Machine Learning Basics

Lecture slides for Chapter 5 of Deep Learning

www.deeplearningbook.org

Definition

- A machine learning algorithm: an algorithm that is able to learn from data.
- Mitchell (1997) "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The Task, T

- Machine learning tasks are usually described in terms of how the machine learning system should process an **example**.
- An example is a collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process
- We typically represent an example as a vector $x \in \mathcal{R}^n$ where each entry x_i of the vector is a feature

Kinds of Tasks

- **Classification**: specify which of k categories some input belongs to. $f: \mathcal{R}^n \rightarrow \{1, \ldots, k\}$
  - When $y = f(x)$, the model assigns an input described by vector x to a category identified by numeric code y.
  - f outputs a probability distribution over classes
- **Classification with missing inputs**: some of the inputs maybe missing,
  - the learning algorithm must learn a set of functions. Each function corresponds to classifying x with a different subset of its inputs missing.
  - One way to efficiently define such a large set of functions is to learn a probability distribution over all of the relevant variables, then solve the classification task by marginalizing out the missing variables.
  - Goodfellow et al. (2013) for an example of a deep probabilistic model applied to such a task

Task Types

- Regression: predict a numerical value given some input. function f: $\mathcal{R}^n \rightarrow \mathcal{R}$
- Transcription: the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form. optical character recognition, speech recognition
- Machine translation
- Structured output: the output is a vector (or other data structure containing multiple values) with important relationships between the different elements
  - Parsing—mapping a natural language sentence into a tree that describes its grammatical structure and tagging nodes of the trees as being verbs, nouns, or adverbs, etc.
  - Image Captioning

- **Anomaly detection:** In this type of task, the computer program sifts through a set of events or objects, and flags some of them as being unusual or atypical.
- **Synthesis and sampling**: The machine learning algorithm is asked to generate new examples that are similar to those in the training data.
  - video games can automatically generate textures for large objects or landscapes, rather than requiring an artist to manually label each pixel (Luo et al., 2013)
  - a speech synthesis task, we provide a written sentence and ask the program to emit an audio waveform containing a spoken version of that sentence.

- Imputation of missing values: In this type of task, the machine learning algorithm is given a new example $x \in \mathcal{R}^n$, but with some entries missing. The algorithm must provide a prediction of the values of the missing entries
- Denoising: In this type of task, the machine learning algorithm is given in input a corrupted example $\bar{x} \in \mathcal{R}^n$ obtained by an unknown corruption process from a clean example $x \in \mathcal{R}^n$. The learner must predict the clean example from its corrupted version $\bar{x}$, or more generally predict the conditional probability distribution $p(x|\bar{x})$

- Density estimation or probability mass function estimation: learn a function $p_{model}: \Re^n \rightarrow \mathcal{R}$, where $p_{model}(x)$ can be interpreted as a probability density function (if x is continuous) or a probability mass function (if x is discrete) on the space that the examples were drawn from.
  - the algorithm needs to learn the structure of the data it has seen. It must know where examples cluster tightly and where they are unlikely to occur

The Performance Measure, *P*

- a quantitative measure of its performance. Usually this performance measure P is specific to the task T
- Classification:
  - Accuracy – the proportion of examples for which the model produces the correct output
  - Equivalently, the error rate, the proportion of examples for which the model produces an incorrect output
    - the expected 0-1 loss
- density estimation task: average log-probability the model assigns to some examples
- how well the machine learning algorithm performs on data that it has not seen before, since this determines how well it will work when deployed in the real world.
- Test set different from trainings set

The Experience, E

- Unsupervised or supervised
- Dataset
- Data points

Linear Regression

- In the case of linear regression, the output is a linear function of the input. Let $\hat{y}$ be the value that our model predicts y should take on. We define the output to be

$$\hat{y} = w^T x$$

$$MSE_{test} = \frac{1}{m} \left\| \hat{y}^{(test)} - y^{(test)} \right\|_2^2$$

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\mathrm{train})} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} ||\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0$$

$$\hat{y} = w^T x$$

$$MSE_{test} = \frac{1}{m} \left\| \hat{y}^{(test)} - y^{(test)} \right\|_2^2$$

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\mathrm{train})} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} ||\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right)^\top \left(\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right) = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2 \boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} + \boldsymbol{y}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} \right) = 0$$

$$\Rightarrow 2 \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2 \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} = 0$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \right)^{-1} \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})}$$

