Introduction to AI for postgraduate students

Lecture Note 7
Regularization

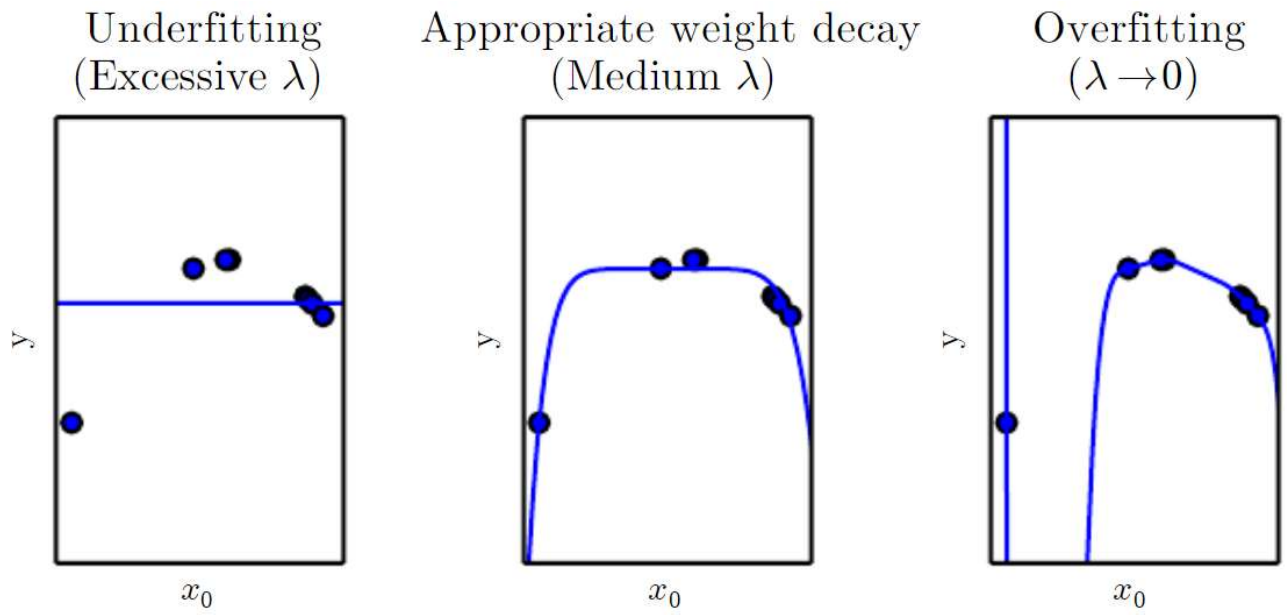
POSTECH

AiSLab

Overview

Regularization

- Strategies designed to reduce the test error, possibly at the expense of increased training error.
- Our focus is on regularization strategies for deep models or models that may be used as building blocks to form deep models.



Agenda

- **L^2 Parameter Regularization**
- L^1 Parameter Regularization
- Others

Parameter Norm Penalties

- limiting the capacity of models by adding a parameter norm penalty $\Omega(\boldsymbol{\theta})$ to the object function J:

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

Hyperparameter

- We leave the biases unregularized.
  - Each bias typically requires less data than the weights to fit accurately, because it controls a single variable.
  - Regularizing the bias parameters can introduce a significant amount of underfitting.
- We therefore use the vector $\boldsymbol{w}$ to indicate all the weights that should be affected by a norm penalty,
  - While the vector $\boldsymbol{\theta}$ denotes all the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.

