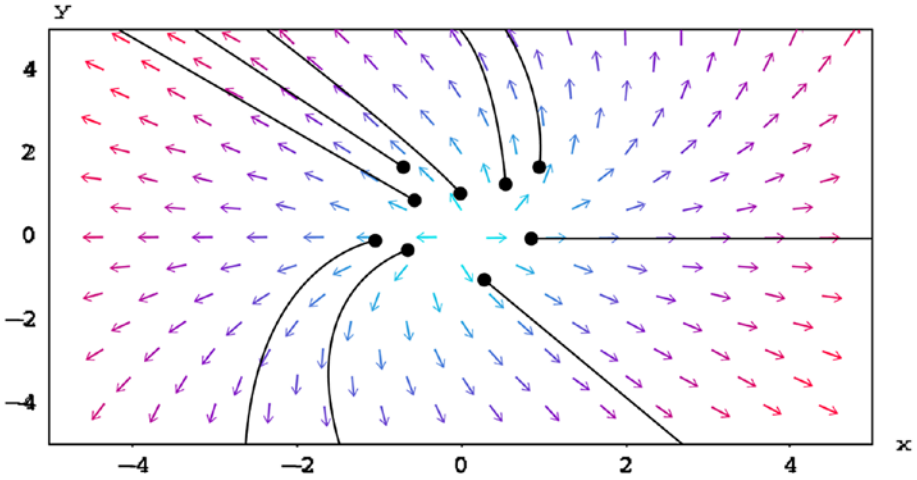

Figure 2-6. *Visulaization of eigenvectors*

Eigenvectors and eigenvalues become an integral part of understanding a technique discussed later in our discussion regarding a variable selection technique called principal components analysis (PCA). The eigendecomposition of a symmetric positive semi-definite matrix yields an orthogonal basis of eigenvectors, each of which has a non-negative eigenvalue. PCA studies linear relations among variables and is performed on the covariance matrix, or the correlation matrix, of the input data set. For the covariance or correlation matrix, the eigenvectors correspond to principal components and the eigenvalues to the variance explained by the principal components. Principal component analysis of the correlation matrix provides an orthonormal eigenbasis for the space of the observed data: in this basis, the largest eigenvalues correspond to the principal components that are associated with containing the most covariability of the observed dataset.

Linear Transformations

A linear transformation is a mapping $V \rightarrow W$ between two modules that preserves the operations of addition and scalar multiplication. When V = W, we call this a linear operator, or endomorphism, of V. Linear transformations always map linear subspaces onto linear subspaces, and sometimes this can be in a lower dimension. These linear maps can be represented as matrices, such as rotations and reflections. An example of where linear transformations are used is specifically PCA. Discussed in detail later, PCA is an orthogonal linear transformation of the features in a data set into uncorrelated principal components such that for K features, we have K principal components. I discuss orthogonality in detail in the following sections, but for now I focus on the broader aspects of PCA. Each principal component retains the variance from the original data set but gives us a representation of it such that we can infer the importance of a given principal component based on the contribution of the variance to the data set it provides. When translating this to the original data set, we then can remove features from the data set that we feel don't exhibit significant amounts of variance.

A function $\mathcal{L}:\mathbb{R}^n \to \mathbb{R}^m$ is called a linear transformation if the following is true:

$$\mathcal{L}(ax) = a\mathcal{L}(x) \text{ for every } x \in \mathbb{R}^n \text{ and } a \in \mathbb{R}$$

$$\mathcal{L}(x+y) = \mathcal{L}(x) + \mathcal{L}(y) \text{ for every } x, y, \in \mathbb{R}^n$$

When we fix the bases for $\mathbb{R}^n$ and $\mathbb{R}^m$, the linear transformation $\mathcal{L}$ can be represented by a matrix A. Specifically, there exists $A \in \mathbb{R}^{mxn}$ such that the following representation holds. Suppose $x \in \mathbb{R}^n$ is a given vector and x' is the representative of x with respect to the given basis for $\mathbb{R}^m$. If $y = \mathcal{L}(x)$ and Y' is the representative of y with respect to the given basis for $\mathbb{R}^m$, then

$$y' = Ax'$$

We call A the matrix representation of $\mathcal{L}$ with respect to the given bases for $\mathbb{R}^n$ and $\mathbb{R}^m$.