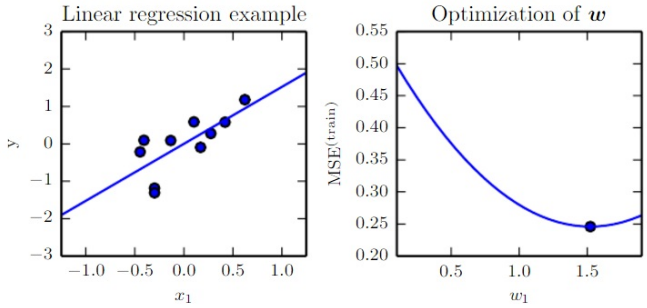


Figure 5.1: A linear regression problem, with a training set consisting of ten data points, each containing one feature. Because there is only one feature, the weight vector $\boldsymbol{w}$ contains only a single parameter to learn, w_1. *(Left)*Observe that linear regression learns to set w_1 such that the line $y = w_1 x$ comes as close as possible to passing through all the training points. *(Right)*The plotted point indicates the value of w_1 found by the normal equations, which we can see minimizes the mean squared error on the training set.

Underfitting and Overfitting in Polynomial Estimation

- The central challenge in machine learning is that we must perform well on new, previously unseen inputs – generalization
- Reduce training error – an optimization problem
- ML: we want the generalization error (test error) to be low.
- The train and test data are generated by a probability distribution over datasets called the data generating process.
- We typically make a set of assumptions – the i.i.d. assumptions.
  - The examples in each dataset are independent from each other
  - the trainset and test set are identically distributed, drawn from the same probability distribution as each other
- We sample the training set, then use it to choose the parameters to reduce training set error, then sample the test set.
- Under this process, the expected test error is greater than or equal tothe expected value of training error.

Underfitting and Overfitting

- We sample the training set, use it to choose the parameters to reduce training set error, then sample the test set.
  - the expected test error is greater than or equal to the expected value of training error.
- The factors determining how well a machine learning algorithm will perform are its ability to:
  1. Make the training error small
  2. Make the gap between training and test error small

1. Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.
2. Overfitting occurs when the gap between the training error and test error is too large

- We can control whether a model is more likely to overfit or underfit by altering its capacity.
- A model's capacity is its ability to fit a wide variety of functions.
  - Models with low capacity may struggle to fit the training set.
  - Models with high capacity can overfit by memorizing properties of the training set
- One way to control the capacity of a learning algorithm is by choosing its hypothesis space
- Generalizing linear regression to include polynomials in its hypothesis space increases the model's capacity.

- A polynomial of degree one gives us the linear regression model
$$\hat{y} = b + wx$$
- Quadratic model
$$\hat{y} = b + w_1 x + w_2 x^2$$
- The output is still a linear function of the parameters, so we can still use the normal equations to train the model in closed form
- Polynomial of degree 10
$$\hat{y} = b + \sum_{i=1}^{10} w_i x^i$$
- ML algorithms will generally perform best when their capacity is appropriate for
  - the true complexity of the task and
  - the amount of training data
- Models with insufficient capacity are unable to solve complex tasks.
- Models with high capacity may overfit

Underfitting and Overfitting in Polynomial Estimation

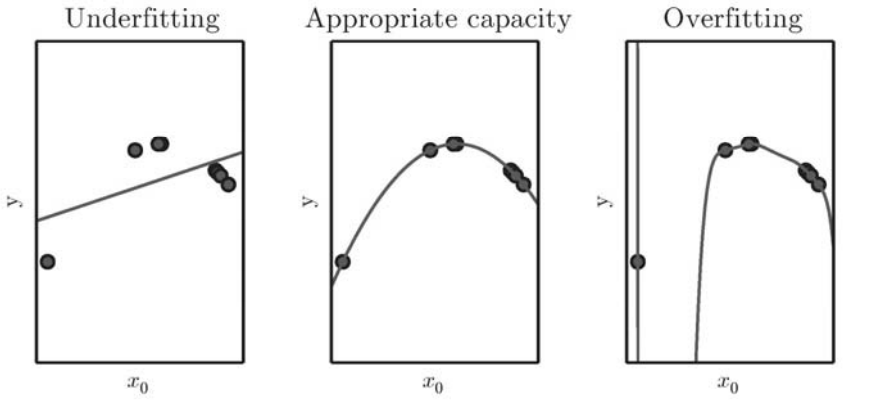


Figure 5.2

(Goodfellow 2016)