L^2 Parameter Regularization

- Commonly known as weight decay
  - drives the weights closer to the origin by adding a regularization term $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2$

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

- The gradient descent update:

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

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon\left(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})\right)$$

$$\boldsymbol{w} \leftarrow (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

multiplicatively shrink the weight vector by a constant factor on each step, just before performing the usual gradient update

L^2 Parameter Regularization

- Quadratic approximation of to J around $\boldsymbol{w}^* = \arg\min J(\boldsymbol{w})$:

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

  - $\boldsymbol{H}$ is the Hessian matrix of J
  - no first-order term in this quadratic approximation, because $\boldsymbol{w}^*$ is defined to be a minimum, where the gradient vanishes.
- Minimum of the approximated $\hat{J}$ occurs when its gradient $\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = H(\boldsymbol{w} - \boldsymbol{w}^*) = \boldsymbol{0}$.
- Adding the weight decay, we get the minimum location $\tilde{w}$ of the regularized and approximated J:

Replaced by $\hat{J}$

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

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

Replaced by $\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w})$

$$\alpha\tilde{\boldsymbol{w}} + \boldsymbol{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) = 0$$

$$(\boldsymbol{H} + \alpha \boldsymbol{I})\tilde{\boldsymbol{w}} = \boldsymbol{H}\boldsymbol{w}^*$$

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

L^2 Parameter Regularization

- From $\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1} \boldsymbol{H} \boldsymbol{w}^*$, as α approaches 0, the regularized solution $\tilde{w}$ approaches the unregularized solution w^*.
- What happens as α grows?
- Because $\boldsymbol{H}$ is real and symmetric, we can decompose it into a diagonal matrix $\boldsymbol{\Lambda}$ and an orthonormal basis of eigenvectors $\boldsymbol{Q}$ such that $\boldsymbol{H} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^H$.
- Applying the decomposition, we get

$$\begin{aligned}
\tilde{\boldsymbol{w}} &= (\boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top + \alpha \boldsymbol{I})^{-1} \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^* \\
&= \left[\boldsymbol{Q} (\boldsymbol{\Lambda} + \alpha \boldsymbol{I}) \boldsymbol{Q}^\top \right]^{-1} \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^* \\
&= \boldsymbol{Q} (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^*.
\end{aligned}$$

L^2 Parameter Regularization

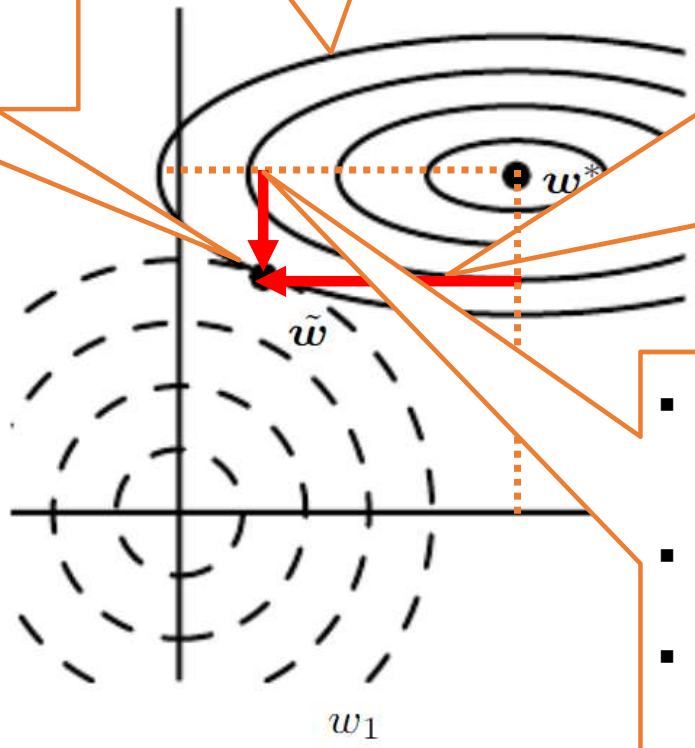
- From $\tilde{\boldsymbol{w}} = \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{w}^*$, we see that
  - the component of $\boldsymbol{w}^*$ that is aligned with the i-th eigenvector of $\boldsymbol{H}$ is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$.
  - Along the directions where the eigenvalues of $\boldsymbol{H}$ are relatively large, for example, where $\lambda_i \gg \alpha$, the effect of regularization is relatively small.
  - Components with $\lambda_i \ll \alpha$ will be shrunk to have nearly zero magnitude.

L^2 Parameter Regularization

At $\tilde{w}$, the two competing objectives reach an equilibrium.

