Colección Completa de Análisis de Datos

Norah Jones

2025-11-21

Tabla de contenidos

Preface **6**

1 Introduction **7**

I Métodos de Optimización **8**

2 Introducción a la Optimización para Ciencia de Datos **9**
2.1 Introducción . . . 9
2.2 Análisis de Datos y Optimización . . . 10
2.2.1 El Marco de Ciencia de Datos . . . 10
2.2.2 El Objetivo de Aprendizaje . . . 11
2.2.3 Parametrización y Formulación de Optimización . . . 11
2.2.4 Formulación de Suma Finita . . . 11
2.2.5 Predicción y Propiedades del Modelo . . . 12
2.2.6 Tipos de Problemas de Análisis de Datos . . . 12
2.2.7 Complicaciones de Datos y Robustez . . . 13
2.2.8 Regularization and Overfitting . . . 13
2.2.9 Resumen del Marco . . . 15
2.3 Mínimos Cuadrados . . . 15
2.3.1 Motivación Estadística . . . 15
2.3.2 Regresión Ridge . . . 16
2.3.3 Formulación LASSO . . . 16
2.4 Problemas de Factorización de Matrices . . . 17
2.4.1 Problema Básico de Sensado de Matrices . . . 17
2.4.2 Regularización de Norma Nuclear . . . 17
2.4.3 Representación Factorizada . . . 18
2.4.4 Factorización de Matrices No Negativas . . . 19
2.5 Máquinas de Vectores de Soporte . . . 19
2.5.1 Formulación de Pérdida Bisagra . . . 20
2.5.2 Métodos de Kernel . . . 21
2.5.3 Formulación Dual . . . 21
2.6 Regresión Logística . . . 22
2.6.1 Logaritmo Negativo de Verosimilitud . . . 23
2.6.2 Selección de Características con LASSO . . . 23

2.6.3 Regresión Logística Multiclase 23
2.7 Aprendizaje Profundo 24
2.7.1 Estructura de la Red Neuronal 25
2.7.2 Transformaciones de Capa 25
2.7.3 Capa de Salida y Función de Pérdida 26
2.7.4 Problema de Optimización en Aprendizaje Profundo 26
2.7.5 Variantes e Ingeniería 27
2.7.6 Características Distintivas del Aprendizaje Profundo 27
2.8 Énfasis 27
2.8.1 Énfasis del Tratamiento 28
2.8.2 Preocupaciones Prácticas 28

II Resumen y Referencias **29**

3 Summary **30**

References **31**

Listado de Figuras

Listado de Tablas

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

Parte I

Métodos de Optimización

2 Introducción a la Optimización para Ciencia de Datos

Fundamentos de Optimización Continua

Resumen del Capítulo

Este capítulo introduce los fundamentos de los **algoritmos de optimización continua** para aplicaciones de ciencia de datos. Exploramos cómo los problemas de aprendizaje automático, estadística y análisis de datos pueden formularse como desafíos de optimización.
Temas Principales:

- Análisis de datos a través de la lente de optimización
- Problemas clásicos de optimización (Mínimos Cuadrados, LASSO)
- Factorización de matrices y problemas de bajo rango
- Formulaciones de aprendizaje automático (SVM, Regresión Logística)
- Desafíos de optimización en aprendizaje profundo

2.1. Introducción

Este libro se enfoca en los **fundamentos de algoritmos** para resolver problemas de optimización continua, que involucran:

- **Minimizar funciones** de múltiples variables de valores reales
- **Manejar restricciones** en los valores de las variables
- **Enfatizar problemas convexos** con aplicaciones en ciencia de datos
- **Conectar teoría y práctica** en aprendizaje automático, estadística y análisis de datos

! Enfoque Principal

Nuestra selección de temas está **motivada por la relevancia para la ciencia de datos** — las formulaciones y algoritmos discutidos son directamente útiles para resolver

problemas del mundo real en aprendizaje automático, estadística y análisis de datos.

Este capítulo describe varios **paradigmas clave de la ciencia de datos** y demuestra cómo pueden formularse como problemas de optimización continua. Comprender las **propiedades de suavidad y estructura** de estas formulaciones es crucial para seleccionar algoritmos apropiados.

2.2. Análisis de Datos y Optimización

El problema de optimización típico en análisis de datos implica encontrar un **modelo que equilibre** dos objetivos competitivos:

1. **Concordancia con los datos recopilados**
2. **Adhesión a restricciones estructurales** que reflejan nuestras creencias sobre buenos modelos

2.2.1. El Marco de Ciencia de Datos

En un problema de análisis típico, nuestro **conjunto de datos** consiste en m objetos:

$$D := \{(a_j, y_j), j = 1, 2, \ldots, m\} \tag{2.1}$$

donde:

Características (a_j)

- Vector o matriz de características
- Variables de entrada
- Predictores

Etiquetas/Observaciones (y_j)

- Valores objetivo
- Respuestas
- Resultados

Preprocesamiento de Datos

Asumimos que los datos han sido **limpiados** de modo que todos los pares (a_j, y_j) tienen tamaño y forma consistentes.