Capacity

- Many ways of changing a model's capacity.
  - changing the number of input features and adding corresponding parameters
- Representational capacity – the model specifies which family of functions the learning algorithm can choose from
- Finding the best function within this family is an optimization problem
- Imperfection of the optimization => the effective capacity may be less than the representational capacity of the model family
- Occam's razor(c. 1287-1347) among competing hypotheses that explain known observations equally well, one should choose the "simplest" one
- The problem ofdetermining the capacity of a deep learning model is especially difficult because theeffective capacity is limited by the capabilities of the optimization algorithm, andwe have little theoretical understanding of the very general non-convex optimizationproblems involved in deep learning

Generalization and Capacity

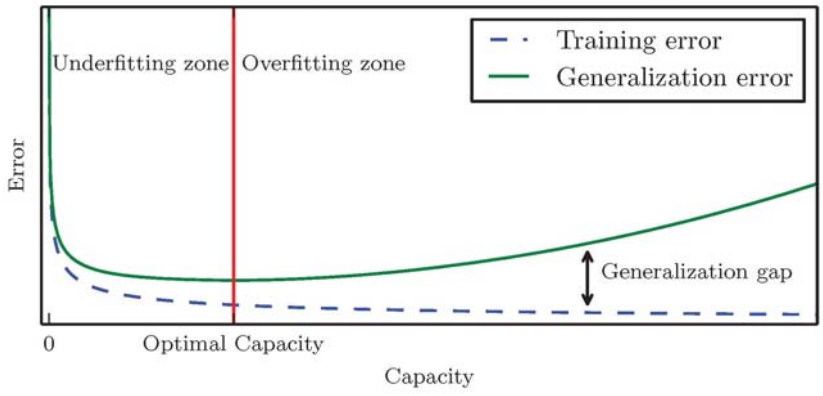


Figure 5.3

(Goodfellow 2016)

Non-parametric

- Parametric models learn a function described by a parameter vector whose size is finite and fixed
- Non-parametric models have no such limitation
- To reach the most extreme case of arbitrarily high capacity, non-parametric models.
- nearest neighbour regression: simply stores the X, y from the training set. When asked to classify a test point x, the model looks up the nearest entry in the training set and returns the associated regression target
- we can also create a non-parametric learning algorithm by wrapping a parametric learning algorithm inside another algorithm that increases the number of parameters as needed.

- The ideal model is an oracle that simply knows the true probability distribution that generates the data.
- Even such a model will still incur error, because there may still be some noise in the distribution.
- The mapping may be inherently stochastic, or y may be a deterministic function that involves other variables besides those included in x.
- The error incurred by an oracle making predictions from the true distribution p(x, y) is called the Bayes error.

Generalization error

- Expected generalization error can never increase as the number of training examples increases.
- non-parametric models, more data yields better generalization until the best possible error is achieved.
- Any fixed parametric model with less than optimal capacity will asymptote to an error value that exceeds the Bayes error.
- model with optimal capacity may still have a large gap between training and generalization error
  - may be able to reduce this gap by gathering more training examples.

The No Free Lunch Theorem

- Inductive reasoning, or inferring general rules from a limited set of examples, is not logically valid.
- To logically infer a rule describing every member of a set, one must have information about every member of that set
- ML avoids this problem by offering only probabilistic rules, rather than the entirely certain rules used in purely logical reasoning.
- promises to find rules that are probably correct about most members of the set they concern
- The no free lunch theorem (Wolpert, 1996) states that, averaged overall possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.
- our goal is to understand what kinds of distributions are relevant to the "real world" that an AI agent experiences, and what kinds of machine learning algorithms perform well on data drawn from the kinds of data generating distributions we care about

