Fast Exact Multiplication by the Hessian

Reporter: Yuri Sapronov

Introduction

- Extracting second-order information (Hessian matrix) from large neural networks is critical for:
  - Analyzing convergence (Widrow et al., 1979; Le Cun et al., 1991).
  - Enhancing generalization (MacKay, 1991; Hassibi and Stork, 1993).
  - Full second-order optimization (Watrous, 1987).
- Full Hessian calculation is impractical for large networks.
- Diagonal or trace approximations of the Hessian are sometimes used.

Objective

- Goal: Efficient technique to calculate $\mathbf{H}v$ (product of Hessian and vector) without full Hessian computation.
- Application: Estimating Hessian via products with vectors (complexity $O(n^2)$ reduced to $O(n)$ in time and space).

Gradient and Hessian Relation

- The Hessian matrix appears in the expansion of the gradient:

$$\nabla_{\mathbf{w}}(w + \Delta w) = \nabla_{\mathbf{w}}(w) + \mathbf{H}\Delta w + O(\|\Delta w\|^2)$$

- Let $\Delta w = r\mathbf{v}$, where r is a small scalar.
- Compute $\mathbf{Hv}$ using:

$$\mathbf{Hv} = \frac{\nabla_{\mathbf{w}}(w + r\mathbf{v}) - \nabla_{\mathbf{w}}(w)}{r} + O(r)$$

Approximation Algorithm

- This approximation is effective when r is small enough to minimize numerical issues.
- But small r may lead to numerical precision problems due to tiny differences in gradients.

The $R\{\cdot\}$ Technique

- Define the operator:

$$R\{\nabla_{\mathbf{w}}(f(\mathbf{w}))\} = \frac{\partial}{\partial r}\nabla_{\mathbf{w}}(w + r\mathbf{v})\Big|_{r=0}$$

- This operator transforms gradient computation to Hessian-vector product computation.

Properties of $R\{\cdot\}$

- Basic rules for $R\{\cdot\}$:

  $R\{cf(\mathbf{w})\} = cR\{f(\mathbf{w})\}, \quad R\{f(\mathbf{w})+g(\mathbf{w})\} = R\{f(\mathbf{w})\}+R\{g(\mathbf{w})\}$

- Differential properties:

$$R\left\{\frac{df(\mathbf{w})}{dt}\right\} = \frac{dR\{f(\mathbf{w})\}}{dt}$$

- With $R\{\mathbf{w}\} = \mathbf{v}$.

Backpropagation with $R\{\text{backprop}\}$

- $R\{\text{backprop}\}$ is an algorithm to efficiently calculate $\mathbf{Hv}$ for backpropagation networks.
- This method uses the gradient of the network output with respect to the input.
- The direction of equations is reversed from the gradient backpropagation algorithm.

Indexing Weights

- Let w_{ij} denote the weight from unit i to unit j.
- v has the same dimensionality as w.
- Forward computation (topological order):

$$x_i = \sum_j w_{ij} y_j$$

Forward Pass

- Total input to unit i:

$$y_i = \sigma(x_i) + I_i$$

- Error function $E = E(y)$, where:

$$e_i = \frac{dE}{dy_i}$$

- Common error measures include squared error and cross-entropy.

Backward Pass (Gradient Computation)

- Backward pass equations:

$$\frac{\partial E}{\partial y_i} = e_i(y_i) + \sum_j w_{ij} \frac{\partial E}{\partial x_j}$$

$$\frac{\partial E}{\partial x_i} = \sigma_i'(x_i) \frac{\partial E}{\partial y_i}$$

$$\frac{\partial E}{\partial w_{ij}} = y_j \frac{\partial E}{\partial x_i}$$

Applying $R\{\cdot\}$ to Forward and Backward Passes

- For the forward pass:

$$R\{x_i\} = \sum_j (w_{ij} R\{y_j\} + v_{ij} y_j)$$

$$R\{y_i\} = R\{x_i\}\sigma_i'(x_i)$$

- For the backward pass:

$$R\left\{\frac{\partial E}{\partial y_i}\right\} = e_i'(y_i) R\{y_i\} + \sum_j \left(w_{ij} R\left\{\frac{\partial E}{\partial x_j}\right\} + v_{ij} \frac{\partial E}{\partial x_j} \right)$$

Recurrent Backpropagation Algorithm

- The recurrent backpropagation algorithm (Almeida, 1987; Pineda, 1987) uses forward equations that relax to the gradient.
- It provides a dynamic framework to compute $\frac{\partial E}{\partial w_{ij}}$ as the system reaches equilibrium.

Forward Equations

- Total input to unit i:

$$x_i = \sum_j w_{ij} y_j$$

- Dynamics of y_i:

$$\frac{dy_i}{dt} \propto -y_i + \sigma_i(x_i) + I_i$$

- Dynamics of z_i:

$$\frac{dz_i}{dt} \propto -z_i + \sigma_i'(x_i) \sum_j (w_{ij} z_j) + e_i(y_i)$$

where $e_i(y_i) = \frac{\partial E}{\partial y_i}$.

