$

$$P(A) = \frac{1}{4}, \quad P(B) = \frac{1}{4}, \quad P(A \cap B) = \frac{13}{52} \times \frac{12}{51} \neq P(A)P(B)$$

Bayes' Theorem

$$P(H|E) = \frac{P(E \cap H)}{P(E)} = \frac{P(E|H)P(H)}{P(E)}$$

$$\boxed{P(H|E) = \frac{P(E|H)}{P(E)}\, P(H)}$$

- $P(H)$: prior degree of belief in hypothesis H

  $P(H|E)$: posterior degree of belief after evidence E

  $\frac{P(E|H)}{P(E)}$: support that E provides for H

Previously,

$$\begin{aligned} \text{Learning} &\iff \text{Maximizing Certainty} \\ &\iff \text{Minimizing Randomness (Entropy)} \end{aligned}$$

The Bayesian viewpoint is

$$\text{Learning} \iff \text{Updating Belief through Evidences}$$

Since $P(E) = P(E|H)P(H) + P(E|H^c)P(H^c)$, we have

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} = \frac{P(E|H)P(H)}{P(E|H)P(H) + P(E|H^c)P(H^c)}.$$

Example: A new test for cancer is developed. When a person has cancer, this test is positive with probability 0.9. When a person does not have cancer, it is positive with probability 0.1. It is known that a person has cancer with probability 0.01. Bob receives the test and the result is positive. What is the probability that Bob has cancer?

$$P(C|+) = \frac{P(C\cap +)}{P(+)} = \frac{0.01 \times 0.9}{0.01 \times 0.9 + 0.99 \times 0.1} \approx 0.083$$

$$P(NC|+) = 1 - P(C|+) \approx 0.917$$

$$P(C|-) = \frac{P(C\cap -)}{P(-)} = \frac{0.01 \times 0.1}{0.01 \times 0.1 + 0.99 \times 0.9} \approx 0.001$$

$$P(NC|-) = 1 - P(C|-) \approx 0.999$$

- Bayesian interpretation

$$P(C|+) = \frac{P(+|C)}{P(+)} P(C)$$

- prior $P(C) = 0.01$, posterior $P(C|+) \approx 0.083$

  update $\frac{P(+|C)}{P(+)} \approx 8.3$

Random Variables

Let S be a sample space with probability P.

- A function $X : S \to \mathbb{R}$ is said to be a random variable.

Examples:

1. Roll a die, $X =$ the number that appears
2. Roll two dice, $X =$ the sum of two numbers that appear
3. Drive from Storrs to Boston, $X =$ the driving time in minutes

When the values of X is discrete, the probability mass function (pmf) of X is defined by

$$p(x) := P(X = x) = P(\{\omega \in S : X(\omega) = x\}).$$

Example: Consider an experiment whose outcome is a success or a failure.

- Let $Y = \begin{cases} 1 & \text{if a success,} \\ 0 & \text{if a failure.} \end{cases}$
- This random variable Y is called a Bernoulli random variable. Such an experiment is called a Bernoulli trial.
- If we set $p(1) = P(Y = 1) = p$, then $p(0) = P(Y = 0) = 1 - p$, and it defines a pmf.

Example: Repeat a Bernoulli trial n times, and assume each trial is independent.

- Let X = number of successes out of n trials.
- This random variable X is called a binomial random variable with parameters (n, p).
- The pmf is given by

$$p(k) = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \ldots, n,$$

  where $\binom{n}{k} := \frac{n!}{k!(n-k)!}$ is "n choose k".

When the values of X is continuous, the probability density function (pdf) of X is defined to be a function $f(x) \geq 0$ such that

$$P(a \leq X \leq b) = \int_a^b f(x)dx \quad (a \leq b).$$

Then we have

$$f(x) = \frac{d}{dx} P(-\infty \leq X \leq x).$$

Examples:

1. The random variable X with density function

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

is called the standard normal random variable.

2. For $\alpha, \lambda > 0$, the ramdom variable X with density function

$$f(x) = \frac{\lambda}{\Gamma(\alpha)} e^{-\lambda x} (\lambda x)^{\alpha - 1} \quad (x \geq 0)$$

is called a gamma random variable, where $\Gamma(\alpha) = \int_0^\infty e^{-x} x^{\alpha-1} dx$ is the gamma function. We have

$$\Gamma(n) = \int_0^\infty e^{-x} x^{n-1} dx = (n-1)!, \qquad n = 1, 2, 3, \ldots.$$

- X: random variable
- The expected value or mean of X is defined to be

$$E(X) = \sum x\,p(x) \quad \text{or} \quad \int x\,f(x)dx.$$

- The variance of X is defined by

$$\mathrm{Var}(X) = E[(X - E(X))^2] = E(X^2) - E(X)^2.$$

- The standard deviation of X is defined by

$$\sigma(X) = \sqrt{\mathrm{Var}(X)}.$$

- $E(aX+b) = aE(X)+b$
- $\mathrm{Var}(aX+b) = a^2\,\mathrm{Var}(X)$

Example: Roll a die, $X =$ the number that appears

$$p(x) = P(X = x) = \frac{1}{6} \quad \text{for } x = 1, 2, 3, 4, 5, 6$$

$$\begin{aligned} E(X) &= \sum x\, p(x) = 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + \cdots + 6 \times \frac{1}{6} \\ &= \frac{1 + 2 + \cdots + 6}{6} = \frac{21}{6} = \frac{7}{2} \end{aligned}$$

$$\begin{aligned} \mathrm{Var}(X) &= \sum (x - E(X))^2 p(x) = \sum x^2 p(x) - E(X)^2 \\ &= \frac{91}{6} - \left(\frac{7}{2}\right)^2 = \frac{35}{12} \approx 2.9167 \end{aligned}$$

$$\sigma(X) = \sqrt{\mathrm{Var}(X)} = \sqrt{\frac{35}{12}} \approx 1.708$$

Example: $X = \mathrm{binom}(n, p)$

$$E(X) = np \qquad \text{and} \qquad \mathrm{Var}(X) = np(1-p)$$

Example

- X: standard normal, $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$
- Since $xf(x)$ is odd, we have

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx = 0.$$

$$E(X^2) = \int_{-\infty}^{\infty} x^2 \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \int_0^{\infty} \frac{2}{\sqrt{2\pi}} x^2 e^{-\frac{x^2}{2}} dx$$
$$= \frac{2}{\sqrt{2\pi}} (-xe^{-\frac{x^2}{2}}) \Big|_0^{\infty} + \int_0^{\infty} \frac{2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = 1$$

$$\mathrm{Var}(X) = E(X^2) - E(X)^2 = 1$$

- Let Z be the standard normal random variable.

$$\boxed{E(Z) = 0 \qquad \text{and} \qquad \mathrm{Var}(Z) = 1}$$

- Define $X = \sigma Z + \mu$. Then X is a normal random variable.
- $E(X) = E(\sigma Z + \mu) = \sigma E(Z) + \mu = \mu$
- $\mathrm{Var}(X) = \mathrm{Var}(\sigma Z + \mu) = \sigma^2 \mathrm{Var}(Z) = \sigma^2$

$$\boxed{E(X) = \mu \qquad \text{and} \qquad \mathrm{Var}(X) = \sigma^2}$$

- The density function of f is given by

$$\mathcal{N}(x|\mu,\sigma) := f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

- Conversely, we can standardize a noraml random variable by

$$\boxed{Z = \frac{X - \mu}{\sigma}}.$$

Multivariate Distributions

Consider two random variables X and Y at the same time.

- When X and Y are discrete, the joint mass function is defined by

$$p(x, y) = P(X = x, Y = y).$$

- Set $p_X(x) = \sum_y p(x, y)$ and $p_Y(y) = \sum_x p(x, y)$. They are pmfs of X and Y, respectively, and are called the marginal mass functions.

- When X and Y are continuous, the joint density function of X and Y is defined to be a function $f(x, y) \geq 0$ such that, for any $A \subseteq \mathbb{R}^2$,

$$P((X, Y) \in A) = \iint\limits_{A} f(x, y)dxdy.$$

- Define $f_X(x) = \int f(x, y)dy$ and $f_Y(y) = \int f(x, y)dx$. They are pdfs of X and Y, repectively, and are called the marginal density functions.
- In particular, we have

$$\iint\limits_{\mathbb{R}^2} f(x, y)dxdy = 1.$$

Example

- X, Y: two standard normal variables
- Define

$$f(x,y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}.$$

Then $f(x,y)$ is a joint density function.

- $P(X^2+Y^2 \le 1) = ??$

$$P(X^2+Y^2 \le 1) = \iint\limits_{x^2+y^2\le 1} f(x,y)dxdy = \int_0^{2\pi}\int_0^1 \frac{1}{2\pi} e^{-\frac{r^2}{2}} r\,drd\theta$$

$$= \int_0^1 e^{-\frac{r^2}{2}} r\,dr = -e^{-u}\Big|_0^1 = 1 - e^{-1} \approx 0.6321$$

- Two random variables X and Y are said to be independent if

$$P(X \le a, Y \le b) = P(X \le a)P(Y \le b) \qquad \text{for all } a \text{ and } b.$$

- <u>discrete case</u>: X and Y are independent

$$\iff \quad P(X = a, Y = b) = P(X = a)P(Y = b) \qquad \text{for all } a \text{ and } b.$$

- <u>continuous case</u>: X and Y are independent

$$\iff \quad f(x, y) = f_X(x)f_Y(y).$$

- X, Y: random variables

  Then we have
  $$E(X+Y) = E(X) + E(Y).$$
  In general, for random variables $X_1, X_1, \dots, X_n$, we have
  $$E\left(\sum X_i\right) = \sum E(X_i).$$

- X, Y: independent random variables

  Then we have
$$E(XY) = E(X)E(Y).$$

- The covariance of X and Y is defined by
$$\mathrm{Cov}(X, Y) = E[(X - E(X))(Y - E(Y))].$$

  We have
$$\mathrm{Cov}(X, Y) = E(XY) - E(X)E(Y).$$

- If X and Y are independent, we have
$$\mathrm{Cov}(X, Y) = 0.$$

- We have

$$\mathrm{Var}(X+Y) = V(X) + V(Y) + 2\,\mathrm{Cov}(X,Y).$$

  In general,

$$\mathrm{Var}\left(\sum X_i\right) = \sum \mathrm{Var}(X_i) + 2\sum_{i<j} \mathrm{Cov}(X_i, X_j).$$

- If $X_1, X_2, \dots, X_n$ are independent, then

$$\mathrm{Var}\left(\sum X_i\right) = \sum \mathrm{Var}(X_i).$$

Maximum Likelihood Estimate (MLE)

A student practices three-point shots in basketball. He tried 100 shots and made 67 shots successful. What is the prbability of success for the next shot?

Choose your model and compute the likelihood.

$$L(p) = \binom{100}{67} p^{67}(1-p)^{33}$$

Determine the parameter p so that the likelihood function $L(p)$ is maximized.

It is more convenient to consider $\log L(p)$.

$$\log L(p) = 67 \log p + 33 \log(1-p) + C$$

$$(\log L(p))' = \frac{67}{p} - \frac{33}{1-p} = \frac{67(1-p) - 33p}{p(1-p)} = \frac{67 - 100p}{p(1-p)}$$

When $p = \frac{67}{100}$, the likelihood $L(p)$ is maximized.

<u>Prediction</u>: the next shot will be successful with probability $\frac{67}{100}$.

This method is called the maximum likelihood estimate (MLE).

Linear Regression as MLE

- Input: $\boldsymbol{x} = (x_1, x_2 \dots, x_K) \in \mathbb{R}^K$ Output: $t \in \mathbb{R}$

  Observations: $(x_{11}, x_{12}, \dots, x_{1K}; t_1), \dots, (x_{N1}, x_{N2}, \dots, x_{NK}; t_N)$

- Use these observations as training examples.

