A Survey on Supervised Learning

Lanxiao Bai
University of Illinois at Urbana-Champaign
lbai5@illinois.edu

Abstract

As a powerful tool for classification and regression, supervised learning recently captured attention of both academia and industry thanks to the success of Deep Learning algorithms since AlexNet's[19] excellent performance in ImageNet. But long before that, there are also a lot of methods and algorithms that are as well widely used in different fields. In this work, we provides a full exploration of supervised learning methods and we hope that it will provide a better understanding on the history and status quo of supervised learning and discuss some application of these methods.

1. Introduction

Among all tasks of Machine Learning(ML) and Data Mining, two of the most significant and the most researched are classification and regression. As we enter the era of data, we are drowning in so much information so that it is very hard for human to conduct manual search for valuable knowledge and other data with meaningful structures. For an instance, there are about 1 trillion web pages[1] on the Internet, and in each page, there are also great amount of texts, images, videos and some other types of data and that's just the potential information we have active access to.

On the other hand, thanks to the digital notification services, RSS softwares and emails, we also can be bombarded with hundreds even thousands of fragments of information each day, but among them all, we are interested only very small part of them. If we have to manually pick and consume these data every day, a lot of time will be wasted. In other words, it is important that we have an efficient system of classification that can automatically handle all these mixed information. In fact, classification systems are already widely used to construct anti-SPAM and recommendation systems[21].

[1] http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

1.1. Challenges

Although methods of supervised learning is well-researched, applying appropriate algorithms to corresponding problems is still hard and some challenges appear in the process of deployment of these methods:

(i). **Curse of Dimensionality.** When the number of attributes of data is low, it is easy to analyze the contribution of each attribute to the problem and picking the useful attributes that lead to maximization of performance. However, as the number of possible combinations grows exponentially, it becomes very hard to examine if certain attributes are useful[12].

(ii). **Bias-Variance Tradeoff.** Considering mean squared error (MSE)

$$\begin{aligned} MSE &= \mathbb{E}[(\hat{\theta}_m - \theta)^2] \\ &= Bias(\hat{\theta}_m)^2 + Var(\hat{\theta}_m)[12] \end{aligned} \quad (1)$$

we see that both the bias and variance of a model contribute to the generalization error, so it is inherently hard to choose a suitable number of parameters. While too many parameters will lead to overfitting, which means that the performance of model on training dataset is much higher than that on testing (unseen) dataset and vice versa[12] as shown in Figure 1[4].

1.2. Our contribution

Our survey provides the following contribution:

(i). We examine the similarities and differences of methods and algorithms we have for supervised learning and provide a taxonomy of approaches to supervised learning and how they fit problems in different scenes.

(ii). We provide a detailed and comprehensive analysis on various kinds of supervised learning models, specifically, on the properties of the models and how these properties can be utilized or how the models can be modified to handle the challenges we mentioned above.

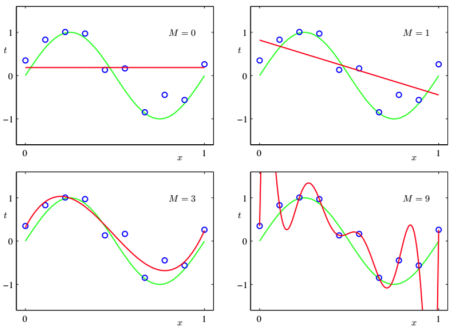


Figure 1. Fitting dataset with different models

1.3. Organization of the Survey

In Section 2, we will introduce the definition required to understand and analysis the models theoretically. Section 3 introduces the classification of the supervised learning methods we have, and Section 4-6 will cover details of approaches in each category. And in Section 7, we will talk about some popular application of supervised learning in some domains.

2. Definition of Tasks

2.1. What is Supervised Learning?

Definition 2.1 *Suppose we have a labeled dataset* $\mathcal{D} = \{x_i, y_i\}$ *in which* x_i *is called instances or inputs and* y_i *is called labels that represent categories, our goal is to find a function* h *which, ideally, have*

$$h(x_i) = y_i$$

for all $x_i \in \mathcal{D}$.

