ECE 449/590 – OOP and Machine Learning
Lecture 20 Regularization for Deep Learning

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

November 2, 2022

Reading Assignment

- This lecture: Deep Learning 7
- Next lecture: We'll move back to C++

Regularization

- Strategies to reduce the test error.
  - Possibly at the cost of increasing training error.
- Typical strategies
  - Add constraints to weights (need to solve constrained optimization problems).
  - Include additional terms in the cost function, working as "soft" constraints so that the optimization problem remains unconstrained.
  - Methods to encode specific kinds of prior knowledge.
  - Emsemble methods that combine alternative models.
- Model bias vs. variance
  - Underfitting models tend to have high bias.
  - Overfitting models tend to have high variance.
  - As we won't have access to the true models for complex learning problems, it is difficult to find models with right capacity to reduce overfitting.
  - Regularization helps to reduce variance of overfitting models so that they could be make use of.

Parameter Norm Penalties

- Limiting model capacity by adding a parameter norm penalty.

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

  - The hyperparameter $\alpha \in [0, \infty)$ controls the relative importance of the penalty to the loss function J.
  - May cause J to increase in order to decrease Ω.
- Usually for parameters of linear layers, only weights $\boldsymbol{w}$ need to be regularized.
  - In comparison to biases, weigths require more data to fit – from the last lecture we know that $\frac{\partial h_y}{\partial w_{i,j}} = x_i \frac{\partial f_y}{\partial v_j}$ while $\frac{\partial h_y}{\partial b_j} = \frac{\partial f_y}{\partial v_j}$.
- You may choose different α's for norms of weights at different layers.

L^2 Parameter Regularization

- Use the L^2 norm as the penalty: $\Omega(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||_2^2$.
  - Commonly known as weight decay.
- Let's ignore the impact of biases,

$$\begin{aligned}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) &= \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),\\ \nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) &= \alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})\end{aligned}$$

- For a step in gradient descent,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J) = (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J$$

  - Recall when there is no weight decay: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J$
  - So the weight vector is shrinked for such regularization.

L^2 Parameter Regularization (Cont.)

- Let $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$
  - Optimal weight vector when there is no weight decay.
- Expand J at $\boldsymbol{w}^*$ to the second order:

$$J(\boldsymbol{w}) \approx J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}_{\boldsymbol{w}} J(\boldsymbol{w}^*)(\boldsymbol{w} - \boldsymbol{w}^*)$$

  - Let $\boldsymbol{H}_{\boldsymbol{w}} J(\boldsymbol{w}^*) = \boldsymbol{Q}\Lambda\boldsymbol{Q}^\top$ be the eigendecomposition.
- (Aproximate) Gradients for weight decay:

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \boldsymbol{Q}\Lambda\boldsymbol{Q}^\top(\boldsymbol{w} - \boldsymbol{w}^*)$$

- We may compute $\tilde{\boldsymbol{w}}^* = \arg\min_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w})$ by solving $\nabla_{\boldsymbol{w}} \tilde{J} = 0$.

$$\tilde{\boldsymbol{w}}^* = (\alpha\boldsymbol{I} + \boldsymbol{Q}\Lambda\boldsymbol{Q}^\top)^{-1}\boldsymbol{Q}\Lambda\boldsymbol{Q}^\top\boldsymbol{w}^* = \boldsymbol{Q}(\Lambda + \alpha\boldsymbol{I})^{-1}\Lambda\boldsymbol{Q}^\top\boldsymbol{w}^*$$

- $\tilde{\boldsymbol{w}}^*$ shrinks in the space of eigenvectors $\boldsymbol{Q}$, especially along the directions where the eigenvalues are smaller relative to α.

Weight Decay as Constrained Optimization

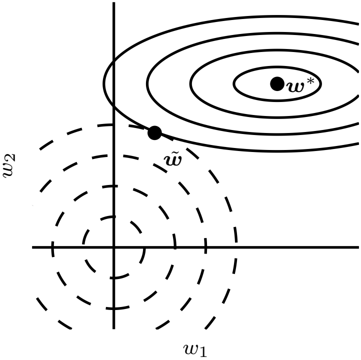


Figure 7.1

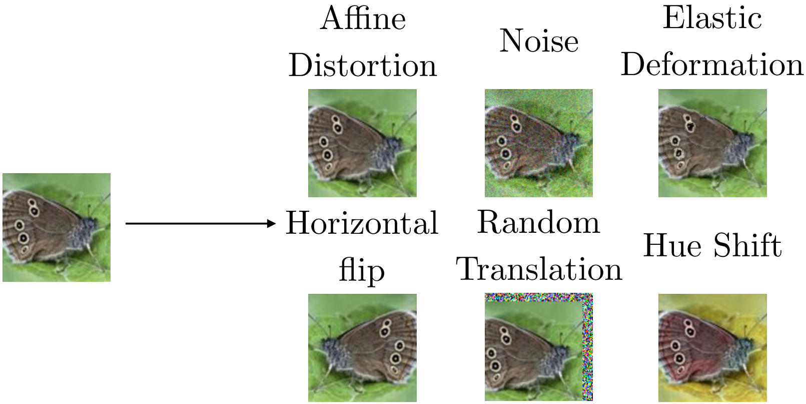
(Goodfellow 2016)