  Task: $\boxed{\text{Given a new input } (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_K)\text{, predict the output } \tilde{t}.}$

- Choose a model:

  Given $\boldsymbol{x}$, the corresponding value of t has a normal distribution with mean

  $$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_K x_K,$$

  where $\boldsymbol{w} = [w_0, w_1, \ldots, w_K]^\top$.

- Then we have

  $$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon,$$

  where ϵ is the Gaussian noise with mean 0.

- The density function is given by

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}),$$

  where β is a parameter corresponding to the inverse variance, called the precision.

- Assume that each observation is independent, and that the variance β^{-1} is all the same.
- Then the likelihood density is

$$\begin{aligned} L(\boldsymbol{w}) &= p(\mathbf{t}|X, \boldsymbol{w}, \beta) \\ &= \prod_{n=1}^{N} \mathcal{N}(t_n | y_n(\boldsymbol{x}_n, \boldsymbol{w}), \beta^{-1}) \\ &= \left(\sqrt{\frac{\beta}{2\pi}} \right)^{N} \exp\left[-\frac{\beta}{2} \sum (t_n - y_n)^2 \right]. \end{aligned}$$

- This is our probabilistic model.

Taking log, we obtain

$$\begin{aligned}\log L(\boldsymbol{w}) &= -\frac{\beta}{2}\sum(t_n - y_n)^2 + (\text{constant}) \\ &= -\frac{\beta}{2}\|\mathbf{t} - X\boldsymbol{w}\|^2 + (\text{constant}).\end{aligned}$$

Thus MLE is equivalent to minimizing the error function

$$E(\boldsymbol{w}) = \sum(t_n - y_n)^2 = \|\mathbf{t} - X\boldsymbol{w}\|^2.$$

- Maximum likelihood

$\iff$ Minimizing $\|\boldsymbol{y} - X\boldsymbol{w}\|^2$

$\iff$ Linear regression

Bayesian Linear Regression

- Bayesian linear regression avoids the over-fitting problem of maximum likelihood.

1. Consider $\boldsymbol{w}$ as a random variable.
2. Choose prior information on $\boldsymbol{w}$ in the form of probability distribution.
3. Given observations $\{(\mathbf{x}_n, t_n)\}$, update $\boldsymbol{w}$ using Bayes' formula to obtain the posterior distribution of $\boldsymbol{w}$.
4. Determine $\boldsymbol{w}$ so that $\underset{\sim}{\text{the posterior probability is maximized}}$; that is, take the mode of the posterior.

- M-dimensional Gaussian distribution:

$$\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu},\Sigma) := \frac{1}{(2\pi)^{M/2}}\frac{1}{|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu})^\top\Sigma^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\right\},$$

  where the M-dimensional vector $\boldsymbol{\mu}$ is the mean, the $M \times M$ matrix Σ is the covariance, and $|\Sigma|$ is the determinant of Σ.

- Choose a prior distribution for $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|0,\alpha^{-1}I).$$

- We have

$$-\ln p(\boldsymbol{w}|\alpha) = \frac{\alpha}{2}\boldsymbol{w}^\top\boldsymbol{w} + (\text{constant}) = \frac{\alpha}{2}\|\boldsymbol{w}\|^2 + (\text{constant}).$$

- Bayes' Theorem gives the posterior

$$p(\boldsymbol{w}|X, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|X, \boldsymbol{w}, \beta)\, p(\boldsymbol{w}|\alpha).$$

Recall $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$.

Task: Determine $\boldsymbol{w}$ so that the posterior is maximized.

This process is called a maximum a posteriori (MAP) estimation.

- Take the negative logarithm of the posterior

$$\begin{aligned} -\log p(\boldsymbol{w}|X,\mathbf{t},\alpha,\beta) &= -\log\left[p(\mathbf{t}|X,\boldsymbol{w},\beta)\,p(\boldsymbol{w}|\alpha)\right] + (\text{constant}) \\ &= \frac{\beta}{2}\|\mathbf{t}-X\boldsymbol{w}\|^2 + \frac{\alpha}{2}\|\boldsymbol{w}\|^2 + (\text{constant}) \end{aligned}$$

- The maximum of the posterior is given by the minimum of

$$\boxed{\tilde{E}(\boldsymbol{w}) = \frac{\beta}{2}\|\mathbf{t}-X\boldsymbol{w}\|^2 + \frac{\alpha}{2}\|\boldsymbol{w}\|^2}.$$

Thus, maximizing the posterior distribution is equivalent to minimizing the regularized sum-of-square error function, i.e., ridge regression.

- We can compute $\boldsymbol{w}$ explicitly:

$$\tilde{E}(\boldsymbol{w}) = \frac{\beta}{2}\|X\boldsymbol{w} - \mathbf{t}\|^2 + \frac{\alpha}{2}\|\boldsymbol{w}\|^2,$$

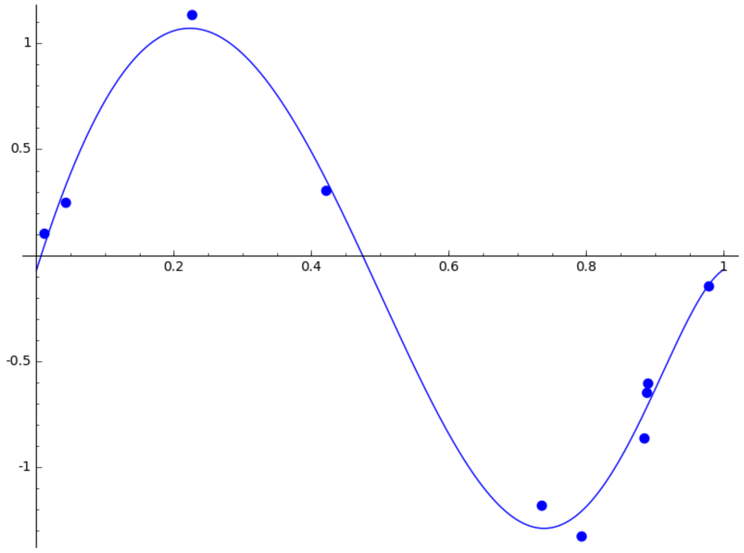
  and

$$\nabla\tilde{E}(\boldsymbol{w}) = \beta X^\top(X\boldsymbol{w} - \mathbf{t}) + \alpha\boldsymbol{w} = 0.$$

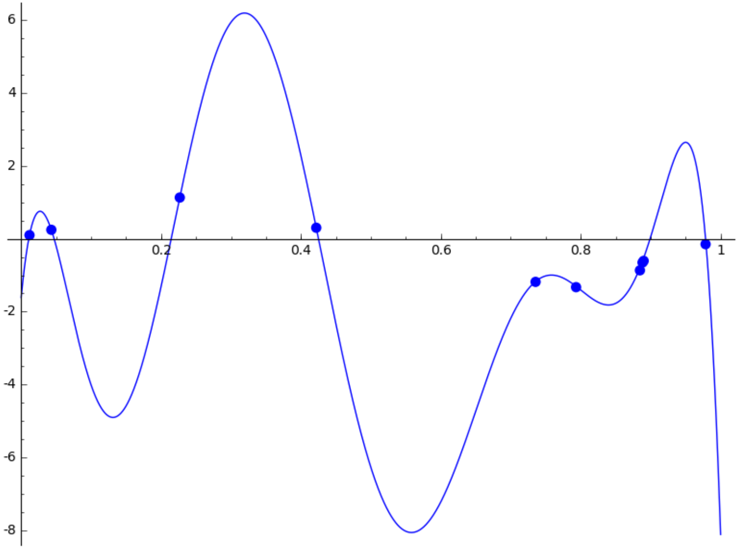
  Thus

$$\boxed{\boldsymbol{w} = \beta S X^\top \mathbf{t} \quad \text{with} \;\; S^{-1} = \alpha I + \beta X^\top X}.$$

$N = 9$, $\alpha = 0.01$ and $\beta = 1000$



Recall the maximum likelihood gave us



- The posterior can be computed explicitly, since the prior and the likelihood are all Gaussian.
- Indeed, we obtain

$$\boxed{p(\boldsymbol{w}|X, \mathbf{t}, \alpha, \beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}, S)},$$

where

$$\boldsymbol{m} = \beta S X^\top \mathbf{t} \quad \text{with} \quad S^{-1} = \alpha I + \beta X^\top X.$$

Naive Bayes Classifier

Consider the binary classification

$$\{\text{sentences}\} \longrightarrow \{\text{positive}, \text{negative}\}.$$

For example, take a dataset of sentences from Amazon reivews on products.

Task: Given a new sentence, determine whether it is positive or negative.

Question: How can we convert a setence into a vector?

- Fix a vocabulary of K words.
- Make an ordered list $w_1, w_2, \dots, w_K$ of the words in the vocabulary.
- Replace a sentece with a binary vector:

$$\text{sentence} \longrightarrow (x_1, x_2, \dots, x_K),$$

  where $x_i = 1$ if w_i occurs and $x_i = 0$ otherwise.

- For simplicity, we don't record how many times a word occurs. Similarly, we ignore the order in which different words appear.

Analize a dataset of 1000 sentences.

Denote by G an occurrence of “great” in a sentence and by G^c no occurrence of “great”.

	$+$	$-$	total
G	92	5	97
G^c	408	495	903
total	500	500	1000

For the classification, we are interested in $P(+|G)$.

How can we compute $P(+|G)$??

- Conditional probability

$$P(+|G) = \frac{P(+\cap G)}{P(G)} = \frac{92/1000}{97/1000} = \frac{92}{97} \approx 0.9485$$

- Bayes' Theorem

$$P(+|G) = \frac{P(G|+)}{P(G)}\ P(+) = \frac{92/500}{97/1000}\ 0.5 \approx 0.9485$$

We have

$$P(-|G) = 1 - 0.9485 = 0.0515,$$
$$P(+|G^c) = \frac{P(+\cap G^c)}{P(G^c)} = \frac{408}{903} = 0.4518,$$
$$P(-|G^c) = 1 - 0.4518 = 0.5482.$$

Denote by W an occurrence of “waste” in a sentence and by W^c no occurrence of “waste”.

	$+$	$-$	total
W	0	14	14
W^c	500	486	986
total	500	500	1000

We have

$$P(+|W) = \frac{P(+\cap W)}{P(W)} = 0, \quad P(-|W) = 1,$$

$$P(+|W^c) = \frac{P(+\cap W^c)}{P(W^c)} = \frac{500}{986} = 0.5071,$$

$$P(-|W^c) = 1 - 0.5071 = 0.4929.$$

We need to consider many different words all together.

Assumption: Given a collection of words, their presence and absence in a (postive or negative) review are independent.

Then we have

$$P(G, W|\pm) = P(G|\pm)P(W|\pm), \quad P(G, W^c|\pm) = P(G|\pm)P(W^c|\pm),$$

$$P(G^c, W|\pm) = P(G^c|\pm)P(W|\pm), \quad P(G^c, W^c|\pm) = P(G^c|\pm)P(W^c|\pm).$$

The above ssumption is naive, but it enables us to compute relavant probabilities.

- Consider a sentence with G and W^c.
- Is this sentence positive or negative?
- To be precise, we compare $P(+|G, W^c)$ with $P(-|G, W^c)$.

Using Bayes' Theorem, we have

$$P(+|G,W^c) = \frac{P(G,W^c|+)P(+)}{P(G,W^c)} = \frac{P(G|+)P(W^c|+)P(+)}{P(G,W^c)},$$

$$P(-|G,W^c) = \frac{P(G,W^c|-)P(-)}{P(G,W^c)} = \frac{P(G|-)P(W^c|-)P(-)}{P(G,W^c)}.$$

Since the denominators are the same, it is enough to consider the numberators for comparison.

$$L(+|G,W^c) := P(G|+)P(W^c|+)P(+) = \frac{92}{500}\frac{500}{500}\frac{1}{2} = 0.092$$

$$L(-|G,W^c) := P(G|-)P(W^c|-)P(-) = \frac{5}{500}\frac{486}{500}\frac{1}{2} = 0.00486$$

- Assume that we have extracted key words $w_1, w_2, \ldots, w_K$ from the dataset.
- Replace a sentece with a binary vector:

$$\text{sentence} \longrightarrow (x_1, x_2, \ldots, x_K),$$

  where $x_i = 1$ if w_i occurs and $x_i = 0$ otherwise.
- By Assumption, x_i's correspond to independent Bernoulli random variables.
- If there are N sentences in the dataset, we obtain a data matrix X of size $N \times K$.