Figure 5.4

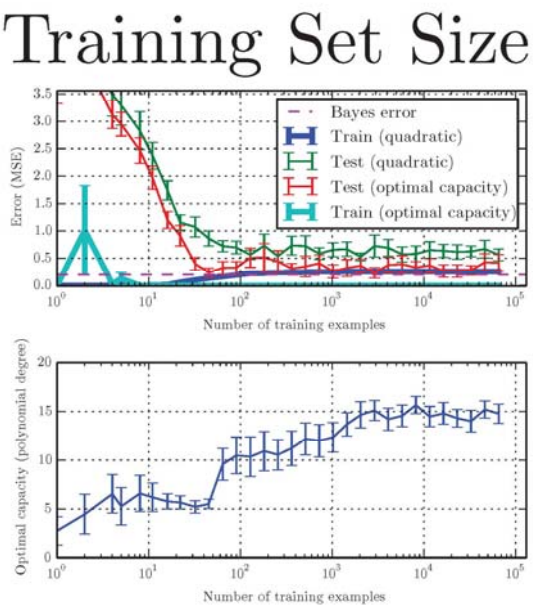


Fig 5.4

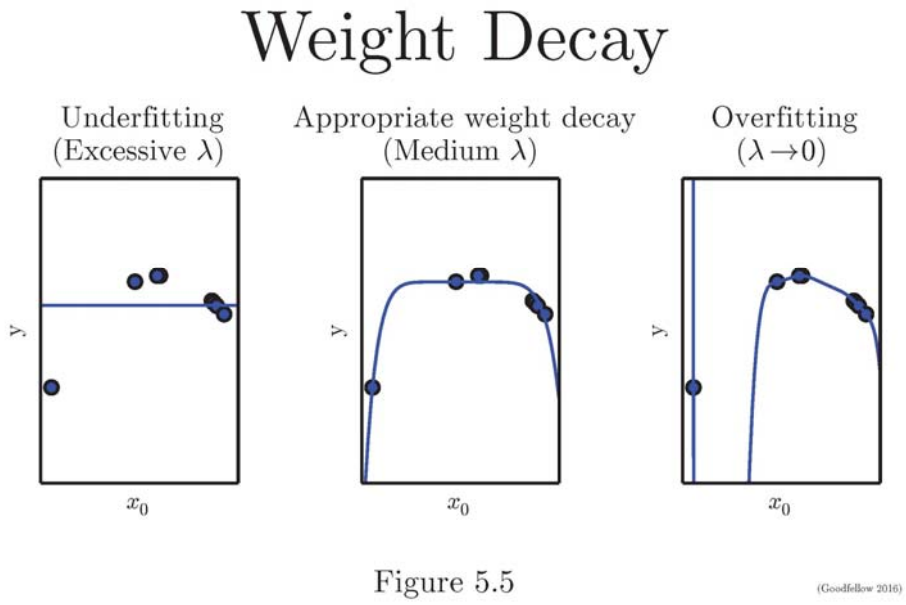
- The effect of the training dataset size on the train and test error, and the optimal model capacity.
- A synthetic regression problem based - add moderate noise to a degree-5 polynomial, generated a single test set, and then generated several different sizes of training set. For each size, generated 40 different training sets ito plot error bars showing 95 percent confidence intervals.
- (Top)The MSE on the training and test set a quadratic model, and a model with degree chosen to minimize the test error. Both are fit in closed form.
- quadratic model: training error increases with the size of the training set because larger datasets are harder to fit.
  the test error decreases, because fewer incorrect hypotheses are consistent with the training data. The quadratic model not have enough capacity, so test error asymptotes to a high value.
- The test error at optimal capacity asymptotes to the Bayes error.
- The training error can fall below the Bayes error, due to the ability of the training algorithm to memorize specific instances of the training set.
- As the training size increases to infinity, the training error of any fixed-capacity model (here, the quadratic model) must rise to at least the Bayes error.
- (Bottom)As the training set size increases, the optimal capacity(shown here as the degree of the optimal polynomial regressor) increases. The optimal capacity plateaus after reaching sufficient complexity to solve the task

