Implementación de Técnicas de Descenso de Gradiente Estocástico y Variantes

Una Comparación Experimental

Brandon Trigueros

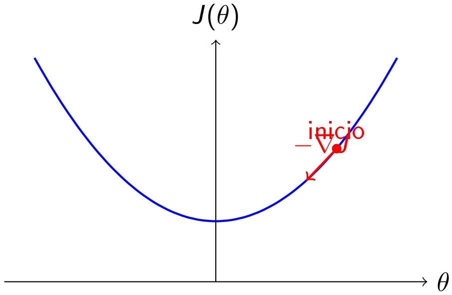
Facultad de Ingeniería
Universidad de Costa Rica

24 de junio de 2025

Índice

Motivación

- El **descenso de gradiente** es fundamental en optimización y aprendizaje automático
- Analogía: como una persona caminando por un paisaje de montañas buscando el valle más bajo
- **Problema**: El método clásico es muy lento con grandes datasets
- **Solución**: Variantes estocásticas más eficientes



Objetivos del Trabajo

Objetivo Principal

Comparar experimentalmente cuatro técnicas de optimización:

- SGD básico
- SGD con Momentum
- RMSProp
- Adam

Metodología

- Implementación en Python desde cero
- Experimento con regresión logística
- Dataset Iris (clasificación binaria)
- Análisis de curvas de convergencia

Descenso de Gradiente Clásico

Fórmula Básica

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$$

Ventajas:

- Convergencia estable
- Garantiza llegar al mínimo (funciones convexas)

Desventajas:

- Muy lento con datasets grandes
- Calcula gradiente completo en cada paso

Donde: η = tasa de aprendizaje, $J(\boldsymbol{\theta})$ = función de costo

SGD: Descenso de Gradiente Estocástico

Idea Principal

Usar solo **una muestra** (o pequeño mini-lote) por iteración:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t; \mathbf{x}_{i(t)}, y_{i(t)})$$

Trade-off Fundamental

- + Mucho más rápido computacionalmente
- + Permite manejar datasets enormes
- - Introduce ruido en las actualizaciones
- - Trayectoria más errática

SGD con Momentum

Analogía Física

Como una bola rodando que **acumula velocidad** y mantiene inercia

Fórmulas

$$\mathrm{v}_t = \gamma \mathrm{v}_{t-1} + \eta \nabla J(\boldsymbol{\theta}_t) \quad (1)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathrm{v}_t \quad (2)$$

Beneficios:

- Acelera en direcciones consistentes
- Amortigua oscilaciones

Riesgo:

- Puede "pasar de largo.el mínimo
- Requiere ajuste cuidadoso de η

Típicamente: $\gamma = 0{,}9$ (retiene 90 % de la velocidad previa)

RMSProp

Problema que Resuelve

Diferentes parámetros pueden necesitar diferentes tasas de aprendizaje

Fórmulas

$$E[g_j^2]_t = \rho E[g_j^2]_{t-1} + (1-\rho) g_{j,t}^2 \tag{3}$$

$$\theta_{j,t+1} = \theta_{j,t} - \frac{\eta}{\sqrt{E[g_j^2]_t + \varepsilon}} g_{j,t} \tag{4}$$

Intuición

- Si un parámetro tiene gradientes grandes $\Rightarrow$ paso más pequeño
- Si un parámetro tiene gradientes pequeños $\Rightarrow$ paso más grande
- **Adaptación automática** por coordenada

Típicamente: $\rho = 0{,}9$, $\varepsilon = 10^{-8}$

Adam: Lo Mejor de Dos Mundos

Combinación Inteligente

Adam = Momentum + RMSProp

Fórmulas (simplificadas)

$$m_{j,t} = \beta_1 m_{j,t-1} + (1-\beta_1) g_{j,t} \quad \text{(momentum)} \tag{5}$$

$$v_{j,t} = \beta_2 v_{j,t-1} + (1-\beta_2) g_{j,t}^2 \quad \text{(normalización)} \tag{6}$$

$$\theta_{j,t+1} = \theta_{j,t} - \frac{\eta}{\sqrt{\hat{v}_{j,t}} + \varepsilon} \hat{m}_{j,t} \tag{7}$$

¿Por qué es Popular?

