# Complexity Regularization and Network Pruning

## with Optimal Brain Surgeon (OBS)

Kiran Bagale

July, 2025

# Outline

# Motivation

**Why do we need model compression?**

- Modern deep networks are overparameterized (millions to billions of parameters)
- **Generalization:** Complex models prone to overfitting
- **Efficiency:** Memory, computational cost, energy consumption
- **Deployment:** Mobile devices, edge computing constraints

**Two complementary approaches:**

1. **Complexity Regularization** (during training)
2. **Network Pruning** (after or during training)

# Empirical Risk & Complexity Regularization

Given dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ and network $f_w$ with parameters $w$:

$$\mathcal{L}_{\mathrm{emp}}(w) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f_w(x_i))$$

**Regularized objective:**

$$\boxed{\mathcal{L}(w) = \mathcal{L}_{\mathrm{emp}}(w) + \lambda\,\Omega(w)}$$

where:

- $\lambda > 0$ is the regularization strength
- $\Omega(w)$ is the complexity penalty

# Common Regularizers

## L2 Regularization (Ridge)

$$\Omega(w) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2}\sum_j w_j^2$$

- Shrinks weights toward zero
- Smoother decision boundaries
- Corresponds to Gaussian prior: $w_j \sim \mathcal{N}(0, \frac{1}{\lambda})$

## Group Sparsity

$$\Omega(w) = \sum_{g \in \mathcal{G}} \|w_g\|_2$$

## L1 Regularization (Lasso)

$$\Omega(w) = \|w\|_1 = \sum_j |w_j|$$

- Induces sparse solutions
- Many weights become exactly zero
- Corresponds to Laplace prior: $p(w_j) \propto e^{-\lambda |w_j|}$

## Elastic Net

$$\Omega(w) = \alpha \|w\|_1 + (1-\alpha)\frac{1}{2}\|w\|_2^2$$

# Bias-Variance Tradeoff

Regularization controls the bias-variance tradeoff:

$$\text{Expected Test Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

| $\lambda$ | Bias | Variance |
|---|---|---|
| Small | Low | High (overfitting) |
| Large | High | Low (underfitting) |
| Optimal | Balanced | |

**Key insight:** Adding $\Omega(w)$ reduces variance by constraining parameter space, at the cost of increased bias.

# Network Pruning: Problem Statement

**Goal:** Starting from trained parameters $w$, find sparse $w'$ such that:

$$\boxed{\begin{aligned} \text{minimize:} \quad & \mathcal{L}_{\mathrm{emp}}(w') \\ \text{subject to:} \quad & \left\|w'\right\|_0 \leq k \end{aligned}}$$

where $\|w'\|_0$ counts non-zero elements and $k \ll \|w\|_0$.

**Challenges:**

- $\ell_0$ constraint is combinatorial (NP-hard)
- Need to maintain network functionality
- Hardware efficiency depends on pruning structure

# Types of Pruning

**1. Unstructured (Weight-level) Pruning:**

$$w'_j = \begin{cases} 0, & \text{if } |w_j| < \tau \text{ or } j \in \mathcal{S} \\ w_j, & \text{otherwise} \end{cases}$$

where $\mathcal{S}$ is the set of pruned weights.

**2. Structured Pruning:**

- **Neuron-level:** Remove entire neurons (rows/columns)
- **Channel-level:** Remove feature map channels
- **Filter-level:** Remove convolutional filters

Ranking criterion: $s_k = \|W_k\|_2$, $\|W_k\|_1$, or other measures.

**3. Pruning Strategies:**

- **One-shot:** Prune once, then fine-tune
- **Iterative:** Gradual pruning with periodic fine-tuning