Regularization

- No free lunch theorem implies we must design ML algorithms to perform well on a specific task. We do so by building a set of preferences into the learning algorithm.
- The behaviour of our algorithm is strongly affected not just by how large we make the set of functions allowed in its hypothesis space, but by the specific identity of those functions
- We can also give a learning algorithm a preference for one solution in its hypothesis space to another
- For example, we can modify the training criterion for linear regression to include weight decay

$$J(w) = MSE_{train} + \lambda w^T w$$

- λ controls the strength of preference for smaller weights



- We fit a high-degree polynomial regression model to our example training set.
- The true function is quadratic, we use models with degree 9.
- We vary the amount of weight decay to prevent these high-degree models from overfitting.
- With very large λ, we can force the model to learn a function with no slope at all. This underfits because it can only represent a constant function.
- (Center) With a medium value of λ, the learning algorithm recovers a curve with the right general shape. weight decay has encouraged it to use a simpler function described by smaller coefficients.
- (Right)With weight decay approaching zero thedegree-9 polynomial overfits significantly

- More generally, we can regularize a model that learns a function f(x;θ) by adding a penalty called a regularizer to the cost function.
- In the case of weight decay, the regularizer is Ω(w) $=w^T w$
- Many other regularizers are possible
- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Hyperparameters and Validation Sets

- Most ML algs have several settings that we can use to control the behaviour - hyperparameters.
- Poly regression problem –
  - degree of the poly is the hyperparameter
  - Value of lambda (weight decay)
- Sometimes a setting is chosen to be a hyper-parameter that the learning algorithm does not learn because it is difficult to optimize.
- Hyper-parameters that control model capacity cannot be learned on training set
- To solve this problem, we need avalidation set

Cross-Validation

Estimators, Bias and Variance

- Function Estimation (or function approximation): predict a variable y given an input vector x. We may assume

$$y = f(x) + \epsilon$$

- Bias of an estimator:

$$bias(\widehat{\theta_m}) = \mathrm{E}[\hat{\theta}_m] - \theta$$

- Variance and Standard Error
  - how much we expect it to vary as a function of the data sample.

$$Var(\hat{\theta})$$

  - the square root of the variance is called the standard error, denoted SE(ˆθ).
  - a measure of how we would expect the estimate we compute from data to vary as we independently resample the dataset from the underlying data generating process.

Trading off Bias and Variance to Minimize Mean Squared Error

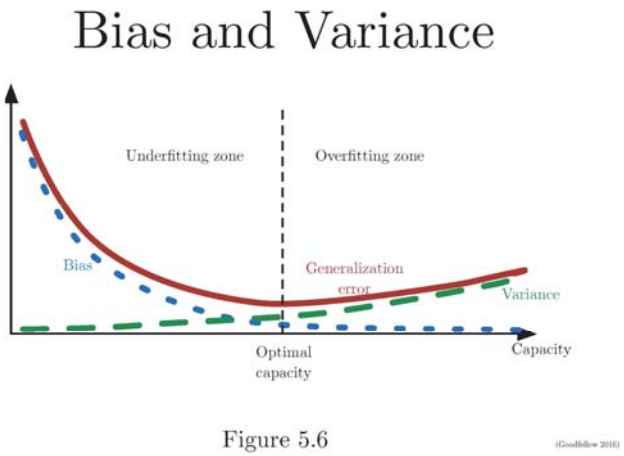


Figure 5.6

Maximum Likelihood Estimation

- derive specific functions that are good estimators for different models
- Consider m examples $\mathbb{X} = \{x^{(1)}, \dots, x^{(m)}\}$ drawn independently from the true but unknown data generating distribution $p_{data}(x)$
- Let $p_{model}(x; \theta)$ be a parametric family of probability distributions over the same space indexed by θ
- MLE for θ is defined as

$$\theta_{ML} = \arg\max_{\theta} p_{model}(\mathbb{X}; \theta)$$

$$= \arg\max_{\theta} \prod_{i=1}^{m} p_{model}(x^{(i)}; \theta)$$

$$\theta_{ML} = \arg\max_{\theta} \sum_{i=1}^{m} \log p_{model}(\mathbb{X}; \theta)$$

