III. Preliminary Knowledge: Statistical Learning

Young-geun Kim

Department of Statistics and Probability

STT 997 (SS 2025)

MICHIGAN STATE
UNIVERSITY

Outline

Statistical Learning

- The advent of large-scale and high-resolution data has brought opportunities to learn from data but also poses challenges due to their high-dimensional and complex nature.
- *Statistical learning* is a key solution that leverages functions, called *machines*, that take appropriately processed data as input. It identifies optimal functions based on desired behaviors and explores their properties to infer patterns in populations of interest.
- This lecture provides preliminary knowledge on statistical learning. The main textbook is:

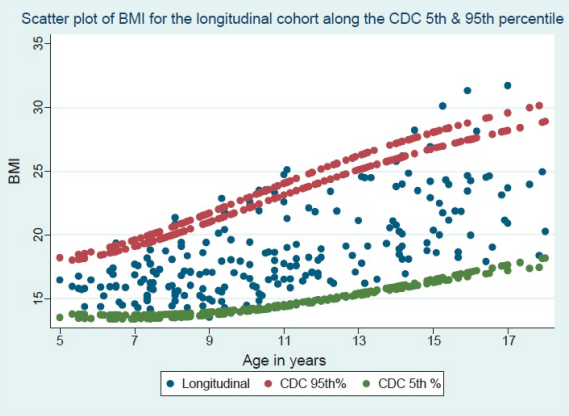
  The Elements of Statistical Learning (Hastie, 2009),

  which is available at `https://hastie.su.domains/ElemStatLearn/`.

Statistical Learning: Toy Example

- Let's begin with a toy example, with the body mass index (BMI) defined as the ratio of weight to the square of height.
- When we want to use weights (denoted as X_1) and heights (denoted as X_2) to calculate BMIs (denoted as Y), a function X_1/X_2^2 provides an accurate value of Y. In this case, $f(X_1, X_2) = X_1/X_2^2 = Y$ where f is known.
- However, in most cases, such function is unknown, and $\vec{X}$ is insufficient to determine Y. For instance, when we only observe weights, due to the uncertainty of height given weight, we cannot predict accurate BMI values.

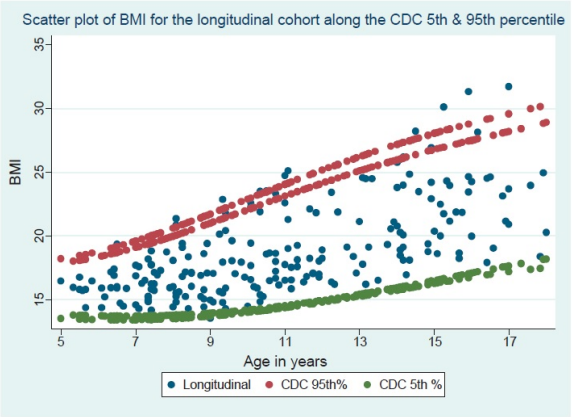
Statistical Learning: Toy Example



Q: When using age (X_3) to predict Y, what should our target function to learn, $f(X_3)$, be?

The figure is from Holland et al. (2014).

Statistical Learning: Toy Example



- One might expect $f(X_3)$ to be bounded by the percentiles of BMI at each age level. Additionally, one might assume monotonicity or piece-wise polynomial structures of f.

The figure is from Holland et al. (2014).

Statistical Learning: Toy Example

- Alternatively, we might expect $f(X_3)$ to minimize some form of error across populations, such as:

$$\mathbb{E}_{(X_3,Y)}(Y - f(X_3))^2.$$

  This is one of widely expected properties of f, and the conditional expectation, $\mathbb{E}_{Y|X_3}(Y|X_3)$, is one of such minimizers.[1]
- Note that the (conditional) expectation is sensitive to outliers, which can make it less representative of each sub-population. Depending on domain knowledge or requirements of the analysis, we may need other desired properties for f.
- Other interesting tasks include:
  - Using (X_1, X_2, X_3) to classify obesity vs. non-obesity. Here, f is a classifier.
  - Using (X_1, X_2, X_3) to identify underlying heterogeneous sub-populations, such as clusters related to demographic and socioeconomic characteristics. Here, f is a cluster indicator or outputs latent factors defining the distinct clustering space.

[1] Details will be discussed in a later subsection for the method of least squares.

Main Contents of This Lecture

- This lecture focuses on:
  1. Introducing several paradigms in Statistical Learning, including supervised and unsupervised learning
  2. Providing details on supervised learning, including the empirical risk minimization principle[2]
  3. Introducing various key neural networks, such as convolutional neural networks.

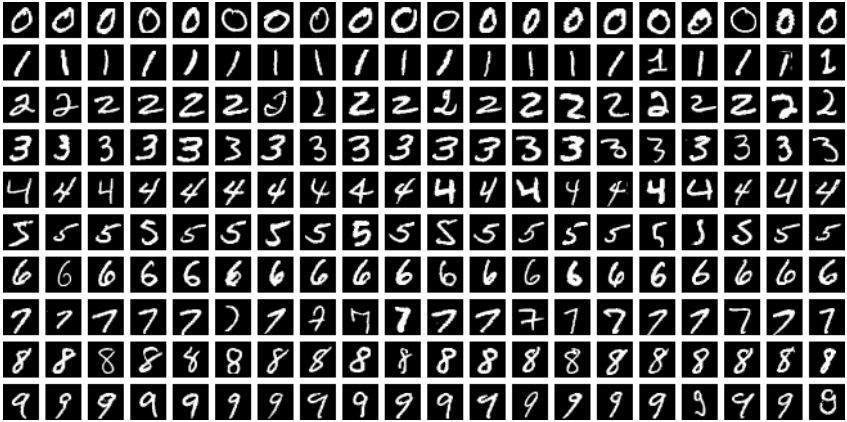
[2] The earliest approaches and basics of unsupervised learning will be covered in a separate lecture.

Outline

Paradigms in Statistical Learning

- In this subsection, we review several paradigms, including supervised and unsupervised learning.
- In *supervised learning*, we find f using observed pairs of explanatory variables (X) and response variables (Y).
- Examples include the previous BMI prediction example, which is a *regression* task.

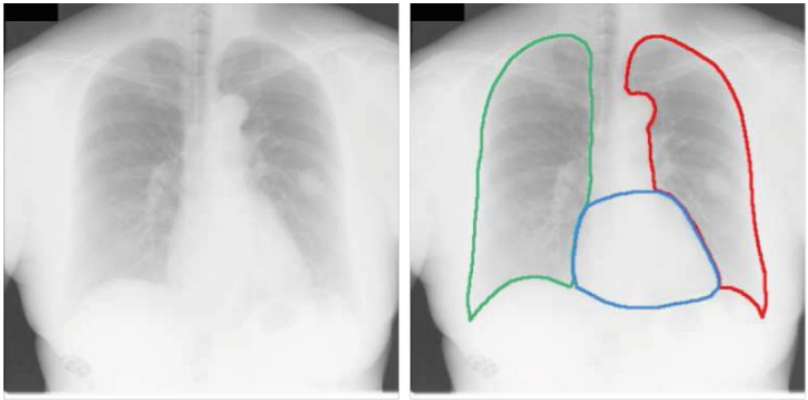
Supervised Learning: Multi-class Classification



- Sample images from Modified National Institute of Standards and Technology (MNIST) database (LeCun, 1998). Each image in $\{0,1\}^{28\times 28}$ is labeled with one of ten digits.

Image source: `https://en.wikipedia.org/wiki/MNIST_database`.

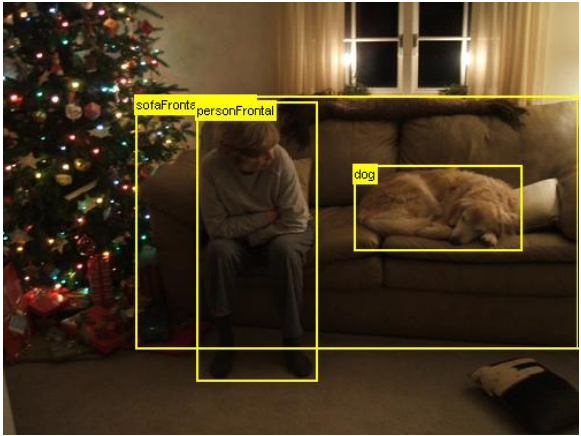
Supervised Learning: Segmentation



- Image segmentation involves a multivariate multi-class classification task, where the label is a multivariate random vector, and each pixel can be assigned some of multiple classes.