Quadratic Forms

A *quadratic form* is a homogenous polynomial of the second degree in a number of variables and have applications in machine learning. Specifically, functions we seek to optimize that are twice differentiable can be optimized using Newton's method. The power in this is that if a function is twice differentiable, we know that we can reach an objective minimum.

A quadratic form $f:\mathbb{R}^n \to \mathbb{R}^m$ is a function such that the following holds true:

$$F(x) = x^T Q x$$

Where Q is an n x n real matrix. There is no loss of generality in assuming Q to be symmetric—that is, $Q = Q^T$.

Minors of a matrix Q are the determinants of the matrices obtained by successively removing rows and columns from Q. The principal minors are detQ itself and the determinants of matrices obtained by removing an ith row and an ith column.

Sylvester's Criterion

Sylvester's criterion is necessary and sufficient to determine whether a matrix is positive semi-definite. Simply, it states that for a matrix to be positive semi-definite, all the leading principal minors must be positive.

Proof: if real-symmetric matrix A has non-negative eigenvalues that are positive, it is called positive-definite. When the eigenvalues are just non-negative, A is said to be positive semi-definite.

A real-symmetric matrix A has non-negative eigenvalues if and only if A can be factored as $A = B^T B$, and all eigenvalues are positive if and only if B is non-singular.

Forward implication: if $A \in R^{nxn}$ is symmetric, then there is an orthogonal matrix P such that $A = PDP^T$, where $\mathrm{D} = \mathrm{diag}(\lambda_1 \lambda_2, \ldots, \lambda_n)$ is a real diagonal matrix with entries such that its columns are the eigenvectors of A. If $\lambda_i \geq 0$ for each I, $D^{\frac{1}{2}}$ exists.

Reverse implication: if A can be factored as A = B^TB, then all eigenvalues of A are non-negative because for any eigenpair (x, λ)

$$\lambda = \left(\frac{x^T Ax}{x^{Tx}} \right) = \left(\frac{x^T B^T Bx}{x^T x} \right) = \left(\frac{\|Bx\|^2}{\|x\|^2} \right) \geq 0$$

Orthogonal Projections

A *projection* is linear transformation P from a vector space to itself such that $P^2 = P$. Intuitively, this means that whenever P is applied twice to any value, it gives the same result as it it were applied once. Its image is unchanged and this definition generalizes the idea of graphical projection moreover. $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ if $\mathrm{x}_1, \mathrm{x}_2 \in \mathcal{V} \rightarrow \alpha x_1 + \beta x_2 \in \mathcal{V}$ for all $\alpha, \beta \in \mathbb{R}$. The dimension of this subspace is also equal o the maximum number of linearly independent vectors in $\mathcal{V}$. If $\mathcal{V}$ is a subspace of R^n, the orthogonal complement of $\mathcal{V}$, demoted $\mathcal{V}^{\perp}$, consists of all vectors that are orthogonal to every vector in $\mathcal{V}$. Thus, the following is true:

$$\mathcal{V}^{\perp} = \left\{ x : v^T x = 0 \ \textit{for all} \ v \in \mathcal{V} \right\}$$