- Let $\mathbf{t} = [t_1, t_2, \ldots, t_N]^\top$ be the column vector, where $t_n = 1$ if the n^{th} review is positive and $t_n = 0$ otherwise.
- The number of positive reviews is equal to $N_+ := \mathbf{t}^\top \mathbf{t}$, and the number of negative reviews is equal to $N_- := N - N_+$.
- Notice that

$$\mathbf{t}^\top X = [\#(+ \cap w_1),\ \#(+ \cap w_2), \ldots,\ \#(+ \cap w_K)],$$

where $\#(+ \cap w_i)$ is the number of positive reviews that contain w_i.
- We have

$$P_+ := \frac{1}{N_+} \mathbf{t}^\top X = [P(w_1|+),\ P(w_2|+), \ldots,\ P(w_K|+)].$$

- Let $\mathbb{1}$ be the matrix with all entries equal to 1, whose size is to be determined by the context.
- We have

$$(\mathbb{1} - \mathbf{t})^\top X = [\#(- \cap w_1),\ \#(- \cap w_2), \dots,\ \#(- \cap w_K)].$$

- It follows that

$$P_- := \frac{1}{N_-}(\mathbb{1} - \mathbf{t})^\top X = [P(w_1|-),\ P(w_2|-), \dots,\ P(w_K|-)].$$

- Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$ be a data vector.
  For example, $\boldsymbol{x} = (1, 1, 0, 0, 1)$. Then

$$P(\boldsymbol{x}|+) = P(w_1|+)P(w_2|+)(1 - P(w_3|+))(1 - P(w_4|+))P(w_5|+).$$

- We see that

$$P(\boldsymbol{x}|\pm) = \prod_i P(w_i|\pm)^{x_i}(1 - P(w_i|\pm))^{(1-x_i)}.$$

- Taking log of both sides, we get

$$\log P(\boldsymbol{x}|\pm) = \sum_i x_i \log P(w_i|\pm) + (1 - x_i)\log(1 - P(w_i|\pm)).$$

- Bayes' Theorem

$$P(\pm|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\pm)P(\pm)}{P(\boldsymbol{x})}$$

- By taking log, we have

$$\log P(\pm|\boldsymbol{x}) = \log P(\boldsymbol{x}|\pm) + \log P(\pm) - \log P(\boldsymbol{x}),$$

  where

$$P(\pm) = \frac{N_\pm}{N}.$$

- A review is positive if $\log P(+|\boldsymbol{x}) > \log P(-|\boldsymbol{x})$ and negative otherwise.
- It follows from

$$\log P(\pm|\boldsymbol{x}) = \log P(\boldsymbol{x}|\pm) + \log P(\pm) - \log P(\boldsymbol{x})$$

that a review is positive if

$$\boxed{\log P(\boldsymbol{x}|+) + \log P(+) > \log P(\boldsymbol{x}|-) + \log P(-)}$$

and negative otherwise.

- In a previous example, we had $P(+|W) = 0$.
  Then $\log P(+|W)$ is not defined!

- Introduce a “fake sentence” into both classes in which every vocabulary word appears.
  This guarantees that no relevant probability is zero.

- Recall

$$\log P(\boldsymbol{x}|\pm) = \sum_i x_i \log P(w_i|\pm) + (1 - x_i)\log(1 - P(w_i|\pm)).$$

- Let X be the data matrix with each row $\boldsymbol{x}$ corresponding to a sentence.
- Define $\log P(X|\pm)$ to be the column vector consising of $\log P(\boldsymbol{x}|\pm)$.
- We have

$$\boxed{\log P(X|\pm) = X(\log P_\pm)^\top + (\mathbb{1} - X)\log(\mathbb{1} - P_\pm)^\top}.$$

Multi-class Logistic Regression

- Assume that there are s different classes.

  Data set: $\{(x_{n,1}, \dots, x_{n,K}; t_n)\}$, $t_n = 1, 2, \dots, s$, $n = 1, \dots, N$
- How to generalize logistic regression?

- $t_n = 1 \Leftrightarrow [1, 0, \dots, 0], \quad t_n = 2 \Leftrightarrow [0, 1, 0 \dots, 0], \dots,$

  $t_n = s \Leftrightarrow [0, \dots, 0, 1]$
- Obtain an $N \times s$ matrix $\mathbf{t} = [t_{n,m}]$ such that

$$t_{n,m} = \begin{cases} 1 & \text{if } t_n = m, \\ 0 & \text{if } t_n \neq m. \end{cases}$$

- Define $\boldsymbol{\sigma} : \mathbb{R}^s \to (0,1)^s$ by

$$\boldsymbol{\sigma}(\boldsymbol{a}) = \left(\frac{e^{a_1}}{\sum_{i=1}^s e^{a_i}}, \dots, \frac{e^{a_s}}{\sum_{i=1}^s e^{a_i}} \right),$$

  where $\boldsymbol{a} = (a_1, a_2, \dots, a_s)$.

  The function $\boldsymbol{\sigma}$ is called the softmax function.

- When $s = 2$, we have

$$\boldsymbol{\sigma}(\boldsymbol{a}) = \left(\frac{e^{a_1}}{e^{a_1} + e^{a_2}}, \frac{e^{a_2}}{e^{a_1} + e^{a_2}} \right) = \left(\frac{1}{1 + e^{a_2 - a_1}}, \frac{e^{a_2 - a_1}}{1 + e^{a_2 - a_1}} \right).$$

- The softmax function is a generalization of the sigmoid function.

- Define $\boldsymbol{y} = [y_1, \dots, y_s] = \boldsymbol{\sigma}(\boldsymbol{a})$.
- For $m, j = 1, \dots, s$,

$$\boxed{\frac{\partial y_m}{\partial a_j} = y_m(\delta_{j,m} - y_j)},$$

where $\delta_{j,m}$ is the Kronecker's delta, i.e.

$$\delta_{j,m} = \begin{cases} 1 & \text{if } j = m, \\ 0 & \text{otherwise.} \end{cases}$$

Proof: Recall $y_m = \dfrac{e^{a_m}}{\sum_i e^{a_i}}$. Then we have

$$\begin{aligned}\frac{\partial y_m}{\partial a_j} &= \frac{\delta_{j,m} e^{a_m} (\sum e^{a_i}) - e^{a_m} e^{a_j}}{(\sum e^{a_i})^2} \\ &= \frac{e^{a_m}(\delta_{j,m} \sum e^{a_i} - e^{a_j})}{(\sum e^{a_i})^2} \\ &= \frac{e^{a_m}}{\sum e^{a_i}} \, \frac{\delta_{j,m} \sum e^{a_i} - e^{a_j}}{\sum e^{a_i}} = y_m(\delta_{j,m} - y_j).\end{aligned}$$

□

- Consider a $(K+1) \times s$ matrix $\boldsymbol{w} = [w_{p,q}]$.
  Define $\boldsymbol{y} = \boldsymbol{\sigma}(X\boldsymbol{w}) = [y_{n,m}]$,
  where X is as before and $\boldsymbol{\sigma}$ is applied to the rows of $X\boldsymbol{w}$.
  Each row of $\boldsymbol{y}$ consists of probabilities for classes 1 through s.
- The likelihood function is given by

$$L(\boldsymbol{w}) = \prod_{n=1}^{N} \prod_{m=1}^{s} y_{n,m}^{t_{n,m}}.$$

- The cross-entropy is

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \sum_{m=1}^{s} t_{n,m} \ln y_{n,m}.$$

- We have

$$\nabla E(\boldsymbol{w}) = \left[\frac{\partial E}{\partial w_{p,q}}\right] = \left[\sum_{n=1}^{N}(y_{n,q} - t_{n,q})x_{n,p}\right] = X^\top(\boldsymbol{y} - \mathbf{t}).$$

<u>Proof</u>: Notice

$$\sum_{m=1}^{s} t_{n,m}(\delta_{q,m} - y_{n,q}) = \sum_{m=1}^{s}(t_{n,m}\delta_{q,m} - t_{n,m}y_{n,q}) = t_{n,q} - y_{n,q}.$$

Write $X\boldsymbol{w} = [a_{n,j}]$ and $\boldsymbol{a}_n = (a_{n,1}, \dots, a_{n,s})$. Then $a_{n,j} = \sum_k x_{n,k}w_{k,j}$ and $\boldsymbol{y}_n = \boldsymbol{\sigma}(\boldsymbol{a}_n) = (y_{n,1}, \dots, y_{n,s})$. We have

$$\frac{\partial y_{n,m}}{\partial a_{n,j}} = y_{n,m}(\delta_{j,m} - y_{n,j}).$$

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{m=1}^{s} t_{n,m}\ln y_{n,m}$$

$$\begin{aligned}
\frac{\partial E}{\partial w_{p,q}} &= -\sum_n\sum_m t_{n,m}\frac{1}{y_{n,m}}\frac{\partial y_{n,m}}{\partial w_{p,q}} \\
&= -\sum_n\sum_m t_{n,m}\frac{1}{y_{n,m}}\sum_j \frac{\partial y_{n,m}}{\partial a_{n,j}}\frac{\partial a_{n,j}}{\partial w_{p,q}} \\
&= -\sum_n\sum_m t_{n,m}\frac{1}{y_{n,m}}\sum_j y_{n,m}(\delta_{j,m} - y_{n,j})\delta_{j,q}x_{n,p} \\
&= -\sum_n\sum_m t_{n,m}(\delta_{q,m} - y_{n,q})x_{n,p} = \sum_n (y_{n,q} - t_{n,q})x_{n,p}
\end{aligned}$$

□

- Gradient Descent

$$\boxed{\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \eta X^\top(\boldsymbol{y} - \mathbf{t})}.$$

- Let $\boldsymbol{w}_i \to \boldsymbol{w}_*$ as $i \to \infty$.

  Given $\boldsymbol{x} = [1, x_1, \dots, x_K]$, the coordinates of the vector

$$\boldsymbol{y} = \sigma(\boldsymbol{x}\boldsymbol{w}_*)$$

  represent the probabilities for the classes.

- The (multi-class) logistic regression is the simplest **neural network**.

Dimensionality Reduction

- Typically, our data points live in a high dimensional space.
- In a high dimensional space,

  - the data set becomes very sparse;

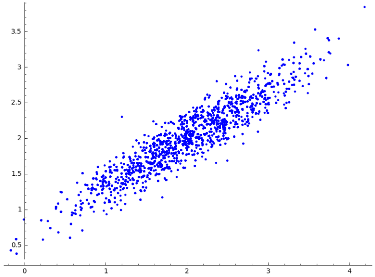
  - our geometric intuitions can fail.
- Severe difficulties that can arise in spaces of high dimensions are usually called the curse of dimensionality (Bellman, 1961).

- Frequently, dimensionality reduction is very helpful and Principal Component Analysis (PCA) is a commond method.
- PCA is an unsupervised learning method.
- Other methods for dimensionality reduction:
  - Linear Discriminatn Analysis (LDA),
  - t-distributed Stochastic Neighbor Embedding (t-SNE),
  - Uniform Manifold Approximation and Projection (UMAP)

Principal Component Analysis (PCA)

- $\{\boldsymbol{x}_n \in \mathbb{R}^K\}$: dataset, $n = 1, \ldots, N$

  For example, when $K = 2$,



  Which direction represents the dataset best?

- Goal: Project the dataset onto a subspace of dimension $D < K$ such that the variance is maximized.
  Typically, we take $D = 2$ or $D = 3$.

