Regularization for Deep Learning

ADVANCED ARTIFICIAL INTELLIGENCE

JUCHEOL MOON

1

Regularization

- Regularization is any modification we make to a learning algorithm that is intended to reduce
  - its generalization(test) error
  - but not its training error
- Regularization strategies
  - put extra constraints on a machine learning model
  - add extra terms in the objective function

2

2

Parameter Norm Penalties

- Limiting the capacity of models by adding a parameter norm penalty $\Omega(\theta)$ to the objective function J.
  - $\tilde{J}(\vec{\theta}; \vec{X}, \vec{y}) =$ $J(\vec{\theta}; \vec{X}, \vec{y}) + \alpha \Omega(\vec{\theta})$
  - where $\alpha \in [0, \infty)$ is a hyperparameter.
  - $\alpha = 0$ results in no regularization,
  - Large α correspond to more regularization.
  - It is sometimes desirable to use a separate penalty with a different α coefficient for each layer of the network.

3

3

L^2 Parameter Regularization

- Also known as weight decay and ridge regularization
- $\tilde{J}(\vec{w}; \vec{X}, \vec{y}) =$ $J(\vec{w}; \vec{X}, \vec{y}) + \underbrace{\frac{\alpha}{2} \vec{w}^T \vec{w}}$
- $\nabla_{\vec{w}} \tilde{J}(\vec{w}; \vec{X}, \vec{y}) =$ $\nabla_{\vec{w}} J(\vec{w}; \vec{X}, \vec{y}) + \alpha \vec{w}$
- $\vec{w} \leftarrow$ $\vec{w} - \varepsilon\left(\alpha \vec{w} + \nabla_{\vec{w}} J(\vec{w}; \vec{X}, \vec{y})\right)$
- $\vec{w} \leftarrow$ $\underline{\vec{w} - \varepsilon \alpha \vec{w}} - \varepsilon \nabla_{\vec{w}} J(\vec{w}; \vec{X}, \vec{y}) = \underbrace{(1 - \varepsilon\alpha)\vec{w}} - \varepsilon \nabla_{\vec{w}} J(\vec{w}; \vec{X}, \vec{y})$
- shrink the weight vector by a constant factor on each step

4

4

Weight Decay as Constrained Optimization

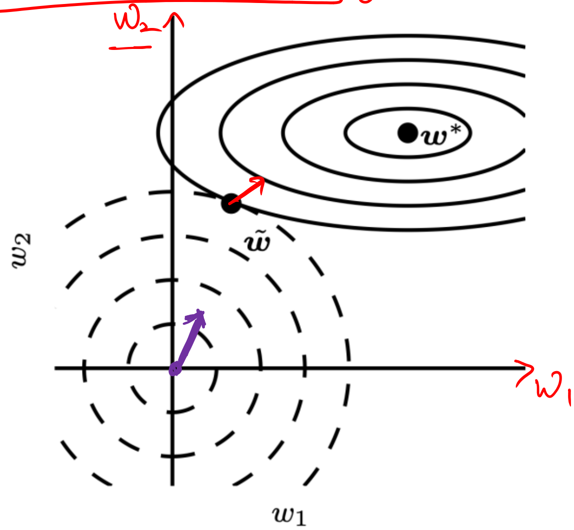
- $\vec{w} \leftarrow \boxed{(1-\epsilon\alpha)\vec{w}}\ \underbrace{\boxed{-\epsilon\nabla_{\vec{w}}J(\vec{w};\vec{X},\vec{y})}}_{\vec{g}}$