The orthogonal complement of $\mathcal{V}$ is also a subspace. Together, $\mathcal{V}$ and $\mathcal{V}^{\perp}$ span R^n in the sense that every vector $x \in \mathbb{R}^n$ can be represented as

$$x = x_1 + x_2$$

where $x_1 \in \mathcal{V}$ and $x_2 \in \mathcal{V}^{\perp}$. We call the above representation the *orthogonal decomposition* of *x* with respect to $\mathcal{V}$. We say that x_1 and x_2 are orthogonal projections of *x* onto the subspaces $\mathcal{V}$ and $\mathcal{V}^{\perp}$ respectively. We write $\mathbb{R}^n = \mathcal{V} \otimes \mathcal{V}^{\perp}$, and say that $\mathbb{R}^n$ is a direct sum of $\mathcal{V}$ and $\mathcal{V}^{\perp}$. We say that a linear transformation of P is an orthogonal projector onto $\mathcal{V}$ for all $x \in \mathbb{R}^n$, we have $Px \in \mathcal{V}$ and $x - Px \in \mathcal{V}^{\perp}$.

Range of a Matrix

The *range* of a matrix defines the number of column vectors it contains.

Let $A \in \mathbb{R}^{mxn}$. The range, or image, of A, is written as the following:

$$\mathcal{R}(A) \triangleq \left\{ Ax : x \in \mathbb{R}^n \right\}$$

Nullspace of a Matrix

The nullspace of a linear map $\mathcal{L}:\mathcal{V}\to\mathcal{W}$ between two vector spaces is the set of all elements of v of $\mathcal{V}$ for which $\mathcal{L}(v)=0$, where zero denotes the zero vector in $\mathcal{W}$.

The nullspace, or kernel, of A is written as the following:

$$\mathcal{N}(A)\triangleq\left\{x\in\mathbb{R}^n : Ax=0\right\}$$

Hyperplanes

Earlier I mentioned the significance of the support vector machine and the hyperplane. In the context of regression problems, the observations within the hyperplane are acceptable as response variable solutions. In the context of classification problems, the hyperplanes form the boundaries between different classes of observations (shown in Figure 2-7).

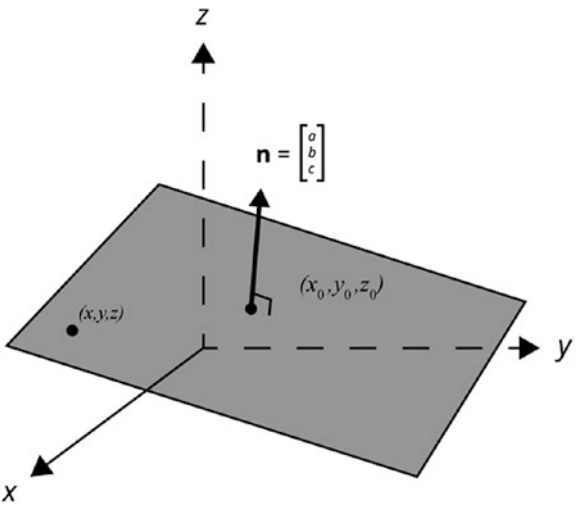

Figure 2-7. *Visualization of hyperplane*

We define a *hyperplane* as a subspace of one dimension less than its ambient space, otherwise known as the feature space surrounding the object.

Let $u=[u_1,u_2,\ldots,u_n]$, $u\in\mathbb{R}$, where at least one of the u_i is non-zero. The set of all points $x=[x_1,x_2,\ldots,x_n]^T$ that satisfy the linear equation

$$u_1x_1+u_2x_2+\ldots+u_nx_n=v$$

is called a hyperplane of the space $\mathbb{R}^n$. We may describe the hyperplane with the following equation:

$$\left\{x \in \mathbb{R}^n : u^T x = v\right\}$$

A hyperplane is not necessarily a subspace of $\mathbb{R}^n$ because, in general, it does not contain the origin. For $n = 2$, the equation of the hyperplane has the form $u_1 x_1 + u_2 x_2 = v$, which is the equation of a straight line. Thus, straight lines are hyperplanes in $\mathbb{R}^2$. In $\mathbb{R}^3$, hyperplanes are ordinary planes. The hyperplane H divides $\mathbb{R}^n$ into two half spaces, denoted by the following:

$$H_+ = \left\{x \in \mathbb{R}^n : u^T x \geq 0\right\},$$

$$H_- = \left\{x \in \mathbb{R}^n : u^T x \leq 0\right\}.$$

Here H_+ is the positive half-space, and H_- is the negative half-space. The hyperplane H itself consists of the points for which $\langle u, x - a\rangle = 0$, where $a = \left[a_1, a_2, \ldots, a_n\right]^T$ is an arbitrary point of the hyperplane. Simply stated, the hyperplane H is all of the points x for which the vectors u and x – a are orthogonal to one another.

Sequences

A *sequence* of real numbers is a function whose domain is the set of natural numbers 1,2,...,k, and whose range is contained in $\mathbb{R}$. Thus, a sequence of real numbers can be viewed as a set of numbers $\{x_1, x_2, \ldots, x_k\}$, which is often also denoted as $\{x_k\}$.

Properties of Sequences

The *length* of a sequence is defined as the number of elements within it. A sequence of finite length n is also called an n-tuple. *Finite* sequences also include sequences that are empty or ones that have no elements. An *infinite* sequence refers to a sequence that is infinite in one direction. It is therefore described as having a first element, but not having a final element. A sequence with neither a first nor a final element is known as a *two-way infinite* sequence or *bi-infinite* sequence.

Moreover, a sequence is said to be monotonically increasing if each term is greater than or equal to the one before it. For example, the sequence $an(n) = 1$ is monotonically increasing if an only if for all $a_{n+1} \geq a_n$. The terms *non-decreasing* and *non-increasing* are often used in place of increasing and decreasing in order to avoid any possible confusion with strictly increasing and strictly decreasing respectively.

If the sequence of real number is such that all the terms are less than some real numbers, then the sequence is said to be bounded from above. This means that there exists M such that for all n, $a_n \leq M$. Any such M is called an upper bound. Likewise, if, for some real m, $a_n \geq m$ for all n greater than some N, then the sequence is bounded from below, and any such m is called the lower bound.

Limits

A *limit* is the value that a function or sequence approaches as the input or index approaches some value. A number $x^* \in \mathbb{R}$ is called the limit of the sequence if for any positive ϵ there is a number K such that for all $k > K$, $\left|xk - x^*\right| < \epsilon$:

$$x^* = \lim_{k \to \infty} xk$$

A sequence that has a limit is called a *convergent* sequence. Informally speaking, a singly infinite sequence has a limit, if it approaches some value L, called the limit, as n becomes very large. If it converges towards some limit, then it is *convergent*. Otherwise it is *divergent*. Figure 2-8 shows a sequence converging upon a limit.