- It is called the Principal Component Analysis (PCA).

- Consider $\boldsymbol{x}_n$ as a row vector.
- Write $\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$.
- Let $\boldsymbol{u}$ be a column vector with $\|\boldsymbol{u}\| = 1$.

  The mean of the $\boldsymbol{u}$-coordinates is

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{u} = \overline{\boldsymbol{x}}\boldsymbol{u}.$$

- The variance of the $\boldsymbol{u}$-coordinates is

$$\frac{1}{N}\sum_{n=1}^{N}\{\boldsymbol{x}_n\boldsymbol{u}-\overline{\boldsymbol{x}}\boldsymbol{u}\}^2 = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{u}^\top(\boldsymbol{x}_n-\overline{\boldsymbol{x}})^\top(\boldsymbol{x}_n-\overline{\boldsymbol{x}})\boldsymbol{u}$$
$$= \boxed{\boldsymbol{u}^\top S\,\boldsymbol{u}},$$

where S is a $K \times K$ symmetric matrix given by

$$\boxed{S = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n-\overline{\boldsymbol{x}})^\top(\boldsymbol{x}_n-\overline{\boldsymbol{x}})}.$$

- Write $\overline{\boldsymbol{x}} = (\overline{x}_1, \dots, \overline{x}_K)$ and $\boldsymbol{x}_n = (x_{n1}, \dots, x_{nK})$.
- Let S_{ij} be the (i,j)-entry of S. Then we have

$$S_{ij} = \frac{1}{N}\sum_{n=1}^{N}(x_{ni} - \overline{x}_i)(x_{nj} - \overline{x}_j).$$

- This is the covariance of the i-th feature and the j-th feature. In particular, the diagonal entry S_{ii} is the variance of the i-th feature.
- The symmetric matrix S is called the covariance matrix.

- Write X_0 be the matrix whose rows are $\boldsymbol{x}_n - \overline{\boldsymbol{x}}$, $n = 1, 2, \ldots, N$. Then we have
$$\boxed{S = \frac{1}{N} X_0^\top X_0}.$$

- The covariance S_{ij} measures the dependency between the i-feature and the j-th feature.
  However, S_{ij} may be big simply because x_{ni} and x_{nj} are big.

- Define the correlation of X and Y by

$$\mathrm{Corr}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sigma(X)\sigma(Y)}.$$

- We have $-1 \le \mathrm{Corr}(X, Y) \le 1$. In particular, $\mathrm{Corr}(X, X) = 1$.

- In our case, the correlation matrix $C = [C_{ij}]$ is given by

$$C_{ij} = \frac{S_{ij}}{\sqrt{S_{ii} S_{jj}}}.$$

- Task: Maximize the variance $\boldsymbol{u}^\top S\,\boldsymbol{u}$ for $\boldsymbol{u}$ such that

$$\|\boldsymbol{u}\|^2 = \boldsymbol{u}^\top \boldsymbol{u} = 1.$$

- It is an optimization problem with contraints:

  maximize $f(\boldsymbol{x})$ for $\boldsymbol{x}$ subject to $g(\boldsymbol{x}) = c$?

- We can use Lagrange Multiplier.

Theorem (Lagrange Multiplier)

Assume that $g(\boldsymbol{x}) = c$ and $\nabla g \neq 0$. If $\boldsymbol{x}_*$ is a maximum or a minimum of $f(\boldsymbol{x})$, then

$$\nabla f(\boldsymbol{x}_*) = \lambda \nabla g(\boldsymbol{x}_*)$$

for some $\lambda \in \mathbb{R}$.

- Let $f(\boldsymbol{u}) = \boldsymbol{u}^\top S\,\boldsymbol{u}$ and $g(\boldsymbol{u}) = \boldsymbol{u}^\top \boldsymbol{u}$.
  Then $\nabla f(\boldsymbol{u}) = 2S\boldsymbol{u}$ and $\nabla g(\boldsymbol{u}) = 2\boldsymbol{u}$.
  From Lagrange multiplier, we obtain
  $$\boxed{S\boldsymbol{u} = \lambda \boldsymbol{u}}.$$
- $\boldsymbol{u}$ is an eigenvector with eigenvalue λ.

- In this case, the variance is

$$\boldsymbol{u}^\top S\boldsymbol{u} = \lambda\, \boldsymbol{u}^\top \boldsymbol{u} = \lambda,$$

  and it is maximized when λ is the largest eigenvalue of S.

- The corresponding eigenvector is the first principal direction.
- Actually, we can always find K prinipal directions, or K linearly independent eigenvectors.

- Recall that S is symmetric.

Theorem (Spectral Theorem)

Let A be a $K \times K$ symmetric matrix. Then the following is true.

1. *The matrix A has K real eigenvalues, counting multiplicities.*
2. *The eigenvectors corresponding to different eigenvalues are orthogonal.*
3. *The matrix A is orthogonally diagonalizable.*

- In particular, the covariance matrix S is diagonalizable with real eigenvalues.

- Let

$$\lambda_1 \geq \cdots \geq \lambda_K$$

be the eigenvalues of S, and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$ are corresponding eigenvectors with $\|\boldsymbol{u}_i\| = 1$.

- The variance in the $\boldsymbol{u}_i$ direction is

$$\boldsymbol{u}_i^\top S \boldsymbol{u}_i = \lambda_i \qquad \text{for } i = 1, \ldots, K.$$

- For $D \leq K$, the D-dimensional projection onto the subspace spanned by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D$ produces the desired dimensionality redution. (Typically, $D = 2$ or 3.)

How to compute the projection:

- In order to obtain a centered picture, use X_0 instead of X.
- The $\boldsymbol{u}_i$-coordinate of $\boldsymbol{x}_n - \overline{\boldsymbol{x}}$ is simply

$$(\boldsymbol{x}_n - \overline{\boldsymbol{x}})\boldsymbol{u}_i.$$

- Let U be the $K \times D$ matrix whose columns are $\boldsymbol{u}_1, \dots, \boldsymbol{u}_D$. Then the coordinates of the projection is given by the $N \times D$ matrix

$$X_0 U = X_0[\boldsymbol{u}_1 \cdots \boldsymbol{u}_D].$$

Question: Can we see the original features in the projection?

- Consider
$$\boldsymbol{e}_1 = (1, 0, \dots, 0), \boldsymbol{e}_2 = (0, 1, 0, \dots, 0), \dots, \boldsymbol{e}_K = (0, \dots, 0, 1).$$
Then $\boldsymbol{e}_i$ is supported only by the i-th feature.

- The projection $\boldsymbol{e}_i U$ is nothing but the i-th row of U.

- The i-th row vector of U is the direction of the i-th feature, and it is called the loading of the i-th feature.

PCA Approximation

- We may use PCA for approximation.
- Let $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K\}$ be an orthonormal basis consiting of eigenvectors with $\lambda_1 \geq \cdots \geq \lambda_K$.
- Then a PCA approximation $\tilde{\boldsymbol{x}}_n$ to a data vector $\boldsymbol{x}_n$ is given by

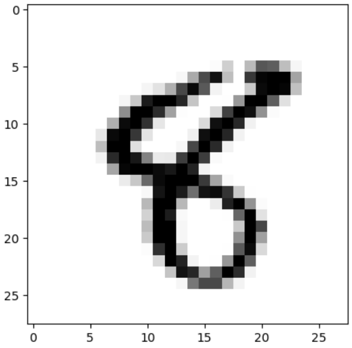
$$\boxed{\tilde{\boldsymbol{x}}_n = \overline{\boldsymbol{x}} + \sum_{i=1}^{D} (\boldsymbol{x}_n - \overline{\boldsymbol{x}}) \boldsymbol{u}_i \, \boldsymbol{u}_i^\top .}$$

Example: Hand-written digits

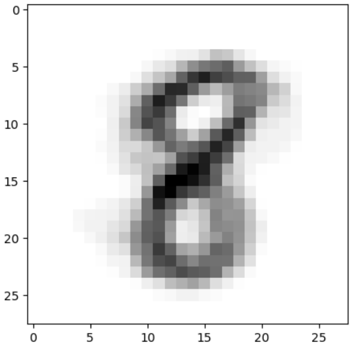
Original Data: $K = 28 \times 28 = 784$

Taking $N = 10$ and $D = 1$, we obtain the following:

Original



PCA approximation with $D = 1$



Singular Value Decomposition (SVD)

- PCA can be understood as SVD.
- SVD also generalizes the notion of eigenvectors from square matrices to matrices of arbitrary sizes.
- Recall that a matrix A is orthogonal if the columns form an orthonormal set, or equivalently,

$$A^\top A = I.$$

Theorem

Any (real) $N \times K$ matrix X can be decomposed as follows:

$$\boxed{X = U \Lambda V^\top},$$

where U is an $N \times N$ orthogonal matrix, V is a $K \times K$ orthogonal matrix, and Λ is an $N \times K$ matrix containing $r := \min(N, K)$ vaules, called singular values, on the main diagonal with 0's elsewhere.

- Write $\Lambda = \operatorname{diag}(\sigma_1, \dots, \sigma_r)$. The pseudoinverse $X^\dagger$ is defined by

$$\boxed{X^\dagger = V \Lambda^\dagger U^\top},$$

where $\Lambda^\dagger = \operatorname{diag}(1/\sigma_1, \dots, 1/\sigma_r)$.

Example

- Let $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$. Then the SVD gives

$$U \approx \begin{bmatrix} -0.3863 & -0.9224 \\ -0.9224 & 0.3863 \end{bmatrix}, \quad \Lambda \approx \begin{bmatrix} 9.508 & 0.0 & 0.0 \\ 0.0 & 0.7729 & 0.0 \end{bmatrix},$$

$$V \approx \begin{bmatrix} -0.4287 & 0.806 & 0.4082 \\ -0.5663 & 0.1124 & -0.8165 \\ -0.7039 & -0.5812 & 0.4082 \end{bmatrix}.$$

- The pseudoinverse is given by

$$X^\dagger = V\Lambda^\dagger U^\top = \begin{bmatrix} -17/18 & 4/9 \\ -1/9 & 1/9 \\ 13/18 & -2/9 \end{bmatrix}, \quad \text{where } \Lambda^\dagger \approx \begin{bmatrix} 1/9.508 & 0.0 \\ 0.0 & 1/0.7729 \\ 0.0 & 0.0 \end{bmatrix}.$$

- We have $XX^\dagger = I$, but $X^\dagger X \neq I$.

- Let $X = \begin{bmatrix} 3 & 1 \\ 2 & 7 \\ 1 & 5 \end{bmatrix}$. Then the SVD gives

$$U \approx \begin{bmatrix} -0.2068 & 0.9702 & 0.1261 \\ -0.803 & -0.0947 & -0.5885 \\ -0.559 & -0.2229 & 0.7986 \end{bmatrix}, \quad \Lambda \approx \begin{bmatrix} 9.0613 & 0.0 \\ 0.0 & 2.6255 \\ 0.0 & 0.0 \end{bmatrix},$$

$$V \approx \begin{bmatrix} -0.3074 & 0.9516 \\ -0.9516 & -0.3074 \end{bmatrix}.$$

Eigenvalues and Eigenvectors

- Suppose that $X = U\Lambda V^\top$ is a SVD.

  Then

  $$X^\top X = V\Lambda^\top U^\top U\Lambda V^\top = V(\Lambda^\top\Lambda)V^\top,$$

  where $\Lambda^\top\Lambda$ is a diagonal matrix.

  Hence

  $$(X^\top X)V = V(\Lambda^\top\Lambda).$$

- eigenvectors of $X^\top X =$ columns of V

  eigenvalues of $X^\top X =$ diagonal entries of $\Lambda^\top\Lambda$
- Recall that we used an eigenvector of $X^\top X$ in the PCA example.