2.2.2. El Objetivo de Aprendizaje

La **tarea de análisis de datos** consiste en descubrir una función ϕ tal que:

$$\phi(a_j) \approx y_j \quad \text{para la mayoría } j = 1, 2, \ldots, m$$

> **Terminología**
>
> El proceso de descubrir el mapeo ϕ se suele llamar **"aprendizaje"** o **"entrenamiento"**.

2.2.3. Parametrización y Formulación de Optimización

La función ϕ a menudo se define en términos de un **vector o matriz de parámetros**, que denotamos por x o X. Con estas parametrizaciones, el problema de identificar ϕ se convierte en un **problema tradicional de ajuste de datos**:

> **Objetivo de Optimización**
>
> Encontrar los parámetros x que definen ϕ tal que $\phi(a_j) \approx y_j$ para $j = 1, 2, \ldots, m$ en algún sentido óptimo.

Una vez que definimos "óptimo" (y posiblemente agregamos restricciones en los valores permitidos de los parámetros), tenemos un problema de optimización.

2.2.4. Formulación de Suma Finita

Frecuentemente, estas formulaciones de optimización tienen funciones objetivo del **tipo suma finita**:

$$L_D(x) := \frac{1}{m} \sum_{j=1}^{m} \ell(a_j, y_j; x) \tag{2.2}$$

donde:

- $\ell(a, y; x)$ representa la **"pérdida"** incurrida por no alinear correctamente nuestra predicción $\phi(a)$ con y
- $L_D(x)$ mide la **pérdida promedio** acumulada sobre todo el conjunto de datos cuando el vector de parámetros es igual a x

2.2.5. Predicción y Propiedades del Modelo

Una vez que se ha aprendido un valor apropiado de x (y por lo tanto ϕ) de los datos, podemos usarlo para hacer predicciones sobre otros elementos de datos que no están en el conjunto D (Ecuación 2.1).

Dado un elemento de datos no visto $\hat{a}$ del mismo tipo que a_j, $j = 1, 2, \ldots, m$, predecimos que la etiqueta $\hat{y}$ asociada con $\hat{a}$ será $\phi(\hat{a})$.

El mapeo ϕ también puede exponer otras estructuras y propiedades en el conjunto de datos:

- **Selección de Características**: Revela que solo una pequeña fracción de las características en a_j son necesarias para predecir confiablemente la etiqueta y_j
- **Descubrimiento de Subespacios**: Cuando el parámetro x es una matriz, podría revelar un subespacio de baja dimensión que contiene la mayoría de los vectores a_j
- **Matrices Estructuradas**: Podría revelar una matriz con estructura particular (bajo rango, dispersa) tal que las observaciones de X motivadas por los vectores de características a_j produzcan resultados cercanos a y_j

2.2.6. Tipos de Problemas de Análisis de Datos

La forma de las etiquetas y_j difiere según la naturaleza del problema de análisis de datos:

2.2.6.1. Regresión

Si cada y_j es un **número real**, típicamente tenemos un **problema de regresión**.

2.2.6.2. Clasificación

Cuando cada y_j es una **etiqueta** (un entero del conjunto $\{1, 2, \ldots, M\}$) indicando que a_j pertenece a una de M clases:

- **Clasificación Binaria**: $M = 2$
- **Clasificación Multiclase**: $M > 2$

> **Nota**
>
> En problemas de análisis de datos que surgen en reconocimiento de voz e imagen, M puede ser muy grande, del orden de miles o más.

2.2.6.3. Aprendizaje No Supervisado

Las etiquetas y_j pueden ni siquiera existir; el conjunto de datos puede contener solo los vectores de características a_j, $j = 1, 2, \ldots, m$. Todavía hay problemas interesantes de análisis de datos asociados con estos casos. Por ejemplo, podemos desear agrupar los a_j en **clústeres** (donde los vectores dentro de cada clúster se consideran funcionalmente similares) o identificar un **subespacio de baja dimensión** (o una colección de subespacios de baja dimensión) que aproximadamente contenga los a_j.

En tales problemas, esencialmente estamos aprendiendo las etiquetas y_j junto con la función ϕ. Por ejemplo, en un problema de clustering, y_j podría representar el clúster al cual se asigna a_j.

2.2.7. Complicaciones de Datos y Robustez

Incluso después de la limpieza y preparación, la configuración anterior puede contener muchas complicaciones:

- **Ruido y Corrupción**: Las cantidades (a_j, y_j) pueden contener ruido o estar corrompidas de otra manera, requiriendo que el mapeo ϕ sea robusto a tales errores
- **Datos Faltantes**: Partes de los vectores a_j pueden estar faltando, o podemos no conocer todas las etiquetas y_j
- **Datos en Flujo**: Los datos pueden estar llegando de forma continua en lugar de estar disponibles todos a la vez, requiriendo **aprendizaje en línea** de ϕ

2.2.8. Regularization and Overfitting

One consideration that arises frequently is that we wish to avoid **overfitting** the model to the data set D in (Ecuación 2.1).

> **! Generalization Goal**
>
> The particular data set D available to us can often be thought of as a finite sample drawn from some underlying larger (perhaps infinite) collection of possible data points. We wish the function ϕ to perform well on the **unobserved data points** as well as the observed subset D.