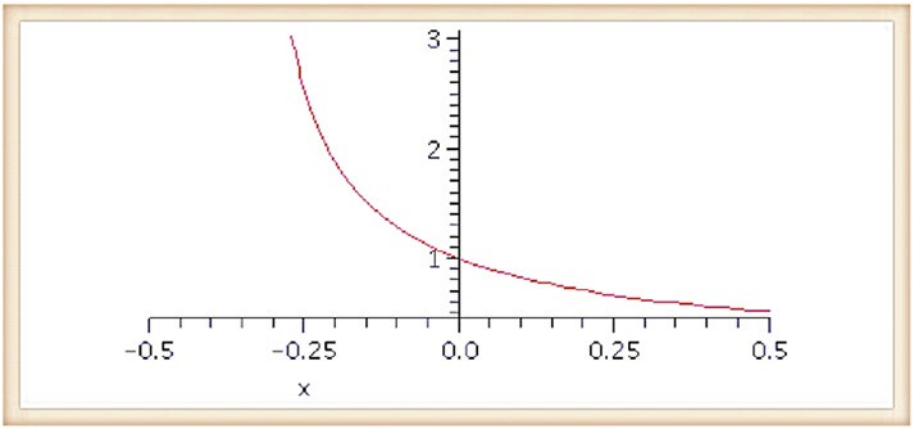

Figure 2-8. *A function converging upon 0 as x increases*

We typically speak of convergence within the context of machine learning and deep learning with reaching an *optimal solution*. This is ultimately the goal of all of our algorithms, but this becomes more ambiguous with the more difficult use cases readers encounter. Not every solution has a single global optimum—instead it could have local optima. Methods of avoiding these local optima are more specifically addressed in later chapters. Typically this requires parameter tuning of machine learning and deep learning algorithms, the most difficult part of the algorithm training process.

Derivatives and Differentiability

Differentiability becomes an important part of machine learning and deep learning, most specifically for the purpose of parameter updating. This can be seen via the back-propagation algorithm used to train multilayer perceptrons and the parameter updating of convolutional neural networks and recurrent neural networks. A derivative of a function measures the degree of change in one quantity to the degree of another. One of the most common examples of a derivative is a slope (change in y over x), or the return of a stock (price percentage change over time). This is a fundamental tool for calculus but is also the basis of many of the models we will study in the latter part of the book.

A function is considered to be *affine* if there exists a linear function $\mathcal{L}:\mathbb{R}^n \to \mathbb{R}^m$ and a vector $y \in \mathbb{R}^m$ such that

$$A(x) = \mathcal{L}(x) + y$$

for every $x \in \mathbb{R}^n$. Consider a function $f:\mathbb{R}^n \to \mathbb{R}^m$ and a point $x_0 \in \mathbb{R}^n$. We want to find an affine function A that approximates f near the point x0. First, it's natural to impose this condition:

$$A(x_0) = f(x_0)$$

We obtain $y = f(x_0) - \mathcal{L}(x_0)$. By the linearity of L,

$$\mathcal{L} + y = \mathcal{L}(x) - \mathcal{L}(x_0) + f(x_0) = \mathcal{L}(x - x_0) + f(x_0)$$

$$A(x) = \mathcal{L}(x - x_0) + f(x_0)$$

We also require that $A(x)$ approaches $f(x)$ faster than x approaches x_0.

Partial Derivatives and Gradients

Also utilized heavily in various machine learning derivations is the *partial derivative*. It is similar to a derivative, except we only take the derivative of one of the variables in the function and hold the others constant, whereas in a total derivative all the variables are evaluated. The gradient descent algorithm is discussed in Chapter 3, but we can discuss the broader concept of the gradient itself now. A *gradient* is the generalization of the concept of a derivative when applied to functions of several variables. The gradient represents the point of greatest rate of increase in the function, and its magnitude is the slope of the graph in that direction. It's a vector field whose components in a coordinate system will transform when going from one system to another:

$$\nabla f(x) = grad\ f(x) = \frac{df(x)}{dx}$$

Hessian Matrix

Functions can be differentiable more than once, which leads us to the concept of the Hessian matrix. The *Hessian* is a square matrix of second-order partial derivatives of a scalar values function, or scalar field:

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

If the gradient of a function is zero at some point x, then f has a critical point at x. The determinant of the Hessian at x is then called the *discriminant*. If this determinant is zero, then x is called a degenerate critical point of *f*, or a non-Morse critical point of *f*. Otherwise, it is non-degenerate.

A Jacobian matrix is the matrix of first-order partial derivatives of a vector values function. When this is a square matrix, both the matrix and its determinant are referred to as the *Jacobian*:

$$\mathbf{J} = \frac{\mathrm{df}}{\mathrm{dx}} = \left[\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial f 1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Summary

This brings us to the conclusion of the basic statistics and mathematical concepts that will be referenced in later chapters. Readers should feel encouraged to check back with this chapter when unsure about anything in later chapters. Moving forward, we'll address the more advanced optimization techniques that power machine learning algorithms, as well as those same machine learning algorithms that formed the inspiration of the deep learning methods we'll tackle afterwards.