- Similarly,

$$XX^\top = U\Lambda V^\top V\Lambda^\top U^\top = U(\Lambda\Lambda^\top)U^\top$$

$$(XX^\top)U = U(\Lambda\Lambda^\top)$$

- eigenvectors of $XX^\top =$ columns of U
  eigenvalues of $XX^\top =$ diagonal entries of $\Lambda\Lambda^\top$

Example

- Let $X = \begin{bmatrix} 3 & 1 \\ 2 & 7 \\ 1 & 5 \end{bmatrix}$. Then $X^\top X = \begin{bmatrix} 14 & 22 \\ 22 & 75 \end{bmatrix}$.

  Recall that $\Lambda \approx \begin{bmatrix} 9.0613 & 0.0 \\ 0.0 & 2.6255 \\ 0.0 & 0.0 \end{bmatrix}$.

  The eigenvalues are given by $\Lambda^\top \Lambda \approx \begin{bmatrix} 82.1065 & 0.0 \\ 0.0 & 6.8935 \end{bmatrix}$;

  the eigenvectors are given by $V \approx \begin{bmatrix} -0.3074 & 0.9516 \\ -0.9516 & -0.3074 \end{bmatrix}$.

Linear Discriminant Analysis

Multi-class classification

$$\boldsymbol{x} = (x_1, \dots, x_k) \rightsquigarrow P(t|\boldsymbol{x}), \qquad t = 1, 2, \dots, s$$

- We studied logistic regression.
- The Linear Discriminant Analysis (LDA) is based on Bayesian inference.
- LDA can be used for both supervised and unsupervised learning.

- Recall the density function f of the normal distribution is given by
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$
- K-dimensional Gaussian distribution:
$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{K/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^\top \right\}$$
where the K-dimensional vector $\boldsymbol{\mu}$ is the mean, the $K \times K$ matrix Σ is the covariance, and $|\Sigma|$ is the determinant of Σ.

- $\pi_t := P(t)$ prior probability that an observation belongs to class t

  $f_t(\boldsymbol{x}) := P(\boldsymbol{x}|t)$ likelihood

  Bayes' Theorem:

$$P(t|\boldsymbol{x}) \propto P(t)P(\boldsymbol{x}|t) = \pi_t \, f_t(\boldsymbol{x})$$

- LDA assumes

$$\boxed{\begin{array}{l}\text{(1) } f_t(\boldsymbol{x}) \text{ is normal, i.e., } f_t(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_t, \Sigma_t) \\ \text{(2) all the covariances are the same,} \\ \qquad\qquad\qquad\qquad\qquad\text{i.e., } \Sigma := \Sigma_1 = \cdots = \Sigma_s.\end{array}}$$

- Recall

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_t, \Sigma) = \frac{1}{(2\pi)^{K/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_t)\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_t)^\top\right\}.$$

- We have

$$\begin{aligned}\ln P(t|\boldsymbol{x}) &= \ln \pi_t + \ln f_t(\boldsymbol{x}) + \text{(terms without } t\text{)} \\ &= \ln \pi_t - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_t)\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_t)^\top + \text{(terms without } t\text{)} \\ &= \ln \pi_t - \frac{1}{2}\boldsymbol{\mu}_t\Sigma^{-1}\boldsymbol{\mu}_t^\top + \boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_t^\top + \text{(terms without } t\text{)}.\end{aligned}$$

(Note that $\boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_t^\top$ is scalar and $\boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_t^\top = (\boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_t^\top)^\top = \boldsymbol{\mu}_t\Sigma^{-1}\boldsymbol{x}^\top$.)

- The part (terms without t) does not contribute when we compare $\ln P(t|\boldsymbol{x})$ for $t = 1, \dots, s$.

- Define the discriminant function by

$$\delta_t(\boldsymbol{x}) := \ln \pi_t - \frac{1}{2}\boldsymbol{\mu}_t \Sigma^{-1} \boldsymbol{\mu}_t^\top + \boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_t^\top$$

  for $t = 1, \dots, s$.

- Given $\boldsymbol{x}$, if $\delta_{t_*}(\boldsymbol{x})$ is the largest,

  observation $\boldsymbol{x}$ belongs to class t_* with largest probability.

  $\leadsto$ Classification

- In LDA, we use the training data to approximate $\delta_t(\boldsymbol{x})$.

Given $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2 \sqcup \cdots \sqcup \mathcal{D}_s$ (disjoint union),

set

$$N_t := \#(\mathcal{D}_t), \quad t = 1, 2, \dots, s, \qquad N := N_1 + \cdots + N_s.$$

We make the following estimates:

$$\begin{aligned}
\hat{\pi}_t &= N_t/N, \\
\hat{\boldsymbol{\mu}}_t &= \frac{1}{N_t} \sum_{\boldsymbol{x} \in \mathcal{D}_t} \boldsymbol{x}, \\
\hat{\Sigma} &= \frac{1}{N-s} \sum_{t=1}^{s} \sum_{\boldsymbol{x} \in \mathcal{D}_t} (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_t)^\top (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_t).
\end{aligned}$$

The use of $N - s$ in $\hat{\Sigma}$ is called Bessel's correction.

Let $\hat{X}_0$ be the $N \times K$ matrix whose rows are $\boldsymbol{x} - \hat{\boldsymbol{\mu}}_t$.

Then we have

$$\boxed{\hat{\Sigma} = \frac{1}{N-s} \hat{X}_0^\top \hat{X}_0}.$$

- An approximation of the discriminant function is given by

$$\hat{\delta}_t(\boldsymbol{x}) := \ln \hat{\pi}_t - \frac{1}{2}\hat{\boldsymbol{\mu}}_t \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_t^\top + \boldsymbol{x} \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_t^\top .$$

- When $\pi_1 = \pi_2 = \cdots = \pi_s$, the decision boundaries are given by

$$-\frac{1}{2}\hat{\boldsymbol{\mu}}_i \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_i^\top + \boldsymbol{x} \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_i^\top = -\frac{1}{2}\hat{\boldsymbol{\mu}}_j \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_j^\top + \boldsymbol{x} \hat{\Sigma}^{-1} \hat{\boldsymbol{\mu}}_j^\top$$

  for $i \neq j$.

Dimensionality Reduction through LDA

- We studied dimensionality reduction through PCA.
- $\boxed{\text{LDA can be used for dimensionality reduction.}}$
  It can be considered as a “supervised PCA”.

- Given $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2 \sqcup \cdots \sqcup \mathcal{D}_s$ (disjoint union),
  set

$$N_t := \#(\mathcal{D}_t), \quad t = 1, 2, \ldots, s, \qquad N := N_1 + \cdots + N_s.$$

- Main Idea: Shrink class $\mathcal{D}_t$ into a single point $\boldsymbol{\mu}_t$ and do PCA.

$$\rightsquigarrow \quad \text{Principal directions for classes.}$$

Set

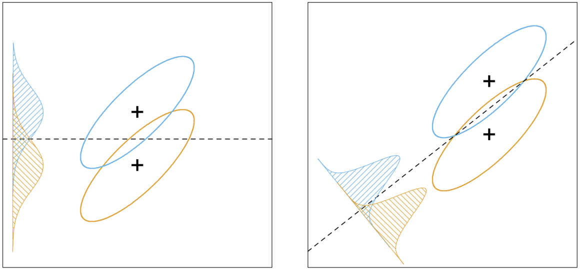
$$\boldsymbol{\mu}_t = \frac{1}{N_t} \sum_{\boldsymbol{x} \in \mathcal{D}_t} \boldsymbol{x} \quad \text{and} \quad \boldsymbol{\mu} = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{D}} \boldsymbol{x}.$$

However, the point $\boldsymbol{\mu}_t$ must have multiplicity N_t.

- Find a vector $\boldsymbol{a}$ which maximizes

$$\boldsymbol{a}^\top \left(\sum_{t=1}^{s} N_t (\boldsymbol{\mu}_t - \boldsymbol{\mu})^\top (\boldsymbol{\mu}_t - \boldsymbol{\mu}) \right) \boldsymbol{a}.$$

- Constraint: Overlap between classes needs to be controlled to achieve better class separation.



[Hastie, Tibshirani and Friedman, *The elements of statistical learning*]

Although the line joining the centroids defines the direction of greatest centroid spread, there is considerable overlap between the projected classes.

- A vector $\boldsymbol{a}$ may capture more variability from one class than from the other classes.
- To control this, we fix the (weighted) sum of variances. That is,

$$\sum_{t=1}^{s} \boldsymbol{a}^\top \left(\sum_{\boldsymbol{x}\in\mathcal{D}_t} (\boldsymbol{x}-\boldsymbol{\mu}_t)^\top (\boldsymbol{x}-\boldsymbol{\mu}_t) \right) \boldsymbol{a} = 1.$$

- In short, our goal is to maximize the between-class variance relative to the within-class variance.

- Define

$$B = \sum_{t=1}^{s} N_t(\boldsymbol{\mu}_t - \boldsymbol{\mu})^\top(\boldsymbol{\mu}_t - \boldsymbol{\mu})$$

$$W = \sum_{t=1}^{s} \sum_{\boldsymbol{x}\in\mathcal{D}_t} (\boldsymbol{x} - \boldsymbol{\mu}_t)^\top(\boldsymbol{x} - \boldsymbol{\mu}_t).$$

Let X_0 be the $N \times K$ matrix whose rows are $\boldsymbol{x} - \boldsymbol{\mu}_t$.

Then we have

$$\boxed{W = X_0^\top X_0}.$$

- Task: Maximize $\boldsymbol{a}^\top B\boldsymbol{a}$, subject to $\boldsymbol{a}^\top W\boldsymbol{a} = 1$.
- Lagrange Multiplier

$$f(\boldsymbol{a}) = \boldsymbol{a}^\top B\boldsymbol{a}, \qquad g(\boldsymbol{a}) = \boldsymbol{a}^\top W\boldsymbol{a}$$

- We obtain $\nabla f = 2B\boldsymbol{a}$, $\nabla g = 2W\boldsymbol{a}$, and

$$B\boldsymbol{a} = \lambda W\boldsymbol{a} \quad \text{for some } \lambda \quad \Longleftrightarrow \quad \boxed{W^{-1}B\boldsymbol{a} = \lambda\boldsymbol{a}}.$$

  Critical points are eigenvectors of $W^{-1}B$.

- Note that $W^{-1}B$ is symmetric.

  Thus the Spectral Theorem applies.

- If $\boldsymbol{a}$ is an eigenvector of $W^{-1}B$ such that $\boldsymbol{a}^\top W\boldsymbol{a} = 1$, then

  $$\boldsymbol{a}^\top B\boldsymbol{a} = \boldsymbol{a}^\top WW^{-1}B\boldsymbol{a} = \lambda\boldsymbol{a}^\top W\boldsymbol{a} = \lambda.$$

  Recall that we are maximizing $\boldsymbol{a}^\top B\boldsymbol{a}$.

  Thus an eigenvector of the largest eigenvalue is the first principal direction.

- Take D-many principal directions for $D < K$.

  Project data points onto the subspace of the principal directions.

  $\rightsquigarrow$ Dimensionality Reduction

Support Vector Machines

- Assume that there are two sets $\mathcal{A}_+$ and $\mathcal{A}_-$ of datapoints.
- Suppose that $\mathcal{A}_+$ and $\mathcal{A}_-$ can be separated by a hyperplane.
- That is, there is a hyperplane H given by

$$f(\boldsymbol{x}) = f(x_1, \dots, x_K) = 0,$$

where $\quad f(\boldsymbol{x}) = w_0 + w_1 x_1 + \cdots + w_K x_K = w_0 + \boldsymbol{x}\boldsymbol{w} \quad$ and

$$\boldsymbol{w} = [w_1, \dots, w_K]^\top,$$

such that the points $\boldsymbol{x}$ in $\mathcal{A}_+$ satisfy $f(\boldsymbol{x}) > 0$ and

those $\boldsymbol{x}$ in $\mathcal{A}_-$ satisfy $f(\boldsymbol{x}) < 0$.