In other words, we want ϕ to be not too sensitive to the particular sample D that is used to define empirical objective functions such as (Ecuación 2.2).

One way to avoid this issue is to modify the objective function by adding constraints or penalty terms, in a way that limits the "complexity" of the function ϕ. This process is typically called **regularization**.

An optimization formulation that balances fit to the training data D, model complexity, and model structure is:

$$\min_{x \in \Omega} L_D(x) + \lambda \text{pen}(x) \tag{2.3}$$

where:

- Ω is a set of allowable values for x
- $\text{pen}(\cdot)$ is a regularization function or **regularizer**
- $\lambda \geq 0$ is a **regularization parameter**

The regularizer usually takes lower values for parameters x that yield functions ϕ with lower complexity. For example, ϕ may depend on fewer of the features in the data vectors a_j or may be less oscillatory.

2.2.8.1. Ajuste del Parámetro de Regularización

El parámetro λ puede ser "ajustado" para proporcionar un equilibrio apropiado:

- **Valores más pequeños de** λ: Producen soluciones que se ajustan a los datos de entrenamiento D con mayor precisión
- **Valores más grandes de** λ: Conducen a modelos menos complejos

> **Perspectiva Moderna sobre el Sobreajuste**
>
> Curiosamente, el concepto de sobreajuste ha sido reexaminado en años recientes, particularmente en el contexto del aprendizaje profundo, donde los modelos que se ajustan perfectamente a los datos de entrenamiento a veces se observa que también hacen un buen trabajo al clasificar datos previamente no vistos. Este fenómeno es un tema de intensa investigación actual en la comunidad de aprendizaje automático.

2.2.8.2. Conjuntos de Restricciones

El conjunto de restricciones Ω en (Ecuación 2.3) puede elegirse para excluir valores de x que no son relevantes o útiles en el contexto del problema de análisis de datos. Por ejemplo:

- En algunas aplicaciones, podemos no desear considerar valores de x en los que uno o más componentes sean negativos
- Podríamos establecer Ω como el conjunto de vectores cuyos componentes son todos mayores o iguales a cero

2.2.9. Resumen del Marco

Ahora examinamos algunos problemas particulares en ciencia de datos que dan lugar a formulaciones que son casos especiales de nuestro problema maestro (Ecuación 2.3). Veremos que:

- Una gran variedad de problemas puede formularse usando este marco general
- Dentro de este marco, hay una amplia gama de estructuras que deben tenerse en cuenta al elegir algoritmos para resolver estos problemas de manera eficiente

2.3. Mínimos Cuadrados

Probablemente el **problema de análisis de datos más antiguo y conocido** es el de mínimos cuadrados lineales. Aquí, los puntos de datos (a_j, y_j) se encuentran en $\mathbb{R}^n \times \mathbb{R}$, y resolvemos:

$$\min_x \frac{1}{2m} \sum_{j=1}^{m} (a_j^T x - y_j)^2 = \frac{1}{2m} \|Ax - y\|_2^2 \tag{2.4}$$

donde:

- A es la matriz cuyas filas son a_j^T, $j = 1, 2, \ldots, m$
- $y = (y_1, y_2, \ldots, y_m)^T$

En la terminología anterior, la función ϕ se define por $\phi(a) := a^T x$.

Agregar un Intercepto

Podemos introducir un intercepto no nulo agregando un parámetro adicional $\beta \in \mathbb{R}$ y definiendo $\phi(a) := a^T x + \beta$.

2.3.1. Motivación Estadística

Esta formulación puede motivarse estadísticamente, como una **estimación de máxima verosimilitud** de x cuando las observaciones y_j son exactas excepto por ruido Gaussiano independiente e idénticamente distribuido (i.i.d.).

2.3.2. Regresión Ridge

Podemos agregar una variedad de funciones de penalización a este problema básico de mínimos cuadrados para imponer estructura deseable en x y, por lo tanto, en ϕ. Por ejemplo, la **regresión ridge** agrega una penalización de norma ℓ_2 al cuadrado:

$$\min_x \frac{1}{2m}\|Ax - y\|_2^2 + \lambda\|x\|_2^2 \tag{2.5}$$

para algún parámetro $\lambda > 0$. La solución x de esta formulación regularizada tiene menos sensibilidad a perturbaciones en los datos (a_j, y_j).

2.3.3. Formulación LASSO

La formulación **LASSO** (Least Absolute Shrinkage and Selection Operator):

$$\min_x \frac{1}{2m}\|Ax - y\|_2^2 + \lambda\|x\|_1 \tag{2.6}$$

tiende a producir soluciones x que son **dispersas** – es decir, que contienen relativamente pocos componentes no nulos.

! Selección de Características

Esta formulación realiza **selección de características**: Las ubicaciones de los componentes no nulos en x revelan aquellos componentes de a_j que son instrumentales para determinar la observación y_j.

