MATEMÁTICA COMPLETA PARA MACHINE LEARNING & DEEP LEARNING

Do Zero ao Herói: Fundamentos Matemáticos para Previsão de Demanda

Nova Corrente - Curso Educacional Completo

Objetivo Deste Curso

Este documento fornece uma **jornada completa** da matemática necessária para compreender, implementar e otimizar modelos de previsão de demanda, desde os fundamentos até técnicas avançadas de Deep Learning.

Público-alvo: Engenheiros, cientistas de dados e estudantes que desejam dominar a matemática por trás de ARIMA, Prophet, LSTM, XGBoost e modelos híbridos.

MÓDULO 1: FUNDAMENTOS MATEMÁTICOS

1.1 Álgebra Linear - A Base de Tudo

1.1.1 Vetores e Matrizes

Definição: Um vetor é uma lista ordenada de números.

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^n$$

Exemplo prático (Nova Corrente):

$$\text{Demanda\textbackslash\_Semanal} = \begin{bmatrix} 85 \\ 92 \\ 78 \\ 101 \\ 88 \\ 95 \\ 72 \end{bmatrix} \text{ (conectores por dia)}$$

Matriz: Tabela retangular de números.

$$A = \begin{bmatrix} a_{11} & amp; a_{12} & amp; \cdots & amp; a_{1n} \\ a_{21} & amp; a_{22} & amp; \cdots & amp; a_{2n} \\ \vdots & amp; \vdots & amp; \ddots & amp; \vdots \\ a_{m1} & amp; a_{m2} & amp; \cdots & amp; a_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Exemplo prático:

$$\text{Consumo\textbackslash\_Sites} = \begin{bmatrix} 85 & amp; 92 & amp; 78 \\ 101 & amp; 88 & amp; 95 \\ 72 & amp; 84 & amp; 90 \end{bmatrix} \text{ (3 sites} \times \text{3 materiais)}$$

1.1.2 Operações Fundamentais

Multiplicação Matriz-Vetor:

$$A\vec{x} = \begin{bmatrix} a_{11} & amp; a_{12} \\ a_{21} & amp; a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix}$$

Aplicação: Cálculo de previsão linear.

$$\hat{y} = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

onde:

- $\hat{y}$: Demanda prevista
- w_i: Pesos (importância de cada feature)
- x_i: Features (temperatura, dia da semana, etc.)
- b: Bias (termo constante)

Exemplo numérico:

$$\hat{y} = 0.8 \times 28 + 0.5 \times 1 + 0.3 \times 5 + 10 = 34.4 \text{ conectores}$$

1.1.3 Normas e Distâncias

Norma L2 (Euclidiana):

$$\|\vec{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Aplicação: Medir o erro de previsão.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Norma L1 (Manhattan):

$$\|\vec{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

Aplicação: Mean Absolute Error (MAE).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

1.2 Cálculo Diferencial e Integral

1.2.1 Derivadas - A Essência da Otimização

Definição:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Interpretação: Taxa de variação instantânea.

Regras Fundamentais:

1. **Regra da Potência:**

$$\frac{d}{dx}(x^n) = nx^{n-1}$$

2. **Regra da Cadeia:**

$$\frac{d}{dx}f(g(x)) = f'(g(x)) \cdot g'(x)$$

3. **Regra do Produto:**

$$\frac{d}{dx}[f(x) \cdot g(x)] = f'(x)g(x) + f(x)g'(x)$$

1.2.2 Gradiente - Direção de Maior Crescimento

Para função de múltiplas variáveis $f(x_1, x_2, \ldots, x_n)$:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Aplicação em ML: Gradient Descent.

$$\theta_{new} = \theta_{old} - \alpha \nabla L(\theta)$$

onde:

- θ: Parâmetros do modelo (pesos)
- α: Taxa de aprendizado (learning rate)

- $L(\theta)$: Função de perda (loss function)
- $\nabla L(\theta)$: Gradiente da perda

Exemplo (Regressão Linear):

Função de perda (Mean Squared Error):

$$L(w, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (wx_i + b))^2$$

Gradientes:

$$\frac{\partial L}{\partial w} = -\frac{2}{n} \sum_{i=1}^{n} x_i (y_i - (wx_i + b))$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n} \sum_{i=1}^{n} (y_i - (wx_i + b))$$

Atualização:

$$w_{new} = w_{old} - \alpha \frac{\partial L}{\partial w}$$

$$b_{new} = b_{old} - \alpha \frac{\partial L}{\partial b}$$

1.3 Probabilidade e Estatística

1.3.1 Distribuições de Probabilidade

Distribuição Normal (Gaussiana):