2.2. What is a good model?

In order to evaluate the performance of our models, we define two kinds of metrics.

Definition 2.2 (Risk) *Risk function* R *is a function that satisfy the following rules:*

- $R(h, \mathcal{D}) = 0$ *if* $h(x_i) = y_i$ *for all* $(x_i, y_i) \in \mathcal{D}$
- $R(h, \mathcal{D})$ *grows as the number of* (x_i, y_i) *that* $h(x_i) \neq y_i$ *increases*

Vice versa, we have the utility function defined as following

Definition 2.3 (Utility) *Utility function* U *is a function that satisfy the following rules:*

- $U(h, \mathcal{D}) = 0$ *if* $h(x_i) \neq y_i$ *for all* $(x_i, y_i) \in \mathcal{D}$
- $U(h, \mathcal{D})$ *grows as the number of* (x_i, y_i) *that* $h(x_i) = y_i$ *increases*

In order to design an appropriate metric function, domain knowledge and robustness need to be considered.

In order to evaluate the generalization capacity of model, we define

Definition 2.4 (Generalization Risk)

$$R_G = |R(h_n, \mathcal{P}) - R(h_n, \mathcal{D}_n)|$$

in which $\mathcal{P}$ *is the population.*

A good model has high utility and low risk, and low generalization risk[12], which means that it does not only has good performance on training dataset, but also on testing unseen dataset.

3. Taxonomy

There are many ways we can construct our model for supervised learning as shown in the table.

Category	Algorithms
Probabilistic Model	Naive Bayes, Logistic Regression
Logic based Model	Decision Tree, Random Forest, Boosting
Neural Network	Perceptron, Deep Learning
Kernel based Model	SVM, Kernelized Ridge Regression
Instance-based learning	kNN

Among these methods, statistical approaches is known for having an explicit underlying probability model[18], logical based approaches utilize certain logical rules to select correct labels, neural networks for now work as a black box that captures the features of given database and use backpropagation to adjust weights of neurons and kernel algorithms maps data into high-dimension spaces so that a hyper-plane can be determined to separate data with different labels.

4. Probabilistic Models

Statistical learning algorithms provide a probability that an instance belongs in each class, rather than simply a classification[18], so that our goal is changed to estimating the parameters of the model, so that the likelihood function or a posterior can be maximized.

$$w^* = \arg\max_w p(x \mid w),^2$$

or

$$w^* = \arg\max_w p(x \mid w)p(w),^3$$

[2]Maximum Likelihood Estimation(MLE)
[3]Maximum A Posterior(MAP)

so that our classifier is

$$\arg\max_k p(y_i = k \mid x_i, w)$$

4.1. Naive Bayes

Naive Bayes is derived from the assumption that feature x_i is conditionally dependent of every other feature $x_j \neq x_i$, so that

$$p(\mathbf{x} \mid C) = \prod_i p(x_i \mid C)$$

As a result, with Bayes' Theorem, we have

$$\begin{aligned} p(C_k \mid \mathbf{x}) &= \frac{p(\mathbf{x} \mid C_k)p(C_k)}{p(\mathbf{x})} \\ &= \frac{p(C_k)\prod_i p(x_i \mid C_k)}{p(\mathbf{x} \\ &= \frac{p(C_k)\prod_i p(x_i \mid C_k)}{\sum_j p(C_j)p(\mathbf{x} \mid C_j)} \end{aligned}$$

So that

$$y = \arg\max_k \frac{p(C_k)\prod_i p(x_i \mid C_k)}{\sum_j p(C_j)p(\mathbf{x} \mid C_j)}$$

One of the best known usage of Naive Bayes Classifier is Document classification. Assume that documents are drawn from a number of classes of documents, by calculating the word occurrence vector of texts, we can feed the data into the classifier and predict the category of the document.