2.3.3.1. Ventajas de la Selección de Características

1. **Atractivo Estadístico**: Los predictores que dependen de pocas características son potencialmente más simples y más comprensibles que aquellos que dependen de muchas características
2. **Beneficios Prácticos**: En lugar de recopilar todos los componentes de un nuevo vector de datos $\hat{a}$, solo necesitamos encontrar las características "seleccionadas" porque solo estas son necesarias para hacer una predicción

i LASSO como Prototipo

La formulación LASSO (Ecuación 2.6) es un prototipo importante para muchos problemas en análisis de datos en el sentido de que involucra un término de regularización $\lambda\|x\|_1$

que es no suave y convexo pero tiene una estructura relativamente simple que puede ser potencialmente explotada por algoritmos.

2.4. Problemas de Factorización de Matrices

Hay una variedad de problemas de análisis de datos que requieren estimar una **matriz de bajo rango** a partir de alguna colección dispersa de datos. Tales problemas pueden formularse como una extensión natural de mínimos cuadrados a problemas en los que los datos a_j se representan naturalmente como matrices en lugar de vectores.

2.4.1. Problema Básico de Sensado de Matrices

Cambiando ligeramente la notación, suponemos que cada A_j es una matriz $n \times p$, y buscamos otra matriz $n \times p$ X que resuelva:

$$\min_X \frac{1}{2m} \sum_{j=1}^{m} (\langle A_j, X \rangle - y_j)^2 \tag{2.7}$$

donde $\langle A, B \rangle := \text{trace}(A^T B)$.

Aquí podemos pensar en las A_j como "sondeo" de la matriz desconocida X. Los tipos de observaciones comúnmente considerados son:

- **Combinaciones lineales aleatorias**: Los elementos de A_j se seleccionan i.i.d. de alguna distribución
- **Observaciones de elementos individuales**: Cada A_j tiene 1 en una única ubicación y ceros en otros lugares

2.4.2. Regularización de Norma Nuclear

Una versión regularizada de (Ecuación 2.7), que conduce a soluciones X de bajo rango, es:

$$\min_X \frac{1}{2m} \sum_{j=1}^{m} (\langle A_j, X \rangle - y_j)^2 + \lambda \|X\|_* \tag{2.8}$$

donde $\|X\|_*$ es la **norma nuclear**, que es la suma de valores singulares de X.

Propiedades de la Norma Nuclear

La norma nuclear juega un papel análogo a la norma ℓ_1 en (Ecuación 2.6):

- La norma ℓ_1 favorece vectores dispersos
- La norma nuclear favorece matrices de bajo rango

Aunque la norma nuclear es una función no suave algo compleja, es al menos convexa, por lo que la formulación (Ecuación 2.8) también es convexa.

Esta formulación puede demostrarse que produce una solución estadísticamente válida cuando:

- La verdadera X es de bajo rango
- Las matrices de observación A_j satisfacen una "propiedad de isometría restringida" (comúnmente satisfecha por matrices aleatorias pero no por matrices con solo un elemento no nulo)

La formulación también es válida en un contexto diferente, en el que:

- La verdadera X es incoherente (hablando aproximadamente, no tiene unos pocos elementos que sean mucho más grandes que los otros)
- Las observaciones A_j son de elementos individuales

2.4.3. Representación Factorizada

En otra forma de regularización, la matriz X se representa explícitamente como un producto de dos matrices "delgadas" L y R, donde $L \in \mathbb{R}^{n\times r}$ y $R \in \mathbb{R}^{p\times r}$, con $r \ll \text{mín}(n,p)$. Establecemos $X = LR^T$ en (Ecuación 2.7) y resolvemos:

$$\min_{L,R} \frac{1}{2m} \sum_{j=1}^{m} (\langle A_j, LR^T \rangle - y_j)^2 \tag{2.9}$$

En esta formulación:

- El rango r está "cableado" en la definición de X, por lo que no hay necesidad de incluir un término de regularización
- Esta formulación es típicamente mucho más compacta que (Ecuación 2.8); el número total de elementos en (L,R) es $(n+p)r$, que es mucho menor que np
- Sin embargo, esta función es **no convexa** cuando se considera como una función de (L,R) conjuntamente

No Convexidad Benigna

Una línea activa de investigación actual muestra que la no convexidad es benigna en muchas situaciones y que, bajo ciertas suposiciones sobre los datos (A_j, y_j), $j = 1, 2, \ldots, m$ y una cuidadosa elección de estrategia algorítmica, se pueden obtener buenas soluciones de la formulación (Ecuación 2.9).

Una pista sobre este buen comportamiento es que aunque esta formulación es no convexa, en cierto sentido es una aproximación a un problema tratable: Si tenemos una observación completa de X, entonces una aproximación de rango-r puede encontrarse realizando una descomposición de valores singulares de X y definiendo L y R en términos de los r principales vectores singulares izquierdos y derechos.

2.4.4. Factorización de Matrices No Negativas

Algunas aplicaciones en visión por computadora, quimiometría y agrupamiento de documentos requieren que encontremos factores L y R como aquellos en (Ecuación 2.9) en los que todos los elementos son no negativos. Si se observa la matriz completa $Y \in \mathbb{R}^{n \times p}$, este problema tiene la forma:

$$\min_{L,R} \|LR^T - Y\|_F^2 \quad \text{sujeto a } L \geq 0,\ R \geq 0 \tag{2.10}$$

y se llama **factorización de matrices no negativas**.