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

onde:

- μ: Média
- σ^2: Variância
- σ: Desvio padrão

Aplicação: Intervalos de confiança em previsões.

$$\text{IC}_{95\%} = \hat{y} \pm 1.96 \times \sigma_{\hat{y}}$$

1.3.2 Esperança e Variância

Esperança (Média):

$$\mathbb{E}[X] = \sum_{i=1}^{n} x_i P(X = x_i)$$

ou, para variável contínua:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x) dx$$

Variância:

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

Aplicação no Reorder Point:

Safety Stock com variabilidade:

$$SS = Z_\alpha \times \sigma_d \times \sqrt{LT}$$

onde:

- Z_α: Quantil da normal para nível de serviço α
- σ_d: Desvio padrão da demanda diária
- LT: Lead time

Exemplo:

- Nível de serviço: 95% → $Z_{0.95} = 1.65$
- $\sigma_d = 2.5$ conectores
- $LT = 14$ dias

$$SS = 1.65 \times 2.5 \times \sqrt{14} = 1.65 \times 2.5 \times 3.74 = 15.4 \approx 16 \text{ unidades}$$

1.3.3 Covariância e Correlação

Covariância:

$$\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

Correlação de Pearson:

$$\rho_{XY} = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

onde $\rho \in [-1, 1]$.

Aplicação: Identificar features correlacionadas com demanda.

MÓDULO 2: SÉRIES TEMPORAIS - FUNDAMENTOS

2.1 Conceitos Básicos

2.1.1 Componentes de Série Temporal

Uma série temporal Y_t pode ser decomposta em:

$$Y_t = T_t + S_t + C_t + I_t$$

onde:

- T_t: **Tendência** (trend) - movimento de longo prazo
- S_t: **Sazonalidade** (seasonality) - padrão repetitivo
- C_t: **Ciclo** (cycle) - flutuações de médio prazo
- I_t: **Irregular** (noise) - variação aleatória

Exemplo Nova Corrente:

- T_t: Crescimento de 5G → +15% demanda/ano
- S_t: Verão → +30% demanda refrigeração
- C_t: Ciclo econômico (recessão/crescimento)
- I_t: Variação diária aleatória

2.1.2 Estacionariedade

Definição: Série é estacionária se suas propriedades estatísticas não mudam ao longo do tempo.

Formalmente:

1. $\mathbb{E}[Y_t] = \mu$ (média constante)
2. $\text{Var}(Y_t) = \sigma^2$ (variância constante)
3. $\text{Cov}(Y_t, Y_{t+k}) = \gamma_k$ (depende apenas do lag k)

Teste de estacionariedade: Augmented Dickey-Fuller (ADF).

Hipótese nula H_0: Série tem raiz unitária (não estacionária).

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta Y_{t-i} + \epsilon_t$$

Se , rejeitamos H_0 → série é estacionária.

2.1.3 Diferenciação

Primeira diferença:

$$\Delta Y_t = Y_t - Y_{t-1}$$

Segunda diferença:

$$\Delta^2 Y_t = \Delta Y_t - \Delta Y_{t-1} = Y_t - 2Y_{t-1} + Y_{t-2}$$

Aplicação: Tornar série estacionária para ARIMA.

2.2 Autocorrelação e Parcial Autocorrelação

2.2.1 Função de Autocorrelação (ACF)

Mede correlação entre Y_t e Y_{t-k}:

$$\rho_k = \frac{\text{Cov}(Y_t, Y_{t-k})}{\text{Var}(Y_t)}$$

Interpretação:

- : Correlação positiva (valores similares separados por k períodos)
- : Correlação negativa
- $\rho_k \approx 0$: Não correlacionados

2.2.2 Função de Autocorrelação Parcial (PACF)

Correlação entre Y_t e Y_{t-k} **removendo** influência de lags intermediários.

$$\phi_{kk} = \text{Corr}(Y_t - \hat{Y}_t, Y_{t-k} - \hat{Y}_{t-k})$$

onde $\hat{Y}$ é predição baseada em lags 1 a $k-1$.

Uso: Identificar ordem de modelos AR e MA.

MÓDULO 3: ARIMA - MATEMÁTICA DETALHADA

3.1 Modelo AR (AutoRegressive)

Definição: Regressão do valor atual sobre valores passados.