$0 < \varepsilon \ll 1$

$0 < \alpha \ll 1$

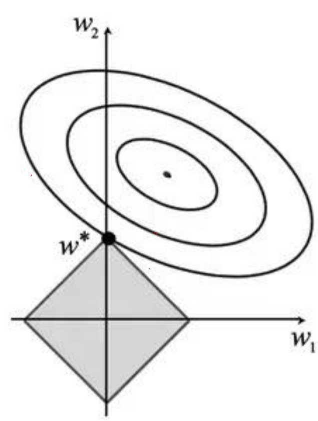
$0 < \varepsilon\alpha \ll 1$

$1 - \varepsilon\alpha < 1$



L^1 Parameter Regularization

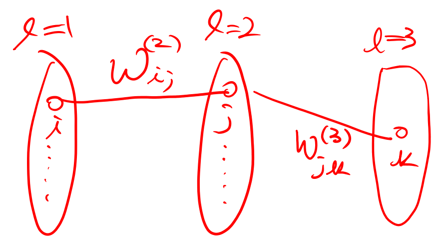
- Also known as LASSO regularization
- $\tilde{J}(\vec{w};\vec{X},\vec{y}) = J(\vec{w};\vec{X},\vec{y}) + \alpha\|\vec{w}\|_1$
- $\nabla_{\vec{w}}\tilde{J}(\vec{w};\vec{X},\vec{y}) = \nabla_{\vec{w}}J(\vec{w};\vec{X},\vec{y}) + \alpha\,\mathrm{Sign}(\vec{w})$
- $\vec{w} \leftarrow \vec{w} - \epsilon\alpha sign(\vec{w})\ \underbrace{\boxed{-\epsilon\nabla_{\vec{w}}J(w;\vec{X},\vec{y})}}_{\vec{g}}$



L^2 Regularization in Neural Network

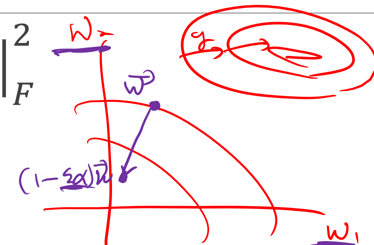
- $\underbrace{J(\vec{W},\vec{b}) = \frac{1}{m}\sum_i^m L\left(\hat{y}^{(i)}, y^{(i)}\right)}_{J} + \underbrace{\frac{\alpha}{2}\sum_l^L \left\|\vec{W}^{(l)}\right\|_F^2}_{\Omega}$
- Frobenius norm
  - $\left\|\vec{W}^{(l)}\right\|_F^2 = \sum_i \sum_j \left(w_{ij}^{(l)}\right)^2$

layer

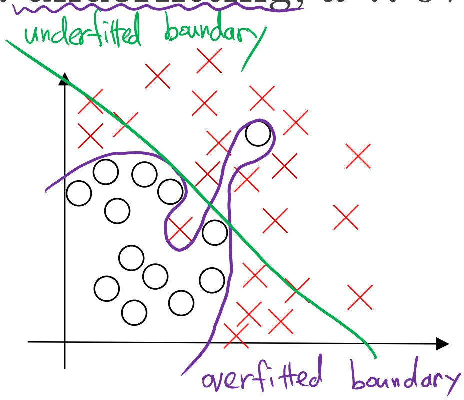


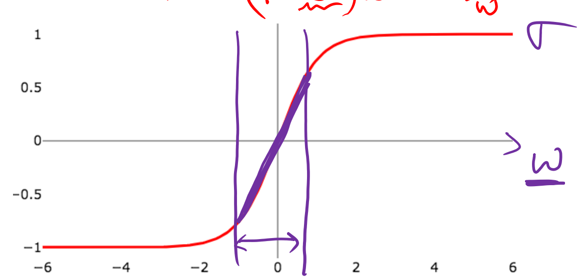
L^2 Regularization in Neural Network

- $J(\vec{W},\vec{b}) = \frac{1}{m}\sum_i^m L\left(\hat{y}^{(i)}, y^{(i)}\right) + \frac{\alpha}{2}\sum_l^L \left\|\vec{W}^{(l)}\right\|_F^2$
- In deep neural network
  - $\alpha \uparrow$: underfitting, $\alpha \downarrow$: overfitting