In application, we can either apply it to a multi-class classification to predict the theme of given texts, so that we can recommend it to the user if their preference matches the prediction. We can also apply it to a binary classification task and predict if an incoming email is a spam and decide whether it should be filtered.

The major advantage of the naive Bayes classifier is its short computational time for training[18]. If a feature is numerical, we can discretize it during data pre-processing [28] ,or it is also possible to use the normal distribution to calculate probabilities[6].

However, the conditionally independent assumption is almost always wrong, so that when in sophisticated learning tasks, it's accuracy is usually lower than other algorithms. There are various attempts to modify the model and improve its performance

4.2. Linear Model and Logistic Regression

Linear Model is the most common model in supervised learning[21], not only because there are a lot of data that are linearly separable, but for the cases that data can not be perfectly separated, kernel tricks or nonlinear activation functions in Neural Networks can be used to introduce non-linearity to the model.

We commonly model the noise as a Gaussian distribution, so that the mean of the model is a hyper-plane.

$$p(y \mid \theta, x) = \mathcal{N}(y \mid \mathbf{w}^T\mathbf{x}, \sigma^2)$$

As mentioned above, one common way to estimate the parameters is to use Maximum Likelihood Estimation[21].

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} \log \mathcal{N}(w^T, \sigma^2) \\ &= \sum_{i=1}^{n} -\frac{1}{2\sigma^2}(y_i - w^T x_i) - \frac{1}{2}\log(2\sigma^2\pi) \end{aligned}$$

Then

$$\min_{w,\sigma^2} -l(\theta) = \min_{\sigma^2}\left[-\frac{1}{2}\log(2\sigma^2\pi) + \frac{1}{2\sigma^2}\min_w RSS(w)\right]$$

in which

$$RSS = \frac{1}{2}||y - xw||_2^2 = \frac{1}{2}(y - xw)^T(y - xw)$$

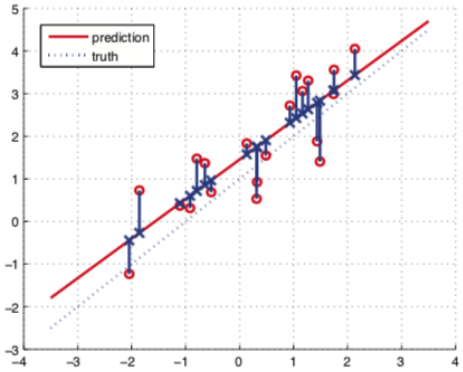


Figure 2. Linear least squares[21]

There're several ways, the minimization can be done:

- Linear least squares, we try to minimize the sum of squared distances from each training point (denoted by a red circle) to its approximation[21]
- Maximum Likelihood Estimation

$$\mathbf{w} = \arg\max_{\mathbf{w}} \log p(\mathbf{x} \mid \mathbf{w})$$

- Gradient Descent

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \gamma\nabla l(\mathbf{w})$$

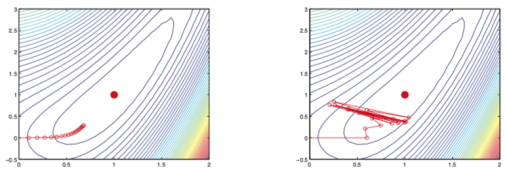


Figure 3. Gradient Descent[21]

In the first two ways, we are able to get a closed form of estimation

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

In order to prevent this model from overfitting, regularization terms can be used to add punishment to the model if it does not satisfy some requirements, so that the loss function is now

$$L(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2^2$$

We see that l_2 norm punishes weights that has too large magnitude and l_1 forces as many weights as possible to be zero[4].

Similarly, we have the estimation of $\mathbf{w}$ to be

$$\mathbf{w}^* = (\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

this result can be derived from MAP as well, since regularization can also be regarded as prior distribution of $\mathbf{w}$[4].