J's contours

L^2-regularizer's contours



- In the first dimension, the eigenvalue of the Hessian of J is small.
- "Components with $\lambda_i \ll \alpha$ will be shrunk to have nearly zero magnitude."
- The regularizer pulls w_1 close to zero.
- The objective function does not increase much when moving horizontally away from $\boldsymbol{w}^*$.

- In the second dimension, the objective function is very sensitive to movements away from w^*.
- The corresponding eigenvalue is large, indicating high curvature.
- "If $\lambda_i \gg \alpha$, the effect of regularization is relatively small."
- weight decay affects the position of w_2 relatively little.

L^2 Parameter Regularization

- We can apply the L^2 parameter regularization to ML with the cost function of the sum of squared errors.

	Original	L^2 parameter regularized
Cost function	$(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^\top(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}).$	$(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^\top(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})+\frac{1}{2}\alpha\boldsymbol{w}^\top\boldsymbol{w}.$
solution	$\boldsymbol{w}=(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$	$\boldsymbol{w}=(\boldsymbol{X}^\top\boldsymbol{X}+\alpha\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}.$

- It represent the covariance matrix $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$, which changes to $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\alpha\boldsymbol{I})$ in the regularized case.
- We can see that L^2 regularization causes the learning algorithm to "perceive" the input $\boldsymbol{X}$ as having higher variance, which makes it shrink the weights on features whose covariance with the output target is low compared to this added variance.

Agenda

- L^2 Parameter Regularization
- **L^1 Parameter Regularization**
- Others

L^1 Parameter Regularization

- L^1 regularization on the model parameter $\boldsymbol{w}$:

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|$$

- Regularized objective function (without a bias):

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha||\boldsymbol{w}||_1 + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

- The corresponding gradient is given by:

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha \text{sign}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{w})$$

L^1 Parameter Regularization

L^2 norm regularization

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$$

L^1 norm regularization

$$\nabla_w \tilde{J}(w; X, y) = \alpha \text{sign}(w) + \nabla_w J(X, y; w)$$

- regularization contribution to the gradient no longer scales linearly with each w_i.
  - instead it is a constant factor with a sign equal to sign(w_i).

- Recall that the quadratic approximation of the cost function and its gradient are given by:

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) \quad \Rightarrow \quad \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = H(\boldsymbol{w} - \boldsymbol{w}^*)$$

- We further apply the assumption of a diagonal Hessian:

$$\boldsymbol{H} = \text{diag}([H_{1,1}, \ldots, H_{n,n}]), \text{ where each } H_{i,i} > 0.$$

  - This assumption holds if the data for the linear regression problem has been preprocessed to remove all correlation between the input features, which may be accomplished using PCA.

L^1 Parameter Regularization

- Then, the approximated quadratic objective function is given by:

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

$$\boldsymbol{H} = \operatorname{diag}([H_{1,1},\ldots,H_{n,n}])$$

$$\hat{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{w}^*;\boldsymbol{X},\boldsymbol{y}) + \sum_i \left[\frac{1}{2}H_{i,i}(\boldsymbol{w}_i-\boldsymbol{w}_i^*)^2 + \alpha|w_i|\right].$$

Minimizing solution

$$w_i = \operatorname{sign}(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$$

- Assuming $w_i^* > 0$ for all i:
  - $w_i^* \leq \frac{\alpha}{H_{i,i}}$: $w_i = 0$, because α is relatively very large, and thus J is overwhelmed by the regularization term.
  - $w_i^* > \frac{\alpha}{H_{i,i}}$: $w_i = |w_i^*| - \frac{\alpha}{H_{i,i}}$, because α is not that large, so the regularization only shits the solution a little bit.