2.5. Máquinas de Vectores de Soporte

Problema Clásico de ML

La clasificación mediante Máquinas de Vectores de Soporte (SVM) es un problema de optimización clásico en aprendizaje automático, que remonta sus orígenes a la **década de 1960**.

Dados los datos de entrada (a_j, y_j) con $a_j \in \mathbb{R}^n$ e $y_j \in \{-1, 1\}$, SVM busca un vector $x \in \mathbb{R}^n$ y un escalar $\beta \in \mathbb{R}$ tales que:

$$a_j^T x - \beta \geq 1 \quad \text{cuando } y_j = +1 \tag{2.11}$$

$$a_j^T x - \beta \leq -1 \quad \text{cuando } y_j = -1 \tag{2.12}$$

Cualquier par (x, β) que satisfaga estas condiciones define un **hiperplano separador** en $\mathbb{R}^n$, que separa los casos "positivos" $\{a_j \mid y_j = +1\}$ de los casos "negativos" $\{a_j \mid y_j = -1\}$.

Entre todos los hiperplanos separadores, el que minimiza $\|x\|_2$ es el que **maximiza el margen** entre las dos clases – es decir, el hiperplano cuya distancia al punto a_j más cercano de cualquier clase es mayor.

2.5.1. Formulación de Pérdida Bisagra

Podemos formular el problema de encontrar un hiperplano separador como un problema de optimización definiendo un objetivo con la forma de suma (Ecuación 2.2):

$$H(x, \beta) = \frac{1}{m} \sum_{j=1}^{m} \text{máx}(1 - y_j(a_j^T x - \beta), 0) \tag{2.13}$$

Nótese que el término j-ésimo en esta suma es cero si se satisfacen las condiciones (Ecuación 2.11)–(Ecuación 2.12), y es positivo en caso contrario. Incluso si no existe ningún par (x, β) para el cual $H(x, \beta) = 0$, un valor (x, β) que minimice (Ecuación 2.13) será el que más se acerque a satisfacer las condiciones en algún sentido.

A menudo se agrega un término $\frac{1}{2\lambda}\|x\|_2^2$ (para algún parámetro $\lambda > 0$) a (Ecuación 2.13), produciendo la siguiente versión regularizada:

$$H(x, \beta) = \frac{1}{m} \sum_{j=1}^{m} \text{máx}(1 - y_j(a_j^T x - \beta), 0) + \frac{1}{2\lambda}\|x\|_2^2 \tag{2.14}$$

> **Pérdida vs Regularizador**
>
> En contraste con los ejemplos presentados hasta ahora, el problema SVM tiene una **función de pérdida no suave** y un **regularizador suave**.

Si λ es suficientemente pequeño, y si existen hiperplanos separadores, el par (x, β) que minimiza (Ecuación 2.14) es el hiperplano separador de margen máximo. La propiedad de margen máximo es consistente con los objetivos de generalizabilidad y robustez.

Por ejemplo, si los datos observados (a_j, y_j) se extraen de una "nube" subyacente de casos positivos y negativos, la solución de margen máximo generalmente hace un trabajo razonable al separar otras muestras de datos empíricos extraídas de las mismas nubes, mientras que un hiperplano que pasa cerca de varios de los puntos de datos observados puede no funcionar tan bien.

2.5.2. Métodos de Kernel

A menudo, no es posible encontrar un hiperplano que separe los casos positivos y negativos lo suficientemente bien como para ser útil como clasificador. Una solución es transformar todos los vectores de datos brutos a_j mediante algún mapeo no lineal ψ y luego realizar la clasificación de máquina de vectores de soporte en los vectores $\psi(a_j)$, $j = 1, 2, \dots, m$. Las condiciones (Ecuación 2.11)–(Ecuación 2.12) serían así reemplazadas por:

$$\psi(a_j)^T x - \beta \geq 1 \quad \text{cuando } y_j = +1 \tag{2.15}$$

$$\psi(a_j)^T x - \beta \leq -1 \quad \text{cuando } y_j = -1 \tag{2.16}$$

conduciendo al siguiente análogo de (Ecuación 2.14):

$$H(x, \beta) = \frac{1}{m} \sum_{j=1}^{m} \text{máx}(1 - y_j(\psi(a_j)^T x - \beta), 0) + \frac{1}{2\lambda}\|x\|_2^2 \tag{2.17}$$

Cuando se transforma de vuelta a $\mathbb{R}^m$, la superficie $\{a \mid \psi(a)^T x - \beta = 0\}$ es no lineal y posiblemente desconectada, y a menudo es un clasificador mucho más poderoso que los hiperplanos resultantes de (Ecuación 2.14).

2.5.3. Formulación Dual

Notamos que SVM también puede expresarse naturalmente como un problema de minimización sobre un conjunto convexo. Al introducir variables artificiales, el problema (Ecuación 2.17) (y (Ecuación 2.14)) puede formularse como un programa cuadrático convexo – es decir, un problema con un objetivo cuadrático convexo y restricciones lineales.