For classification tasks, we introduce sigmoid function[2]

$$\sigma(x) = \frac{1}{1+\exp(\mathbf{w}^T\mathbf{x})}$$

and softmax function[2]

$$f(x) = \frac{\exp(\mathbf{w}_j^T\mathbf{x})}{\sum_{k=1}^{K}\mathbf{w}_k^T\mathbf{x}}$$

to map vector to the probability an instance should be classified as class k respectively for binary and muiticlass classification.

In order to get weights that can minimize the loss function, other than gradient descent, we can also apply Stochastic Gradient Descent(SGD)

$$w = w - \frac{1}{n}\eta\sum_i \nabla l(w, x_i)$$

[4]This means that l_1 norm can give us sparsity. In fact, l_0 norm gives the best sparsity, but that would make the optimization problem a NP problem to solve

The major advantage of SGD over GD is that, SGD take only part of the data(called a batch) to compute the gradient and iteratively update the weights. SGD also allows model to perform online learning, so the model can consistently evolve as new data flows in.[2] And this strategy is very effective in the case of large-scale machine learning problems[5].

One significant challenge is that the loss function is almost never convex[17], which means that SGD only guarantees local minima, so the performance of SGD heavily relies on initialization.

In high-dimensional space, the occasion is a little different. As the dimension goes up, the probability the gradient is zero in any direction become very low, but there are a lot of saddle points[17], and at those points the convergence will be very slow. Also, when the learning rate is fixed, if learning rate is too small, the training will suffer from painfully slow convergence; if learning rate is too large, the loss function will fluctuate around the minimum or even to diverge[25].

In order to overcome those flat areas in weight space, many strategies are designed.

For areas like ravines, namely, areas where the surface curves much more steeply in one dimension than in another[26], momentum is used to modify the direction of descent with the direction of last update

$$v_t = \gamma v_{t-1} + \eta\nabla_w L(w)$$

and

$$w = w - v_t$$

Figure 4. SGD without momentum

Just blindly following the slope is usually unsatisfactory[25], so that we have SGD to have a notion of where to slow down before the hill slopes up again. Nesterov accelerated gradient (NAG)[22] is able to give the momentum such prescience. NAG first makes a big jump in the direction of the previous accumulated gradient, measures the gradient and then makes a correction, which results in the complete NAG update[25].

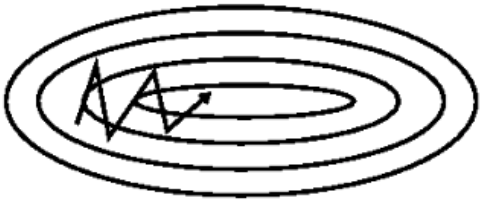


Figure 5. SGD with momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_w L(w - \gamma v_{t-1})$$
$$w = w - v_t$$

On the other hand, in order to handle the problem of learning rate scheduling, some other updating algorithms are invented. Adagrad[10] adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters, so that it greatly improved the robustness of SGD[9] by excellently deal with sparse data.

Geoffery Hinton proposed RMSProp in his lecture[5], which can adaptive learning rate.

5. Logic based Models

5.1. Decision Tree

By recursively splitting datasets for training based on some attributes of the datasets, the decision tree algorithm can partition the input space $\mathcal{X}$ into different regions, so that the all (or at least majority of) points $x_i \in \mathcal{X}_j$ has the same label y. RMSprop divides the learning rate by an exponentially decaying average of squared gradients[25].

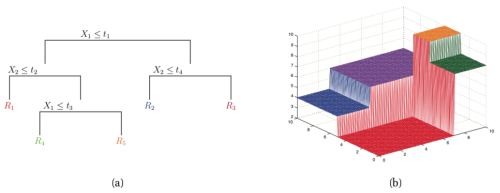


Figure 6. Decision Trees[21]

In order to increase the purity of each branch after splitting, there're several evaluation metrics to determine based on which attribute and the corresponding value can the post-split purity reaches the highest value. In general, there're several properties that the impurity metric ϕ should meet in order to be use:

[5]http://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf

For simplicity, we consider the metric for binary classification, and let p be the probability of a label if uniformly randomly picked, then