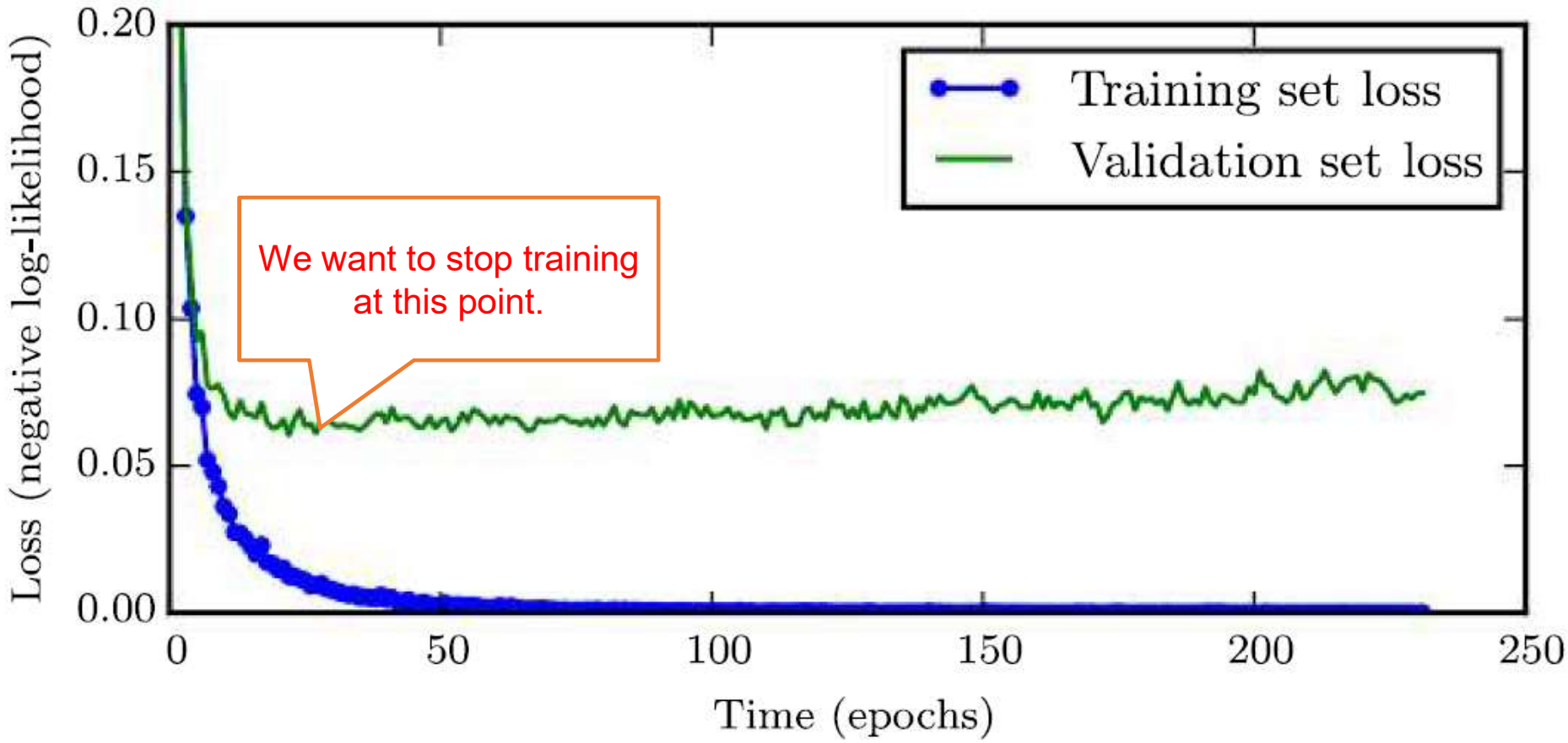
L^1 Parameter Regularization

- Comparison to L^2 regularization
  - L^1 regularization results in a solution that is more sparse, i.e., some parameters have an optimal value of zero.
  - If we apply the same assumption of the diagonal $\boldsymbol{H}$, the L^2 regularization gives us the solution $\tilde{w}_i = \frac{H_{i,i}}{H_{i,i}+\alpha} w_i^*$. cf) $w_i = \text{sign}(w_i^*) \max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$ for L^1 regularization
  - This sparsity is helpful when we want to extract features from inputs.