- The vector $\boldsymbol{w}$ is orthogonal (or normal) to the hyperplane H.

  Proof. Assume $f(\boldsymbol{a}) = f(\boldsymbol{b}) = 0$. Then

$$(\boldsymbol{a}-\boldsymbol{b})\boldsymbol{w} = (w_0 + \boldsymbol{a}\boldsymbol{w}) - (w_0 + \boldsymbol{b}\boldsymbol{w}) = f(\boldsymbol{a}) - f(\boldsymbol{b}) = 0. \qquad \square$$

- For $\boldsymbol{x} \in \mathbb{R}^K$, the distance from $\boldsymbol{x}$ to H is

$$\frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|}.$$

  Proof. Assume $\boldsymbol{a} \in H$. The distance between $\boldsymbol{x}$ and H is the length of $\boldsymbol{x}-\boldsymbol{a}$ in the $\boldsymbol{w}$-direction, i.e.

$$\left|(\boldsymbol{x}-\boldsymbol{a})\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right| = \frac{|w_0 + \boldsymbol{x}\boldsymbol{w} - (w_0 + \boldsymbol{a}\boldsymbol{w})|}{\|\boldsymbol{w}\|} = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|}. \qquad \square$$

- We have $w_0 + \boldsymbol{p}w > 0 > w_0 + \boldsymbol{q}w$ for $\boldsymbol{p} \in \mathcal{A}_+$ and $\boldsymbol{q} \in \mathcal{A}_-$.
  Thus $(\boldsymbol{p} - \boldsymbol{q})\boldsymbol{w} > 0$ for $\boldsymbol{p} \in \mathcal{A}_+$ and $\boldsymbol{q} \in \mathcal{A}_-$.

- Find $\boldsymbol{a} \in \mathcal{A}_+$ such that $\boldsymbol{aw} \leq \boldsymbol{pw}$ for all $\boldsymbol{p} \in \mathcal{A}_+$.
  Define $w_0^+ = -\boldsymbol{aw}$ and $f_{\boldsymbol{w},+}(\boldsymbol{x}) = w_0^+ + \boldsymbol{xw}$.

- Similarly, find $\boldsymbol{b} \in \mathcal{A}_-$ such that $\boldsymbol{bw} \geq \boldsymbol{qw}$ for all $\boldsymbol{q} \in \mathcal{A}_-$.
  Define $w_0^- = -\boldsymbol{bw}$ and $f_{\boldsymbol{w},-}(\boldsymbol{x}) = w_0^- + \boldsymbol{xw}$.

- We have
  $$\boxed{(\boldsymbol{p} - \boldsymbol{q})w \geq (\boldsymbol{a} - \boldsymbol{b})w = w_0^- - w_0^+ > 0}$$
  for $\boldsymbol{p} \in \mathcal{A}_+$ and $\boldsymbol{q} \in \mathcal{A}_-$.

- $H_{\boldsymbol{w},+}$: hyperplane defined by $f_{\boldsymbol{w},+}(\boldsymbol{x}) := w_0^+ + \boldsymbol{x}\boldsymbol{w} = 0$
  We have $f_{\boldsymbol{w},+}(\boldsymbol{x}) \geq 0$ for $\boldsymbol{x} \in \mathcal{A}_+$ and $\exists\ \boldsymbol{x} \in \mathcal{A}_+ \cap H_{\boldsymbol{w},+}$.

- $H_{\boldsymbol{w},-}$: hyperplane defined by $f_{\boldsymbol{w},-}(\boldsymbol{x}) := w_0^- + \boldsymbol{x}\boldsymbol{w} = 0$
  We have $f_{\boldsymbol{w},-}(\boldsymbol{x}) \leq 0$ for $\boldsymbol{x} \in \mathcal{A}_-$ and $\exists\ \boldsymbol{x} \in \mathcal{A}_- \cap H_{\boldsymbol{w},-}$.

Definition

Let $\mathcal{A} \subset \mathbb{R}^K$. A hyperplane H defined by $f(\boldsymbol{x}) = 0$ is called a supporting hyperplane of $\mathcal{A}$ if $f(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathcal{A}$ or $f(\boldsymbol{x}) \leq 0$ for all $\boldsymbol{x} \in \mathcal{A}$ and $\exists$ $\boldsymbol{x} \in \mathcal{A} \cap H$.

- By this definition, $H_{\boldsymbol{w},+}$ and $H_{\boldsymbol{w},-}$ are supporting hyperplanes of $\mathcal{A}_+$ and $\mathcal{A}_-$, respectively.

- Define $d(\boldsymbol{w})$ to be the distance between $H_{\boldsymbol{w},+}$ and $H_{\boldsymbol{w},-}$. Then we have
$$d(\boldsymbol{w}) = \frac{|w_0^+ - w_0^-|}{\|w\|}.$$

Proof. Assume $\boldsymbol{q} \in H_{\boldsymbol{w},-}$. Then $d(\boldsymbol{w})$ is equal to the distance between $\boldsymbol{q}$ and $H_{\boldsymbol{w},+}$. Thus we have

$$\begin{aligned} d(\boldsymbol{w}) &= \frac{|f_{\boldsymbol{w},+}(\boldsymbol{q})|}{\|w\|} = \frac{|w_0^+ + \boldsymbol{q}\boldsymbol{w}|}{\|w\|} = \frac{|w_0^- + \boldsymbol{q}\boldsymbol{w} + (w_0^+ - w_0^-)|}{\|w\|} \\ &= \frac{|w_0^+ - w_0^-|}{\|w\|}. \qquad \square \end{aligned}$$

- The hyperplane $H_{\boldsymbol{w}}$ defined by

$$\frac{w_0^+ + w_0^-}{2} + \boldsymbol{x}\boldsymbol{w} = 0$$

is at the same distance from $H_{\boldsymbol{w},+}$ and $H_{\boldsymbol{w},-}$.

Proof. Assume $\boldsymbol{p} \in H_{\boldsymbol{w}}$.

$$\frac{|w_0^+ + \boldsymbol{p}\boldsymbol{w}|}{\|\boldsymbol{w}\|} = \frac{\left|\frac{w_0^+ + w_0^-}{2} + \frac{w_0^+ - w_0^-}{2} + \boldsymbol{p}\boldsymbol{w}\right|}{\|\boldsymbol{w}\|} = \frac{\left|\frac{w_0^+ - w_0^-}{2}\right|}{\|\boldsymbol{w}\|}$$

$$\frac{|w_0^- + \boldsymbol{p}\boldsymbol{w}|}{\|\boldsymbol{w}\|} = \frac{\left|\frac{w_0^+ + w_0^-}{2} + \frac{w_0^- - w_0^+}{2} + \boldsymbol{p}\boldsymbol{w}\right|}{\|\boldsymbol{w}\|} = \frac{\left|\frac{w_0^- - w_0^+}{2}\right|}{\|\boldsymbol{w}\|} \qquad \square$$

- Now let $\boldsymbol{w}$ vary.
- What is the best $\boldsymbol{w}$ for separating $\mathcal{A}_+$ from $\mathcal{A}_-$?
- Define the optimal margin $d(\mathcal{A}_+, \mathcal{A}_-)$ by

$$d(\mathcal{A}_+, \mathcal{A}_-) := \max_{\boldsymbol{w}} d(\boldsymbol{w}).$$

- When $d(\boldsymbol{w}_*) = d(\mathcal{A}_+, \mathcal{A}_-)$, the hyperplane H_* defined by

$$H_* : \quad \frac{w_0^+ + w_0^-}{2} + \boldsymbol{x}\boldsymbol{w}_* = 0$$

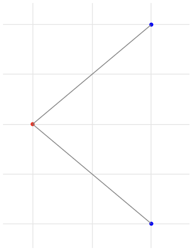
  is called the classifying hyperplane.
- The hyperplane H_* is at the same distance from $H_{\boldsymbol{w}_*,+}$ and $H_{\boldsymbol{w}_*,-}$.

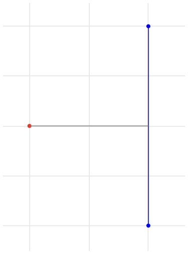
Question: How to find the classifying hyperplane?

Naive Idea: Find $\boldsymbol{a} \in \mathcal{A}_+$ and $\boldsymbol{b} \in \mathcal{A}_-$ be such that

$$\|\boldsymbol{a} - \boldsymbol{b}\| \le \|\boldsymbol{p} - \boldsymbol{q}\| \quad \text{for all } \boldsymbol{p} \in \mathcal{A}_+, \boldsymbol{q} \in \mathcal{A}_-.$$

Set $\boldsymbol{w} = (\boldsymbol{a} - \boldsymbol{b})^\top$ and use $\boldsymbol{w}$ to construct the classifying hyperplane.





This idea works if we consider the line!

- Recall a set $\mathcal{X} \subset \mathbb{R}^K$ is said to be convex if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,

$$\{\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} : 0 \le \lambda \le 1\} \subseteq \mathcal{X},$$

  or equivalently,

$$\{\lambda_1 \boldsymbol{x} + \lambda_2 \boldsymbol{y} : \lambda_1 + \lambda_2 = 1, \lambda_i \in \mathbb{R}_{\ge 0}\} \subseteq \mathcal{X}.$$

- Let $\mathcal{A} \subset \mathbb{R}^K$. The convex hull $C(\mathcal{A})$ of $\mathcal{A}$ is defined to be the smallest convex set that contains $\mathcal{A}$.
- When $\mathcal{A} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, the convex hull $C(\mathcal{A})$ is given by

$$C(\mathcal{A}) = \left\{ \sum_{i=1}^{N} \lambda_i \boldsymbol{x}_i : \sum_{i=1}^{N} \lambda_i = 1,\ \lambda_i \in \mathbb{R}_{\ge 0} \right\}.$$

  Proof. Exercise $\square$

Lemma

Let $\mathcal{A} \subset \mathbb{R}^K$ be a finite set. Then a hyperplane H is a supporting hyperplane of $\mathcal{A}$ if and only if it is a supporting hyperplane of $C(\mathcal{A})$.

Proof. Let $\mathcal{A} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$.

$\Rightarrow$) Assume H is a supporting hyperplane of $\mathcal{A}$ defined by $f(\boldsymbol{x}) = w_0 + \boldsymbol{x}\boldsymbol{w} = 0$. WLOG, suppose $f(\boldsymbol{x}) \geq 0$ for $\boldsymbol{x} \in \mathcal{A}$. Let $\sum \lambda_i \boldsymbol{x}_i \in C(\mathcal{A})$ with $\sum \lambda_i = 1$ and $\lambda_i \geq 0$. Then

$$f\left(\sum \lambda_i \boldsymbol{x}_i\right) = w_0 + \sum \lambda_i \boldsymbol{x}_i \boldsymbol{w} = \sum \lambda_i (w_0 + \sum \boldsymbol{x}_i \boldsymbol{w}) \geq 0.$$

There exists $\boldsymbol{x} \in \mathcal{A} \cap H \subset C(\mathcal{A}) \cap H$. Thus H is a supporting hyperplane of $C(\mathcal{A})$.

