THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

CS 4364/6364

Machine Learning

Fall Semester 10/5/2023
Lecture 13.
Regularization

John Sipple
jsipple@gwu.edu

Announcements

Midterm (20%): Tuesday, 10/10 9:35, this room (through Regularization, L1-L13)

Open books, notes, laptop, Internet. Collaboration or use of LLMs prohibited.

Presentation day (12/7) - Here or Google Reston

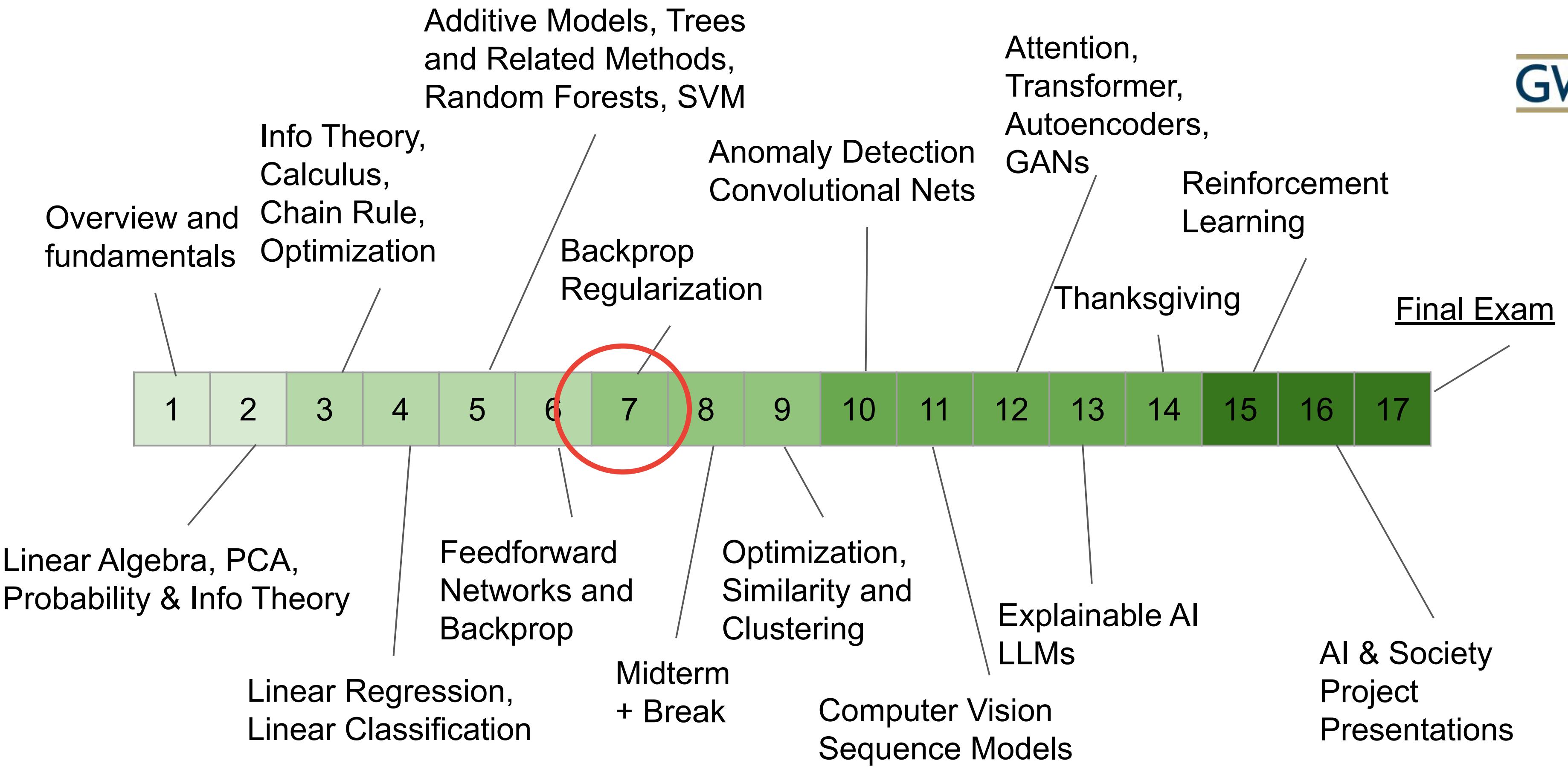
Voting on HW6 - only 7 votes: current lead is Autoencoder/GANs

Great return on HW2!

