Formeln

Lineare Regression

Linearer Zusammenhang zwischen den Eingabevariablen x und der Ausgabevariable y wird modelliert.

Hypothesenfunktion:
$h_{\theta(x)} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$

Kostenfunktion (MSE):
$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Ziel:
Finde Parameter θ um J zu minimieren
$\min J(\theta)$

Multivariat:
Mehrere Features $x_1, x_2, ..., x_n$

Polynom-Regression:
$h_{\theta(x)} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + ...$

Gradient Descent

Update-Regel:
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

Für lineare Regression:
$\theta_j := \theta_j + \alpha \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) \cdot x_j^{(i)}$

Lernrate α:
Zu groß ⟶ Divergenz,
zu klein ⟶ langsame Konvergenz

Logistische Regression

Sigmoidfunktion:
$g(z) = \frac{1}{1+e^{-z}}$

Hypothese:
$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

Klassifikation:
$h_\theta(x) \geq 0.5 \rightarrow$ Klasse 1
$h_\theta(x) < 0.5 \rightarrow$ Klasse 0

Entscheidungsgrenze:
$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$

Nicht-linearität:
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + ...)$

Regularisierung

Kostenfunktion mit L2-Regularisierung:
$J(\theta) = \frac{1}{2n} \sum \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{d} \theta_j^2$

Effekt von λ:
- $\lambda = 0 \rightarrow$ kein Penalty
- großes $\lambda \rightarrow$ starke Bestrafung, Underfitting

Bias-Term θ_0 wird oft nicht regularisiert

Support Vector Machines

Ziel:
$m \underset{\{w,b\}}{\in} (1)\{2\} \ |w|^2 + C \sum xi_i)$

Nebenbedingungen:
$y^{\{(i)\}}\left(w^T x^{\{(i)\}} + b\right) ge 1 - xi_i$ mit $xi_i ge 0$

C kontrolliert Trade-off:
großes $C \rightarrow$ weniger Fehler,
kleines $C \rightarrow$ größerer Margin

Kernel-Trick:
z.B. $K(x, x') = e^{\{-\gamma \ |x-x'|^2\}}$ (RBF-Kernel)

Neuronale Netzwerke

Feedforward:
$z^{\{(l+1)\}} = \theta^{\{(l)\}} a^{\{(l)\}}$
$a^{\{(l+1)\}} = g\left(z^{\{(l+1)\}}\right)$

Backpropagation:
$\delta^{\{(L)\}} = a^{\{(L)\}} - y$
$\delta^{\{(l)\}} = \left(\theta^{\{(l)\}}\right)^T \delta^{\{(l+1)\}} . * g'\left(z^{\{(l)\}}\right)$

Gradientenabstieg:
$\theta^{\{(l)\}} := \theta^{\{(l)\}} - \alpha \delta^{\{(l)\}} a^{\{(l-1)\}}$

Aktivierungsfunktionen:
Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax

Convolutional Neuronal Networks

Hauptidee:
Extraktion lokaler Merkmale durch Faltungsoperationen.

Faltungsschicht (Convolution Layer):
wendet Filter (Kerne) an:
$z = W * x + b$

Pooling-Schicht:
Reduktion der Dimensionalität (z.B. Max-Pooling oder Average-Pooling)

Aktivierung:
Typisch: ReLU $f(x) = \max(0, x)$

Architektur-Beispiel:
Input ⟶ Conv ⟶ ReLU ⟶ Pool ⟶ Dense ⟶ Output

Parameteranzahl:
Abhängig von Filtergröße und Anzahl

Vorteile:
- Translation-Invarianz
- Weniger Parameter als vollständig verbundene Netze

Modell Evaluation

Konfusionsmatrix:
TP = True Positive - Patienten, die krank sind und als krank klassifiziert wurden
FP = False Positive - (Patienten, die gesund sind, aber als krank klassifiziert wurden
TN = True Negative - (Patienten, die gesund sind und als gesund klassifiziert wurden
FN = False Negative - Patienten, die krank sind, aber als gesund klassifiziert wurden

Metriken:
Accuracy = $\frac{\text{TP + TN}}{\text{TP + TN + FP + FN}}$
Precision (Relevanz) = $\frac{\text{TP}}{\text{TP + FP}}$
Recall (Sensitivity) = $\frac{\text{TP}}{\text{TP + FN}}$
F1-Score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision + Recall}}$

Beispiel:
Ein Modell mit hohem Recall identifiziert möglichst viele positive Fälle,
auch auf Kosten von mehr False Positives.

Wann ist Recall wichtiger?
- Krankheitserkennung
- Sicherheitschecks
- Betrugserkennung

Entscheidungsbäume

Grundidee:
Baumstruktur zur schrittweisen Entscheidung basierend auf Features.

Split-Kriterien:
- Entropie: $H(S) = -\sum p_i \log_2(p_i)$
- Informationsgewinn
- Gini-Index: $\text{Gini}(S) = 1 - \sum p_i^2$

Vorteile:
- Interpretierbar
- Keine Skalierung notwendig

Nachteile:
- Overfitting bei tiefen Bäumen
- Instabil bei kleinen Datenänderungen

Pruning (Beschneiden):
Reduziert Komplexität und Overfitting

Ensemble-Methoden:
- Random Forests: viele Bäume, Voting
- Boosting (z.B. AdaBoost): sequentielle Optimierung

Pricipal Component Analysis(PCA)

Ziel:
Reduktion der Dimensionalität bei maximalem Erhalt der Varianz.

Schritte:
1. Zentrieren der Daten
2. Kovarianzmatrix berechnen
3. Eigenvektoren & -werte berechnen
4. Hauptkomponenten auswählen (größte Eigenwerte)
5. Projektion der Daten auf neue Achsen

Mathematisch:
Gegeben Datenmatrix X, berechne $C = 1\}\{n\} X^T X$
Finde Eigenvektoren v mit $Cv = \lambda v$

Eigenschaften:
- Unüberwachtes Verfahren
- Hauptachsen sind orthogonal
- Keine Label nötig

Anwendungen:
- Visualisierung
- Vorverarbeitung für ML
- Rauschreduktion

Reinforcement Learning

Grundidee:
Ein Agent lernt durch Interaktion mit einer Umgebung, um Belohnungen zu maximieren.

Zentrale Begriffe:
- Agent, Environment
- Zustand s, Aktion a, Belohnung r, Politik $\pi(a|s)$
- Ziel: Maximiere erwarteten kumulierten Reward

Belohnungsformel:
$R_t = \sum_{\{k=0\}}^{\{\infty\}} \gamma^k r_{\{t+k+1\}}$ mit Diskontfaktor $\gamma \in [0, 1]$

Q-Learning:
$Q(s,a) := Q(s,a) + \alpha \left[r + \gamma \max_{\{a'\}} Q(s', a') - Q(s,a)\right]$

Strategien:
- Exploration vs. Exploitation
- var ε-greedy Policy

Anwendungen:
- Spiele (z.B. AlphaGo)
- Robotik
- Empfehlungssysteme

von Eva Hölzl

Aufgaben