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t$$

onde:

- p: Ordem do modelo (número de lags)
- ϕ_i: Coeficientes autorregressivos

- c: Constante
- $\epsilon_t \sim N(0, \sigma^2)$: Ruído branco

Exemplo AR(1):

$$Y_t = c + \phi_1 Y_{t-1} + \epsilon_t$$

Se $\phi_1 = 0.8$, $c = 10$:

$$Y_t = 10 + 0.8Y_{t-1} + \epsilon_t$$

Interpretação: Valor hoje = 10 + 80% do valor de ontem + ruído.

Condição de estacionariedade: para AR(1).

3.2 Modelo MA (Moving Average)

Definição: Combinação linear de erros passados.

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

onde:

- q: Ordem do modelo
- θ_i: Coeficientes de média móvel
- μ: Média da série

Exemplo MA(1):

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1}$$

Se $\theta_1 = 0.5$, $\mu = 50$:

$$Y_t = 50 + \epsilon_t + 0.5\epsilon_{t-1}$$

Interpretação: Valor hoje = 50 + erro de hoje + 50% do erro de ontem.

3.3 Modelo ARIMA(p,d,q)

Combinação de AR, I (Integração/Diferenciação), e MA:

$$\phi(B)(1 - B)^d Y_t = \theta(B)\epsilon_t$$

onde:

- B: Operador backshift ($BY_t = Y_{t-1}$)
- d: Ordem de diferenciação
- $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$
- $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$

Exemplo ARIMA(1,1,1):

Após primeira diferença:

$$\Delta Y_t = Y_t - Y_{t-1}$$

Modelo ARMA(1,1):

$$\Delta Y_t = \phi_1 \Delta Y_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1}$$

Seleção de ordem (p,d,q):

1. **AIC (Akaike Information Criterion):**

$$\text{AIC} = 2k - 2\ln(L)$$

onde k = número de parâmetros, L = likelihood.

2. **BIC (Bayesian Information Criterion):**

$$\text{BIC} = k\ln(n) - 2\ln(L)$$

onde n = tamanho da amostra.

Regra: Menor AIC/BIC = melhor modelo.

3.4 SARIMA - Sazonalidade

SARIMA(p,d,q)(P,D,Q)s:

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D Y_t = \theta(B)\Theta(B^s)\epsilon_t$$

onde:

- s: Período sazonal (ex: 7 para semanal, 12 para mensal)
- P, D, Q: Componentes sazonais

Exemplo SARIMA(1,1,1)(1,1,1)7:

Captura:

- Tendência (diferença simples)
- Sazonalidade semanal (diferença de lag 7)
- AR e MA para ambos componentes

MÓDULO 4: FACEBOOK PROPHET - MATEMÁTICA

4.1 Modelo Aditivo

Prophet decompõe série temporal como:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

onde:

- $g(t)$: Tendência (trend)
- $s(t)$: Sazonalidade (seasonality)
- $h(t)$: Feriados/eventos (holidays)
- ϵ_t: Erro (noise)

4.2 Componente de Tendência

Modelo de crescimento logístico:

$$g(t) = \frac{C(t)}{1 + \exp(-k(t - m))}$$

onde:

- $C(t)$: Capacidade de saturação (market cap)
- k: Taxa de crescimento
- m: Ponto de inflexão

Com changepoints (mudanças de tendência):

$$g(t) = \left(k + \vec{a}(t)^T\vec{\delta}\right) \cdot t + \left(m + \vec{a}(t)^T\vec{\gamma}\right)$$

onde:

- $\vec{\delta}$: Vetor de ajustes de taxa em changepoints
- $\vec{\gamma}$: Vetor de ajustes de offset
- $\vec{a}(t)$: Vetor indicador de changepoints até tempo t

4.3 Componente de Sazonalidade

Série de Fourier:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right)\right)$$

onde:

- P: Período (365.25 para anual, 7 para semanal)
- N: Número de termos de Fourier

- a_n, b_n: Coeficientes a estimar

Múltiplas sazonalidades:

$$s(t) = s_{year}(t) + s_{week}(t) + s_{day}(t)$$

4.4 Componente de Feriados

$$h(t) = Z(t)\vec{\kappa}$$

onde:

- $Z(t)$: Matriz de indicadores de feriados
- $\vec{\kappa}$: Vetor de efeitos de feriados

Janela de influência:

Para feriado i com janela $[\ell_i, u_i]$:

$$Z_{it} = \begin{cases} 1 & amp; \text{se } t \in [\ell_i, u_i] \\ 0 & amp; \text{caso contrário} \end{cases}$$