- Funciona bien .ºut-of-the-box"
- Pocos hiperparámetros que ajustar
- Robusto en muchos problemas

Valores por defecto: $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, $\eta = 0{,}001$

Diseño Experimental

Dataset Iris

- 150 muestras de flores Iris
- 4 características: longitud/ancho sépalos y pétalos
- **Clasificación binaria**: Versicolor vs. Virginia
- División: 80 % entrenamiento, 20 % prueba

Modelo: Regresión Logística

- Función sigmoide: $h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$
- Función de costo: Entropía cruzada binaria
- Gradiente analítico: $\nabla J = \frac{1}{N}\sum_i (h(\mathbf{x}_i) - y_i)\mathbf{x}_i$

Implementación en Python

SGD Básico

```
for epoch in range(epochs):
    for i in range(N):
        y_pred = sigmoid(np.dot(w, X[i]))
        grad = (y_pred - y[i]) * X[i]
        w = w - lr * grad
```

SGD con Momentum

```
v = np.zeros(d)  # velocidad inicial
for epoch in range(epochs):
    for i in range(N):
        grad = compute_gradient(w, X[i], y[i])
        v = gamma * v + lr * grad
        w = w - v
```

Configuración de Hiperparámetros

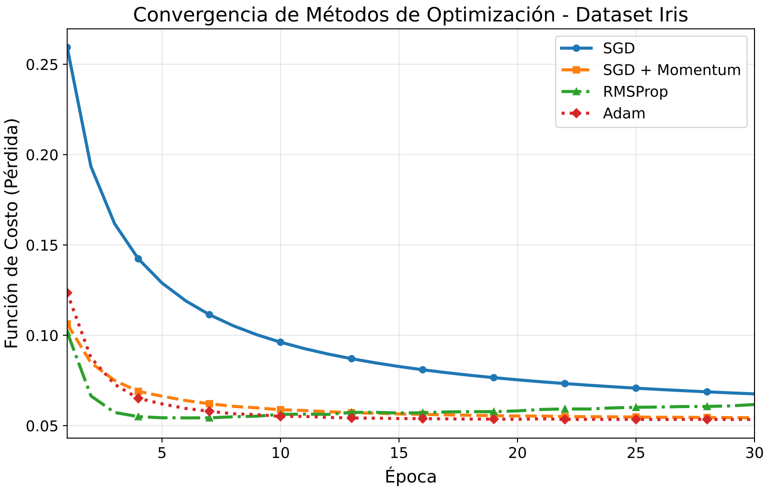
Después de experimentación, se eligieron:

Algoritmo	Tasa de Aprendizaje	Parámetros Adicionales
SGD	0.05	-
SGD + Momentum	0.03	$\gamma = 0{,}9$
RMSProp	0.05	$\rho = 0{,}9,\ \varepsilon = 10^{-8}$
Adam	0.05	$\beta_1 = 0{,}9,\ \beta_2 = 0{,}999$

Nota Importante

Momentum requirió menor tasa de aprendizaje para evitar inestabilidad

Curvas de Convergencia



- **Adam**: Convergencia más rápida (5 épocas)
- **RMSProp**: Buena velocidad, algo oscilante
- **Momentum**: Descenso inicial rápido pero inestable

Análisis Detallado de Resultados

Adam - El Ganador

- Pérdida final: $\sim 0{,}12$ en solo 10 épocas
- Curva suave y estable
- Mínima necesidad de ajuste manual

Momentum - Doble Filo

- Descenso inicial más drástico (época 2: pérdida $\sim 0{,}1$)
- Pero oscilaciones significativas después
- Evidencia del problema de "sobrepaso"

RMSProp vs SGD

- RMSProp: Convergencia acelerada ($\sim 0{,}15$ final)
- SGD: Lento pero confiable ($\sim 0{,}30$ a época 30)