Images of chest X-rays and their organ segmentation labels, edited from (Dai et al., 2018).

Supervised Learning: Object Detection



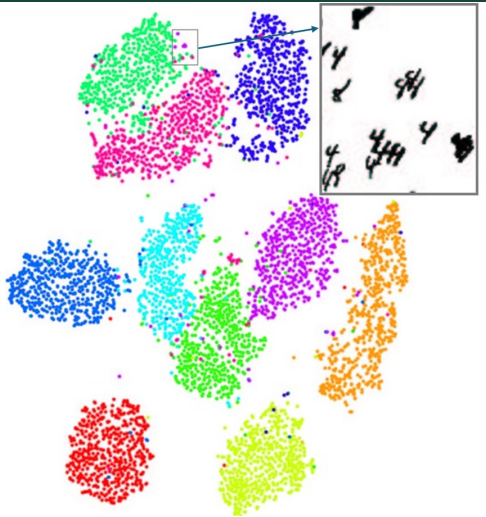
- Some tasks require regression (location) and classification (object types) simultaneously.

An example image of object detection from VOC2012 (Everingham et al., 2015).

Unsupervised Learning

- In *unsupervised learning*, we find f using the explanatory variable (X) only.
- Examples of f include:
  - An embedding map from $\vec{X}$ to lower-dimensional factors $\vec{Z}$ and a corresponding cluster indicator
  - A generator that maps $\vec{Z}$ to synthesize realistic $\vec{X}$
  - A transformation map from one data domain to another data domain

Unsupervised Learning: Clustering



Example of t-SNE embeddings of MNIST, edited from Van der Maaten and Hinton (2008).

Unsupervised Learning: Data Generation

Example of generated images, edited from Li et al. (2024)

Unsupervised Learning: Data-to-Data Translation

- In text-to-image translation, f is a transformation map that inputs a text, e.g.,

  "There is a clean desk in the middle. Outside the window, a whale shark is swimming in the dark night sky above Manhattan."

  to produce images that reflect the semantic information in the text.
- We have images and texts, but usually have little or no information about matching pairs.

Images are generated with DALL-E-2 (Ramesh et al., 2022).

Unsupervised Learning: Visual Representation Learning

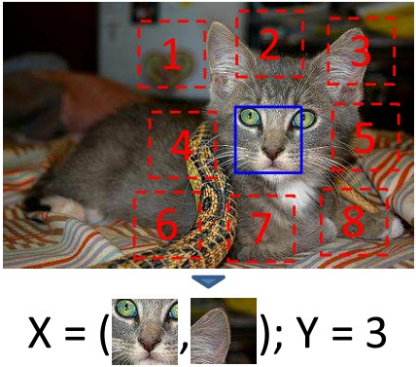
(a) Input context (b) Human artist

- $\vec{X}$: remaining pixel values after artificially removing some of them in images.
- $\vec{Y}$: the pixel values that were removed.

The image is from Pathak et al. (2016).

When labeling is done artificially, rather than by human annotators, as in this example, it is often called *self-supervised learning*.

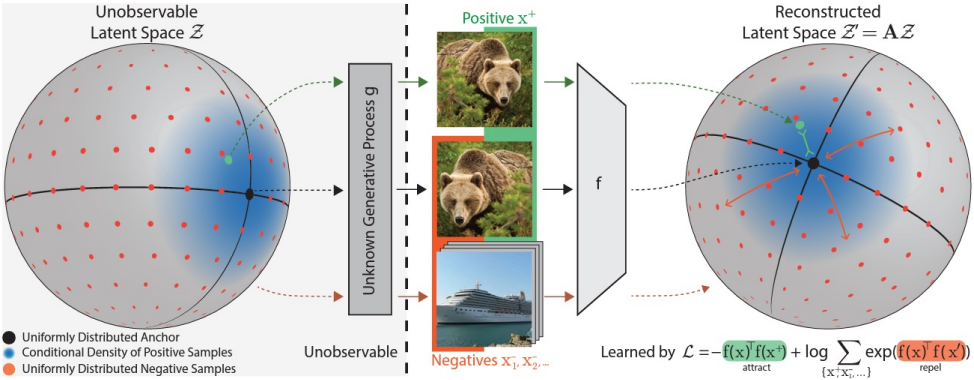
Unsupervised Learning: Visual Representation Learning



- $\vec{X}$: a pair of image patches, where the second one is randomly sampled from one of eight possible relative locations.
- Y: the relative location of the second image patch.

The image is from Doersch et al. (2015).

Unsupervised Learning: Contrastive Learning



- $(\vec{X}, \vec{X}')$: a pair of data; Y: indicates whether the pair is positive (similar) or negative (dissimilar or less similar); $f(\vec{X})$: the representation.

The image is from Zimmermann et al. (2021).

Semi-supervised Learning

- *Semi-supervised learning* is another important paradigm. In semi-supervised learning, some data have both X and Y, and there is usually a much larger number of data that have only X.
- Examples include:
  - Anomaly detection (Chandola et al., 2009) that classifies abnormal data from normal ones. Due to the rarity of anomalies, it typically handles few or no[3] labeled data and a large unlabeled dataset.
  - Positive-unlabeled (PU) learning (Kiryo et al., 2017) that aims to develop binary classifiers when all labeled data are positive. For example, in predicting whether users are interested in a web-ad with features X and click it ($Y = 1$ for clicked ads), negative data cannot be labeled; they are indistinguishable from ads missed by users, regardless of what Y is.

[3] This setting is called *one-class classification* (Tax and Duin, 2004), sometimes assuming that all the observed data are normal.

Outline

Formulation of Regression Problem

- When we want to predict Y using $\vec{X}$, the method of least squares is a popular way to define the learning goal:

$$f^* \in \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(\vec{X},Y)}(Y - f(\vec{X}))^2. \tag{1}$$

  where $\mathcal{F}$ is a class of functions to be considered.
- This formulation shows two key components: (i) the function class $\mathcal{F}$ and (ii) the learning criterion $\mathbb{E}_{(\vec{X},Y)}(Y - f(\vec{X}))^2$.
- For (i), the larger the class, the more capable it is of capturing complex patterns. However, it becomes more prone to learning specific examples in observed data, rather than general patterns in the population (referred to as *overfitting*).
- For (ii), other distances between Y and $f(\vec{X})$ can be considered. For example, when we want to have f to be less sensitive to extreme values, we may use an absolute error $\left|Y - f(\vec{X})\right|$.

Example: Linear Model with Mean Squared Error

- When we have a univariate X, $\mathcal{F}$ is the set of all linear functions of X,

$$\{f_\theta(x) = \beta_0 + \beta_1 x | \theta = (\beta_0, \beta_1)^T \in \mathbb{R}^2\}, \tag{2}$$

and we use the mean squared error, the solution is:

$$f_{\theta^*}(x) = \mathbb{E}_Y(Y) + \frac{\mathrm{Cov}_{(X,Y)}(X, Y)}{\mathrm{Var}_X(X)}(x - \mathbb{E}_X(X)) \tag{3}$$

where $\theta^* = (\mathbb{E}_Y(Y) - (\mathrm{Cov}_{(X,Y)}(X, Y)/\mathrm{Var}_X(X))\mathbb{E}_X(X), \mathrm{Cov}_{(X,Y)}(X, Y)/\mathrm{Var}_X(X))^T$.
Hint: The Hessian matrix of $\mathbb{E}_{(X,Y)}(Y - (\beta_0 + \beta_1 X))^2$, $2\begin{pmatrix} 1 & \mathbb{E}_X X \\ \mathbb{E}_X X & \mathbb{E}_X X^2 \end{pmatrix}$, is positive definite, and the θ^* is the unique solution to $d\mathbb{E}_{(X,Y)}(Y - (\beta_0 + \beta_1 X))^2/d(\beta_0, \beta_1) = 0$.