Agenda

- L^2 Parameter Regularization
- L^1 Parameter Regularization
- **Others**

Early Stopping

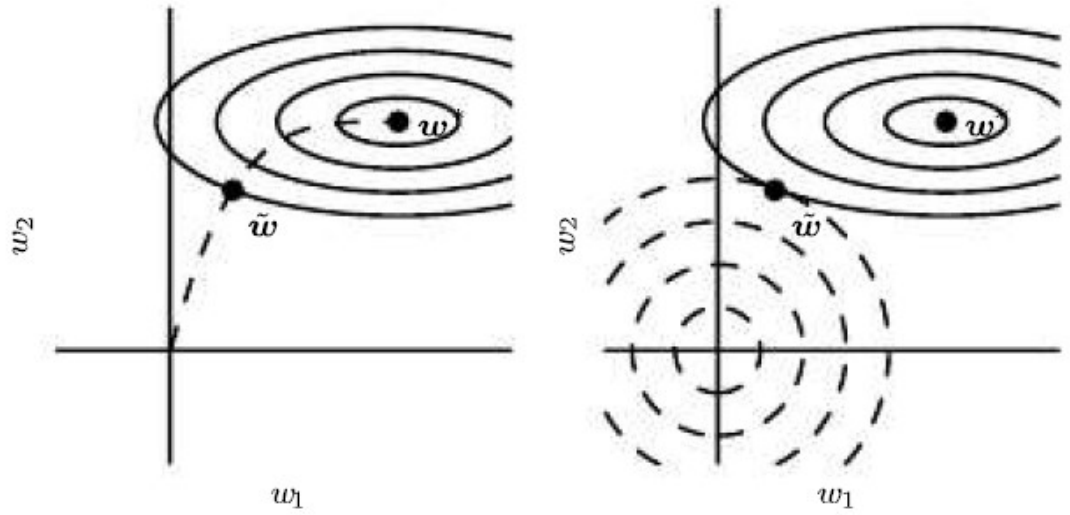


Early Stopping

Algorithm 7.1 The early stopping meta-algorithm for determining the best amount of time to train. This meta-algorithm is a general strategy that works well with a variety of training algorithms and ways of quantifying error on the validation set.

Let n be the number of steps between evaluations.
Let p be the "patience," the number of times to observe worsening validation set error before giving up.
Let $\boldsymbol{\theta}_o$ be the initial parameters.
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_o$
$i \leftarrow 0$
$j \leftarrow 0$
$v \leftarrow \infty$
$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
$i^* \leftarrow i$
while $j < p$ **do**
  Update $\boldsymbol{\theta}$ by running the training algorithm for n steps.
  $i \leftarrow i + n$
  $v' \leftarrow \text{ValidationSetError}(\boldsymbol{\theta})$
  if $v' < v$ **then**
    $j \leftarrow 0$
    $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
    $i^* \leftarrow i$
    $v \leftarrow v'$
  else
    $j \leftarrow j + 1$
  end if
end while
Best parameters are $\boldsymbol{\theta}^*$, best number of training steps is i^*.