# Second-Order Taylor Approximation

To estimate the impact of removing weight $w_q$, use Taylor expansion around current parameters $w$:

$$\Delta\mathcal{L} = \mathcal{L}(w + \Delta w) - \mathcal{L}(w)$$

$$\approx \nabla\mathcal{L}(w)^T \Delta w + \frac{1}{2}\Delta w^T H \Delta w$$

At a local minimum, $\nabla\mathcal{L}(w) = 0$, so:

$$\boxed{\Delta\mathcal{L} \approx \frac{1}{2}\Delta w^T H \Delta w}$$

where $H$ is the Hessian matrix:

$$H_{ij} = \frac{\partial^2 \mathcal{L}}{\partial w_i \partial w_j}$$

# Optimal Brain Damage (OBD)

**Assumption:** Use only diagonal terms of Hessian (ignore correlations):

$$\Delta\mathcal{L} \approx \frac{1}{2} H_{jj} (\Delta w_j)^2$$

For removing weight $w_q$ (setting $\Delta w_q = -w_q$):

$$\boxed{\text{OBD Saliency: } S_q^{\text{OBD}} = \frac{1}{2} H_{qq} w_q^2}$$

**Algorithm:**

1. Compute diagonal Hessian elements $H_{jj}$
2. Calculate saliency for each weight: $S_j^{\text{OBD}}$
3. Remove weight with smallest saliency
4. Fine-tune network

Limitation: Ignores weight interactions (off-diagonal terms)

# Optimal Brain Surgeon (OBS)

**Key insight:** Use full Hessian to capture weight correlations and optimally compensate remaining weights.

**Constrained optimization:** Remove $w_q$ while minimally increasing loss:

$$\begin{aligned} \text{minimize:} \quad & \frac{1}{2}\Delta w^T H \Delta w \\ \text{subject to:} \quad & w_q + \Delta w_q = 0 \end{aligned}$$

**Solution using Lagrange multipliers:**

### OBS Formulas

$$\boxed{S_q^{\text{OBS}} = \frac{w_q^2}{2(H^{-1})_{qq}}} \quad \text{(Saliency)}$$

$$\boxed{\Delta w = -\frac{w_q}{(H^{-1})_{qq}} H^{-1}_{:,q}} \quad \text{(Compensation)}$$

# OBS Algorithm

**Algorithm 1** Optimal Brain Surgeon (OBS)

```
Train network to convergence, obtaining parameters $w$
Compute Hessian $H$ of loss w.r.t. parameters
Compute inverse Hessian $H^{-1}$ (or approximation)
while target sparsity not reached do
    for each remaining weight $w_q$ do
        Compute saliency: $S_q = \frac{w_q^2}{2(H^{-1})_{qq}}$
    end for
    $q^* = \arg\min_q S_q$ (find least salient weight)
    Remove weight $w_{q^*}$ (set to zero)
    Update remaining weights: $w \leftarrow w - \frac{w_{q^*}}{(H^{-1})_{q^* q^*}} H^{-1}_{:,q^*}$
    Fine-tune network (optional)
    Update Hessian (if necessary)
end while
```

# OBS vs OBD: Mathematical Comparison

**Optimal Brain Damage**

- Uses diagonal Hessian only
- $S_q^{\text{OBD}} = \frac{1}{2} H_{qq} w_q^2$
- No weight compensation
- Fast computation: $O(n)$
- Assumes weights are uncorrelated

**Optimal Brain Surgeon**

- Uses full inverse Hessian
- $S_q^{\text{OBS}} = \frac{w_q^2}{2(H^{-1})_{qq}}$
- Optimal weight compensation
- Expensive: $O(n^3)$ for inversion
- Accounts for weight correlations

**Relationship:** If $H$ is diagonal, then $(H^{-1})_{qq} = 1/H_{qq}$ and:

$$S_q^{\text{OBS}} = \frac{w_q^2}{2/H_{qq}} = \frac{1}{2} H_{qq} w_q^2 = S_q^{\text{OBD}}$$

# Regularization ↔ Pruning Connection

**Mathematical connection:** L1 regularization naturally induces sparsity.

The regularized objective:

$$\mathcal{L}(w) = \mathcal{L}_{\mathrm{emp}}(w) + \lambda \|w\|_1$$

**Subgradient at $w_j = 0$:**

$$\frac{\partial \mathcal{L}_{\mathrm{emp}}}{\partial w_j} + \lambda \cdot \mathsf{sign}(w_j) = 0$$

For $w_j$ to remain zero: $\left|\frac{\partial \mathcal{L}_{\mathrm{emp}}}{\partial w_j}\right| \leq \lambda$

**Interpretation:**

- L1 regularization performs implicit pruning during training
- Explicit pruning removes small weights post-hoc
- Both aim to find sparse solutions, but through different mechanisms

# Practical Hessian Approximations

Computing full Hessian inverse is expensive for large networks. **Approximations:**

**1. Diagonal approximation:** $H \approx \text{diag}(H_{11}, \dots, H_{nn})$

**2. Block-diagonal:** $H \approx \text{block-diag}(H_1, \dots, H_k)$

**3. Gauss-Newton approximation:**

$$H \approx J^T J$$

where $J$ is the Jacobian of network outputs w.r.t. parameters.