Other Examples

- When $\mathcal{F}$ is the set of all functions of $\vec{X}$, the conditional expectation $\mathbb{E}_{Y|\vec{X}}(Y|\vec{X})$ is a solution.
  Hint: $Y - f(\vec{X}) = (Y - \mathbb{E}_{Y|\vec{X}}(Y|\vec{X})) + (\mathbb{E}_{Y|\vec{X}}(Y|\vec{X}) - f(\vec{X}))$.
- When $\mathcal{F}$ is the set of all functions of $\vec{X}$ and we use mean absolute error, $\mathbb{E}_{(\vec{X},Y)}|Y - f(\vec{X})|$, the conditional median $F^{-1}_{Y|\vec{X}}(0.5)$ is the minimizer.
  Hint: Derive $\frac{\partial}{\partial c}\mathbb{E}_{Y|\vec{X}}(|Y - c|\,|\vec{X}) = \int_{-\infty}^{c} f_{Y|\vec{X}}(y|\vec{X})dy - \int_{c}^{\infty} f_{Y|\vec{X}}(y|\vec{X})dy$.

Risk Minimization

- We formulate risk minimization principle, which interprets all of the aforementioned examples.
- We denote *loss function* by $l(f(\vec{x}), y)$, a non-negative measure of discrepancy between y and its predicted value $f(\vec{x})$.[4]
- With the loss, we define the *risk*, the population mean of losses: $\mathbb{E}_{(\vec{X},Y)} l(f(\vec{X}), Y)$. That is, the risk measures a form of prediction error across population by averaging the datum-wise errors $l(f(\vec{X}), Y)$ using their occurrence possibilities as weights.
- The risk minimization aims to find

$$f^* \in \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(\vec{X},Y)} l(f(\vec{X}), Y). \tag{4}$$

[4] The f is often referred to as a *hypothesis*.

Risk Minimization: Loss Function

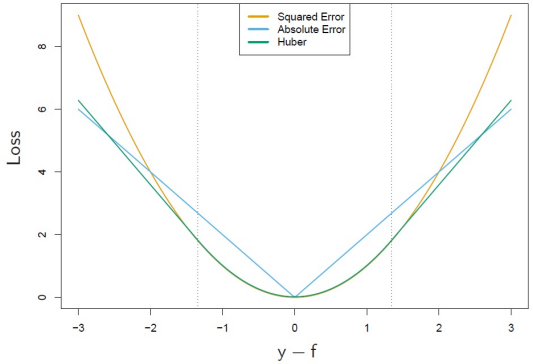
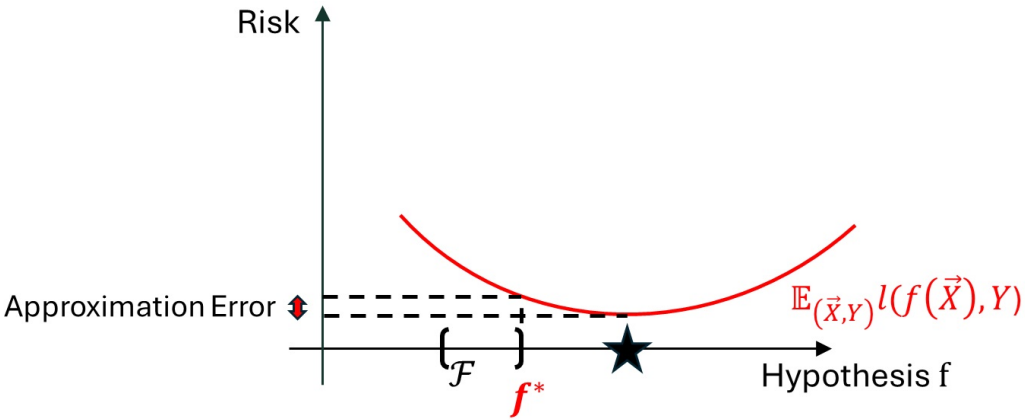


FIGURE 10.5. *A comparison of three loss functions for regression, plotted as a function of the margin $y-f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y-f|$ is large.*

The figure is from Hastie (2009).

Risk Minimization: Approximation Error



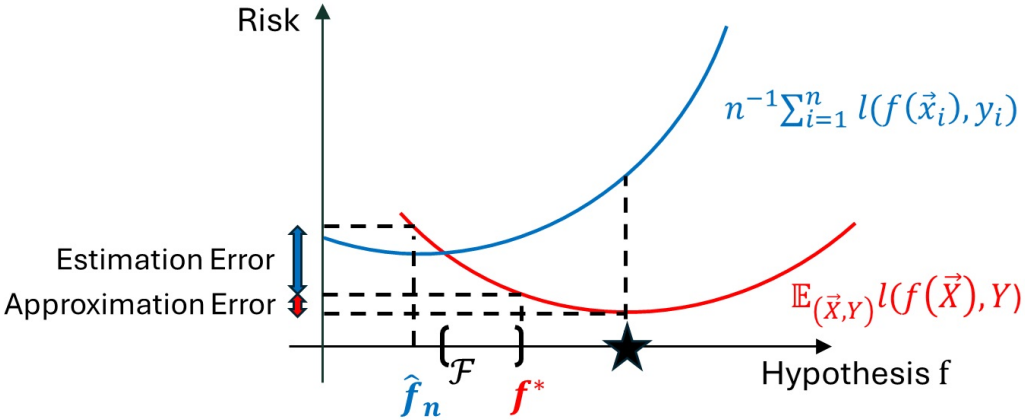
Empirical Risk Minimization

- The population of data is not available or infinite in most situations, and we have access to only a finite number of samples.
- *Empirical risk minimization* (ERM) is a principle for leveraging observed data to find an estimate of f^* in Equation (4).
- Let $\{(\vec{x}_i, y_i)\}_{i=1}^n$ be the dataset of n observed data points. The *empirical risk* is an empirical estimate of the risk, which is defined as:

$$n^{-1} \sum_{i=1}^{n} l(f(\vec{x}_i), y_i). \tag{5}$$

- By the law of large numbers, the empirical risk converges to the (population) risk. Given this, the minimizer of Equation (5) can be considered as a reasonable estimate of f^*.

Empirical Risk Minimization: Estimation Error



The summation of estimation and approximation errors is referred to as *excess risk*.

Simple Regression Approach: Linear Regression

- When $\mathcal{F}$ is the set of all linear functions of X and we use square loss, the minimizer of the empirical risk is:

$$(\hat{f} =) f_{\hat{\theta}}(x) = \bar{y} + \frac{s_{XY}}{s_X^2}(x - \bar{x}) \tag{6}$$

  where $\hat{\theta} = (\bar{y} - (s_{XY}/s_X^2)\bar{x}, s_{XY}/s_X^2)^T$.

- *Estimation Error:*

$$\begin{aligned}
&\mathbb{E}_{(X,Y)}(Y - f_{\hat{\theta}}(X))^2 - \mathbb{E}_{(X,Y)}(Y - f_{\theta^*}(X))^2 \\
&= \left(\mathbb{E}_Y(Y - \bar{y})^2 - \mathrm{Var}_Y(Y)\right) \\
&\quad - 2\left(\frac{s_{XY}}{s_X^2}\mathbb{E}_{(X,Y)}(X - \bar{x})(Y - \bar{y}) - \frac{\sigma_{XY}^2}{\sigma_X^2}\right) + \left(\frac{s_{XY}^2}{s_X^4}\mathbb{E}_X(X - \bar{x})^2 - \frac{\sigma_{XY}^2}{\sigma_X^2}\right) \\
&\xrightarrow{p} 0
\end{aligned} \tag{7}$$

  by the convergence of first and second moments in probability.