- A simple case where $\boldsymbol{Q} = \boldsymbol{I}$ and $\lambda_1 < \lambda_2$.
  - $J(w_1, w_2) = J(w_1^*, w_2^*) + \frac{\lambda_1}{2}(w_1 - w_1^*)^2 + \frac{\lambda_2}{2}(w_2 - w_2^*)^2$

Dataset Augmentation

- Train with more data with limited amount of data.
  - Need to create "fake" data for training that "like" the original training data.
  - Loss on original training data may increase.
- For classification, with original training example $(\boldsymbol{x}, y)$,
  - Transform $\boldsymbol{x}$ into $\boldsymbol{x}'$ and use $(\boldsymbol{x}', y)$.
  - E.g. for images, moving, rotating, scaling, or even mirroring are effective.
- Indeed, prior knowledge of data is incorporated during such transformations.
  - we are aware certain transformations CANNOT be applied if there are "b" and "d", or "6" and "9", etc.

Dataset Augmentation



(Goodfellow 2016)

Multi-Task Learning

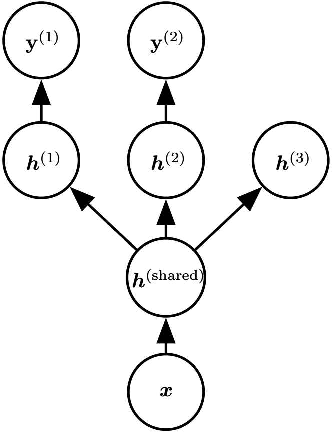


Figure 7.2

(Goodfellow 2016)

Early Stopping and Weight Decay

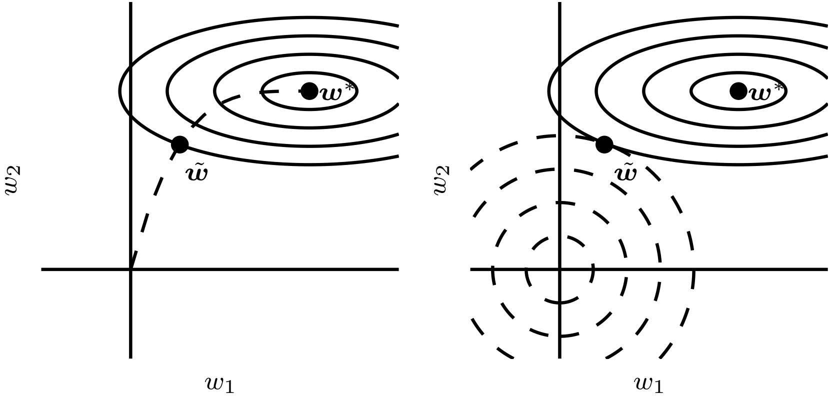


Figure 7.4

(Goodfellow 2016)

Bagging

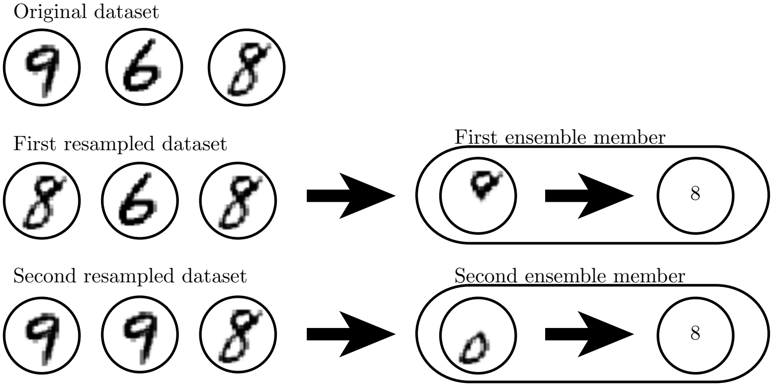


Figure 7.5

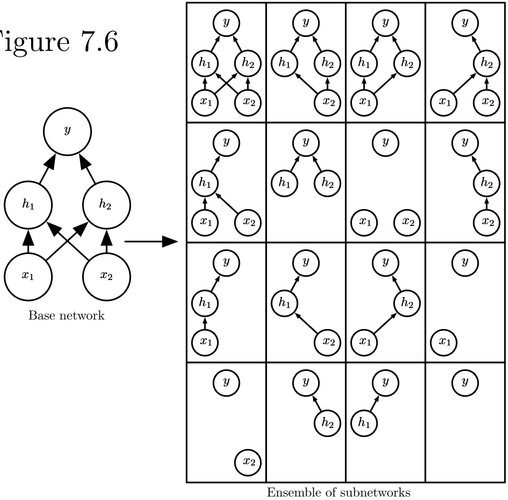
(Goodfellow 2016)

Dropout

- Widely used when training neural network models.
- Inputs to nodes are set to 0 according to a predefined probability p.
  - Applied at training time, per each SGD step, for both forward and back propagation.
  - Implementations usually scale up the inputs by $\frac{1}{1-p}$ so that the network can remain the same for inference where no dropout is applied.
- Why does it work?
  - Combine many emsembles like bagging while force emsembles to share weights.
  - The whole network is trained to work with noisy nodes.

Dropout

Figure 7.6



(Goodfellow 2016)

Summary

- Regularization helps machine learning models to achieve better test error.
- Typical regularization strategies.