$\vec{w} \leftarrow (1-\epsilon\alpha)\vec{w} - \epsilon D_{\vec{w}} J$

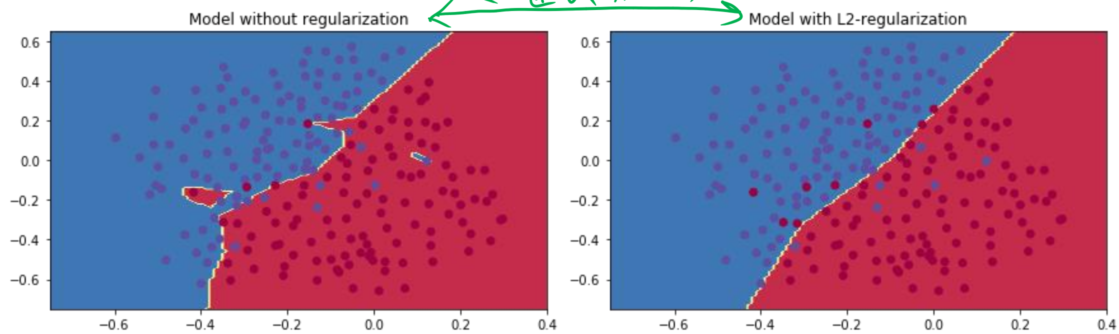




L^2 Regularization in Neural Network

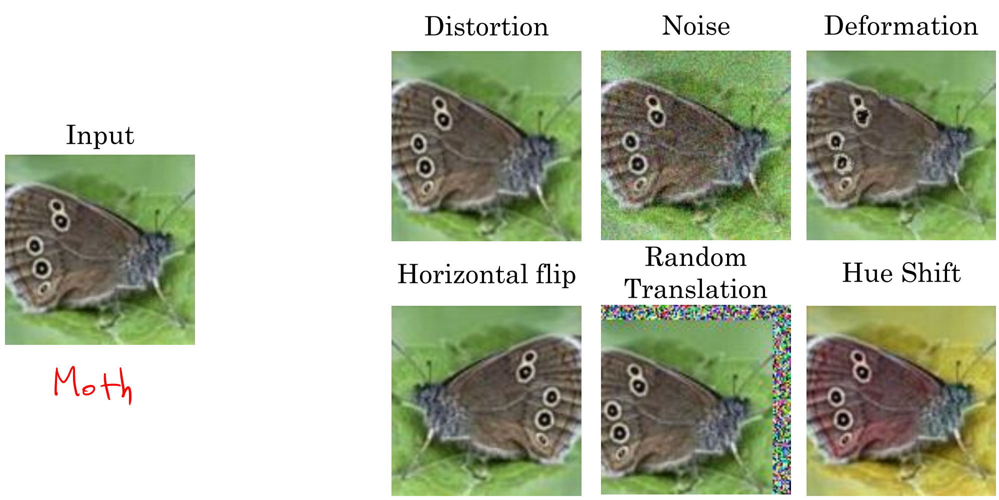
- $J(\vec{W}, \vec{b}) = \frac{1}{m}\sum_i^m L(\hat{y}^{(i)}, y^{(i)}) + \frac{\alpha}{2}\sum_l^L \left\|\vec{W}^{(l)}\right\|_F^2$
- In deep neural network
  - $\alpha \uparrow$: underfitting, $\alpha \downarrow$: overfitting

$\Leftarrow \Downarrow \alpha \Uparrow \Rightarrow$



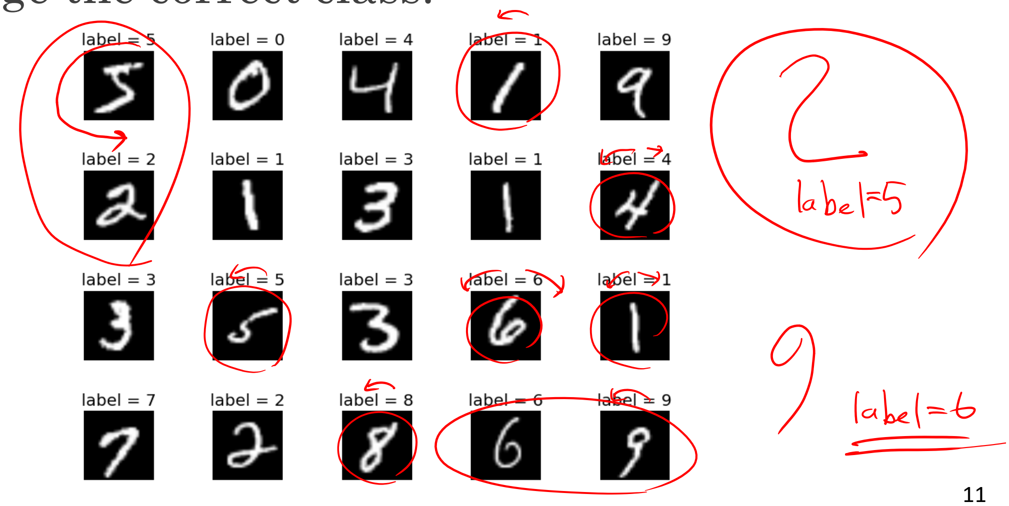
Dataset Augmentation

- The best way to make a machine learning model generalize better is to train it on more data.