Al tomar el dual de este problema, obtenemos otro programa cuadrático convexo, en m variables:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^m} \quad & \frac{1}{2}\alpha^T Q \alpha - \mathbf{1}^T \alpha \\ \text{sujeto a} \quad & 0 \leq \alpha \leq \frac{1}{\lambda}\mathbf{1}, \quad y^T \alpha = 0 \end{aligned} \tag{2.18}$$

donde:

- $Q_{kl} = y_k y_l \psi(a_k)^T \psi(a_l)$
- $y = (y_1, y_2, \dots, y_m)^T$

- $\mathbf{1} = (1, 1, \ldots, 1)^T$

> **El Truco del Kernel**
>
> Curiosamente, el problema (Ecuación 2.18) puede formularse y resolverse sin conocimiento o definición explícita del mapeo ψ. Solo necesitamos una técnica para definir los elementos de Q. Esto se puede hacer con el uso de una **función kernel** $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, donde $K(a_k, a_l)$ reemplaza $\psi(a_k)^T \psi(a_l)$. Este es el llamado **truco del kernel**.

La función kernel K también puede usarse para construir una función de clasificación ϕ a partir de la solución de (Ecuación 2.18). Una elección particularmente popular de kernel es el **kernel Gaussiano**:

$$K(a_k, a_l) := \exp\left(-\frac{1}{2\sigma^2}\|a_k - a_l\|_2^2\right) \tag{2.19}$$

donde σ es un parámetro positivo.

2.6. Regresión Logística

La regresión logística puede verse como una forma suavizada de clasificación binaria de máquina de vectores de soporte en la que, en lugar de que la función de clasificación ϕ dé una predicción sin calificar de la clase en la que se encuentra un nuevo vector de datos a, devuelve una estimación de las **probabilidades** de que a pertenezca a una clase u otra.

Buscamos una "función de probabilidades" p parametrizada por un vector $x \in \mathbb{R}^n$:

$$p(a; x) := (1 + \exp(a^T x))^{-1} \tag{2.20}$$

y buscamos elegir el parámetro x de modo que:

- $p(a_j; x) \approx 1$ cuando $y_j = +1$
- $p(a_j; x) \approx 0$ cuando $y_j = -1$

Nótese la similitud con (Ecuación 2.11)–(Ecuación 2.12).

2.6.1. Logaritmo Negativo de Verosimilitud

El valor óptimo de x puede encontrarse minimizando una **función de logaritmo negativo de verosimilitud**:

$$L(x) := -\frac{1}{m}\left[\sum_{j:y_j=-1} \log(1-p(a_j;x)) + \sum_{j:y_j=1} \log p(a_j;x)\right] \tag{2.21}$$

Nótese que la definición (Ecuación 2.20) asegura que $p(a;x) \in (0,1)$ para todos a y x; por lo tanto, $\log(1-p(a_j;x)) < 0$ y $\log p(a_j;x) < 0$ para todos j y todos x. Cuando se satisfacen las condiciones anteriores, estos términos logarítmicos serán solo ligeramente negativos, por lo que los valores de x que los satisfacen estarán cerca del óptimo.

2.6.2. Selección de Características con LASSO

Podemos realizar selección de características usando el modelo (Ecuación 2.21) introduciendo un regularizador $\lambda\|x\|_1$ (como en la técnica LASSO para mínimos cuadrados):

$$\min_x -\frac{1}{m}\left[\sum_{j:y_j=-1} \log(1-p(a_j;x)) + \sum_{j:y_j=1} \log p(a_j;x)\right] + \lambda\|x\|_1 \tag{2.22}$$

donde $\lambda > 0$ es un parámetro de regularización. Este término tiene el efecto de producir una solución en la que pocos componentes de x son no nulos, haciendo posible evaluar $p(a;x)$ conociendo solo aquellos componentes de a que corresponden a los no nulos en x.

2.6.3. Regresión Logística Multiclase

Una extensión importante de esta técnica es a la **regresión logística multiclase (o multinomial)**, en la que los vectores de datos a_j pertenecen a más de dos clases. Tales aplicaciones son comunes en el análisis de datos moderno.

> **Ejemplo: Reconocimiento de Voz**
>
> En un sistema de reconocimiento de voz, las M clases podrían representar cada una un fonema del habla, uno de los potencialmente miles de sonidos elementales distintos que pueden ser pronunciados por humanos en unas pocas decenas de milisegundos.

Un problema de regresión logística multinomial requiere una función de probabilidades distinta p_k para cada clase $k \in \{1,2,\ldots,M\}$. Estas funciones se parametrizan mediante vectores $x^{[k]} \in \mathbb{R}^n$, $k = 1,2,\ldots,M$, definidos como sigue:

$$p_k(a; X) := \frac{\exp(a^T x^{[k]})}{\sum_{l=1}^{M} \exp(a^T x^{[l]})}, \quad k = 1, 2, \ldots, M \tag{2.23}$$

donde definimos $X := \{x^{[k]} \mid k = 1, 2, \ldots, M\}$.

Como en el caso binario, tenemos $p_k(a) \in (0, 1)$ para todos a y todos $k = 1, 2, \ldots, M$ y, además, que $\sum_{k=1}^{M} p_k(a) = 1$. Las funciones (Ecuación 2.23) realizan un **"softmax"** en las cantidades $\{a^T x^{[l]} \mid l = 1, 2, \ldots, M\}$.