Gradient Computation

- Gradient of the error with respect to weight w_{ij}:

$$\frac{\partial E}{\partial w_{ij}} = y_i z_j \Big|_{t \to \infty}$$

- As $t \to \infty$, z_j stabilizes, allowing the gradient calculation.

Adjoint Equations and the $R\{\cdot\}$ Operator

- Applying the $R\{\cdot\}$ operator to calculate $\mathbf{Hv}$:

$$R\{x_i\} = \sum_j (w_{ij} R\{y_j\} + v_{ij} y_j)$$

$$\frac{dR\{y_i\}}{dt} \propto -R\{y_i\} + \sigma_i'(x_i) R\{x_i\}$$

- Equation for $\frac{dR\{z_i\}}{dt}$:

$$\frac{dR\{z_i\}}{dt} \propto -R\{z_i\} + \sigma_i'(x_i) \sum_j (v_{ij} z_j + w_{ij} R\{z_j\})$$

$$+\sigma_i''(x_i) R\{x_i\} \sum_j (w_{ij} z_j) + e_i'(y_i) R\{y_i\}$$

Gradient with $R\{\cdot\}$ Applied

- Gradient with the $R\{\cdot\}$ operator applied:

$$R\left\{\frac{\partial E}{\partial w_{ij}}\right\} = y_i R\{z_j\} + R\{y_i\} z_j \Big|_{t\to\infty}$$

- This process specifies a relaxation approach for computing **Hv**, similar to gradient relaxation.

Boltzmann Machines: Setup

- For Boltzmann machines, we aim to compute Hv by examining the quantity $p_{ij} = \langle s_i s_j \rangle$.
- Here, s_i and s_j represent the states of neurons i and j, respectively.
- The expected value p_{ij} can be expressed as:

$$p_{ij} = \sum_{\alpha} P(\alpha) s_i^{(\alpha)} s_j^{(\alpha)},$$

where $P(\alpha)$ is the probability of configuration α.

Step 1: Defining p_{ij} in Terms of State Probabilities

- Define the probability $P(\alpha)$ using the Boltzmann distribution:

$$P(\alpha) = \frac{1}{Z} e^{-E_\alpha / T},$$

  where Z is the partition function, and E_α is the energy of configuration α:

$$E_\alpha = \sum_{i<j} s_i^{(\alpha)} s_j^{(\alpha)} w_{ij}.$$

- The partition function Z is given by:

$$Z = \sum_\alpha e^{-E_\alpha / T}.$$

Step 2: Applying R to p_{ij}

- To find the rate of change of p_{ij} along v, apply R:

$$R\{p_{ij}\} = R\left(\sum_{\alpha} P(\alpha) s_i^{(\alpha)} s_j^{(\alpha)}\right).$$

- Expanding, we get:

$$R\{p_{ij}\} = \sum_{\alpha} R\{P(\alpha)\} s_i^{(\alpha)} s_j^{(\alpha)}.$$

Step 3: Calculating $R\{P(\alpha)\}$

- Use the definition of $P(\alpha)$:

$$R\{P(\alpha)\} = R\left(\frac{1}{Z}e^{-E_\alpha/T}\right).$$

- Applying the product rule of R:

$$R\{P(\alpha)\} = \frac{1}{Z}R\left(e^{-E_\alpha/T}\right) + e^{-E_\alpha/T}R\left(\frac{1}{Z}\right).$$

Step 3: Part 1 - Calculating $R\{e^{-E_\alpha/T}\}$

- Using the chain rule:

$$R\left\{e^{-E_\alpha/T}\right\} = e^{-E_\alpha/T} \cdot \left(-\frac{1}{T}R\{E_\alpha\}\right).$$

- The directional derivative $R\{E_\alpha\}$ is:

$$R\{E_\alpha\} = \sum_{i<j} s_i^{(\alpha)} s_j^{(\alpha)} v_{ij} = D_\alpha.$$

- Therefore,

$$R\left\{e^{-E_\alpha/T}\right\} = -\frac{1}{T} D_\alpha e^{-E_\alpha/T}.$$

Step 3: Part 2 - Calculating $R\left\{\frac{1}{Z}\right\}$

- Since $Z = \sum_\alpha e^{-E_\alpha/T}$, we have:

$$R\{Z\} = \sum_\alpha R\left\{e^{-E_\alpha/T}\right\} = -\frac{1}{T}\sum_\alpha D_\alpha e^{-E_\alpha/T}.$$

- Thus,

$$R\left\{\frac{1}{Z}\right\} = -\frac{1}{Z^2}R\{Z\} = \frac{\langle D\rangle}{ZT},$$

where $\langle D\rangle = \sum_\alpha P(\alpha)D_\alpha$.