1. $\forall p \in [0, 1], \phi(1/2, 1/2) \geq \phi(p, 1-p)$
2. $\phi(0, 1) = \phi(1, 0) = 0$
3. p is increasing on $[0, 1/2]$ and decreasing on $[1/2, 1]$

These properties and be easily extend to k-ary classification, and some of the most commonly used methods are

- Entropy: $\phi(p) = -(p \log_2 p + (1-p) \log_2 (1-p))$
- Gini index: $\phi(p) = 2p(1-p)$

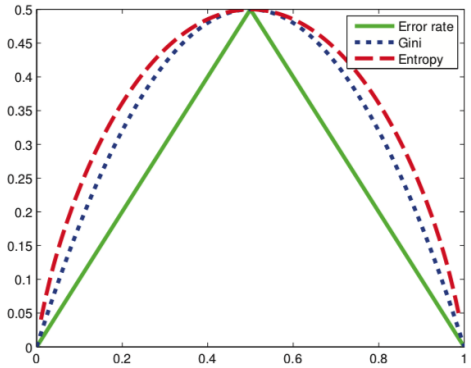


Figure 7. Impurity measures[21]

One problem of Decision Tree is that, when completely grown, a decision tree suffers seriously from overfitting. In order to prevent that, we can stop growing the tree if the decrease in the error is not su cient to justify the extra complexity of adding an extra subtree[21] or prune the fully grown tree by using a scheme that prunes the branches giving the least increase in the error[7].

Decision Tree is popular for the following reasons:

(i). Good intepretability[6]

(ii). Scalability, they can easily handle mixed discrete and continuous inputs and insensitive to monotone transformations of the inputs[21]

(iii). Robust to outliers[21]

However, the decision tree algorithm also has some serious disadvantages:

(i). The accuracy of prediction is not satisfactory comparing to other approaches

[6]We can postprocess the tree to derive a series of logical rules[24]

(ii). Small changes to the input data can have large effects on the structure of the tree because it's a high variance estimator[21]

5.2. Random Forest

In order to reduce the variance of the model, we can apply random forest algorithm. By uniformly sampling data from training dataset with replacement and randomly choosing f attributes for a node to pick and grow, we made use of bagging strategy to randomize the training process to train M different trees. Then we can compute the ensemble by averaging it

$$f(x) = \frac{1}{M}\sum_{m=1}^{M} f_m(x)$$

5.3. Boosting

Another ensemble method to reduce variance and improve performance of single decision tree is boosting.

By consecutively train a series of trees with training data and those instances that fails to be predicted on the previous trained trees.

This strategy is equivalent to adjusting the weights of each weak classifier (stump in this case), with the following rules:

$$err_m = \frac{\sum_{i=1}^{N} w_{i,m}\mathbb{I}(y_i \neq \phi_m(x_i))}{\sum_{i=1}^{N} w_{i,m}}$$

$$\alpha_m = \log[(1 - err_m)/err_m]$$

$$w_i = w_i exp[\alpha_m \mathbb{I}(y_i \neq \phi_m(x_i)))]$$

then finally we have our classifier by weighted averaging the stumps

$$f(x) = sgn\left[\sum_{m=1}^{M} \alpha_m \phi_m(x))\right]$$

6. Neural Networks

The development of Neural Network and, especially, of Deep Learning is explosive in recent years since the success of AlexNet[19] and the development of computational hardwares like GPU and TPU give us ability to deal with huge amount of data, so that the algorithm is widely used to construct artificial intelligence systems in various kinds of industries.

6.1. Single Layer Perceptron

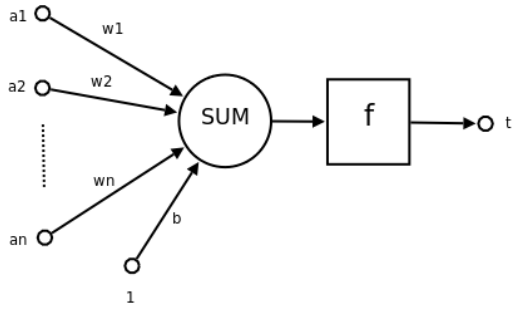
The start of the development of deep learning is single layer perceptron in late 1950s[11]. It is an algorithm for supervised learning of binary linear classifiers.