Updated Zoom policy:

- By exception only, notify me at least 24 hrs in advance via Slack
- Must be an exceptional circumstance

After the exam, the pace will significantly increase, but will be more interesting


Overview and fundamentals
Linear Algebra, PCA, Probability & Info Theory
Info Theory, Calculus, Chain Rule, Optimization
Linear Regression, Linear Classification
Additive Models, Trees and Related Methods, Random Forests, SVM
Feedforward Networks and Backprop
Backprop Regularization
Midterm + Break
Optimization, Similarity and Clustering
Anomaly Detection Convolutional Nets
Computer Vision Sequence Models
Attention, Transformer, Autoencoders, GANs
Explainable AI LLMs
Thanksgiving
Reinforcement Learning
AI & Society Project Presentations
Final Exam
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

Definition

“ **Regularization**: Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. “

Regularization Tradoff

Increase Bias for Decreased Variance

Parameter Norm Penalties

Regularized objective function:

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

- Standard objective function J
- Norm penalty term Ω
- Regularization parameter $\alpha \in [0, \infty)$

L2 Regularization

L2 Regularization

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

Parameter Gradient:

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

Gradient step update, size ϵ:

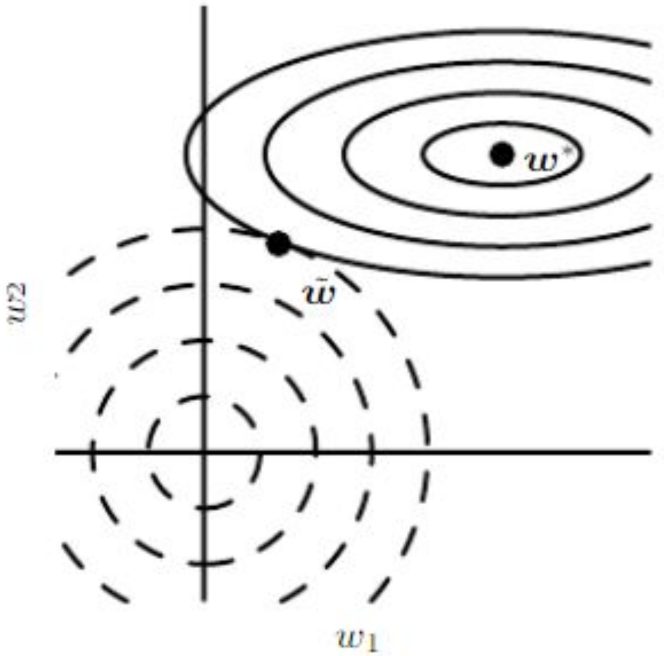
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}))$$

Rearranging:

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

Shrinks the weights proportionally by $\epsilon\alpha$

L2 Regularization



L1 Regularization

Penalty Term

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

Regularized Objective Function

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

And the corresponding gradient:

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

Shrinks the parameters $\boldsymbol{w}$ by a fixed amount $\epsilon\alpha$

Norm Penalties and Constrained Optimization

We've seen the application of constrained optimization in the lecture on SVM using Lagrange function.

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

If we wanted to constraint $\Omega(\boldsymbol{\theta})$ to be less than some constant k:

$$\mathcal{L}(\boldsymbol{\theta}, \alpha; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha(\Omega(\boldsymbol{\theta}) - k)$$

Many options for $\Omega(\boldsymbol{\theta})$:

- Frobenius Norm of the Weight Matrix of each layer
- Or constraining the norm of the columns of the weight matrix

where weight matrix at layer l with columns $\boldsymbol{h}_{out}^{(l-1)}$, and rows $\boldsymbol{h}_{in}^{(l)}$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \max_{\alpha, \alpha \geq 0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha})$$

Regularization and Under-Constrained Problems

Many linear models in ML (regression, PCA, etc.) rely on inverting $\boldsymbol{X}^\top \boldsymbol{X}$

If any two dimensions have too little variance, $\boldsymbol{X}^\top \boldsymbol{X}$ is singular and cannot be inverted

The fix: $\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I}$ is guaranteed to be invertible

Revisit the Moore-Penrose pseudoinverse:

$$\boldsymbol{X}^{+} = \lim_{\alpha \to 0} (\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^\top$$

Enables Linear Regression with *weight decay*

Dataset Augmentation

More data, generates better models, in general

With limited data, we might add new data via transforming $(\boldsymbol{x}, y)$ in some way:

- Rotation, Translation, Scale (works well for image data)
- Random noise works surprisingly well with neural networks

Noise Robustness

In addition to noise applied to input data noise can be added to the model weights

Used in Recurrent Networks

Reflects uncertainty of the layer

Label Smoothing compensates for label errors

- assigns y values between 0 and 1, rather than hard 0, 1

Semi-Supervised Learning

Often labeled data is hard to get in large quantities, and unlabeled data is available

Semi-Supervised Learning Combines Unsupervised Techniques to Augment Datasets:

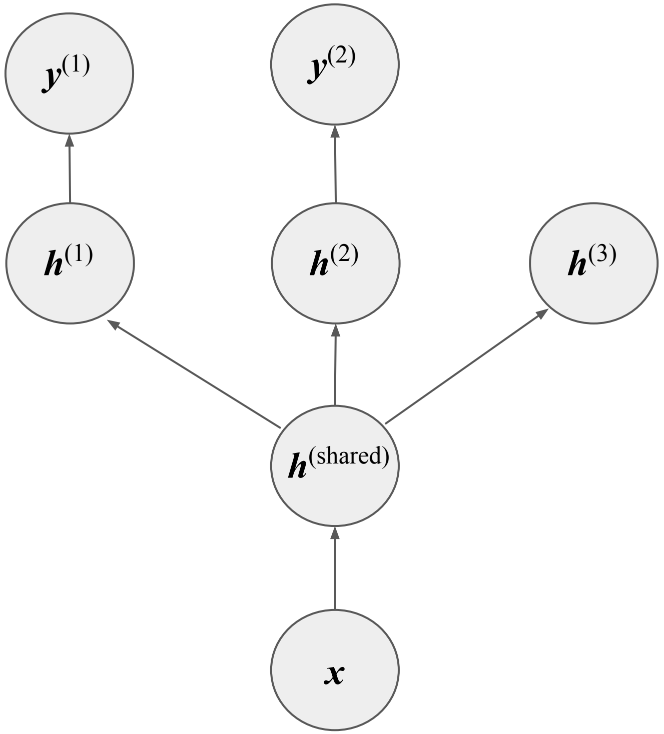
- Clustering
- Dimensionality Reduction (PCA)

Multitask Learning

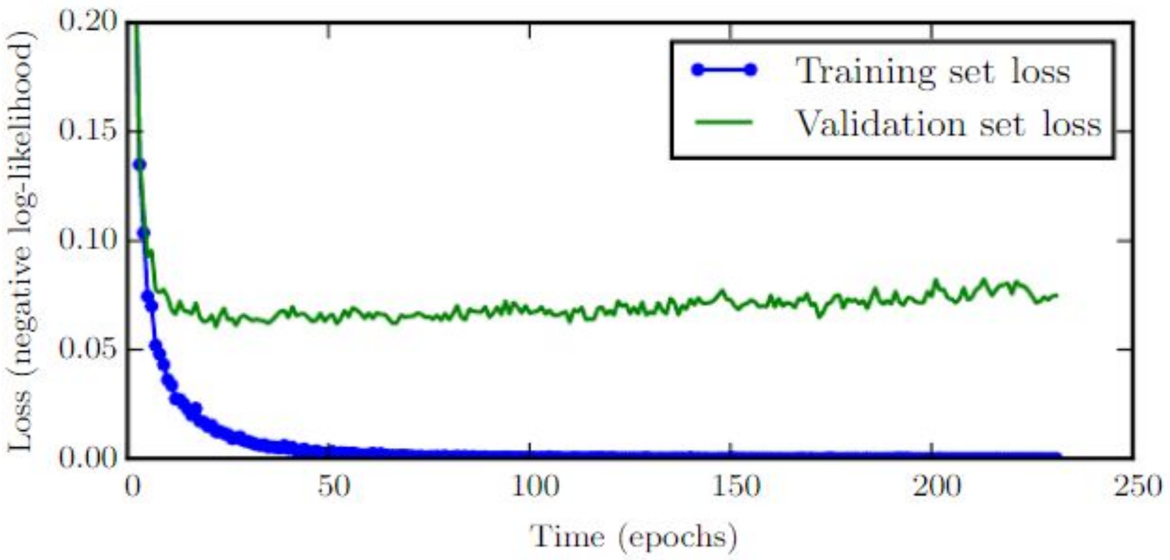
Idea: some aspects of one task are transferable to another task