En el contexto de la regresión logística multiclase, las etiquetas y_j son vectores en $\mathbb{R}^M$ cuyos elementos se definen como sigue:

$$y_{jk} = \begin{cases} 1 & \text{cuando } a_j \text{ pertenece a la clase } k, \\ 0 & \text{en caso contrario.} \end{cases} \tag{2.24}$$

De manera similar al caso binario, buscamos definir los vectores $x^{[k]}$ de modo que:

- $p_k(a_j; X) \approx 1$ cuando $y_{jk} = 1$
- $p_k(a_j; X) \approx 0$ cuando $y_{jk} = 0$

El problema de encontrar valores de $x^{[k]}$ que satisfagan estas condiciones puede formularse nuevamente como uno de minimizar un logaritmo negativo de verosimilitud:

$$L(X) := -\frac{1}{m} \sum_{j=1}^{m} \left[\sum_{\ell=1}^{M} y_{j\ell} (x^{[\ell]T} a_j) - \log \left(\sum_{\ell=1}^{M} \exp(x^{[\ell]T} a_j) \right) \right] \tag{2.25}$$

Se pueden incluir términos de regularización de "dispersión grupal" en esta formulación para seleccionar un conjunto de características en los vectores a_j, común a cada clase, que distingan efectivamente entre las clases.

2.7. Aprendizaje Profundo

Las redes neuronales profundas a menudo están diseñadas para realizar la misma función que la regresión logística multiclase – es decir, clasificar un vector de datos a en una de M clases posibles, a menudo para M grande. La innovación principal es que el mapeo ϕ del vector de datos a la predicción es ahora una **función no lineal**, parametrizada explícitamente por un conjunto de transformaciones estructuradas.

2.7.1. Estructura de la Red Neuronal

La red neuronal mostrada en forma conceptual ilustra la estructura de una red neuronal particular. En esta estructura:

- El vector de datos a_j entra por la izquierda de la red
- Cada caja (más a menudo referida como una "capa") representa una transformación que toma un vector de entrada y aplica una transformación no lineal de los datos para producir un vector de salida
- La salida de cada operador se convierte en la entrada para una o más capas subsiguientes
- Cada capa tiene su propio conjunto de parámetros, y la colección de todos los parámetros sobre todas las capas comprende nuestra variable de optimización
- Los diferentes tipos de transformaciones pueden diferir entre capas, pero podemos componerlas de la manera que se adapte a nuestra aplicación

2.7.2. Transformaciones de Capa

Una transformación típica, que convierte el vector a_j^{l-1} que representa la salida de la capa $l-1$ al vector a_j^l que representa la salida de la capa l, es:

$$a_j^l = \sigma(W^l a_j^{l-1} + g^l) \tag{2.26}$$

donde:

- W^l es una matriz de dimensión $|a_j^l| \times |a_j^{l-1}|$
- g^l es un vector de longitud $|a_j^l|$
- σ es una transformación no lineal componente a componente, usualmente llamada **función de activación**

Las formas más comunes de la función de activación σ actúan independientemente en cada componente de su vector de argumento como sigue:

- **Sigmoide**: $t \to 1/(1+e^{-t})$
- **Unidad Lineal Rectificada (ReLU)**: $t \to \text{máx}(t, 0)$

Se necesitan transformaciones alternativas cuando la entrada a la caja l proviene de dos o más cajas precedentes.

2.7.3. Capa de Salida y Función de Pérdida

La capa más a la derecha de la red neuronal (la **capa de salida**) típicamente tiene M salidas, una para cada una de las posibles clases a las que la entrada (a_j, digamos) podría pertenecer. Estas se comparan con las etiquetas y_{jk}, definidas como en (Ecuación 2.24) para indicar a cuál de las M clases pertenece a_j. A menudo, se aplica un softmax a las salidas en la capa más a la derecha, y se obtiene una función de pérdida similar a (Ecuación 2.25).

2.7.4. Problema de Optimización en Aprendizaje Profundo

Consideremos el caso especial (pero no infrecuente) en el que la estructura de la red neuronal es un grafo lineal de D niveles, en el que la salida de la capa $l-1$ se convierte en la entrada de la capa l (para $l = 1, 2, \ldots, D$) con $a_j = a_j^0$, $j = 1, 2, \ldots, m$, y la transformación dentro de cada caja tiene la forma (Ecuación 2.26). Se aplica un softmax a la salida de la capa más a la derecha para obtener un conjunto de probabilidades.

Los parámetros en esta red neuronal son los pares matriz-vector (W^l, g^l), $l = 1, 2, \ldots, D$ que transforman el vector de entrada $a_j = a_j^0$ en la salida a_j^D de la capa final. Buscamos elegir todos estos parámetros de modo que la red haga un buen trabajo al clasificar correctamente los datos de entrenamiento.