$\Leftarrow$) Assume H is a supporting hyperplane of $C(\mathcal{A})$ defined by $f(\boldsymbol{x}) = w_0 + \boldsymbol{x}\boldsymbol{w} = 0$. WLOG, suppose $f(\boldsymbol{x}) \geq 0$ for $\boldsymbol{x} \in C(\mathcal{A})$. Then $f(\boldsymbol{x}) \geq 0$ for $\boldsymbol{x} \in \mathcal{A} \subset C(\mathcal{A})$. Suppose $f(\sum \lambda_i \boldsymbol{x}_i) = 0$ with $\sum \lambda_i = 1$ and $\lambda_i \geq 0$. Then

$$f\left(\sum \lambda_i \boldsymbol{x}_i\right) = \sum \lambda_i (w_0 + \boldsymbol{x}_i \boldsymbol{w}) = 0.$$

We must have $\lambda_{i_*} > 0$ for some i_*, and $w_0 + \boldsymbol{x}_{i_*}\boldsymbol{w} = 0$. Thus H is a supporting hyperplane of $\mathcal{A}$. $\square$

Proposition

Let $\boldsymbol{p} \in C(\mathcal{A}_+)$ and $\boldsymbol{q} \in C(\mathcal{A}_-)$. Then

$$\|\boldsymbol{p} - \boldsymbol{q}\| \geq d(\mathcal{A}_+, \mathcal{A}_-).$$

Proof. For any $\boldsymbol{w}$, we have

$$\|\boldsymbol{p} - \boldsymbol{q}\| \, \|\boldsymbol{w}\| \geq |(\boldsymbol{p} - \boldsymbol{q})\boldsymbol{w}| \geq |w_0^+ - w_0^-|$$

(the proof the second inequality is almost the same as the one for $\mathcal{A}_+$ and $\mathcal{A}_-$ since $C(\mathcal{A}_+)$ and $C(\mathcal{A}_-)$ are compact) and hence

$$\|\boldsymbol{p} - \boldsymbol{q}\| \geq \frac{|w_0^+ - \boldsymbol{w}_0^-|}{\|\boldsymbol{w}\|} = d(\boldsymbol{w}).$$

In particular, $\|\boldsymbol{p} - \boldsymbol{q}\| \geq d(\mathcal{A}_+, \mathcal{A}_-)$. $\square$

Theorem

Let $\boldsymbol{a} \in C(\mathcal{A}_+)$ *and* $\boldsymbol{b} \in C(\mathcal{A}_-)$ *be such that*

$$\|\boldsymbol{a} - \boldsymbol{b}\| \leq \|\boldsymbol{p} - \boldsymbol{q}\| \quad \text{for all } \boldsymbol{p} \in C(\mathcal{A}_+), \boldsymbol{q} \in C(\mathcal{A}_-).$$

Set $\boldsymbol{w} = (\boldsymbol{a} - \boldsymbol{b})^\top$, $w_0^+ = -\boldsymbol{a}\boldsymbol{w}$ *and* $w_0^- = -\boldsymbol{b}\boldsymbol{w}$. *Then the supporting hyperplanes* $H_{\boldsymbol{w},+}$ *and* $H_{\boldsymbol{w},-}$ *of* $\mathcal{A}_+$ *and* $\mathcal{A}_-$ *are respectively defined by*

$$w_0^+ + \boldsymbol{x}\boldsymbol{w} = 0 \quad \text{and} \quad w_0^- + \boldsymbol{x}\boldsymbol{w} = 0,$$

and the optimal margin is equal to

$$d(\mathcal{A}_+, \mathcal{A}_-) = \|\boldsymbol{a} - \boldsymbol{b}\| = \frac{|w_0^+ - w_0^-|}{\|\boldsymbol{w}\|}.$$

Proof. For the first part, we need to show $\boldsymbol{aw} \le \boldsymbol{pw}$ for all $\boldsymbol{p} \in C(\mathcal{A}_+)$ and $\boldsymbol{bw} \ge \boldsymbol{qw}$ for all $\boldsymbol{q} \in C(\mathcal{A}_-)$. Then, from the previous discussion, the hyperplanes $H_{\boldsymbol{w},+}$ and $H_{\boldsymbol{w},-}$ are the supporting hyperplanes of $\mathcal{A}_+$ and $\mathcal{A}_-$, respectively. Suppose $\exists\, \boldsymbol{p} \in C(\mathcal{A}_+)$ such that $\boldsymbol{aw} > \boldsymbol{pw}$. Consider $(1-\lambda)\boldsymbol{a} + \lambda\boldsymbol{p} \in C(\mathcal{A}_+)$ for $0 \le \lambda \le 1$. Define

$$g(\lambda) = \|(1-\lambda)\boldsymbol{a} + \lambda\boldsymbol{p} - \boldsymbol{b}\|^2 = \|\lambda(\boldsymbol{p}-\boldsymbol{a}) + \boldsymbol{a} - \boldsymbol{b}\|^2.$$

Then

$$g'(0) = 2(\boldsymbol{p}-\boldsymbol{a})(\boldsymbol{a}-\boldsymbol{b})^\top = 2(\boldsymbol{p}-\boldsymbol{a})\boldsymbol{w} < 0.$$

This is a contradiction because $g(\lambda)$ has a minimum when $\lambda = 0$. Thus $\boldsymbol{aw} \le \boldsymbol{pw}$ for all $\boldsymbol{p} \in C(\mathcal{A}_+)$. Similarly, $\boldsymbol{bw} \ge \boldsymbol{qw}$ for all $\boldsymbol{q} \in C(\mathcal{A}_-)$.

For the second part, it follows from Proposition that

$$d(\mathcal{A}_+, \mathcal{A}_-) \geq \frac{|w_0^+ - w_0^-|}{\|\boldsymbol{w}\|} = \frac{|(\boldsymbol{a} - \boldsymbol{b})\boldsymbol{w}|}{\|\boldsymbol{w}\|}$$

$$= \frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} = \|\boldsymbol{w}\| = \|\boldsymbol{a} - \boldsymbol{b}\| \geq d(\mathcal{A}_+, \mathcal{A}_-).$$

$\square$

Task: Find $\boldsymbol{a} \in C(\mathcal{A}_+)$ and $\boldsymbol{b} \in C(\mathcal{A}_-)$ such that

$$\|\boldsymbol{a} - \boldsymbol{b}\| \leq \|\boldsymbol{p} - \boldsymbol{q}\| \quad \text{for all } \boldsymbol{p} \in C(\mathcal{A}_+), \boldsymbol{q} \in C(\mathcal{A}_-).$$

- There is an algorithm for this task called Sequential Minimal Optimization invented by John Platt in 1998.
- We will use the implementation in scikit-learn which in turn uses the LIBSVM library.

Inseparable sets

Frequently, two sets $\mathcal{A}_+$ and $\mathcal{A}_-$ are not separable by a hyperplane.

- Option 1: Shrink the convex hulls by cosidering points $\sum \lambda_i \boldsymbol{x}_i$ where $\sum_i \lambda_i = 1$ and $0 \le \lambda_i \le \alpha$ for some $\alpha < 1$. This excludes outliers.
- Option 2: Use a nonlinear kernel. That is, use nonlinear hypersurfaces.

Multi-class classification

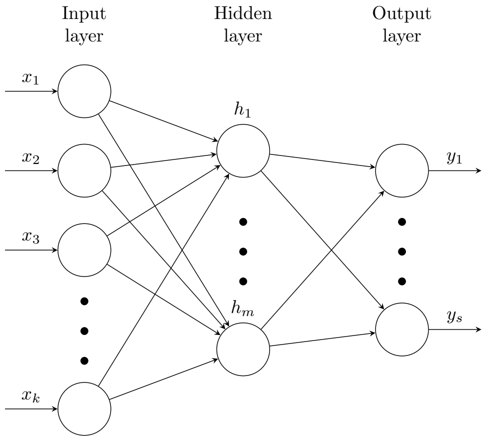
Question: How can we generalize SVM to multi-class cases?

Assume that there are s classes.

- One-versus-One: Compute all $\binom{s}{2}$ pairwise classifiers. For each test point, the predicted class is the one that wins the most pairwise contests.
- One-versus-All: Each class is compared to all the others in s two-class comparisons. For each test point, compute the value of the separating hyperplane function $f(\boldsymbol{x})$ for each of the s classifiers. The predicted class is the one with the largest value.

Neural Networks

- Neural Networks: mimicking the operation of neurons in the brain.



- Form a graph with input nodes and output nodes,
  and with hidden layers of extra nodes between them.
- The output of each layer is the input to the next layer.
  The forwarded input goes through an activation at each hidden layer.
- Each node is meant to play the role of a neuron in the brain.
- The neural network is also called the multi-layer perceptron or MLP, in short.

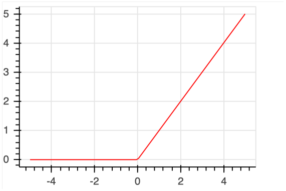
- In each layer of a typical neural network, the process is similar to a logistic regression.

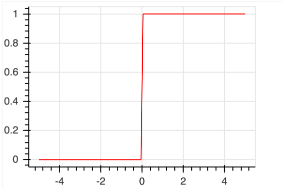
  Neural networks $\approx$ multiple layers of logistic regressions

- Activation functions are given

  by a sigmoid or a rectified linear unit (relu).

$$r(x) = \max(x, 0), \quad r'(x) = u_0(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x < 0. \end{cases}$$





Set-up

- Multi-class Classification:

$$\mathcal{D} = \{(x_1, \dots, x_k; t)\}, \quad t \in \{1, 2, \dots, s\}$$

- Build a network with k input nodes and s output nodes.
  Put one hidden layer with m nodes.
- Each output node will produce probability for the input to be in the corresponding class.
  (Multi-class logistic regression = one-layer neural network)

For simplicity, we take $N = 1$ in what follows.

$$\begin{aligned}
\text{input:} \quad & \boldsymbol{x} = [1, x_1, \ldots, x_k] \\
\text{hidden layer:} \quad & \boldsymbol{x}\boldsymbol{w}^{(1)} = [z_1, \ldots, z_m] \\
& \qquad\qquad \text{with } \boldsymbol{w}^{(1)} \text{ of size } (k+1) \times m \\
& \text{activation: } \sigma(z_i) \quad (\sigma : \text{sigmoid}) \\
& \boldsymbol{h} = [1, h_1, \cdots, h_m] \\
& \quad = [1, \sigma(z_1), \ldots, \sigma(z_m)] \\
\text{output:} \quad & \boldsymbol{y} = [y_1, \ldots, y_s] = \boldsymbol{\sigma}(\boldsymbol{h}\boldsymbol{w}^{(2)}) \quad (\boldsymbol{\sigma} : \text{softmax}) \\
& \qquad\qquad \text{with } \boldsymbol{w}^{(2)} \text{ of size } (m+1) \times s
\end{aligned}$$

As in logistic regression, we want to learn
the best values of $\boldsymbol{w}^{(\ell)}$ ($\ell = 1, 2$) using the training data.

- We have
$$\boxed{\boldsymbol{x} \rightsquigarrow \boldsymbol{h} \rightsquigarrow \boldsymbol{y}}.$$
This process is considered as forward propagation.
- We can develop more general neural networks by considering more complex directed graphs with many layers and by adopting activation functions different from σ.
- However, there should be <u>no oriented cycles</u> in the directed graph to ensure that the outputs are deterministic functions of the inputs. In other words, a network must be feed-forward.

- The likelihood function is given by

$$L(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}) = \prod_{i=1}^{s} y_i^{t_i},$$

  where t is identified with a binary vector $\mathbf{t} = [t_1, \dots, t_n]$.

- Cross-entropy

$$\boxed{E(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}) = -\sum_{i=1}^{s} t_i \ln y_i}$$

- We will use gradient descent and need to take derivatives.

  In particular, we need to use the chain rule inductively.

$$\nabla_{\boldsymbol{w}^{(2)}} E \rightsquigarrow \nabla_{\boldsymbol{w}^{(1)}} E$$

- This process is called backpropagation.

- Recall that the softmax $\boldsymbol{\sigma} : \mathbb{R}^s \to (0,1)^s$ is defined by

$$\boldsymbol{\sigma}(\boldsymbol{a}) = \left(\frac{e^{a_1}}{\sum_{i=1}^{s} e^{a_i}}, \dots, \frac{e^{a_s}}{\sum_{i=1}^{s} e^{a_i}} \right),$$

  where $\boldsymbol{a} = (a_1, a_2, \dots, a_s)$.