Simple Regression Approach: Linear Regression

- *Approximation Error:*

$$\begin{aligned}
&\mathbb{E}_{(X,Y)}(Y - f_{\theta^*}(X))^2 - \mathbb{E}_{(X,Y)}(Y - \mathbb{E}_{Y|X}(Y|X))^2 \\
&= \Big(\operatorname{Var}_Y(Y) - \operatorname{Cov}^2_{(X,Y)}(X,Y)/\operatorname{Var}_X(X)\Big) - \mathbb{E}_X \operatorname{Var}_{Y|X}(Y|X) \\
&= (\operatorname{Var}_X(X)\operatorname{Var}_X(\mathbb{E}_{Y|X}(Y|X)) - \operatorname{Cov}_{(X,Y)}(X,Y)^2)/\operatorname{Var}_X(X).
\end{aligned} \tag{8}$$

Here, by the Cauchy-Schwarz inequality, the approximation error is zero if and only if $\mathbb{E}_{Y|X}(Y|X)$ is a linear function of X.

Formulation of Classification Problem

- When Y is binary, the problem becomes a binary classification task. Let's encode positive and negative class labels as $Y = 1$ and $Y = -1$, respectively.
- Since classification is a special case of regression, ERM is still applicable. For loss functions tailored to the classification, an intuitive choice is the 0-1 loss:

$$l_{0-1}(f(\vec{X}), Y) := I(Y \neq f(\vec{X})) \tag{9}$$

  where $f(\vec{x})$ is the predicted class label.
- However, even for a class of simple functions, ERM equipped with 0-1 loss is known to be a non-deterministic polynomial time (NP)-hard problem (Feldman et al., 2012).

Surrogate Loss

- How about using other losses, such as the square loss $(Y - f(\vec{X}))^2$, where $f(\vec{x})$ now represents the prediction of $\mathbb{P}(Y = 1|\vec{X} = \vec{x})$ and $\text{sgn}(f(\vec{x}))$ is the predicted class label?
- The conditional expectation $\mathbb{E}_{Y|\vec{X}}(Y|\vec{X})$ is still a solution, which can be implemented as a k-nearest neighborhood (k-NN) classifier:

  Given $\vec{x}$, first find k closest data points (referred to as k-NN), and then determine their majority class as the predicted class (approximating the average of Y at $\vec{X} = \vec{x}$).
- How such alternative solution can relate with f^* with 0-1 loss in general?

Surrogate Loss: Bayes Consistency

- When $\mathcal{F}$ is the set of all functions of $\vec{X}$ and we use the 0-1 loss,

$$f^*_{\text{Bayes}}(\vec{x}) := \arg\max_y \mathbb{P}(Y = y|\vec{X} = \vec{x}) \tag{10}$$

  is a solution known as the *Bayes classifier*, and its error rate is referred to as the *Bayes rate*.
  Hint: $\mathbb{E}_{(\vec{X},Y)} I(Y \neq f(\vec{X})) = \mathbb{E}_{\vec{X}} \left(\sum_y I(y \neq f(\vec{X})) \mathbb{P}(Y = y|\vec{X}) \right)$.

- Note that $\text{sgn}(\mathbb{E}_{Y|\vec{X}}(Y|\vec{X} = \vec{x})) = f^*_{\text{Bayes}}(\vec{x})$ holds. That is, when we use the sign of a solution of the risk corresponding to the square loss, it is consistent with the Bayes classifier. Such losses are said to be *Bayes consistent*.

Surrogate Loss: Bayes Consistency

- We define margin-based losses $l(v)$ where $v := yf(\vec{x})$. Examples include:
  1. 0-1 loss: $l(y \neq f(\vec{x})) = l(yf(\vec{x}) \neq 1)$
  2. Square loss: $(y - f(\vec{x}))^2 = (1 - yf(\vec{x}))^2$
- A non-negative and convex $l(v)$, such that $l(0) = 1$, is Bayes consistent if and only if it is differentiable at 0 and $l'(0) < 0$.[5]

[5] See Theorem 2 in Bartlett et al. (2006) for details.

Example: Cross-Entropy Loss

- Let $\sigma(u) := 1/(1+\exp(-u))$ be the *logistic function* (or *sigmoid function*). It is strictly increasing, $\lim_{u\to-\infty}\sigma(u) = 0$, and $\lim_{u\to\infty}\sigma(u) = 1$. The *logit* (or *log-odds*) given $\vec{x}$ is defined as:

$$\log\frac{\mathbb{P}(Y=1|\vec{X}=\vec{x})}{\mathbb{P}(Y=-1|\vec{X}=\vec{x})} = \sigma^{-1}(\mathbb{P}(Y=1|\vec{X}=\vec{x})). \tag{11}$$

- *Cross-entropy loss* is defined as:

$$-I(y=1)\log_2\sigma(f(\vec{x})) - I(y=-1)\log_2(1-\sigma(f(\vec{x}))) = \log_2(1+\exp(-yf(\vec{x}))), \tag{12}$$

  which is margin-based.[6]

- When $\mathcal{F}$ is the set of all functions and we use the logistic loss, the optimal function $f^*(\vec{x}) = \sigma^{-1}(\mathbb{P}(Y=1|\vec{X}=\vec{x}))$ is the true logit.
  Hint: The (population) risk can be expressed as
  $-\mathbb{E}_{\vec{X}}\left(\mathbb{P}(Y=1|\vec{X})\log_2\sigma(f(\vec{X})) + (1-\mathbb{P}(Y=1|\vec{X}))\log_2(1-\sigma(f(\vec{X})))\right)$.

[6] In Machine Learning, the natural logarithm (base e) is commonly used instead of $\log_2(\cdot)$.

Example: Logistic Model with Cross-Entropy Loss

- When $\mathcal{F}$ is the set of all f such that its logit $\sigma^{-1}(f(x))$ is linear w.r.t. x (*logistic model*),

$$\{f_\theta(x) = \sigma(\beta_0 + \beta_1 x) | \theta = (\beta_0, \beta_1)^T \in \mathbb{R}^2\}, \tag{13}$$

  and we use the cross-entropy loss, the $(f^* =) f_{\theta^*}$ uniquely exists where the θ^* is the solution to:

$$\begin{pmatrix} \mathbb{E}_X\Big(\sigma(\beta_0 + \beta_1 X) - I(Y = 1)\Big) \\ \mathbb{E}_X X\Big(\sigma(\beta_0 + \beta_1 X) - I(Y = 1)\Big) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{14}$$

  Due to the non-linearity of σ, there is no closed-form expression of the solution.
  Hint: The Hessian is positive definite, and the risk diverges to positive infinity at the boundaries. This property can also be derived as a characteristic of MLEs.
- Finding the solution to empirical risk minimization leads to *logistic regression*. The solution can be numerically approximated using iteratively reweighted least squares methods.

Generative Model vs. Discriminative Model

- The term 'Generative Model' has been used in classification since before the emergence of deep generative models:
  1. Generative Model: Models $p(X, Y)$, typically by modeling $p(X|Y)$ and $p(Y)$ separately.
  2. Discriminative Model: Models $p(Y|X)$ only and uses it directly.
- Since the generative model learns $p(X, Y)$, it can generate data.

Generative Model vs. Discriminative Model

- Generative models are statistical models of (X, Y), so they naturally define discriminative models of Y given X.
- For example, let $Y \sim 2\text{Ber}(p) - 1$ and $X|Y = y \sim N(\mu_y, \sigma_\epsilon^2)$. Then, $\mathbb{P}(Y = 1|X = x)$ follows a logistic model with $(\beta_0, \beta_1)^T = ((\mu_{-1}^2 - \mu_1^2)/2\sigma_\epsilon^2, (\mu_1 - \mu_{-1})/\sigma_\epsilon^2)^T$.
  Hint: By Bayes' theorem, $\mathbb{P}(Y = 1|X = x) = \mathbb{P}(X = x|Y = 1)\mathbb{P}(Y = 1)/\mathbb{P}(X = x)$.
- This generative model can express all (non-trivial) logistic models: for any $(\beta_0, \beta_1)^T$ where $\beta_1 \neq 0$, the $(\mu_{-1}, \mu_1, \sigma_\epsilon)^T = (-\beta_0/\beta_1 - \beta_1\sigma_\epsilon^2/2, -\beta_0/\beta_1 + \beta_1\sigma_\epsilon^2/2, \sigma_\epsilon)^T$ with arbitrary σ_ϵ defines the logistic model with $(\beta_0, \beta_1)^T$.

Q: Why do we need discriminative models?

