BOSTON UNIVERSITY

Deep Learning for Data Science
DS 542

Lecture 09
Regularization

Slides originally by Thomas Gardos.
Images from [Understanding Deep Learning] unless otherwise cited.

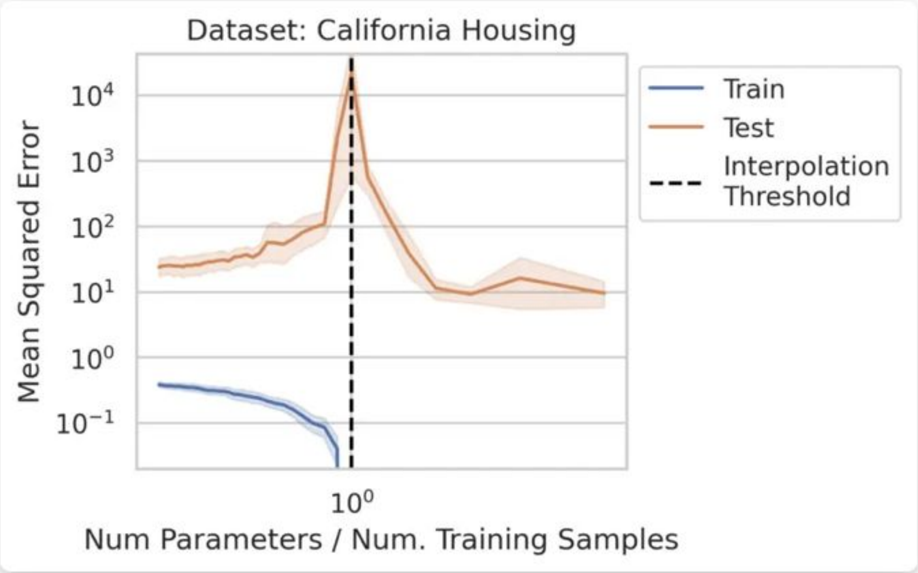
Administrivia

- Homework 9 will be out this afternoon.

- This networking event is Friday.
  - CDS hosted but open to all majors.
  - Please register if interested.

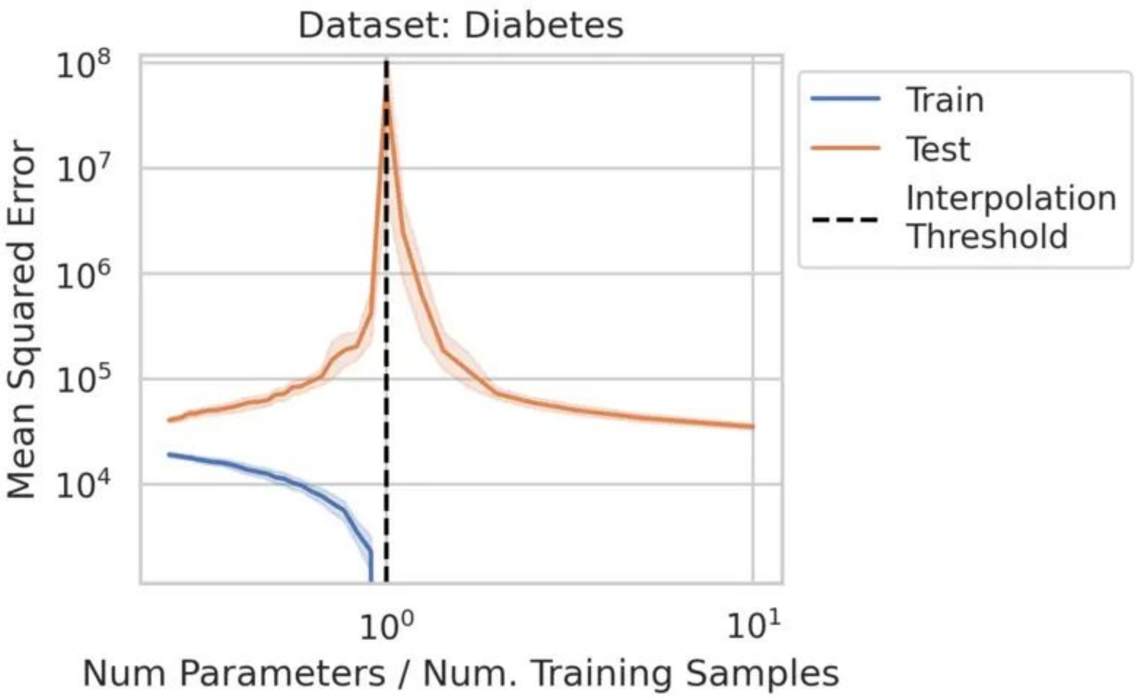
Double Descent Demystified

- Linear regression too?



https://iclr-blogposts.github.io/2024/blog/double-descent-demystified/

Reproduced with Classic Data Sets



Double Descent Test Setup

- Used polynomial features to generate arbitrary numbers of features
- When more polynomial features than training samples, regression has multiple parameters for exact fit.
    - Pick parameters minimizing a norm ← this is a regularization.