By training a combined network with shared layers, both tasks benefit

Unsupervised layers may also support representation learning and explainability



Early Stopping



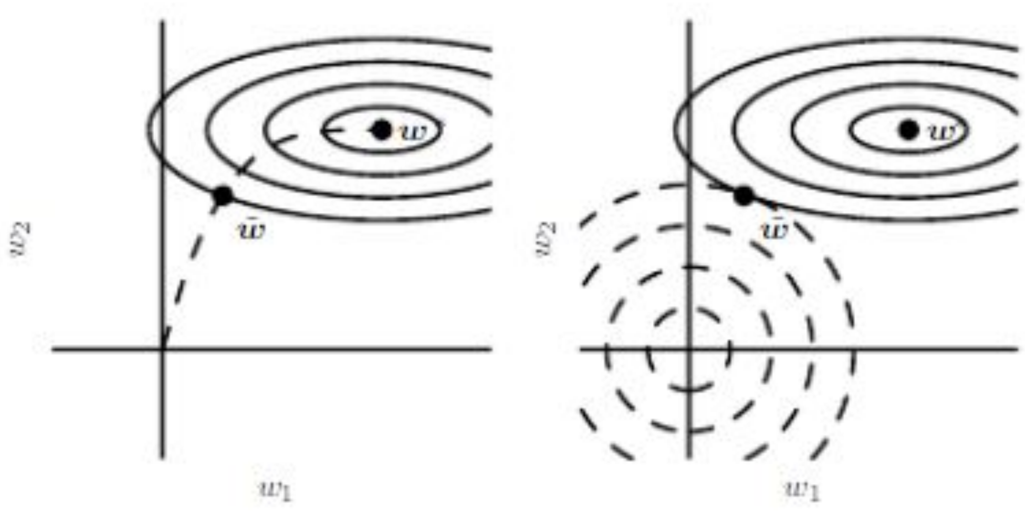
When should we stop learning?

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 and Other Ensemble Methods

In Lecture 7, we introduced bootstrap aggregation (aka bagging).

Model Averaging

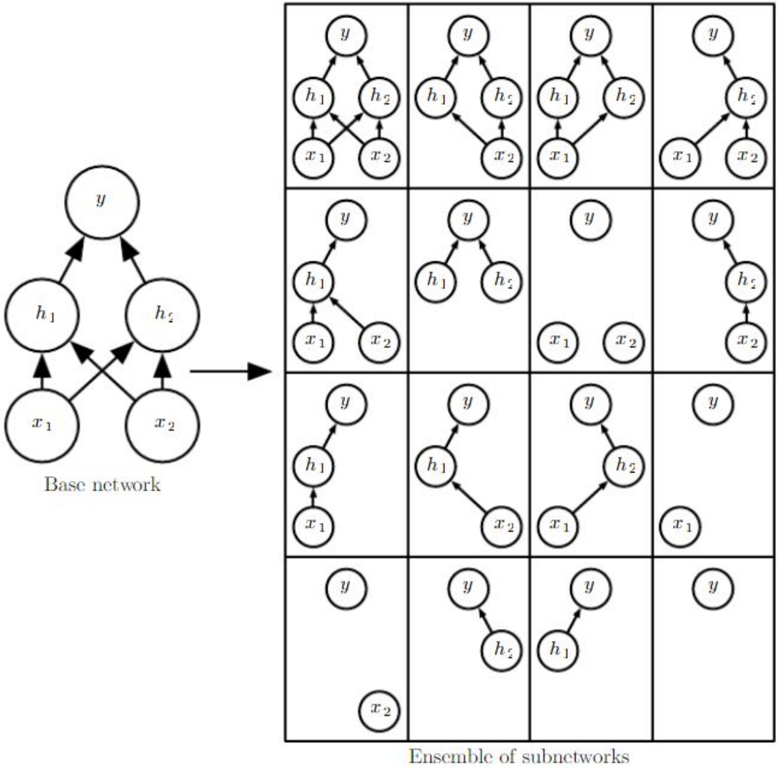
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i\left(\epsilon_i^2 + \sum_{j\neq i}\epsilon_i\epsilon_j\right)\right]$$
$$= \frac{1}{k}v + \frac{k-1}{k}c$$

Dropout

During training, with probability $p_{dropout}$, remove a node in the layer

During prediction, all nodes contribute the response

Common setting $0.1 \leq p_{dropout} \leq 0.3$



Readings

- Goodfellow - Chapter 8 (Optimization)