CHAPTER 3

A Review of Optimization and Machine Learning

Before we dive into the models and components of deep learning in depth, it's important to address the broader field it fits into, which is machine learning. But before that, I want to discuss, if only briefly, optimization. *Optimization* refers to the selection of a best element from some set of available alternatives. The objective of most machine learning algorithms is to find the optimal solution given a function with some set of inputs. As already mentioned, this often comes within the concept of a supervised learning problem or an unsupervised learning problem, though the procedures are roughly the same.

Unconstrained Optimization

Unconstrained optimization refers to a problem in which we much reach an optimal solution. In contrast to constrain optimization, there are constraints placed on what value of x we choose, allowing us to approach the solution from significantly more avenues. An example of an unconstrained optimization problem is the following toy problem:

$$\text{Minimize } f(x), \text{ where } f(x) = x^2, x \in [-100,100]$$

Figure 3-1 visualizes this function.

T. Beysolow II, *Introduction to Deep Learning Using R*, DOI 10.1007/978-1-4842-2734-3_3

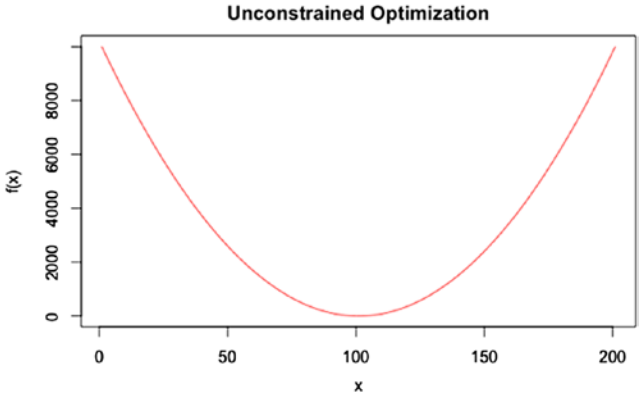

Figure 3-1. *Visualization of f(x)*

In this problem, because there are no constraints, we are allowed to choose whatever number for x is within the bounds defined. Given the equation we seek to minimize, the answer for x is 100. As we can see, we minimize the value of f(x) globally when we choose x. Therefore, we state that x = 100 = x*, which is a global minimizer of f(x). In contrast, here's a constrained optimization problem:

$$\text{Minimize } \mathrm{f}(\mathrm{x}),\ \text{s.t.}(\text{subject to})\,\mathrm{x}\in\Omega$$

$$\text{where } \mathrm{f}(\mathrm{x})=\mathrm{x}^2 \text{ Subject to } \mathrm{x}\in\Omega$$

The function $f:\mathbb{R}^n\to\mathbb{R}$ that we want to minimize is a real-valued function and is called the objective/cost function. The vector x is a vector of length *n* consisting of independent variables where $\mathrm{x}=\left[\mathrm{x}_1,x_2,\ldots,x_n\right]^T\in\mathbb{R}^n$. The variables within this vector commonly are referred to as *decision variables*. The set Ω is a subset of R called the constraint/feasible set. We say that the preceding optimization problem is a decision problem in which we must find the best vector of x that satisfies the objective subject to the constraint. Here, the best vector of x would result in a minimization of the objective function. In this function, because we have a constraint placed, we call this a constrained optimization problem. $\mathrm{x}\in\Omega$ is known as the set constraint. Often, this takes the form of

$$\Omega=\left\{\mathrm{x}:h(x)=0,g(x)\le 0\right\}$$

where *h* and *g* are some given functions. *h* and *g* are referred to as the *functional constraints*.

Imagine that we are still viewing the same function displayed in Figure 3-1, except that our feasible set is *Ω*. For simplicity's sake, let's say that *h*(*x*) and *g*(*x*) are equal to the following:

$$h(x)=g(x)=10-x$$