Step 3: Final Expression for $R\{P(\alpha)\}$

- Putting it all together:

$$R\{P(\alpha)\} = P(\alpha)\left(-\frac{D_\alpha}{T} + \frac{\langle D \rangle}{T}\right).$$

Step 4: Substitute $R\{P(\alpha)\}$ Back into $R\{p_{ij}\}$

- Substituting, we get:

$$R\{p_{ij}\} = \sum_{\alpha} P(\alpha) \left(-\frac{D_\alpha}{T} + \frac{\langle D \rangle}{T} \right) s_i^{(\alpha)} s_j^{(\alpha)}.$$

- This simplifies to:

$$R\{p_{ij}\} = \frac{1}{T} \left(p_{ij} \langle D \rangle - \langle s_i s_j D \rangle \right).$$

Final Formula for $R\{p_{ij}\}$

- Thus, we have shown that:

$$R\{p_{ij}\} = \frac{1}{T}\left(p_{ij}\langle D\rangle - \langle s_i s_j D\rangle\right).$$

Weight Perturbation and Gradient Estimation

- Weight perturbation is used to approximate the gradient ∇_w by adding a random perturbation vector Δw to the weights.
- We observe the resulting change in error, which is given by:

$$E(w + \Delta w) = E(w) + \Delta E = E(w) + \nabla_w \cdot \Delta w$$

- This approach lets us estimate the individual partial derivatives $\frac{\partial E}{\partial w_i}$.

Derivation of $\frac{\partial E}{\partial w_i}$ Using Least Squares

- For each weight w_i, we assume the change in error ΔE is related to the change Δw_i as:

$$\Delta E = \Delta w_i \frac{\partial E}{\partial w_i} + \text{noise}$$

- To solve for $\frac{\partial E}{\partial w_i}$, we minimize the squared error between ΔE and $\Delta w_i \frac{\partial E}{\partial w_i}$, giving an exact solution:

$$\frac{\partial E}{\partial w_i} = \frac{\langle \Delta w_i \Delta E \rangle}{\langle \Delta w_i^2 \rangle}$$

- Here, $\langle \cdot \rangle$ represents averaging over multiple perturbations. This approximation relies on the central limit theorem to provide a stable mean estimate with enough samples.

Improving the Approximation with Hessian Terms

- A better approximation for the change in error, ΔE, includes second-order terms involving the Hessian H:

$$E(w + \Delta w) = E(w) + \nabla_w \cdot \Delta w + \frac{1}{2}\Delta w^T H \Delta w$$

- Since we don't know H, we approximate it with $\hat{H}$ as an estimate of the Hessian.
- This gives us the modified expression:

$$E(w + \Delta w) \approx E(w) + \nabla_w \cdot \Delta w + \frac{1}{2}\Delta w^T \hat{H} \Delta w$$

Derivation of $\hat{H}$ via Minimization

- We want $\hat{H}$ to capture the essential properties of H without needing the full matrix.
- Let's define $\mathbf{z} = H\mathbf{v}$, where $\mathbf{v}$ is a direction vector.
- If we want $\hat{H}\mathbf{v} = \mathbf{z}$, a least-squares solution without additional constraints would be:

$$\hat{H} = \frac{\mathbf{z}\mathbf{v}^T}{\|\mathbf{v}\|^2}$$

- However, this solution is not symmetric, so it does not serve as a valid Hessian approximation.
- Therefore, we add a symmetry constraint: we require that $\hat{H}$ satisfies $\mathbf{v}^T\hat{H} = \mathbf{z}^T$.

Adding a Symmetry Constraint to $\hat{H}$

- To ensure $\hat{H}$ is symmetric, we add a symmetry constraint: $\mathbf{v}^T \hat{H} = \mathbf{z}^T$.
- The new least-squares problem with this constraint leads to the solution:

$$\hat{H} = \frac{1}{\|\mathbf{v}\|^2} \left(\mathbf{z}\mathbf{v}^T + \mathbf{v}\mathbf{z}^T - \frac{\mathbf{v} \cdot \mathbf{z}}{\|\mathbf{v}\|^2} \mathbf{v}\mathbf{v}^T \right)$$

- This formula ensures that $\hat{H}$ is symmetric, satisfies $\hat{H}\mathbf{v} = \mathbf{z}$, and approximates H in a least-squares sense.

Final Formula for ΔE with $\hat{H}$

- Now that we have a symmetric approximation for $\hat{H}$, we substitute it back into our expression for ΔE:

$$\Delta E = \Delta w_i \frac{\partial E}{\partial w_i} + \frac{\Delta w \cdot \mathbf{v}}{\|\mathbf{v}\|^2} \left(\Delta w_i - \frac{\Delta w \cdot \mathbf{v}}{2\|\mathbf{v}\|^2} v_i \right) z_i + \text{noise}$$

- This allows us to estimate $\frac{\partial E}{\partial w_i}$ and z_i for each weight w_i, using only local perturbations and globally broadcast values like ΔE and $\Delta w \cdot \mathbf{v}$.

Conclusion

Key Properties of $\mathcal{R}\{\}$ Technique:

- **Exact:** No approximations are made.
- **Numerically Accurate:** Maintains precision without significant loss.
- **Efficient:** Comparable computation cost to gradient calculation.
- **Flexible:** Compatible with all gradient calculation methods.
- **Robust:** Provides an unbiased estimate if the gradient calculation is unbiased.