Dataset Augmentation

- One must be careful not to apply transformations that would change the correct class.

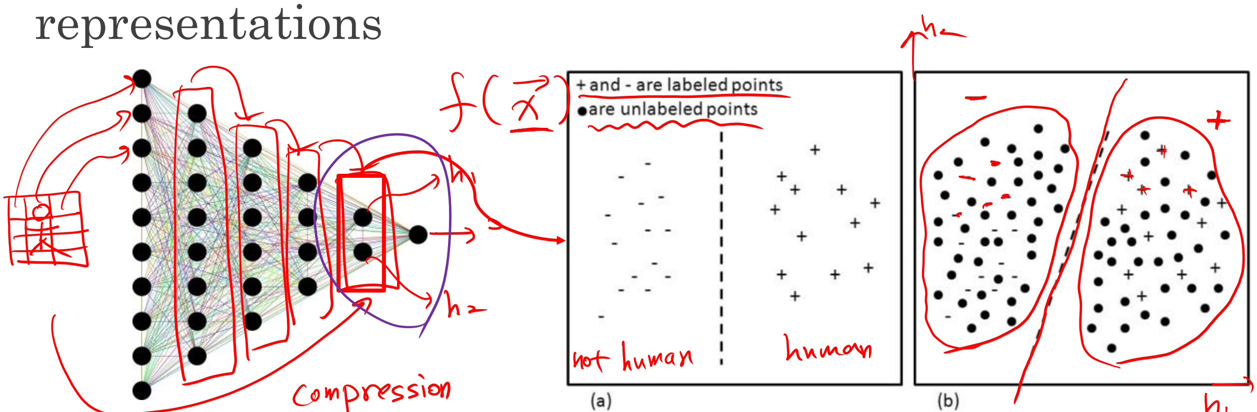


11

11

Semi-Supervised Learning

- Semi-supervised learning usually refers to learning a representation $h = f(x)$
  - Examples from the same class have similar representations

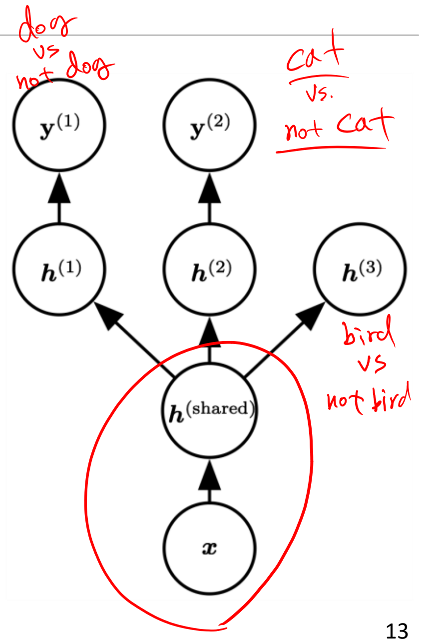


12

12

Multi-Task Learning

- The model can generally be divided into two kinds of parts and associated parameters:
  - Task-specific parameters (which only benefit from the examples of their task to achieve good generalization)
  - Generic parameters, shared across all the tasks (which benefit from the pooled data of all the tasks).