4.5 Estimação de Parâmetros

Prophet usa **Maximum A Posteriori (MAP)** com priors:

$$\vec{\theta}^* = \arg\max_{\vec{\theta}} \left[\log P(\vec{\theta}|\vec{y})\right]$$

$$= \arg\max_{\vec{\theta}} \left[\log P(\vec{y}|\vec{\theta}) + \log P(\vec{\theta})\right]$$

Prior para changepoints:

$$\delta_j \sim \text{Laplace}(0, \tau)$$

Prior para sazonalidade:

$$\beta \sim N(0, \sigma^2)$$

MÓDULO 5: REDES NEURAIS - FUNDAMENTOS

5.1 Neurônio Artificial

Operação matemática:

$$z = \sum_{i=1}^{n} w_i x_i + b = \vec{w}^T \vec{x} + b$$

$$a = \sigma(z)$$

onde:

- x_i: Inputs
- w_i: Pesos
- b: Bias
- σ: Função de ativação
- a: Output (activation)

5.2 Funções de Ativação

5.2.1 Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Propriedades:

- Range: (0, 1)
- Suave e diferenciável
- Problema: Vanishing gradient para $|z|$ grande

Derivada:

$$\frac{d\sigma}{dz} = \sigma(z)(1 - \sigma(z))$$

5.2.2 Tanh

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Propriedades:

- Range: (-1, 1)
- Centrada em zero (melhor que sigmoid)

Derivada:

$$\frac{d\tanh}{dz} = 1 - \tanh^2(z)$$

5.2.3 ReLU (Rectified Linear Unit)

$$\text{ReLU}(z) = \max(0, z) = \begin{cases} z & amp; \text{se } z & gt; 0 \\ 0 & amp; \text{se } z \leq 0 \end{cases}$$

Propriedades:

- Não satura para

- Computacionalmente eficiente
- Problema: Dead neurons ($z \leq 0$ sempre)

Derivada:

$$\frac{d\text{ReLU}}{dz} = \begin{cases} 1 & amp; \text{se } z \qquad gt; 0 \\ 0 & amp; \text{se } z \leq 0 \end{cases}$$

5.2.4 Leaky ReLU

$$\text{LeakyReLU}(z) = \begin{cases} z & amp; \text{se } z \qquad gt; 0 \\ \alpha z & amp; \text{se } z \leq 0 \end{cases}$$

onde α é pequeno (ex: 0.01).

5.3 Propagação Forward (Feedforward)

Para rede com L camadas:

Camada ℓ:

$$\vec{z}^{[\ell]} = W^{[\ell]} \vec{a}^{[\ell-1]} + \vec{b}^{[\ell]}$$

$$\vec{a}^{[\ell]} = \sigma^{[\ell]}(\vec{z}^{[\ell]})$$

onde:

- $W^{[\ell]} \in \mathbb{R}^{n^{[\ell]} \times n^{[\ell-1]}}$: Matriz de pesos
- $\vec{b}^{[\ell]} \in \mathbb{R}^{n^{[\ell]}}$: Vetor de bias
- $n^{[\ell]}$: Número de neurônios na camada ℓ

Input layer: $\vec{a}^{[0]} = \vec{x}$

Output layer: $\hat{y} = \vec{a}^{[L]}$

5.4 Função de Perda

5.4.1 Mean Squared Error (Regressão)

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

5.4.2 Cross-Entropy (Classificação)

Binária:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

Multi-classe:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic})$$

5.5 Backpropagation

Objetivo: Calcular gradientes $\frac{\partial L}{\partial W^{[\ell]}}$ e $\frac{\partial L}{\partial \vec{b}^{[\ell]}}$.

Passo 1: Gradiente da saída

$$\delta^{[L]} = \frac{\partial L}{\partial \vec{a}^{[L]}} \odot \sigma'^{[L]}(\vec{z}^{[L]})$$

Passo 2: Backpropagate erro

Para $\ell = L-1, L-2, \ldots, 1$:

$$\delta^{[\ell]} = (W^{[\ell+1]})^T \delta^{[\ell+1]} \odot \sigma'^{[\ell]}(\vec{z}^{[\ell]})$$