Outline

Challenges in High-dimensional Data Analysis

- Let $\vec{X}|Y = y \sim N(\vec{\mu}_y, \Sigma)$ where $y \in \{-1, 1\}$ and $Y \sim 2\text{Ber}(0.5) - 1$. The Bayes classifier can be expressed as:

$$f^*(\vec{x}) = 2I\left((\mu_1 - \mu_{-1})^T \Sigma^{-1}\left(\vec{x} - \frac{\mu_{-1} + \mu_1}{2}\right) > 0\right) - 1, \tag{15}$$

  which is referred to as the linear discriminant analysis.

  Hint: $f^*(\vec{x}) = 2I\left(\log\left(\mathbb{P}(Y = 1|\vec{X} = \vec{x})/\mathbb{P}(Y = -1|\vec{X} = \vec{x})\right) > 0\right) - 1.$

- Let $f_{\hat{\theta}}(\vec{x})$ be its empirical estimate using sample means $\hat{\mu}_y$ and the sample covariance matrix $\hat{\Sigma}$.
- The accuracy of $f_{\hat{\theta}}(\vec{x})$ can converge to 0.5 as $p/n \to \infty$ (Bickel and Levina, 2004). In this case, $f_{\hat{\theta}}$ is no better than the random guesser. One of such cases is when $\hat{\Sigma}$ is singular, the ratio of the largest to the smallest eigenvalues diverges.
- Several approach has imposed regularity conditions such as sparsity of covariance structure to alleviate this issue (Cai and Liu, 2011).

Challenges in High-dimensional Data Analysis

- Kernel density estimation (KDE) is one of popular non-parametric methods:

$$\hat{f}_n(\vec{x}; H) = n^{-1} \sum_{i=1}^{n} K_H(\vec{x}, \vec{x}_i) \tag{16}$$

where H is $p \times p$ bandwidth matrix and $K_H(\vec{x}, \vec{x}') = |H|^{-1/2} K(H^{-1/2}\vec{x}, H^{-1/2}\vec{x}')$. Here, K is the kernel function such as Gaussian kernel: $K(\vec{x}, \vec{x}') = (2\pi)^{-p/2} \exp(-\|\vec{x} - \vec{x}'\|^2/2)$.

- The convergence rate of mean integrated squared error (MISE) can be slow as the data dimension p increases:

$$\begin{aligned} \text{MISE} &:= \mathbb{E}_{(\vec{X}_1, \ldots, \vec{X}_n)} \left(\int \left(\hat{f}_n(\vec{x}; H) - p(\vec{x}) \right)^2 d\vec{x} \right) \\ &= O(n^{-4/(p+4)}) \end{aligned} \tag{17}$$

when an optimal bandwidth $h_{\text{opt}} = O(n^{-1/(p+4)})$ is used. This result assumes a uniform bandwidth is applied across all data features to an appropriately scaled $\vec{X}$.[7]

[7] For example, Scott's rule suggests using $H = \text{diag}((n^{-1/(p+4)} \text{Var}(X_j)^{1/2})_{j=1}^{p})$.

Projection Pursuit Regression

- One common assumption to address issues in high-dimensional settings is that there is dependency among features of $\vec{X}$, such as a sparse covariance structure and the presence of a lower-dimensional manifold.
- Under this assumption, projection pursuit regression (PPR) can be a prominent statistical approach:

$$f(\vec{x}) = \sum_{h=1}^{H} g_h(\vec{w}_h^T \vec{x}) \tag{18}$$

  where g_h is a non-linear function and $\vec{w}_h \in \mathbb{R}^p$. PPR represents an additive function of H (non-linearly processed) linear embeddings of $\vec{x}$.
- The class of PPR is flexible and is known as a *universal approximator*: For any $\epsilon > 0$ and continuous function g defined on $\mathbb{R}^p$, there exists a PPR that approximates g with a difference less than ϵ.

Example: Principal Component Regression

- Principal component regression is an example of PPR that uses linear transformations for g_h:

$$f(\vec{x}) = \sum_{h=1}^{H} \beta_h(\vec{w}_h^T \vec{x}) \tag{19}$$

  where $\beta_h \in \mathbb{R}$. Here, $\vec{w}_1^T \vec{x}, \ldots, \vec{w}_H^T \vec{x}$ represent the first H principal components.
- This approach first performs principal component analysis (PCA) to select several PCs, and then conducts regression using the selected PCs.
- It addresses multicollinearity better than the ordinary least square method.

Example: Neural Network

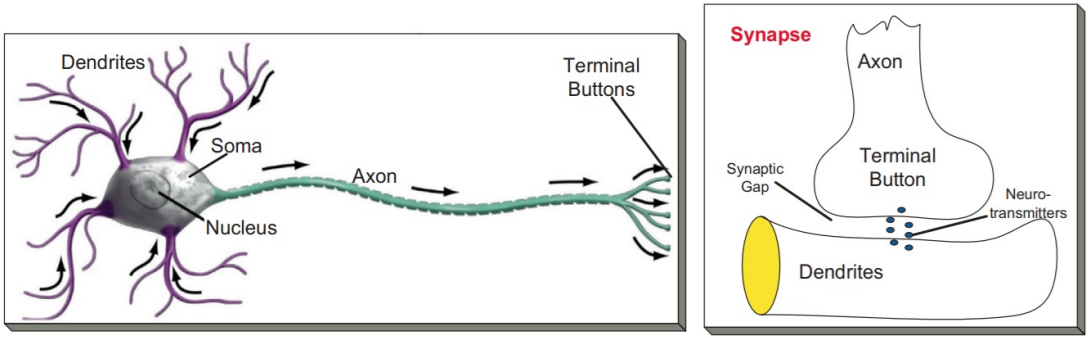
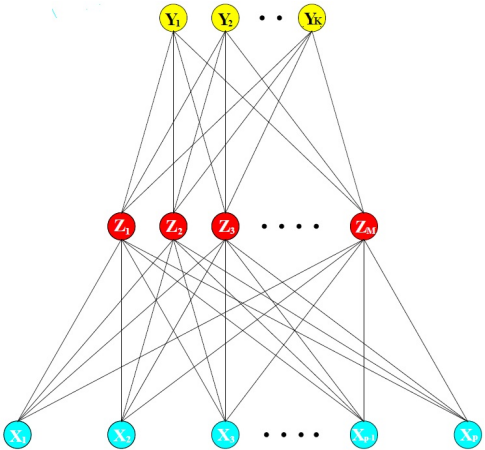


Fig. 1. A biological neuron (left) with the direction of the signal flow and a synapse (right) [7].

The figure is from Kiranyaz et al. (2021).

Example: Neural Network with One Hidden Layer



The figure is from Hastie (2009).

Example: Neural Network with One Hidden Layer

- Another example of PPR is neural networks with one hidden layer:[8]

$$f_\theta(\vec{x}) = \beta_0 + \sum_{h=1}^{H} \beta_h \sigma(\alpha_h + \vec{w}_h^T \vec{x}) \tag{20}$$

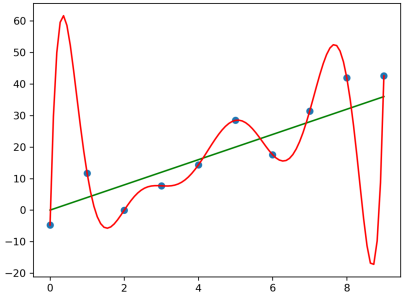
  where $\theta := (\beta_0, \ldots, \beta_H, \alpha_1, \ldots, \alpha_H, \vec{w}_1, \ldots, \vec{w}_H)^T$ and $g_h(\cdot) = \beta_h \sigma(\cdot)$ where σ is the sigmoid function. Here, $\alpha_h + \vec{w}_h^T \vec{x}$ represents the h-th hidden node value, and $(\beta_1, \cdots, \beta_H)^T$ and β_0 are weight vector and bias term, respectively, for the output layer.
- Although the flexibility is reduced from the general formulation of PPR by fixing the non-linearity using sigmoid (activation) functions, one-hidden-layer neural networks remain universal approximators (Funahashi, 1989).

[8] This interpretation focuses on the mathematical formulations of PPR and neural networks. However, they have substantial differences, including their perspectives on the nonlinear component (non-parametric vs. parametric) and their implementations (backfitting vs. backpropagation). See Hwang et al. (1994) and Sections 11.2 and 11.3 in Hastie (2009) for details.