Conditional Log-Likelihood and Mean Squared Error

- The maximum likelihood estimator can readily be generalized to the case where our goal is to estimate a conditional probability $P(y|x; \theta)$
- If X represents all our inputs and Y all our observed targets, then the conditional maximum likelihood estimator is

$$\theta_{ML} = \underset{\theta}{argmax}\, P(Y|X; \theta)$$

- If the examples are assumed to be i.i.d., then this can be decomposed into

$$\theta_{ML} = \underset{\theta}{argmax} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}; \theta)$$

Example: Linear Regression as Maximum Likelihood

- Previously, we motivated linear regression as an algorithm that learns to take an input x and produce an output value $\hat{y}$. The mapping from x to y is chosen to minimize mean squared error
- Instead of producing a single prediction $\hat{y}$, we now think of the model as producing a conditional distribution p(y | x)
- We define $p(y|x) = \mathcal{N}(y; \hat{y}(x, w), \sigma^2)$
- $\hat{y}(x, w)$ gives the prediction of the mean of the Gaussian. In this example, we assume that the variance is fixed to some constant σ^2

- Since the examples are assumed to be i.i.d., the conditional log-likelihood is given by

$$\sum_{i=1}^{m} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \tag{5.64}$$

$$= -m \log \sigma - \frac{m}{2} \log(2\pi) - \sum_{i=1}^{m} \frac{\|\hat{y}^{(i)} - y^{(i)}\|^2}{2\sigma^2}, \tag{5.65}$$

where $\hat{y}^{(i)}$ is the output of the linear regression on the i-th input $\boldsymbol{x}^{(i)}$ and m is the number of the training examples. Comparing the log-likelihood with the mean squared error,

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{m} \sum_{i=1}^{m} ||\hat{y}^{(i)} - y^{(i)}||^2, \tag{5.66}$$

we immediately see that maximizing the log-likelihood with respect to $\boldsymbol{w}$ yields the same estimate of the parameters $\boldsymbol{w}$ as does minimizing the mean squared error. The two criteria have different values but the same location of the optimum.

Supervised Learning Algorithms

- Linear regression
- Logistic Regression
- Support Vector Machines
- kNN
- Decision Tree

Unsupervised Learning Algorithms

- Informally, unsupervised learning refers to most attempts to extract information from a distribution that do not require human labour to annotate examples.
  - density estimation,
  - learning to draw samples from a distribution,
  - learning to denoise data from some distribution,
  - finding a manifold that the data lies near,
  - clustering the data into groups of related examples
- A classic unsupervised learning task is to find the “best” representation of the data
  - preserves as much information about x as possible while obeying some penalty or constraint aimed at keeping the representation simpler

Simpler representation

1. lower dimensional representations
   - compress as much information about x as possible in a smaller representation
2. sparse representations
   - embed the dataset into a representation whose entries are mostly zeroes for most inputs
3. independent representations
   - Disentangle the sources of variation underlying the data distribution such that the dimensions of the representation are statistically independent

- The notion of representation is one of the central themes of deep learning

Principal Components Analysis

- PCA provides a means of compressing data
  - We can view PCA as an unsupervised learning algorithm that learns a representation of data.
  - Learn lower dimension representation
  - elements have no linear correlation with each other
- a first step toward the criterion of learning representations whose elements are statistically independent.
- To achieve full independence, a representation learning algorithm must also remove the nonlinear relationships between variables
- PCA learns an orthogonal, linear transformation of the data that projects an input x to a representation z

k-means Clustering

- An example of a simple representation learning algorithm
- We can thus think of the algorithm as providing a k-dimensional one-hot code vector h representing an input x
- The one-hot code provided by k-means clustering is an example of a sparse representation
- we may prefer a distributed representation to a one-hot representation.
- A distributed representation could have two attributes for each vehicle—one representing its color and one representing whether it is a car or a truck.
- having many attributes reduces the burden on the algorithm to guess which single attribute we care about, and allows us to measure similarity between objects in a fine-grained way by comparing many attributes

Stochastic Gradient Descent

- Nearly all of deep learning is powered by one very important algorithm: stochastic gradient descent or SGD
- large training sets computationally expensive but necessary for generalization
- The cost function used by a machine learning algorithm often decomposes as a sum over training examples of some per-example loss function.
- The insight of stochastic gradient descent is that the gradient is an expectation. The expectation may be approximately estimated using a small set of samples

Building a ML Algorithm

- Nearly all deep learning algorithms can be described as particular instances of a fairly simple recipe:
- combine a specification of
  - a dataset,
  - a cost function,
  - An optimization procedure and
  - a model.