Passo 3: Gradientes dos parâmetros

$$\frac{\partial L}{\partial W^{[\ell]}} = \delta^{[\ell]}(\vec{a}^{[\ell-1]})^T$$

$$\frac{\partial L}{\partial \vec{b}^{[\ell]}} = \delta^{[\ell]}$$

Passo 4: Atualização (Gradient Descent)

$$W^{[\ell]} := W^{[\ell]} - \alpha\frac{\partial L}{\partial W^{[\ell]}}$$

$$\vec{b}^{[\ell]} := \vec{b}^{[\ell]} - \alpha\frac{\partial L}{\partial \vec{b}^{[\ell]}}$$

MÓDULO 6: LSTM - MATEMÁTICA PROFUNDA

6.1 Problema do Vanishing Gradient

RNNs tradicionais:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Gradiente através do tempo:

$$\frac{\partial h_t}{\partial h_{t-k}} = \prod_{i=t-k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}$$

Se , gradiente → 0 (vanishing).

Se , gradiente → ∞ (exploding).

Solução: LSTM com cell state C_t.

6.2 Arquitetura LSTM

Equações fundamentais:

6.2.1 Forget Gate

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Decide o que esquecer do cell state anterior.

6.2.2 Input Gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Decide quais novos valores adicionar ao cell state.

6.2.3 Cell State Update

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

onde $\odot$ é produto elemento-a-elemento (Hadamard).

Interpretação:

- $f_t \odot C_{t-1}$: Esquece seletivamente
- $i_t \odot \tilde{C}_t$: Adiciona nova informação

6.2.4 Output Gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

Decide o que enviar como output.

6.3 Backpropagation Through Time (BPTT) para LSTM

Cell state gradient:

$$\frac{\partial L}{\partial C_{t-1}} = \frac{\partial L}{\partial C_t} \odot f_t + \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t)) \odot f_t$$

Vantagem: Gradiente flui através de C_t com multiplicação por f_t (controlada), reduzindo vanishing.

6.4 Variantes de LSTM

6.4.1 Peephole Connections

Gates acessam C_{t-1}:

$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

6.4.2 GRU (Gated Recurrent Unit)

Simplificação do LSTM:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Vantagens:

- Menos parâmetros (mais rápido)
- Performance similar ao LSTM

MÓDULO 7: XGBOOST - MATEMÁTICA

7.1 Gradient Boosting Framework

Modelo aditivo:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

onde:

- f_t: Nova árvore na iteração t
- $\hat{y}_i^{(t-1)}$: Predição acumulada até $t-1$

Objetivo: Minimizar perda total.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

onde:

- l: Função de perda
- $\Omega(f_k)$: Regularização da árvore k

7.2 Aproximação de Segunda Ordem

Taylor expansion:

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) \approx l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)$$

onde:

- $g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$: Gradiente
- $h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)2}}$: Hessian

Objetivo simplificado (removendo constantes):

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

7.3 Estrutura de Árvore

Definição da árvore:

$$f_t(x) = w_{q(x)}$$

onde:

- $q(x)$: Função que mapeia x para folha
- w_j: Peso da folha j

- T: Número de folhas

Regularização:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

onde:

- γ: Penalidade por folha
- λ: Regularização L2 nos pesos

7.4 Otimização de Peso das Folhas

Para folha j, instâncias $I_j = \{i|q(x_i) = j\}$:

$$\tilde{\mathcal{L}}_j^{(t)} = \left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2$$

Derivando e igualando a zero:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Perda mínima:

$$\tilde{\mathcal{L}}^{(t)*} = -\frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$$

7.5 Critério de Split

Ganho ao dividir folha I em I_L (esquerda) e I_R (direita):

$$\text{Gain} = \frac{1}{2}\left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}\right] - \gamma$$

Algoritmo:

1. Para cada feature e threshold, calcule Gain
2. Escolha split com maior Gain
3. Se Gain < 0, pare de dividir

MÓDULO 8: MODELOS HÍBRIDOS

8.1 Ensemble por Média

Simples:

$$\hat{y}_{ensemble} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$$

Ponderada:

$$\hat{y}_{ensemble} = \sum_{m=1}^{M} w_m \hat{y}_m, \quad \sum_{m=1}^{M} w_m = 1$$

onde w_m são pesos otimizados (ex: por validação cruzada).

8.2 Stacking

Camada 1: Modelos base

$$\hat{y}_1^{(1)}, \hat{y}_2^{(1)}, \ldots, \hat{y}_M^{(1)}$$

Camada 2: Meta-learner

$$\hat{y}_{final} = g(\hat{y}_1^{(1)}, \hat{y}_2^{(1)}, \ldots, \hat{y}_M^{(1)})$$

onde g é regressão linear, XGBoost, etc.