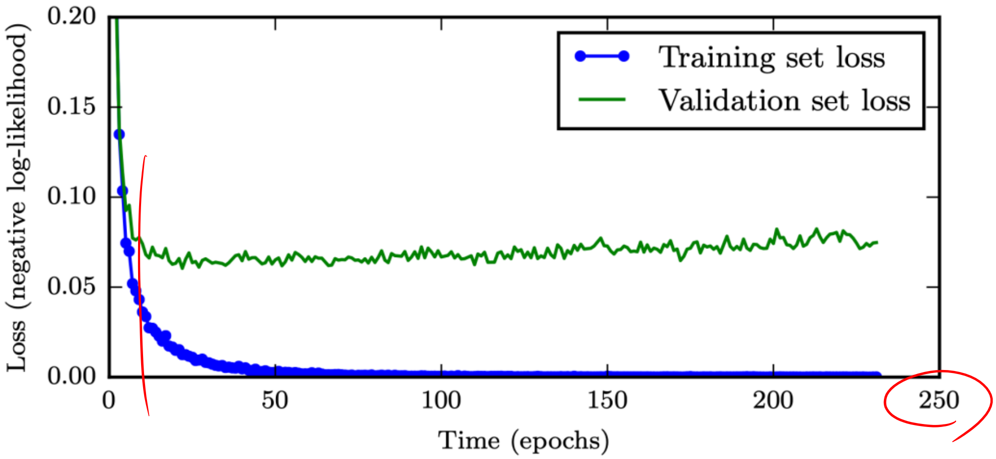


13

13

Early Stopping

- We often observe that training error decreases steadily over time, but validation set error begins to rise again.



14

14

Early Stopping version 1

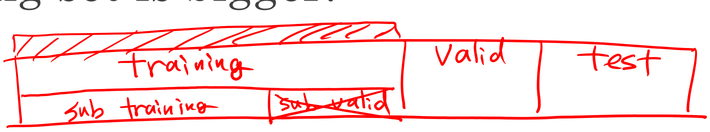
- Every time the error on the validation set improves, we store a copy of the model parameters.
- When the training algorithm terminates, we return these parameters, rather than the latest parameters.
- An additional cost to early stopping is the need to maintain a copy of the best parameters.
  - This cost is generally negligible.
- Early stopping requires a validation set, which means some training data is not fed to the model.

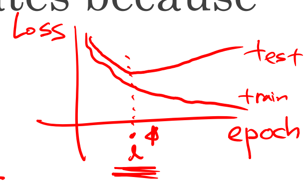
15

15

Early Stopping version 2

- Perform extra training after (the initial training with early stopping has completed.)
- In this second training pass, we train for the same number of steps as the early stopping procedure determined was optimal in the first pass.
- On the second round of training, each pass through the dataset will require more parameter updates because the training set is bigger.





16

16

Early Stopping version 2

Algorithm 7.2 A meta-algorithm for using early stopping to determine how long to train, then retraining on all the data.

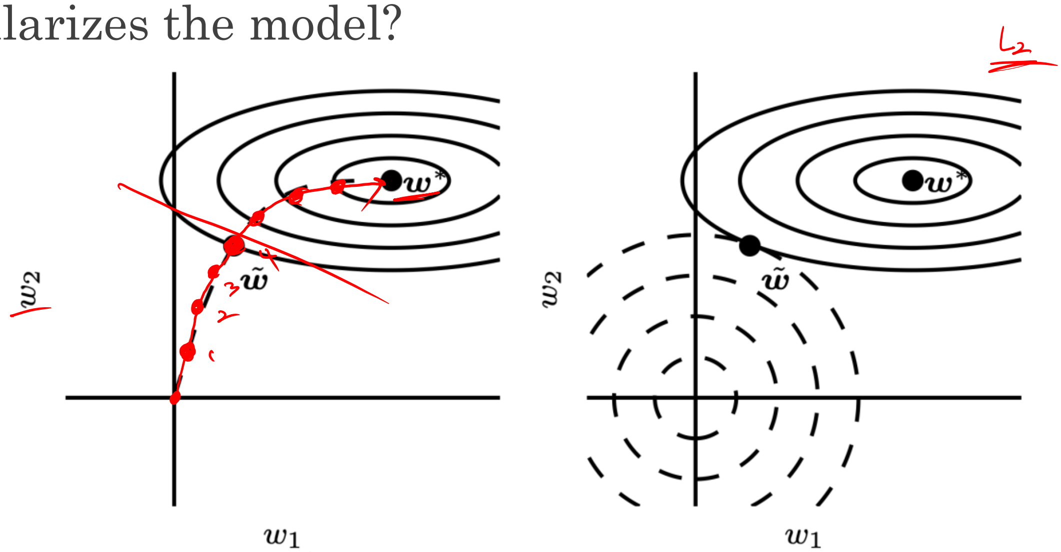
Let $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ be the training set.
Split $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ into $(\boldsymbol{X}^{(\text{subtrain})}, \boldsymbol{X}^{(\text{valid})})$ and $(\boldsymbol{y}^{(\text{subtrain})}, \boldsymbol{y}^{(\text{valid})})$ respectively.
Run early stopping (algorithm 7.1) starting from random $\boldsymbol{\theta}$ using $\boldsymbol{X}^{(\text{subtrain})}$ and $\boldsymbol{y}^{(\text{subtrain})}$ for training data and $\boldsymbol{X}^{(\text{valid})}$ and $\boldsymbol{y}^{(\text{valid})}$ for validation data. This returns i^*, the optimal number of steps.
Set $\boldsymbol{\theta}$ to random values again.
Train on $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ for i^* steps.