Métricas de Rendimiento Final

Algoritmo	Costo Final	Precisión Train	Precisión Test
SGD	0.067	96.25 %	95.00 %
SGD + Momentum	0.054	96.25 %	95.00 %
RMSProp	0.062	96.25 %	90.00 %
Adam	0.053	97.50 %	95.00 %

Observaciones Importantes

- Precisión final similar en todos los métodos
- Diferencias principales en **velocidad de convergencia**
- Adam ligeramente superior en precisión de entrenamiento

Ventajas y Desventajas por Método

SGD

Pros:

- Simple de implementar
- Estable y confiable
- Buena generalización

Cons:

- Convergencia lenta
- Sensible a tasa de aprendizaje

Momentum

Pros:

- Acelera convergencia inicial
- Supera valles estrechos

Cons:

- Puede oscilar mucho

RMSProp

Pros:

- Adaptación automática
- Maneja bien gradientes dispersos

Cons:

- Algo más complejo
- Requiere ajuste de ρ

Adam

Pros:

- Mejor de ambos mundos
- Funciona .ºut-of-the-box"
- Robusto y rápido

Cons:

- Posible overfitting

Conclusiones Principales

1. **Adam es el claro ganador** para convergencia rápida y facilidad de uso
2. **Momentum acelera** pero requiere cuidado en la calibración
3. **RMSProp** ofrece buen compromiso entre velocidad y estabilidad
4. **SGD básico** sigue siendo válido para casos que priorizan generalización

Recomendación Práctica

- **Para empezar**: Adam con parámetros por defecto
- **Para ajuste fino**: Considerar híbrido (Adam inicial + SGD final)
- **Para datos grandes**: RMSProp o Adam
- **Para interpretabilidad**: SGD con momentum controlado

Limitaciones y Trabajo Futuro

Limitaciones del Estudio

- Experimento en problema relativamente simple (Iris)
- Solo regresión logística (modelo lineal)
- Conjunto de datos pequeño

Extensiones Propuestas

- Probar en redes neuronales profundas
- Evaluar generalización en datasets más grandes
- Implementar variantes adicionales (Nadam, AdamW, AMSGrad)
- Estudiar estrategias de learning rate scheduling
- Análisis de mínimos "planos"vs .ªfilados"

Impacto en la Práctica

En Aprendizaje Automático Moderno

- Adam es estándar en deep learning
- SGD sigue siendo importante para generalización
- Momentum útil en problemas mal condicionados
- RMSProp popular en procesamiento de lenguaje natural

Aplicaciones Reales

- **Visión por computadora**: Adam para entrenar CNNs
- **NLP**: RMSProp/Adam para RNNs y Transformers
- **Investigación**: SGD para estudios de generalización
- **Producción**: Híbridos para mejor rendimiento

Mensajes Clave

No existe un optimizador universal

- La elección depende del problema específico
- Adam es un excelente punto de partida
- Siempre monitorear tanto entrenamiento como validación
- La implementación correcta es tan importante como la elección del algoritmo

El entendimiento teórico guía las decisiones prácticas

¿Preguntas?

Gracias por su atención

Brandon Trigueros
brandon.trigueros@ucr.ac.cr

Código disponible en: github.com/usuario/gradient-descent-research

Respaldo: Fórmulas Detalladas de Adam

Algoritmo Completo

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{8}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{9}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \text{(corrección de sesgo)} \tag{10}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad \text{(corrección de sesgo)} \tag{11}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{12}$$

- m_t: estimación del primer momento (media)
- v_t: estimación del segundo momento (varianza no centrada)
- Las correcciones de sesgo son importantes en las primeras iteraciones

Respaldo: Datos del Experimento

Época	SGD	Momentum	RMSProp	Adam
1	0.259	0.106	0.102	0.124
2	0.193	0.085	0.067	0.088
5	0.129	0.066	0.054	0.062
10	0.099	0.058	0.055	0.054
20	0.078	0.053	0.060	0.053
30	0.068	0.054	0.062	0.053

Cuadro: Evolución del costo de entrenamiento

- Adam converge más rápido en las primeras épocas
- Momentum muestra la mayor reducción inicial pero luego oscila
- SGD mejora de manera más gradual y consistente