It is formed like a linear discriminant model

$$y = sign(\mathbf{w}^T\mathbf{x} + b)$$

and use SGD to update the weights until convergence.

The major and significant disadvantage of the perceptron is that it is equivalent to a linear discriminant model, so it is impossible for it to classify data that's not linearly separable. This means that a perception can't even fit xor function[2].



6.2. Multilayer Perceptron

In order to handle the challenges, some modifications are added into the model.

First, instead of only one layer, more layers are connected to the first layer to form the so called Multilayer Perceptron or Artificial Neural Network. Second, in order to add nonlinearity to fit sophisticated dataset, nonlinear activation function. Activation function is required because without it, the multilayer model will be equivalent to a linear model[12].

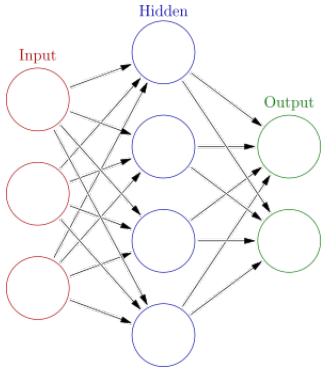


Figure 8. Neural Network

There are a lot of options for activation function, some of them are listed in the table below.

Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

Figure 9. Common Activation Functions

The most common activation function is sigmoid, this is a smooth function and is continuously differentiable. The biggest advantage that it has over step and linear function is that it is non-linear[13], and the analytical property of this function is really good that its derivative has a simple closed form.

The tanh function is a scaled version of the sigmoid function, and is symmetric over the origin, so it basically solves our problem of the values all being of the same sign[13].

The big disadvantage of those stair-like functions is that sigmoid can be saturated with large weights and the learning will be slowed down[18]. To solve this problem, Hinton[18] proposed to use Rectified Linear Units(ReLU). The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time[13] because if the input is negative it will convert it to zero and the neuron is not activated.

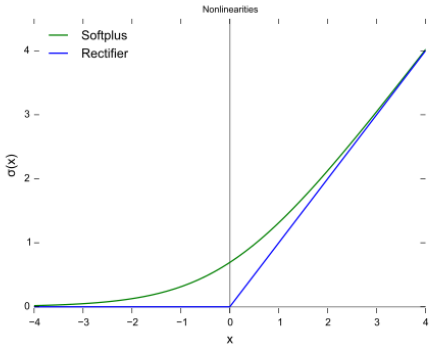


Figure 10. ReLU vs. Softplus

The problem of ReLU is that when using it, we should be careful because when the activation is trapped by the negative side, since the gradient there is constantly zero, the weights will not be updated anymore. Though this is not a big problem because we can add a small amount to the left side to avoid the problem (Leaky ReLU). Because of the advantages of ReLU, it is still the most popular activation function in Machine Learning community[13].

6.3. Convolution Neural Network (CNN)

Another significant problem of traditional Neural Network is that as the scale of network grows, the number of parameters grows exponentially and thus the convergence will be pretty slow.

Inspired by the research about Receptive fields[16], LeCun introduced the well-known LeNet-5 structure[8] to recognize handwritten numbers in MNIST dataset[7]

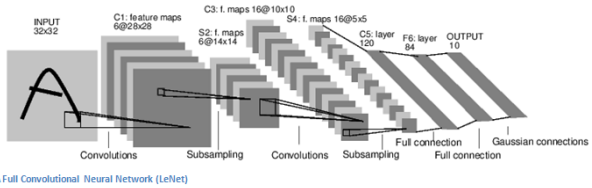


Figure 11. LeNet-5

The symbol of the success of CNN in computer vision is made by the excellent performance made by AlexNet[18], GoogLeNet[27] etc. in competitions like ImageNet.