Deep Neural Network

- The neural network in the previous example is called *fully-connected* (or *dense*) because all input nodes are connected to the hidden nodes, and all hidden nodes are connected to the output nodes. When we stack multiple hidden layers, it becomes a (dense) *deep neural network*.
- Although neural networks can approximate any continuous function, they involve a large number of parameters, which often leads to *overfitting*, a phenomenon where a model fits well to observed data but performs poorly on new samples from the same population.
- The number of parameter in a (dense) L-hidden-layer neural network, in regression task, is $(p+1)H_1 + \cdots + (H_{L-1}+1)H_L + (H_L+1)$ where H_l denotes the number of hidden nodes at the l-th hidden layer.
- For example, when we want to classify MNIST images using a one-hidden-layer network with 256 hidden nodes, the network has about 203K parameters. For four-hidden-layer networks with 256 nodes in each hidden layer, the number of parameter is about 401K.

Deep Neural Network: Overfitting



- The term *overfitting* describes any model that has an unnecessarily large number of parameters. For example, given n data points following a linear pattern (with observation noises), there exists a $(n+1)$-th order polynomial function to interpolate all the points.
- However, predictions based on the (overfit) polynomial function can be poor, as evidenced by the gaps caused by large curvatures.

Image source: `https://en.wikipedia.org/wiki/Overfitting`.

Deep Neural Network: Overfitting

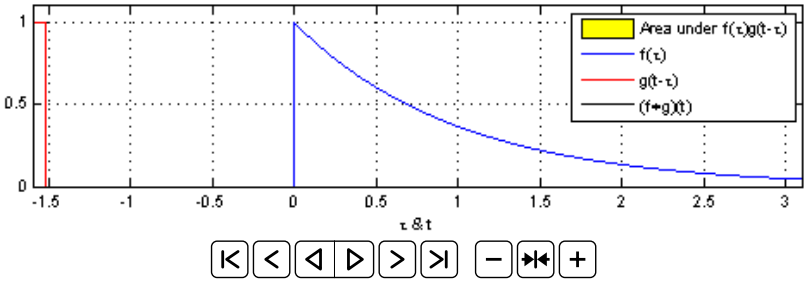
- In Machine Learning, the data used to fit models are called *training data*. Examples include samples for constructing empirical risks to identify their minimizers and n data points in the previous slide.
- In contrast, samples that are not used for fitting, where we expect the trained model to perform as well as it did on the training data, are called *test data*. The performance of a machine on the test data is often referred to as its *generalization performance*.[9]
- The term *overfitting* frequently refers to cases where test performance is notably lower than training performance (regardless of whether the class of models indeed includes the true underlying model).
- Dense deep neural networks are susceptible to the overfitting issue.

[9] The test data are usually assumed to come from the same population as the training data. However, some fields, such as domain adaptation and few-shot learning, study scenarios where the distributions differ. In such contexts, overfitting and generalization may have different interpretations.

Convolutional Neural Network

- To overcome overfitting issues, several methods have been proposed to regularize the flexibility of neural networks. Reducing the number of parameters is one of the most popular approaches.
- Sparse neural networks have been developed by removing many connections in fully-connected networks. We review a foundational model, the *convolutional neural network* (CNN) (LeCun et al., 1998; Krizhevsky et al., 2012).
- CNNs introduce a novel layer called the *convolutional layer* to effectively capture spatial patterns in image pixel values.

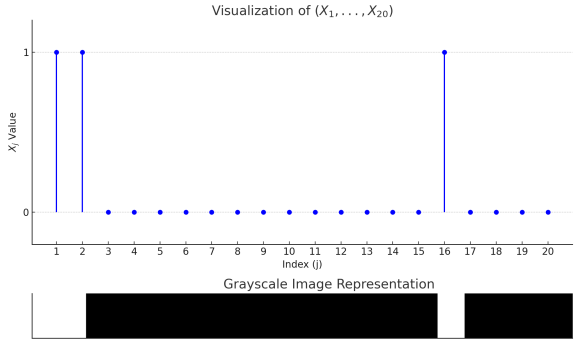
Convolutional Neural Network: Convolution Operation



- For any two functions f and g defined on $\mathbb{R}$, their convolution is defined as: $(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$.
- The convolution measures the area under $f(\tau)$, using $g(t - \tau)$ as weights. To compute $(f * g)(t)$, g is first flipped vertically, then shifted to the right by t, and the (weighted) area is calculated. In this context, g is referred to as a *(kernel) filter*.

Video source: `https://en.wikipedia.org/wiki/Convolution`.

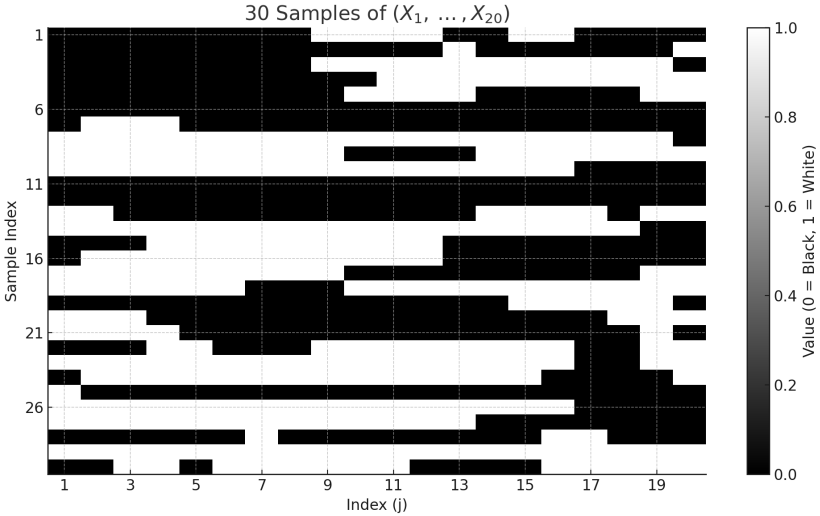
Convolutional Neural Network: Convolutional Layer



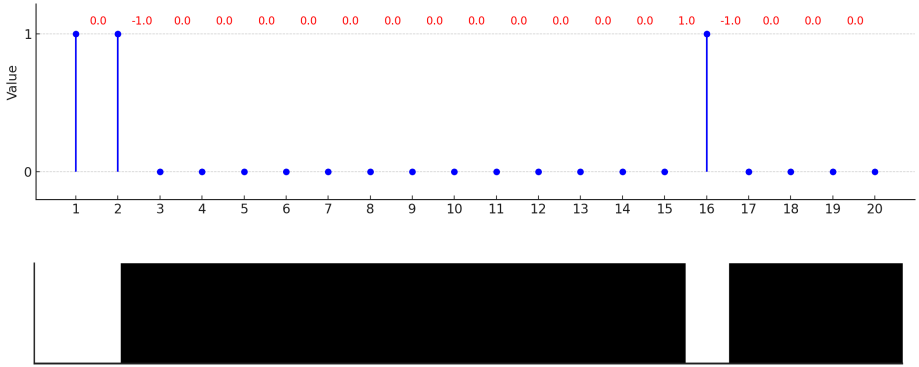
- In the context of neural networks with one hidden layer, features in $\vec{x}$ can be viewed as a sequence of input features, e.g., $f(j) = x_j$ when $j \in \{1, \ldots, p\}$ and is zero otherwise.
- Let's consider the following simulation data:

$$X_1 \sim \text{Ber}(0.5) \text{ and } X_{j+1}|(X_1, \ldots, X_j) \sim \text{Ber}(0.1 + 0.8X_j) \text{ for } j = 1, \ldots, 19. \tag{21}$$