**4. Fisher Information Matrix:**

$$H \approx \mathbb{E}[\nabla \log p(y|x, w) \nabla \log p(y|x, w)^T]$$

**5. Kronecker-Factored approximation (K-FAC):**

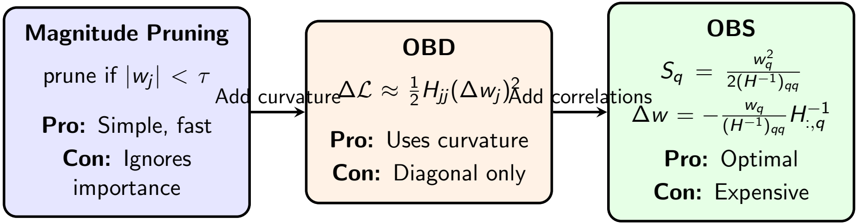
$$H_l \approx A_l \otimes G_l$$

for layer $l$ with activation covariance $A_l$ and gradient covariance $G_l$.

# Summary Comparison Table

| **Method** | **Key Formula** | **Effect** | **Complexity** |
|---|---|---|---|
| L2 Regularization | $\lambda\|w\|_2^2$ | Shrinks weights smoothly | $O(n)$ |
| L1 Regularization | $\lambda\|w\|_1$ | Induces sparsity during training | $O(n)$ |
| Magnitude Pruning | prune if $\lvert w_j\rvert < \tau$ | Removes smallest weights | $O(n)$ |
| Structured Pruning | rank by $\|W_k\|_2$ | Removes neurons/channels/filters | $O(n)$ |
| OBD | $S_q = \frac{1}{2} H_{qq} w_q^2$ | Uses diagonal curvature | $O(n^2)$ |
| OBS | $S_q = \frac{w_q^2}{2(H^{-1})_{qq}}$ | Full curvature + compensation | $O(n^3)$ |

# TikZ Flowchart: Evolution of Pruning Methods



**Magnitude Pruning**

prune if $|w_j| < \tau$

**Pro:** Simple, fast

**Con:** Ignores importance

Add curvature

**OBD**

$$\Delta\mathcal{L} \approx \frac{1}{2} H_{jj} (\Delta w_j)^2$$

**Pro:** Uses curvature

**Con:** Diagonal only

Add correlations

**OBS**

$$S_q = \frac{w_q^2}{2(H^{-1})_{qq}}$$

$$\Delta w = -\frac{w_q}{(H^{-1})_{qq}} H^{-1}_{:,q}$$

**Pro:** Optimal

**Con:** Expensive

# When to Use What?

**Choose based on your constraints:**

- **Magnitude pruning:**
    - When computational budget is very limited
    - As a simple baseline for comparison
    - When hardware supports efficient sparse operations
- **OBD:**
    - When curvature varies significantly across weights
    - Moderate computational budget available
    - Good compromise between accuracy and speed
- **OBS:**
    - When highest accuracy is crucial
    - High sparsity levels required
    - Sufficient computational resources for Hessian operations

Practical tip: Often combine approaches - use OBS for critical layers, simpler methods elsewhere.

# Key Takeaways

1. **Complementary approaches:** Regularization prevents overfitting during training; pruning removes redundancy post-training.
2. **Sparsity connection:** L1 regularization creates implicit sparsity; explicit pruning achieves structured sparsity.
3. **Second-order methods matter:** OBS generalizes OBD by using full Hessian information for correlation-aware pruning and optimal compensation.
4. **Practical considerations:** Trade-off between accuracy and computational cost determines method choice.
5. **Future directions:**
    - Dynamic sparsity during training
    - Hardware-aware pruning strategies
    - Lottery ticket hypothesis and pruning at initialization