Early Stopping



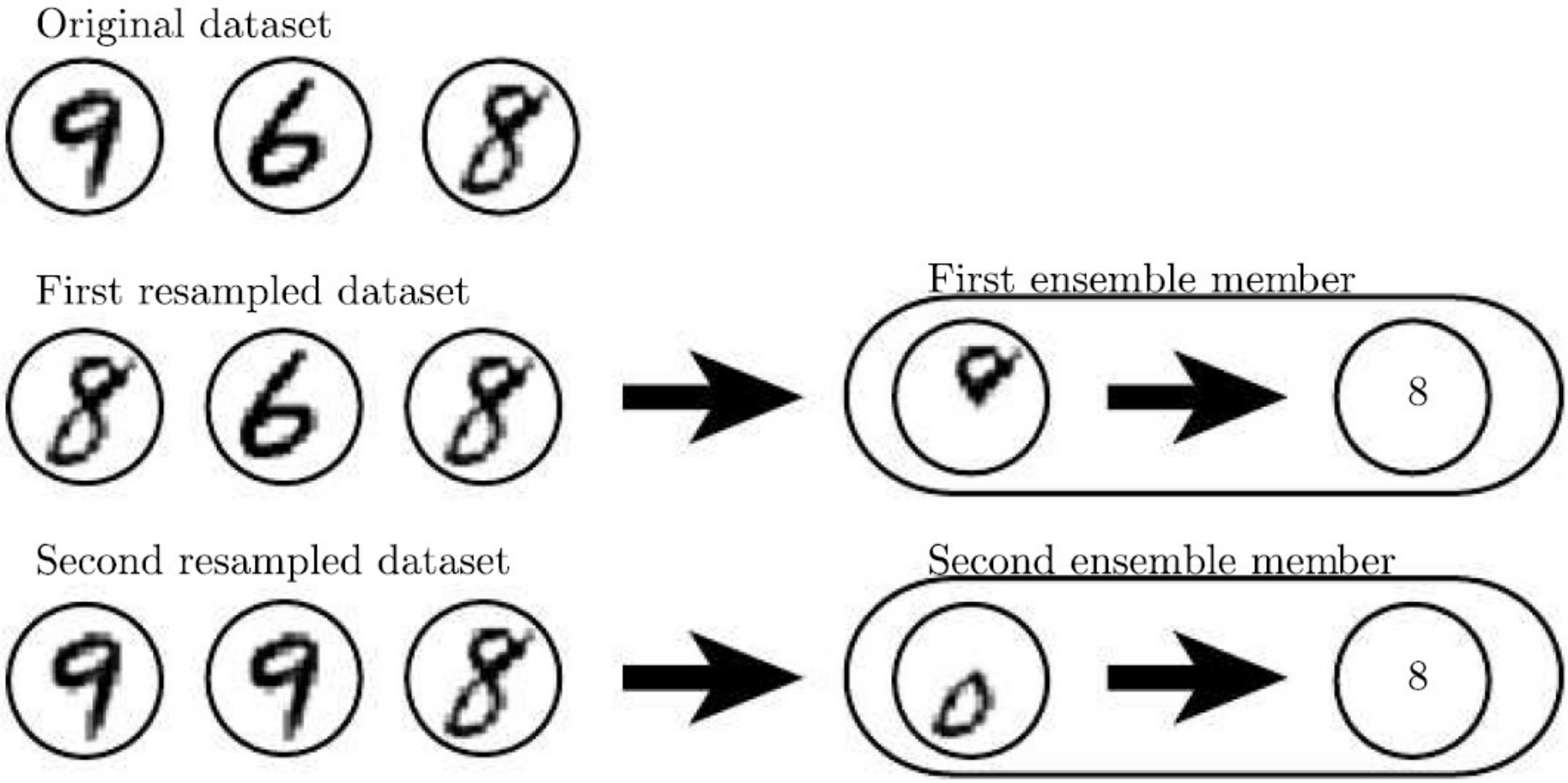
Bagging

- Bagging (Bootstrap aggregating)
  - Reduces generalization error by combining several models.
  - Trains several different models separately, then have all the models vote on the output for test examples.
  - The technique works because different models will usually not make all the same errors on the test set.