Convolutional Neural Network: Convolutional Layer

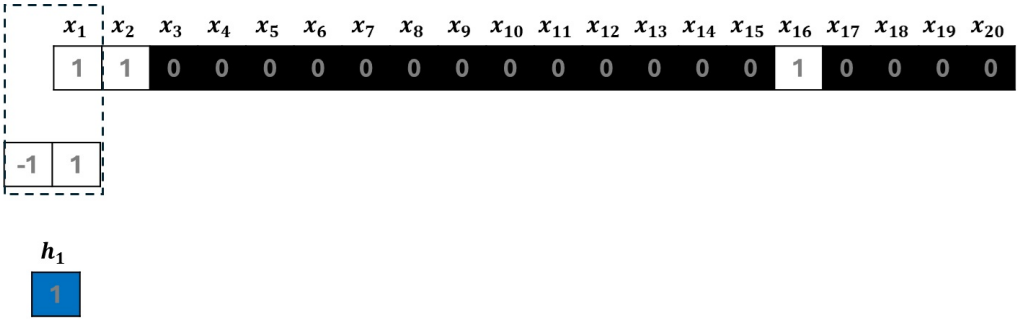


Convolutional Neural Network: Convolutional Layer

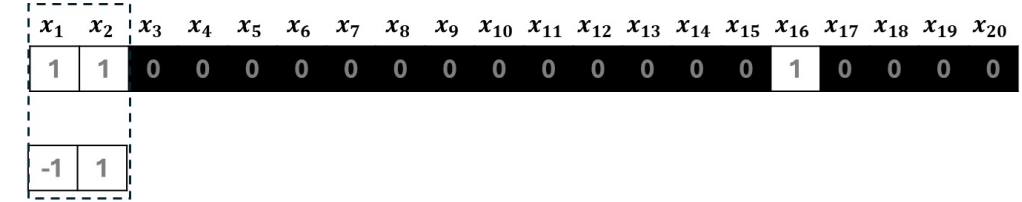


- The value $(f * g)(h)$ can be interpreted as the h-th hidden node value, computed as a summation of features of $\vec{x}$, weighted by $(g(h-j))_{j=1}^{p}$: $(f * g)(h) = \sum_{j=-\infty}^{\infty} f(j)g(h-j)$.
- *Differentiation:* Let $g(0) = 1$, $g(1) = -1$, and $g(j) = 0$ for otherwise. Then, $(f * g)(h) = x_h - x_{h-1}$ where $h = 1, \ldots, p+1$.

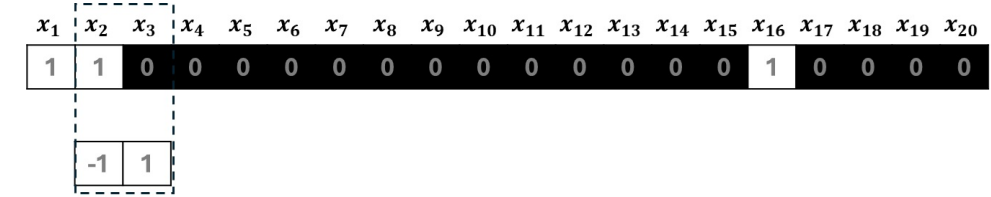
Convolutional Neural Network: Convolutional Layer



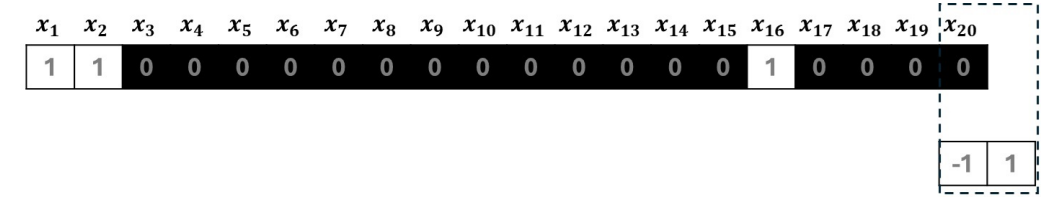
Convolutional Neural Network: Convolutional Layer



Convolutional Neural Network: Convolutional Layer



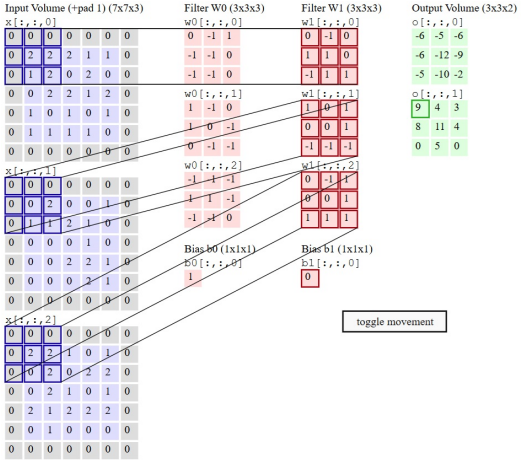
Convolutional Neural Network: Convolutional Layer



Convolutional Neural Network: Convolutional Layer

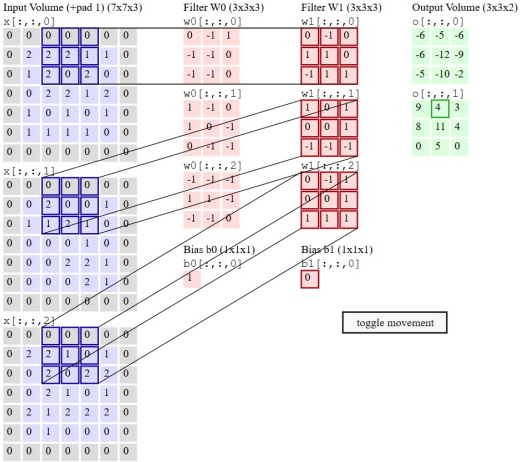
- *Moving Average:* Let $g(0) = g(1) = 1/2$ and $g(j) = 0$ for otherwise. Then, $(f * g)(h) = (x_{h-1} + x_h)/2$ where $h = 1, \ldots, p+1$, with $x_0 = x_{p+1} := 0$ for convenience.
- Note that both examples are special cases of a fully-connected layer with $(p+1)^2$ parameters, where only $2(p+1)$ elements are non-zero.
- Furthermore, all the non-zero elements are parameterized by just two values, $g(0)$ and $g(1)$, while the corresponding convolutions provide informative summaries of two consecutive input features.

Convolutional Neural Network: 2D Convolutional Layer



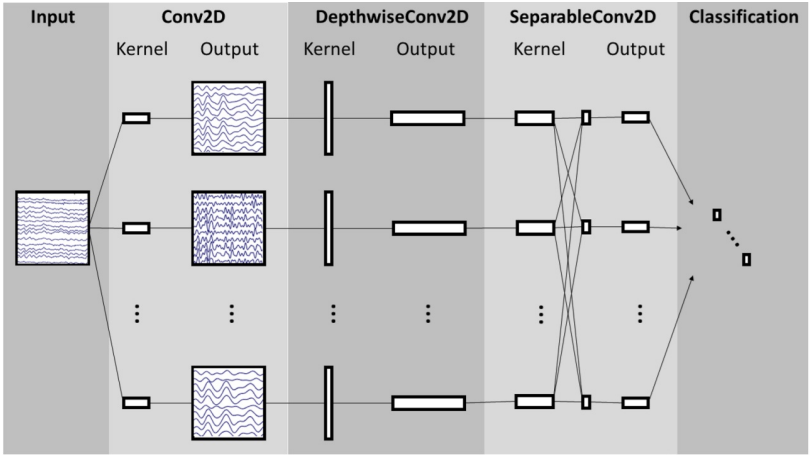
The figure is from `https://cs231n.github.io/convolutional-networks/`.

Convolutional Neural Network: 2D Convolutional Layer



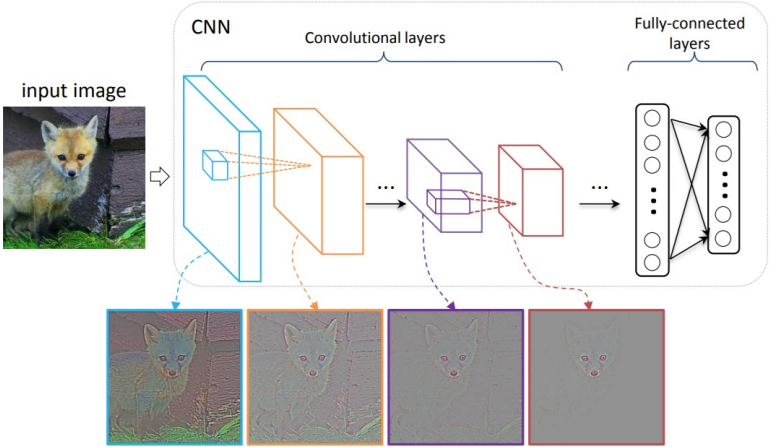
The figure is from `https://cs231n.github.io/convolutional-networks/`.