17

17

Early Stopping

- What is the actual mechanism by which early stopping regularizes the model?

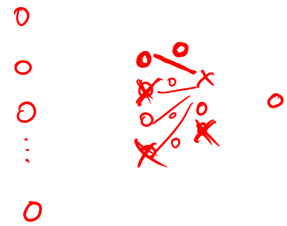


18

18

Sparse Representations

- Meaning of L^1 (or L^2) regularization?
  - Many of the parameters become zero (or close to zero)
- $\hat{y} = f(h) = f(f(x))$
- Representational sparsity describes a representation where many of the elements of the representation are zero (or close to zero)
- $\tilde{J}(\theta; X, y) = J + \alpha \Omega(h)$
  - For example, $\Omega(h) = \|h\|_1$



19

19

Ensemble Methods

- Consider for example a set of k regression models.
  - Suppose that each model makes an error ϵ_i on each example
  - with variances $\mathbb{E}[\epsilon_i^2] = v$, covariances $\mathbb{E}[\epsilon_i \epsilon_j] = c$
  - The error made by the average prediction: $\frac{1}{k}\sum_i \epsilon_i$
- The expected squared error
  - $\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2} E\left[\left(\sum_i \varepsilon_i\right)^2\right] = \frac{1}{k^2} E\left[\sum_i \left(\varepsilon_i^2 + \sum_{i \neq j} \varepsilon_i \varepsilon_j\right)\right]$

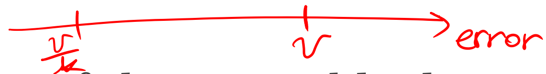
$$= \frac{1}{k^2}\left(\sum_i^k E[\varepsilon_i^2] + \sum_i \sum_{i\neq j} E[\varepsilon_i \varepsilon_j]\right) = \frac{1}{k} v + \frac{1}{k^2} k(k-1) c$$

20

20

Ensemble Methods

- The expected squared error
  - $\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \underbrace{\frac{1}{k}v + \frac{k-1}{k}c}$
- When the errors are perfectly correlated: c $= v$
  - $\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k}\left(\cancel{v} + (k-1)v\right) = \underline{v}$
- When the errors are perfectly uncorrelated: $\underline{c = 0}$
  - $\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{v}{k}$



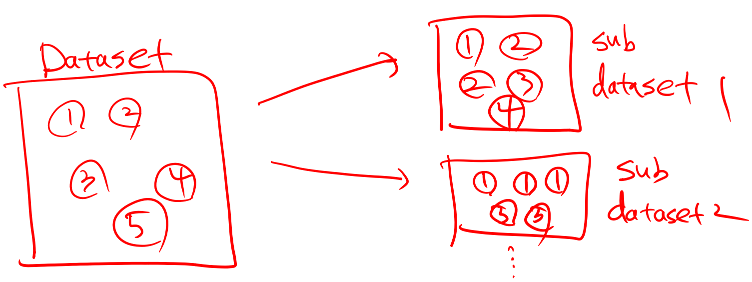
- The expected squared error of the ensemble decreases linearly with the ensemble size.