Usando la notación w para las transformaciones capa a capa, es decir:

$$w := (W^1, g^1, W^2, g^2, \ldots, W^D, g^D)$$

podemos escribir la función de pérdida para aprendizaje profundo como:

$$L(w) = -\frac{1}{m} \sum_{j=1}^{m} \left[\sum_{\ell=1}^{M} y_{j\ell} a_{j,\ell}^D(w) - \log \left(\sum_{\ell=1}^{M} \exp a_{j,\ell}^D(w) \right) \right] \tag{2.27}$$

donde $a_{j,\ell}^D(w) \in \mathbb{R}$ es la salida del elemento ℓ-ésimo en la capa D correspondiente al vector de entrada a_j^0. (Aquí escribimos $a_{j,\ell}^D(w)$ para hacer explícita la dependencia de las transformaciones w así como del vector de entrada a_j.)

Podemos ver la regresión logística multiclase como un caso especial de aprendizaje profundo con $D = 1$, de modo que $a_{j,\ell}^1 = W_{\ell,\cdot}^1 a_j^0$, donde $W_{\ell,\cdot}^1$ denota la fila ℓ de la matriz W^1.

2.7.5. Variantes e Ingeniería

Las redes neuronales en uso para aplicaciones particulares (por ejemplo, en reconocimiento de imagen y reconocimiento de voz, donde han tenido bastante éxito) incluyen muchas variantes del diseño básico. Estas incluyen:

- **Conectividad restringida** entre las cajas (que corresponde a imponer estructura de dispersión en las matrices W^l, $l = 1, 2, \ldots, D$)
- **Compartir parámetros**, que corresponde a forzar subconjuntos de los elementos de W^l a tomar el mismo valor
- **Arreglos complejos** de las cajas, con salidas provenientes de varias capas, conexiones a través de capas no adyacentes, diferentes transformaciones componente a componente σ en diferentes capas, y así sucesivamente

Las redes neuronales profundas para aplicaciones prácticas son objetos altamente diseñados.

2.7.6. Características Distintivas del Aprendizaje Profundo

La función de pérdida (Ecuación 2.27) comparte con muchas otras aplicaciones la forma de suma finita (Ecuación 2.2), pero tiene varias características que la distinguen de las otras aplicaciones discutidas anteriormente:

1. **No Convexidad**: Lo más importante, es no convexa en los parámetros w
2. **Gran Escala**: El número total de parámetros en w es usualmente muy grande

El entrenamiento efectivo de clasificadores de aprendizaje profundo típicamente requiere una gran cantidad de datos y poder de cómputo. Enormes clústeres de computadoras potentes – a menudo usando procesadores multinúcleo, GPUs, e incluso unidades de procesamiento especialmente arquitecturadas – se dedican a esta tarea.

2.8. Énfasis

Muchos problemas pueden formularse como en el marco (Ecuación 2.3), y sus propiedades pueden diferir significativamente. Podrían ser convexos o no convexos, y suaves o no suaves. Pero hay características importantes que todos comparten:

Características Compartidas de los Problemas de Optimización

- Pueden formularse como **funciones de variables reales**, que típicamente organizamos en un vector de longitud n
- Las funciones son **continuas**. Cuando aparece no suavidad en la formulación, lo

hace de manera estructurada que puede ser explotada por el algoritmo
- Las **propiedades de suavidad** permiten a un algoritmo hacer buenas inferencias sobre el comportamiento de la función basándose en el conocimiento obtenido en puntos cercanos que han sido visitados previamente
- El objetivo a menudo se compone en parte de una **suma de muchos términos**, donde cada término depende de un único elemento de datos
- El objetivo es a menudo una **suma de dos términos**: un "término de pérdida" (a veces surgiendo de una expresión de máxima verosimilitud para algún modelo estadístico) y un "término de regularización" cuyo propósito es imponer estructura y "generalizabilidad" en el modelo recuperado

2.8.1. Énfasis del Tratamiento

Nuestro tratamiento enfatiza **algoritmos** para resolver estos diversos tipos de problemas, con análisis de las propiedades de convergencia de estos algoritmos. Prestamos atención a las **garantías de complejidad**, que son límites en la cantidad de esfuerzo computacional requerido para obtener soluciones de una precisión dada. Estos límites usualmente dependen de propiedades fundamentales de la función objetivo y los datos que la definen, incluyendo:

- Las dimensiones del conjunto de datos
- El número de variables en el problema

Este énfasis contrasta con gran parte de la literatura de optimización, en la que los resultados de convergencia global usualmente no involucran límites de complejidad. (Una excepción notable es el análisis de métodos de punto interior.)

2.8.2. Preocupaciones Prácticas

Al mismo tiempo, tratamos en la medida de lo posible de enfatizar las **preocupaciones prácticas** asociadas con la resolución de estos problemas. Hay una variedad de compromisos presentados por cualquier problema, y el optimizador tiene que evaluar qué herramientas son más apropiadas para usar. Además de la formulación del problema, es imperativo tener en cuenta:

- El **presupuesto de tiempo** para la tarea en cuestión
- El **tipo de computadora** en la que se resolverá el problema
- Las **garantías necesarias** para la solución

Parte II

Resumen y Referencias

3 Summary

In summary, this book has no content whatsoever.

References

Knuth, Donald E. 1984. «Literate Programming». *Comput. J.* 27 (2): 97-111. https://doi.org/10.1093/comjnl/27.2.97.