Exemplo: ARIMA + LSTM + XGBoost → Meta-learner.

8.3 ARIMA + LSTM Híbrido

Decomposição:

$$Y_t = L_t + N_t$$

onde:

- L_t: Componente linear (ARIMA)
- N_t: Componente não-linear (LSTM)

Passo 1: Treinar ARIMA

$$\hat{L}_t = \text{ARIMA}(Y_t)$$

Passo 2: Calcular resíduos

$$R_t = Y_t - \hat{L}_t$$

Passo 3: Treinar LSTM nos resíduos

$$\hat{N}_t = \text{LSTM}(R_t)$$

Passo 4: Previsão final

$$\hat{Y}_t = \hat{L}_t + \hat{N}_t$$

MÓDULO 9: MÉTRICAS DE AVALIAÇÃO

9.1 Métricas de Erro

9.1.1 MAE (Mean Absolute Error)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Propriedades:

- Unidade igual a y
- Robusto a outliers

9.1.2 RMSE (Root Mean Squared Error)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Propriedades:

- Penaliza erros grandes mais que MAE
- Unidade igual a y

9.1.3 MAPE (Mean Absolute Percentage Error)

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Propriedades:

- Interpretável (%)
- Problema: Indefinido se $y_i = 0$
- Assimétrico (penaliza over-prediction mais)

9.1.4 sMAPE (Symmetric MAPE)

$$\text{sMAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

Propriedades:

- Range: [0%, 200%]
- Mais simétrico que MAPE

9.2 Métricas de Classificação (para alertas binários)

9.2.1 Confusion Matrix

$$\begin{array}{c|cc} & amp; \text{Pred Positivo} & amp; \text{Pred Negativo} \\ \hline \text{Real Positivo} & amp; TP & amp; FN \\ \text{Real Negativo} & amp; FP & amp; TN \end{array}$$

9.2.2 Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

9.2.3 Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

9.2.4 Recall (Sensitivity)

$$\text{Recall} = \frac{TP}{TP + FN}$$

9.2.5 F1-Score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

MÓDULO 10: OTIMIZAÇÃO AVANÇADA ⚙

10.1 Gradient Descent Variants

10.1.1 Batch Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

onde $J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i))$.

Problemas:

- Lento para datasets grandes
- Requer toda dataset na memória

10.1.2 Stochastic Gradient Descent (SGD)

$$\theta := \theta - \alpha \nabla_\theta L(y_i, f_\theta(x_i))$$

Atualização por **amostra individual**.

Vantagens:

- Rápido
- Pode escapar de mínimos locais

Problemas:

- Convergência oscilante
- Variância alta

10.1.3 Mini-Batch Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta \frac{1}{b} \sum_{i \in \text{batch}} L(y_i, f_\theta(x_i))$$

Tamanho de batch: 32, 64, 128, 256.

Melhor compromisso: Velocidade + estabilidade.

10.2 Momentum

$$v_t = \beta v_{t-1} + (1 - \beta) \nabla_\theta L$$

$$\theta := \theta - \alpha v_t$$

onde $\beta \approx 0.9$ (típico).

Efeito: Acelera na direção consistente, amortece oscilações.

10.3 Adam (Adaptive Moment Estimation)

Momento de primeira ordem (média):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta L$$

Momento de segunda ordem (variância):

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta L)^2$$

Correção de bias:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Atualização:

$$\theta := \theta - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Hiperparâmetros típicos:

- $\beta_1 = 0.9$
- $\beta_2 = 0.999$
- $\epsilon = 10^{-8}$
- $\alpha = 0.001$

REFERÊNCIAS E PRÓXIMOS PASSOS

Leitura Adicional

1. **Time Series Analysis:** Box, G.E.P., Jenkins, G.M. (1976)
2. **Deep Learning:** Goodfellow, I., Bengio, Y., Courville, A. (2016)
3. **XGBoost Paper:** Chen, T., Guestrin, C. (2016)
4. **LSTM Original:** Hochreiter, S., Schmidhuber, J. (1997)
5. **Prophet Paper:** Taylor, S.J., Letham, B. (2018)

Implementação Prática

Consulte os documentos complementares para:

- Código Python completo
- Datasets reais da Nova Corrente
- Notebooks Jupyter
- Diagramas Mermaid expandidos

FIM DO CURSO MATEMÁTICO

Este documento cobre os fundamentos matemáticos necessários. Para diagramas Mermaid, exemplos práticos e expansão para todas as áreas da Nova Corrente, consulte os próximos documentos da série.