The basic design of CNN are the following:

(i) Shared weights: Shared weights is directly inspired by the idea of Receptive fields. In the network, an input image will be scanned by a window (so called convolution) to generate the next level of feature maps. This gives CNN ability to tolerate translation of the input image. By using only a set of shared weights (filters).

(ii) Pooling: Another important design of CNN is pooling, which is a kind of non-linear down-sampling. The intuition is that the exact location of a feature is less important than its rough location relative to other features[12]. Pooling also serves to reduce the spatial size of the represention, which means the number of parameters.

6.4. Recurrent Neural Network (RNN)

While CNN can excellently captures spatial structures, RNN is proposed to dig in sequential and temporal data. They are networks with loops in them, allowing information to persist[1]

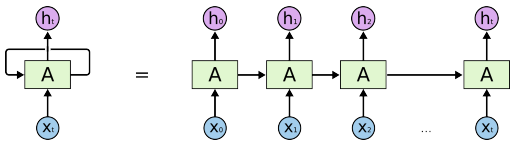
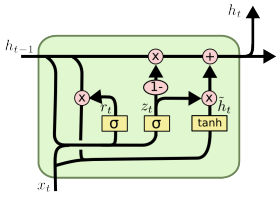


Figure 12. RNN

[7] http://yann.lecun.com/exdb/mnist/

Traditional RNN suffers from problem of gradient explosion and gradient vanishing because of the chain rule of derivative[12]. In order to handle this problem and enable the network to detect long-term dependencies, Long Short Term Memory networks(LSTM) is introduced[15]. The horizontal line running through the top of the diagram transports the state of cell and input gate and forget gate make the network able to decide which of the information should be remembered and which should be forgot.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 13. LSTM

6.5. Residue Network

Just as mention above, as the depth of neural network grows, they became more and more difficult to train due to long dependencies. Kaiming He proposed a residual learning framework to ease the training of networks that are substantially deeper than those used previously[14].

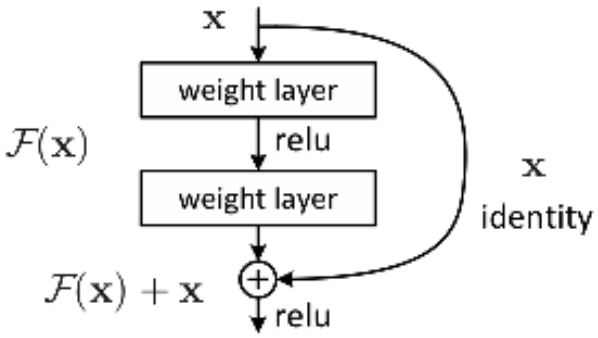


Figure 14. ResNet[14]

By adding the skip structure, the gradient can be passed and maintained to really deep layers and accelerate the training of very massive network. Empirically speaking, the deeper a network is, the more parameters it has, the more capacity the model can show. By solving this problem, massive networks can be trained to solve pretty sophisticated tasks, so the framework became very popular.

7. Kernel

When data is not linearly separable, another classic trick is to define a kernel function to map the data to a higher dimensional space, where the data can be separated with a hyper-plane.

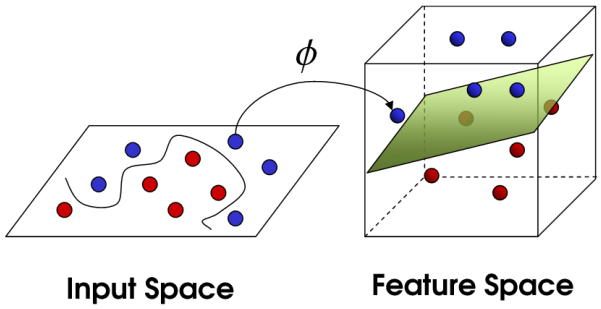


Figure 15. Kernel Function

7.1. Kernel Ridge Regression

Recall that in ridge regression, we have our primal problem

$$L(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||^2$$

whose solution is given by

$$\mathbf{w}^* = (\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

then we can define the the following dual variables

$$\alpha = (\mathbf{K} + \lambda\mathbf{I})\mathbf{y}$$

then we can write

$$\mathbf{w} = \mathbf{X}\alpha$$

which gives

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i)$$

The cost of computing the dual variables α is $O(N^3)$, whereas the cost of computing the primal variables w is $O(D^3)$. Hence the kernel method can be useful in high dimensional settings[21].