21

21

Bagging

- Bagging involves constructing k different datasets.
  - Each dataset has the same number of examples as the original dataset
  - but each dataset is constructed by sampling with replacement from the original dataset

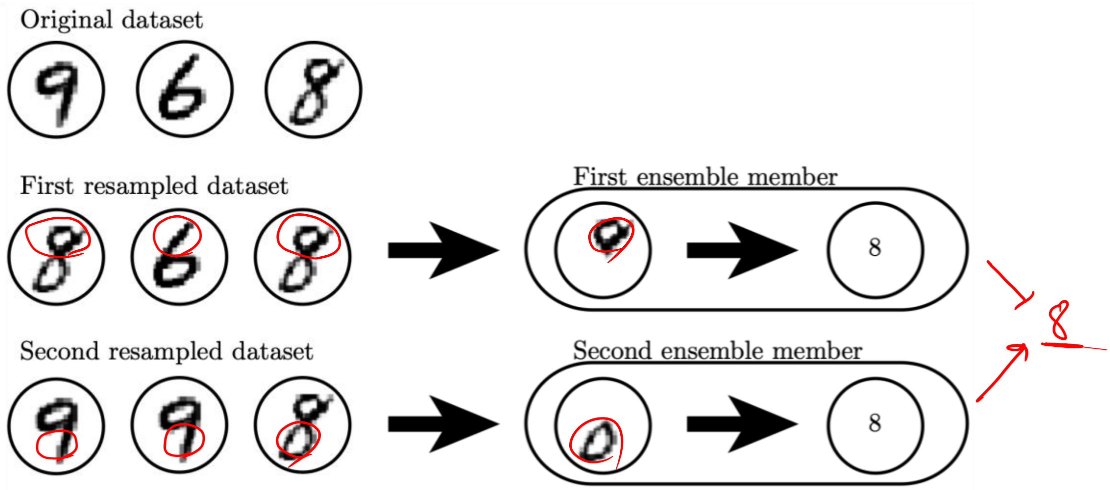


22

22

Bagging

- How bagging works



23

23

Ensemble in Neural networks

- Neural networks reach a wide enough variety of solution points that they can often benefit from model averaging even if all of the models are trained on the **same dataset**.
  - differences in random initialization
  - random selection of minibatches,
  - differences in hyperparameters

24

24

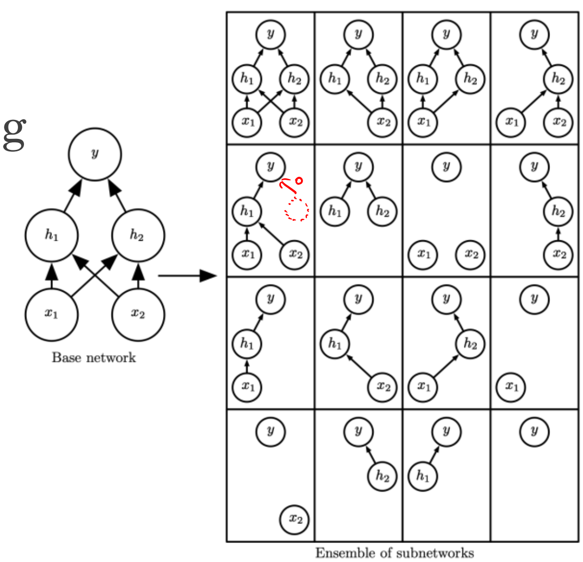
Dropout

- Bagging involves training multiple models, and evaluating multiple models on each test example.
  - impractical when each model is a large neural network, since training and evaluating such networks is costly in terms of runtime and memory
- Dropout provides an inexpensive approximation to training and evaluating a bagged ensemble of exponentially many neural networks

25

Dropout

- Dropout trains the ensemble consisting of all sub-networks that can be formed by removing non-output units from an underlying base network.
- In most modern neural networks, we can effectively remove a unit from a network by multiplying its output value by zero



26

Dropout

- Learn with bagging
  - we define k different models
  - construct k different datasets by sampling from the training set with replacement
  - then train model i on dataset i.
- Dropout aims to approximate this process
  - Each time we load an example into a minibatch,
  - we randomly sample a different binary mask to apply to all of the input and hidden units in the network.
  - The mask for each unit is sampled independently.
  - The probability of sampling a mask value of one is a **hyperparameter** fixed before training begins

27

Dropout

- The models share parameters, with each model inheriting a different subset of parameters from the parent neural network.
- The models can have different dropout probabilities for the layers.
- Predictions at test phase,
  - No dropout! (base network)
- Significant advantage of dropout is that it does not significantly limit the type of model or training procedure that can be used.

28