- Write $\boldsymbol{y} = [y_1, \dots, y_s] = \boldsymbol{\sigma}(\boldsymbol{a})$, and

  for $i, j = 1, \dots, s$,

$$\boxed{\frac{\partial y_j}{\partial a_i} = y_j(\delta_{i,j} - y_i)}.$$

- Write $\boldsymbol{x}\boldsymbol{w}^{(1)} = [z_1^{(1)}, \dots, z_m^{(1)}]$ and $\boldsymbol{h}\boldsymbol{w}^{(2)} = [z_1^{(2)}, \dots, z_s^{(2)}]$. In particular,

$$z_i^{(2)} = \sum_r h_r w_{r,i}^{(2)}.$$

- As in multi-class logistic regression,

$$\frac{\partial E}{\partial z_i^{(2)}} = -\sum_{j=1}^{s} t_j \frac{1}{y_j} \frac{\partial y_j}{\partial z_i^{(2)}} = -\sum_{j=1}^{s} t_j(\delta_{i,j} - y_i) = y_i - t_i,$$

$$\frac{\partial E}{\partial w_{p,q}^{(2)}} = \sum_{i=1}^{s} \frac{\partial E}{\partial z_i^{(2)}} \frac{\partial z_i^{(2)}}{\partial w_{p,q}^{(2)}} = h_p(y_q - t_q),$$

and

$$\nabla_{\boldsymbol{w}^{(2)}} E = \boldsymbol{h}^\top(\boldsymbol{y} - \mathbf{t}).$$

Next we have

$$\frac{\partial E}{\partial z_i^{(1)}} = \sum_{j=1}^{s} \frac{\partial E}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial z_i^{(1)}} = \sum_{j=1}^{s} (y_j - t_j) \sum_{\ell=1}^{m} \frac{\partial z_j^{(2)}}{\partial h_\ell} \frac{\partial h_\ell}{\partial z_i^{(1)}}$$

$$= \sum_{j=1}^{s} (y_j - t_j) \sum_{\ell=1}^{m} w_{\ell j}^{(2)} \delta_{i,\ell} h_i (1 - h_i) = h_i(1-h_i) \sum_{j=1}^{s} (y_j - t_j) w_{ij}^{(2)},$$

$$\frac{\partial E}{\partial w_{p,q}^{(1)}} = \sum_{i=1}^{m} \frac{\partial E}{\partial z_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial w_{p,q}^{(1)}} = x_p h_q (1 - h_q) \sum_{j=1}^{s} (y_j - t_j) w_{qj}^{(2)},$$

$$\text{(Note that} \quad z_i^{(1)} = \sum_r x_r w_{r,i}^{(1)}. \text{)}$$

and

$$\nabla E_{\boldsymbol{w}^{(1)}} = \boldsymbol{x}^T \left[h_q (1 - h_q) \sum_{j=1}^{s} (y_j - t_j) w_{qj}^{(2)} \right]_{q=1,\dots,m} .$$

- Write

$$\delta_i^{(a)} = \frac{\partial E}{\partial z_i^{(a)}} \quad \text{for } a = 1, 2.$$

Then

$$\delta_i^{(2)} = \frac{\partial E}{\partial z_i^{(2)}} = y_i - t_i,$$

$$\delta_i^{(1)} = \frac{\partial E}{\partial z_i^{(1)}} = h_i(1 - h_i) \sum_{j=1}^{s} (y_j - t_j) w_{ij}^{(2)}.$$

Notice that $h_i = \sigma(z_i^{(1)})$ and $h_i(1 - h_i) = \sigma'(z_i^{(1)})$. The formula

$$\boxed{\delta_i^{(1)} = \sigma'(z_i^{(1)}) \sum_{j=1}^{s} \delta_j^{(2)} w_{ij}^{(2)}}$$

is called the backpropagation formula.

$$\nabla E_{\boldsymbol{w}^{(1)}} \text{ and } \nabla E_{\boldsymbol{w}^{(2)}} \quad \rightsquigarrow \quad \text{Gradient Decent}$$

Caveats:

- The error function E is non-convex (and non-concave).
- Initialization – don't take the zero matrix for $\boldsymbol{w}^{(1)}$
- More susceptible to over-fitting – a lot more of parameters

Remarks

- Complex neural networks with multiple layers are usually called deep neural networks.
- Most deep learning models are based on convolutional neural networks.
- TensorFlow is an open-source software library for machine learning with a particular focus on deep neural networks, and Keras provides a Python interface for the TensorFlow library.
- PyToch is an open-source machine learning library designed to provide flexibility and speed for deep neural network implementation.

Convolution Neural Networks (CNN)

- Introduced by Yann LeCun in 1989
- Convolution

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-t)g(t)\,dt$$

$$(f * g)(x) = \sum_{t=-\infty}^{\infty} f(x-t)g(t)$$

- In Math 3160, when X and Y are independent,

$$f_{X+Y} = f_X * f_Y.$$

- Convolution is translation-invariant, i.e., when $\tau_a f(x) = f(x-a)$,

$$\tau_a(f * g) = \tau_a(f) * g.$$

- In Machine Learning, the convolution of 2-D arrays is defined by

$$\text{input: } X = [X_{i,j}] \qquad \text{kernel (or filter): } \boldsymbol{w} = [w_{i,j}]$$

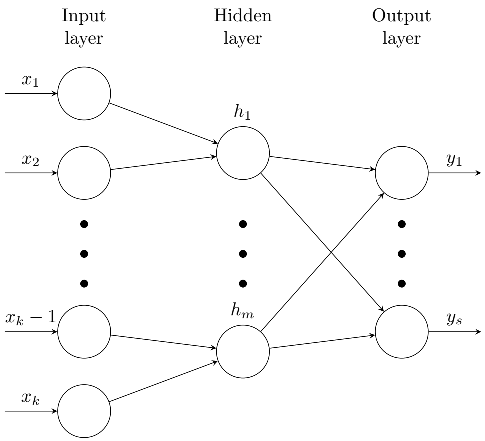
$$\text{output:} \quad (X * \boldsymbol{w})_{i,j} = \sum_{c,d} X_{i+c,j+d} w_{c,d}$$

- For example,

$$\begin{array}{|c|c|c|c|}\hline 1 & 2 & 0 & 3 \\ \hline 0 & 2 & 1 & 2 \\ \hline 1 & 0 & 0 & 2 \\ \hline 3 & 0 & 2 & 0 \\ \hline \end{array} * \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 1 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 5 & 6 & 5 \\ \hline 1 & 2 & 5 \\ \hline 4 & 4 & 2 \\ \hline \end{array}$$

- Clearly, this operation is translation-invariant.

- Convolution brings about parameter sharing and sparse connectivity.



- After convolution, pooling is performed. Two typical methods are the max pooling and the average pooling.
- Pooling is approximately translation-invariant and helps reduce noise.

  For example,

1	2	0
3	1	0
0	0	0

0	0	0
0	1	2
0	3	1

  Notice that the maximum or the average does not change.
- The result of convolution and pooling will go through activation and then move forward to the next layer.

Ideas of CNN

- Sparse connectivity

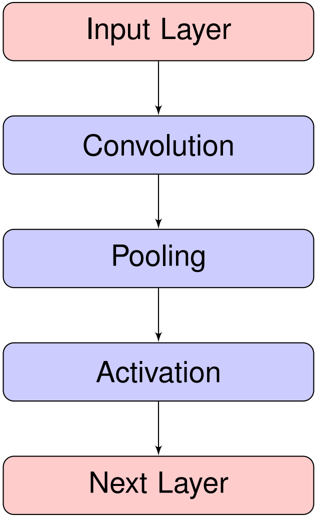
  Nearby pixels on an image are more strongly correlated than more distant pixels. Recognizing local features can enhance the performance of a classifier.

- Parameter sharing

  Any useful features that are detected in some portion of the input may be valid in other parts. The weights are shared across the layer, and the number of parameters is reduced.

- Translation invariance

  Features of images are invariant under translation.


Input Layer
Convolution
Pooling
Activation
Next Layer

Some specifics for code

- We use the max pooling and the relu function for activation.
- We do not perform padding for simplicity. In general, padding brings several benefits.
- Multiple for-loops make computations very slow.

  We need to vectorize our computations.

  For a convolution with a 2×2 kernel, we convert

$$\begin{array}{|c|c|c|c|}\hline x_{00} & x_{01} & x_{02} & x_{03} \\ \hline x_{10} & x_{11} & x_{12} & x_{13} \\ \hline x_{20} & x_{21} & x_{22} & x_{23} \\ \hline x_{30} & x_{31} & x_{32} & x_{33} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline x_{00} & x_{01} & x_{02} & x_{10} & x_{11} & x_{12} & x_{20} & x_{21} & x_{22} \\ \hline x_{01} & x_{02} & x_{03} & x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} \\ \hline x_{10} & x_{11} & x_{12} & x_{20} & x_{21} & x_{22} & x_{30} & x_{31} & x_{32} \\ \hline x_{11} & x_{13} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \\ \hline \end{array}$$

  and use a matrix multiplication once.
- We need to record where the maxima occur in the max pooling.

- We may use multiple kernels (or filters).

  Different kernels learn different features.
- In order to boost speed, we use stochastic gradient descent.

input: $\boldsymbol{x} = [x_1, \ldots, x_k]$ (no bias here)

hidden layer: convolution & pooling with filter $\boldsymbol{w}^{(1)}$

$$\rightsquigarrow [z_1^{(1)}, \ldots, z_m^{(1)}]$$

activation: $\sigma(z_i^{(1)}) = r(w_0^{(1)} + z_i^{(1)})$

$r(x)$ is the relu and $w_0^{(1)}$ is a bias.

$$\boldsymbol{h} = [1, h_1, \cdots, h_m] = [1, \sigma(z_1^{(1)}), \ldots, \sigma(z_m^{(1)})]$$

output: $\boldsymbol{y} = [y_1, \ldots, y_s] = \boldsymbol{\sigma}(\boldsymbol{hw}^{(2)})$ ($\boldsymbol{\sigma}$: softmax)

with $\boldsymbol{w}^{(2)}$ of size $(m+1) \times s$

and $\boldsymbol{hw}^{(2)} = [z_1^{(2)}, \ldots, z_s^{(2)}]$

For $\boldsymbol{w}^{(2)}$, we have the same formula

$$\nabla_{\boldsymbol{w}^{(2)}} E = \boldsymbol{h}^\top (\boldsymbol{y} - \mathbf{t}).$$

Recall the backpropagation:

$$\boxed{\delta_i^{(1)} = \sigma'(z_i^{(1)}) \sum_{j=1}^{s} \delta_j^{(2)} w_{ij}^{(2)}}.$$

$$\delta_i^{(2)} = \frac{\partial E}{\partial z_i^{(2)}} = y_i - t_i,$$

$$\delta_i^{(1)} = \frac{\partial E}{\partial z_i^{(1)}} = u_0(w_0^{(1)} + z_i^{(1)}) \sum_{j=1}^{s} (y_j - t_j) w_{ij}^{(2)}.$$

Since $\dfrac{\partial h_\ell}{\partial w_0^{(1)}} = u_0(w_0^{(1)} + z_\ell^{(1)})$ for any ℓ, we see that

$$\frac{\partial E}{\partial w_0^{(1)}} = \sum_{j=1}^{s} \frac{\partial E}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial w_0^{(1)}} = \sum_{j=1}^{s} (y_j - t_j) \sum_{\ell=1}^{m} \frac{\partial z_j^{(2)}}{\partial h_\ell} \frac{\partial h_\ell}{\partial w_0^{(1)}}$$

$$= \sum_{j=1}^{s} (y_j - t_j) \sum_{\ell=1}^{m} w_{\ell j}^{(2)} u_0(z_\ell^{(1)} + w_0^{(1)}) = \sum_{i=1}^{m} \delta_i^{(1)},$$

$$\frac{\partial E}{\partial w_{p,q}^{(1)}} = \sum_{i=1}^{m} \frac{\partial E}{\partial z_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial w_{p,q}^{(1)}} = \sum_{i=1}^{m} \delta_i^{(1)} x_{i,p,q},$$

where $x_{i,p,q}$ are determined by pooling.