Challenges Motivating Deep Learning

- motivated in part by the failure oftraditional algorithms to generalize well on such AI tasks as speech recognition, object recognition NLP
- the challenge of generalizing to new examples becomes exponentially more difficult when working with high-dimensional data
- the mechanisms used to achieve generalization in traditional ML are insufficient to learn complicated functions in high-dimensional spaces.

The Curse of Dimensionality

- The number of possible distinct configurations of a set of variables increases exponentially as the number of variables increases
- the number of possible configurations of x is much larger than the number of training examples.
- The core idea in deep learning is that we assume that the data was generated by the composition of factors or features, potentially at multiple levels in a hierarchy
- Many other similarly generic assumptions can further improve deep learning algorithms.
- The exponential advantages conferred by the use of deep, distributed representations counter the exponential challenges posed by the curse of dimensionality

Manifold Learning

- A manifold is a connected region.
- Mathematically, it is a set of points, associated with a neighborhood around each point.
- The definition of a neighborhood surrounding each point implies the existence of transformations that can be applied to move on the manifold from one position to a neighboring one.
- In ML it tends to be used more loosely to designate a connected set of points that can be approximated well by considering only a small number of degrees of freedom, or dimensions, embedded in a higher-dimensional space.
- Each dimension corresponds to a local direction of variation

Manifold Learning

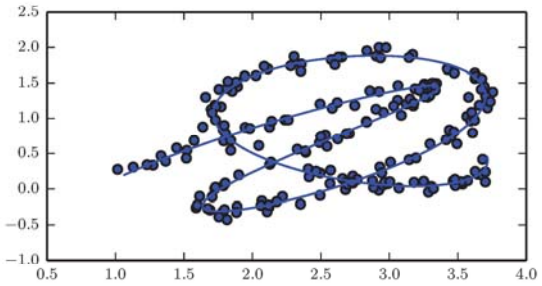


Figure 5.11

(Goodfellow 2016)

- Manifold learning algorithms assume that
  - most of R^n consists of invalid inputs
  - interesting inputs occur only along a collection of manifolds containing a small subset of points,
  - with interesting variations in the output of the learned function occurring only along directions that lie on the manifold,
  - or with interesting variations happening only when we move from one manifold to another.

Manifold hypothesis

- We argue that in the context of AI tasks, such as those that involve processing images, sounds, or text, the manifold assumption is at least approximately correct.
- Obs1: the probability distribution over images, text strings, and sounds that occur in real life is highly concentrated
- 2. we can also imagine such neighbourhoods and transformations informally.
  - images
  - Images: we can think of transformations that allow us to trace out a manifold in image space: we can gradually dim or brighten the lights, gradually move or rotate objects in the image, gradually alter the colors on the surfaces of objects, etc.
  - It remains likely that there are multiple manifolds involved in most applications. For example, the manifold of images of human faces may not be connected to the manifold of images of cat faces.

- When the data lies on a low-dimensional manifold, it can be most natural for machine learning algorithms to represent the data in terms of coordinates on the manifold, rather than in terms of coordinates in $\mathcal{R}^n$.

Optimization (Chapter 4) Sec 4.3

Gradient-Based Optimization

- Most deep learning algorithms involve optimization of some sort.
- Optimization refers to the task of minimizing or maximizing some function $f(x)$ by altering x
  - Called the objective function (cost function, loss function)
- Derivative of $f(x)$ denoted as $f'(x)$or $\frac{dy}{dx}$
  - Gives slope of $f(x)$ at x
- We canthus reduce $f(x)$ by moving x in small steps with opposite sign of the derivative. This technique is called gradient descent

- For functions with multiple inputs, we must make use of the concept of partial derivatives. The partial derivative $\frac{\partial}{\partial x_i} f(x)$ measures how f changes as only the variable x_i increases at point x. The gradient generalizes the notion of derivative to the case where the derivative is with respect to a vector: the gradient of f is thevector containing all of the partial derivatives, denoted $\nabla_x f(x)$
- In multiple dimensions, critical points are points where every element of the gradient is equal to zero