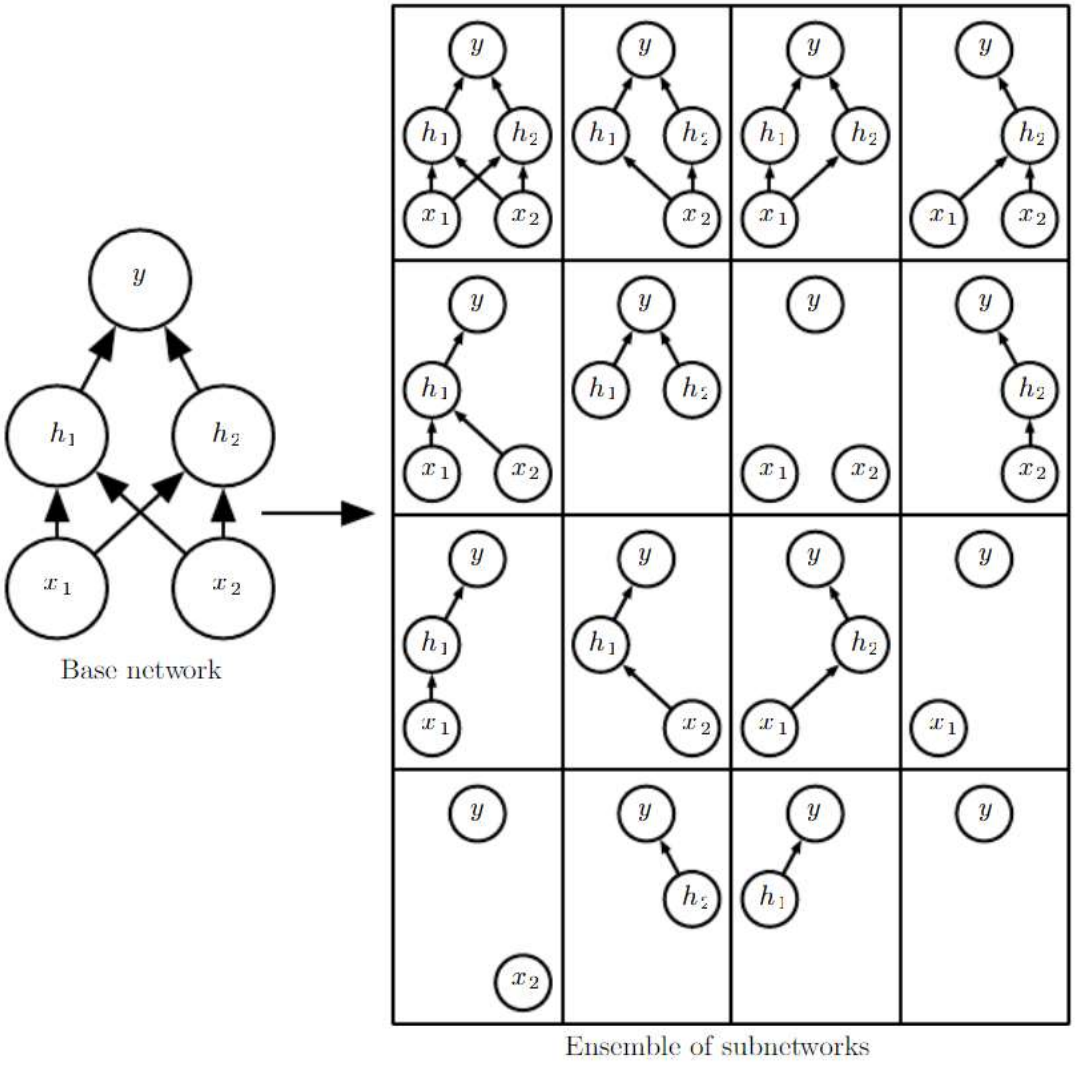
- Consider k regression models, each of which makes an error ϵ_i on each sample, with the errors drawn from a zero-mean multivariate normal distribution with $E[\epsilon^2] = v$ and $E[\epsilon_i \epsilon_j] = c$.
- Then the error made by the average prediction of all the ensemble models is $\frac{1}{k}\sum_i \epsilon_i$.
- The expected squared error of the ensemble predictor is:

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{j \neq i} \epsilon_i \epsilon_j\right)\right],$$
$$= \frac{1}{k} v + \frac{k-1}{k} c.$$

Bagging


Original dataset
First resampled dataset
First ensemble member
8
Second resampled dataset
Second ensemble member
8

Dropout



- Difference from the bagging
  - Models can share parameters, with each model inheriting a different subset of parameters from the parent neural network.
    - ✓ This parameter sharing makes it possible to represent an exponential number of models.
  - Most models are not explicitly trained at all
    - ✓ A tiny fraction of the possible subnetworks are each trained for a single step
    - ✓ The parameter sharing causes the remaining subnetworks to arrive at good settings of the parameters.

Dropout

- $\boldsymbol{\mu}$: specifics which units to include (0.5 for a hidden unit to be included, 0.8 for an input unit)