1D Convolutional Neural Network: EEGNet



The figure is from Lawhern et al. (2018).

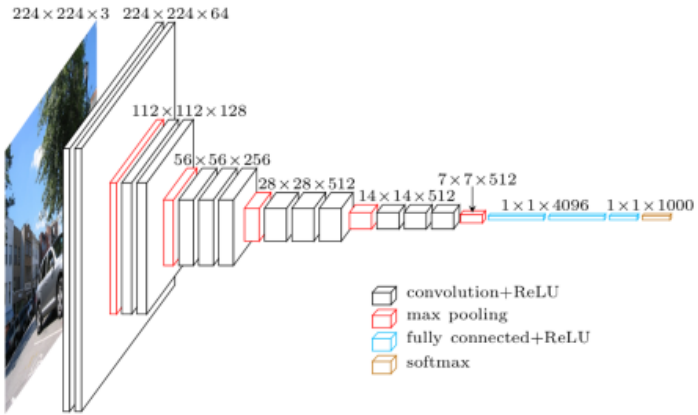
2D Convolutional Neural Network: Visualization



class name	prob
kit fox	0.5956
red fox	0.3576
grey fox	0.0439
coyote	0.0013
Arctic fox	0.0003

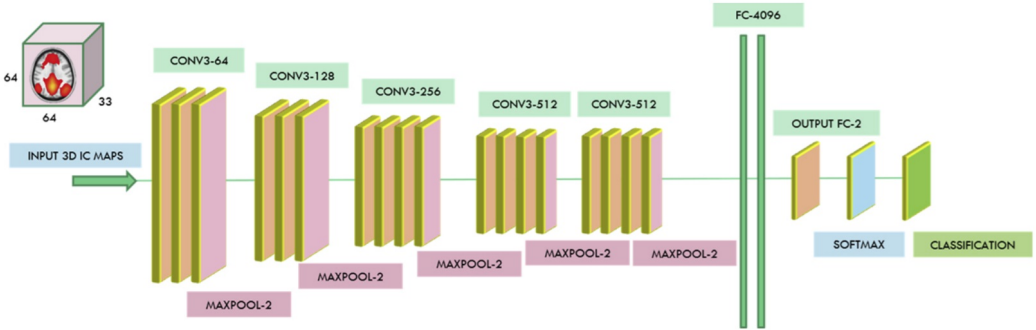
The figure is from Yu et al. (2016).

2D Convolutional Neural Network: Visual Geometry Group (VGG) Network



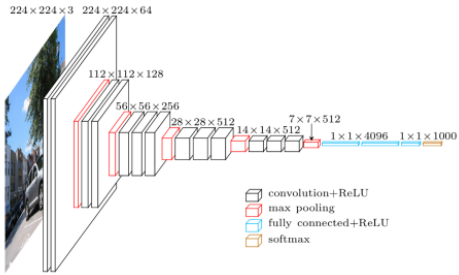
The figure is from Simonyan (2014).

3D Convolutional Neural Network



The figure is from Qureshi et al. (2019).

CNN as a Sparse Neural Network

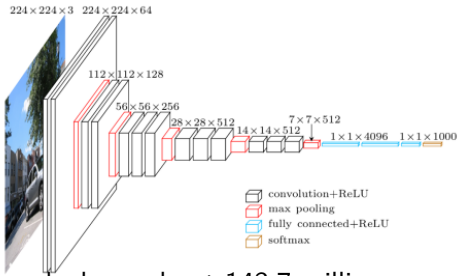


- CNNs efficiently reduce the number of parameters. For example, building a fully-connected network with nodes $224 \times 224 \times 3$-$7 \times 7 \times 512$-4096-4096-1000 would result in approximately 3.9 billion parameters:

$$(224 \times 224 \times 3 + 1) \cdot (7 \times 7 \times 512) + (7 \times 7 \times 512 + 1) \cdot 4096 + (4096 + 1) \cdot 4096 + (4096 + 1) \cdot 1000 = 3,900,114,408.$$

The figure is from Simonyan (2014).

CNN as a Sparse Neural Network

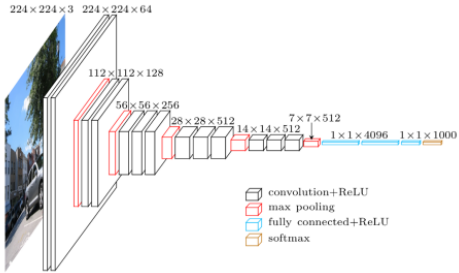


- In contrast, VGG-19 networks have about 143.7 million parameters:

$$(3 \times 3 \times 3 + 1) \cdot 64 + (3 \times 3 \times 64 + 1) \cdot 64 + (3 \times 3 \times 64 + 1) \cdot 128$$
$$+ (3 \times 3 \times 128 + 1) \cdot 128 + (3 \times 3 \times 128 + 1) \cdot 256 + (3 \times 3 \times 256 + 1) \cdot 256 * 3$$
$$+ (3 \times 3 \times 256 + 1) \cdot 512 + (3 \times 3 \times 512 + 1) \cdot 512 * 7$$
$$+ (7 \times 7 \times 512 + 1) \cdot 4096 + (4096 + 1) \cdot 4096 + (4096 + 1) \cdot 1000 = 143,667,240.$$

The figure is from Simonyan (2014).

CNN as a Sparse Neural Network



- Given the number of extracted features for images, $7 \times 7 \times 512$, CNNs reduce the number of parameters by approximately 96.3%.
- Despite the reduced flexibility, CNNs retain their property as universal approximators (Zhou, 2020).

The figure is from Simonyan (2014).

References I

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Bickel, P. J. and Levina, E. (2004). Some theory for fisher's linear discriminant function,naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, 10(6):989–1010.

Cai, T. and Liu, W. (2011). A direct estimation approach to sparse linear discriminant analysis. *Journal of the American statistical association*, 106(496):1566–1577.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

Dai, W., Dong, N., Wang, Z., Liang, X., Zhang, H., and Xing, E. P. (2018). Scan: Structure correcting adversarial network for organ segmentation in chest x-rays. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 263–273. Springer.

References II

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136.

Feldman, V., Guruswami, V., Raghavendra, P., and Wu, Y. (2012). Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192.

Hastie, T. (2009). The elements of statistical learning: data mining, inference, and prediction.

References III

Holland, S. K., Altaye, M., Robertson, S., Byars, A. W., Plante, E., and Szaflarski, J. P. (2014). Data on the safety of repeated mri in healthy children. *NeuroImage: Clinical*, 4:526–530.

Hwang, J.-N., Lay, S.-R., Maechler, M., Martin, R. D., and Schimert, J. (1994). Regression modeling in back-propagation and projection pursuit learning. *IEEE Transactions on neural networks*, 5(3):342–353.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., and Inman, D. J. (2021). 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398.

Kiryo, R., Niu, G., Du Plessis, M. C., and Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

References IV

Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., and Lance, B. J. (2018). Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, T., Tian, Y., Li, H., Deng, M., and He, K. (2024). Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544.

References V

Qureshi, M. N. I., Oh, J., and Lee, B. (2019). 3d-cnn based discrimination of schizophrenia using resting-state fmri. *Artificial intelligence in medicine*, 98:10–17.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Simonyan, K. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tax, D. M. and Duin, R. P. (2004). Support vector data description. *Machine learning*, 54:45–66.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H., and Rui, Y. (2016). Visualizing and comparing alexnet and vgg using deconvolutional layers. In *Proceedings of the 33 rd International Conference on Machine Learning*.

References VI

Zhou, D.-X. (2020). Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2):787–794.

Zimmermann, R. S., Sharma, Y., Schneider, S., Bethge, M., and Brendel, W. (2021). Contrastive learning inverts the data generating process. In *International Conference on Machine Learning*, pages 12979–12990. PMLR.