7.2. Support Vector Machine

In order to have high robustness, SVM does not only requires the complete separate of data of different labels, but also requires that the hyper-plain can have the maximized decision margin[21]. We can convert it to an constrained optimization problem

$$\min_{\mathbf{w}} \frac{1}{2}||w||^2, s.t. y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1$$

and sometimes we may which the SVM to have some tolerance, We replace the hard constraints that $y_i f_i \geq 0$ with the soft margin constraints that $y_i f_i \geq 1\xi_i$, so that the optimization problem can become

$$\min_{\mathbf{w},\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i, s.t. \xi_i \geq 0, y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i$$

Both problems can be solved using SGD, or SMO algorithm[23].

8. Instance-based learning

Instance-based learning algorithms are lazy-learning algorithms[20] because they delay the induction or generalization process until classification is perform[18]. Instance-based learning requires little training time but a lot of evaluation time during the process of classification[18]. The most straight-forward and common example is the nearest neighbor algorithm.

k nearest neighbor algorithm is based on the assumption that spatial closeness leads to categorical similarity, thus it decides the class of a data point by calculating k most close points and make decision by majority vote.

In order to define spatial dissimilarity, metric functions are required. A general metric function $d : M \times M \to \mathbb{R}$ should satisfy the following requirements:

(i).

$$d(x, y) \geq 0$$

and $d(x, y) = 0$ if and only if $x = y$.

(ii).

$$d(x, y) = d(y, x)$$

(iii).

$$d(x, z) \leq d(x, y) + d(y, z)$$

Commonly used metrics are Minkowsky distance, Euclidean distance etc.[18]

Computing the nearest neighbor can be really costly when the number of data is large, some data structures are introduced to mitigate this issue. For instance, k-d tree[3] is very commonly used to compute nearest neighbor quickly.

9. Conclusion

The review of supervised learning explored all commonly used algorithms and models for classification and regression. By analyzing the pros and cons of each algorithms and present how these models can be fixed to deal with certain challenges, it is reasonable to think this paper presented a relatively detailed guide to researchers on the selection and use of models that are best-known in the domain of supervised learning.

References

[1] Understanding lstm networks.

[2] E. Alpaydin. *Introduction to machine learning*. The MIT Press, 2004.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509517, Jan 1975.

[4] C. Bishop. *Pattern recognition and machine learning*. Springer Verlag.

[5] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc), 2008.

[6] R. Bouckaert. Evaluating the replicability of significance tests for comparing learning algorithms. *Advances in knowledge discovery and data*, 2004.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[8] Y. L. Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. *Neurocomputing*, page 303318, 1990.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng. Large scale distributed deep networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1223–1231. Curran Associates, Inc., 2012.

[10] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[11] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. The MIT Press, 2017.

[13] D. Gupta, S. Bansal, P. Srivastava, and S. Jain. Fundamentals of deep learning - activation functions and their use, Oct 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[16] D. Hubel and T. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[17] K. Kawaguchi. Deep learning without poor local minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 586–594. Curran Associates, Inc., 2016.

[18] S. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[20] T. M. Mitchell. *Machine learning*. McGraw Hill, 2017.

[21] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2013.

[22] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). *Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543 547*, 1983.

[23] J. Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[24] J. R. Quinlan. Decision trees and decision-making. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(2):339–346, 1990.

[25] S. Ruder. An overview of gradient descent optimization algorithms, Dec 2017.

[26] R. S. Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1986.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[28] Y. Yang and G. Webb. On why discretization works for naive-bayes classifiers, 12 2003.