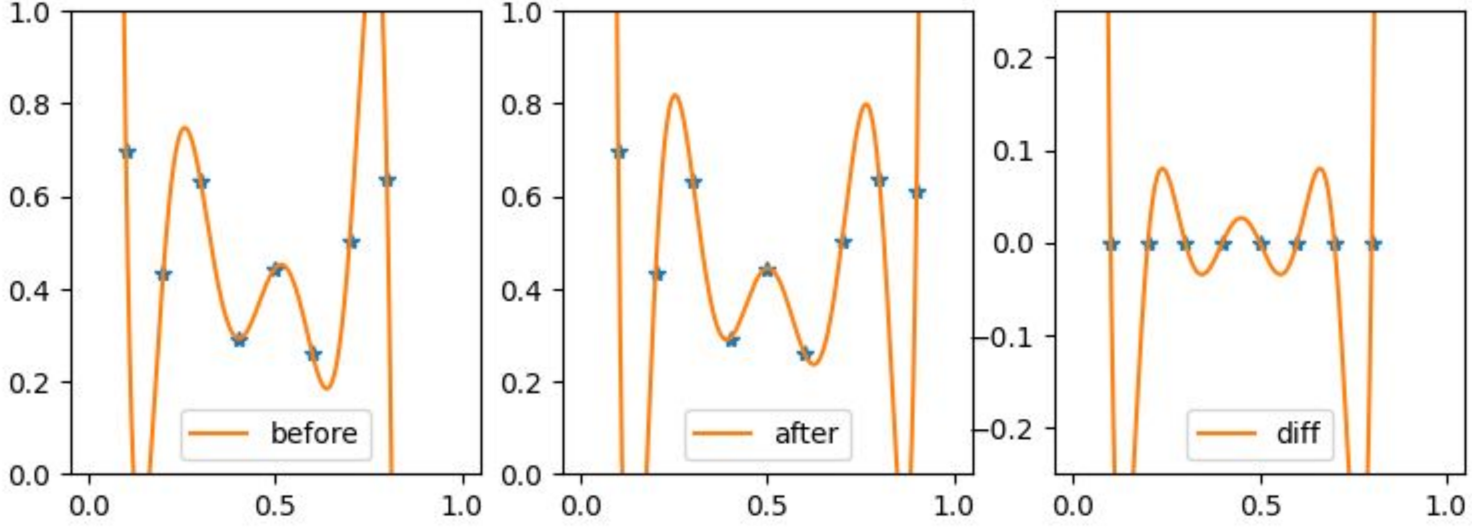
Rough Explanation for Double Descent

Three cases:

- Parameters << samples
  - Model can only fit overall trends. Cannot fit individual points particularly well.
  - Training and test loss improve with more parameters.
- Parameters ~ samples
  - Model can barely / not quite fit all training points.
  - Contortions are likely.
  - Detailed analysis says “singular values” a lot. Read the paper if curious.
- Parameters >> samples
  - Model can easily fit all training points.
  - Lots of freedom to make parameter norms smaller.
  - Some intuitive and proven connections to better generalization from smaller norms.
  - Regularization!

Re: Contortions are Likely

Original slide title: What Happens When You Add Another Point?



Regularization

- Why is there a generalization gap between training and test data?
  - Overfitting (model describes statistical peculiarities)
  - Model unconstrained in areas where there are no training examples
- Regularization = methods to reduce the generalization gap
- Technically means adding terms to loss function
- But colloquially means any method (hack) to reduce gap between training and test data

- Related question: how do we pick between all the possible models with the same training loss?

Reading Check

Takeaways from extra readings?

- [For Valid Generalization the Size of the Weights is More Important than the Size of the Network]
- [Train faster, generalize better: Stability of stochastic gradient descent]

Watch for these themes as we go through regularization examples.

Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Explicit regularization

- **Standard loss function:**

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[\mathrm{L}[\boldsymbol{\phi}]\Big] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i]\right]$$

- Regularization adds and extra term

- Favors some parameters, disfavors others.
- $\lambda>0$ controls the strength

Explicit regularization

- **Standard loss function:**

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[\mathrm{L}[\boldsymbol{\phi}]\Big] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i]\right]$$

- **Regularization adds an extra term**

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}]\right]$$

- Favors some parameters, disfavors others
- λ>0 controls the strength

Technically talking about cost functions, but usually just say “loss”...

Explicit regularization

- Standard loss function:

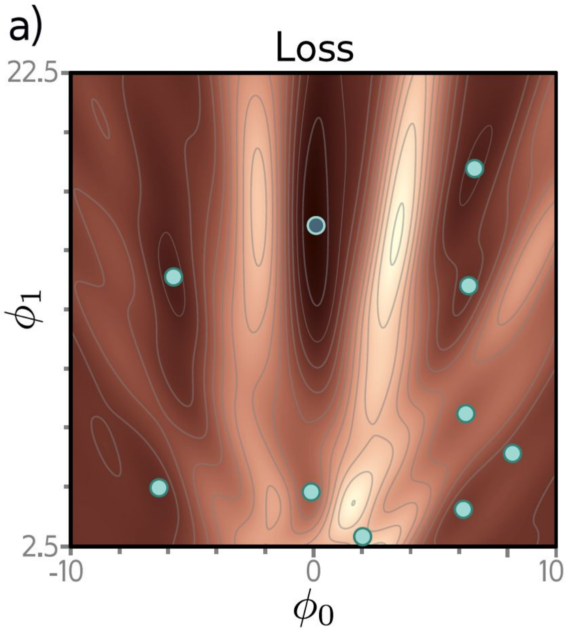
$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\big[\mathrm{L}[\boldsymbol{\phi}]\big] \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i, \mathbf{y}_i]\right]\end{aligned}$$

- Regularization adds an extra term

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}]\right]$$

- Where $\mathrm{g}[\phi]$ is smaller for preferred parameters
- $\lambda > 0$ controls the strength of influence

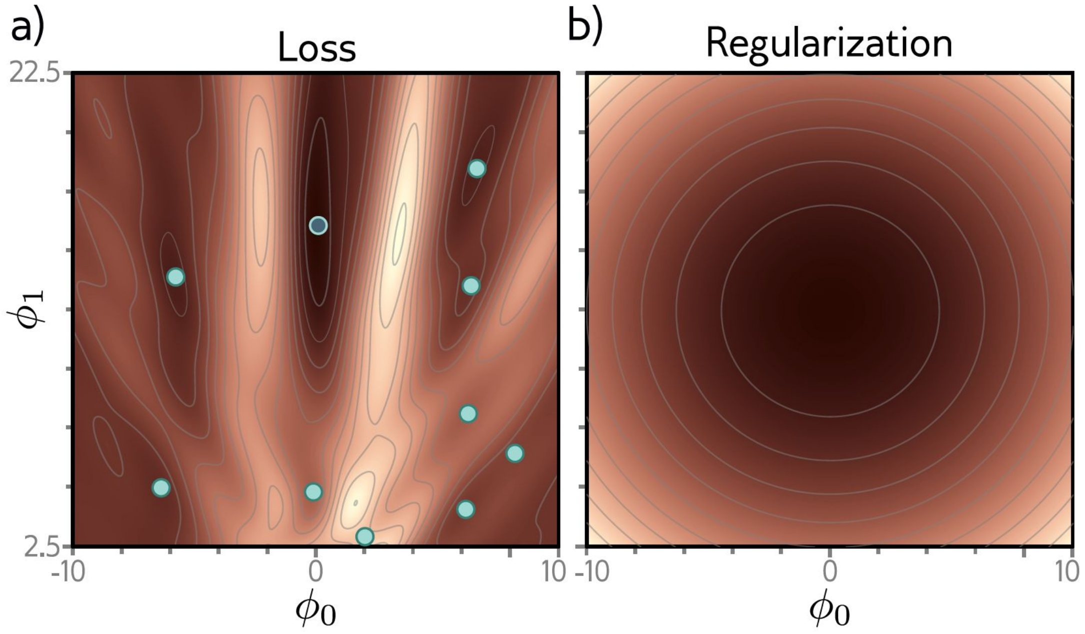
Explicit regularization



Loss function for Gabor model of Lecture 6 and Chapter 6.

 denotes local minima

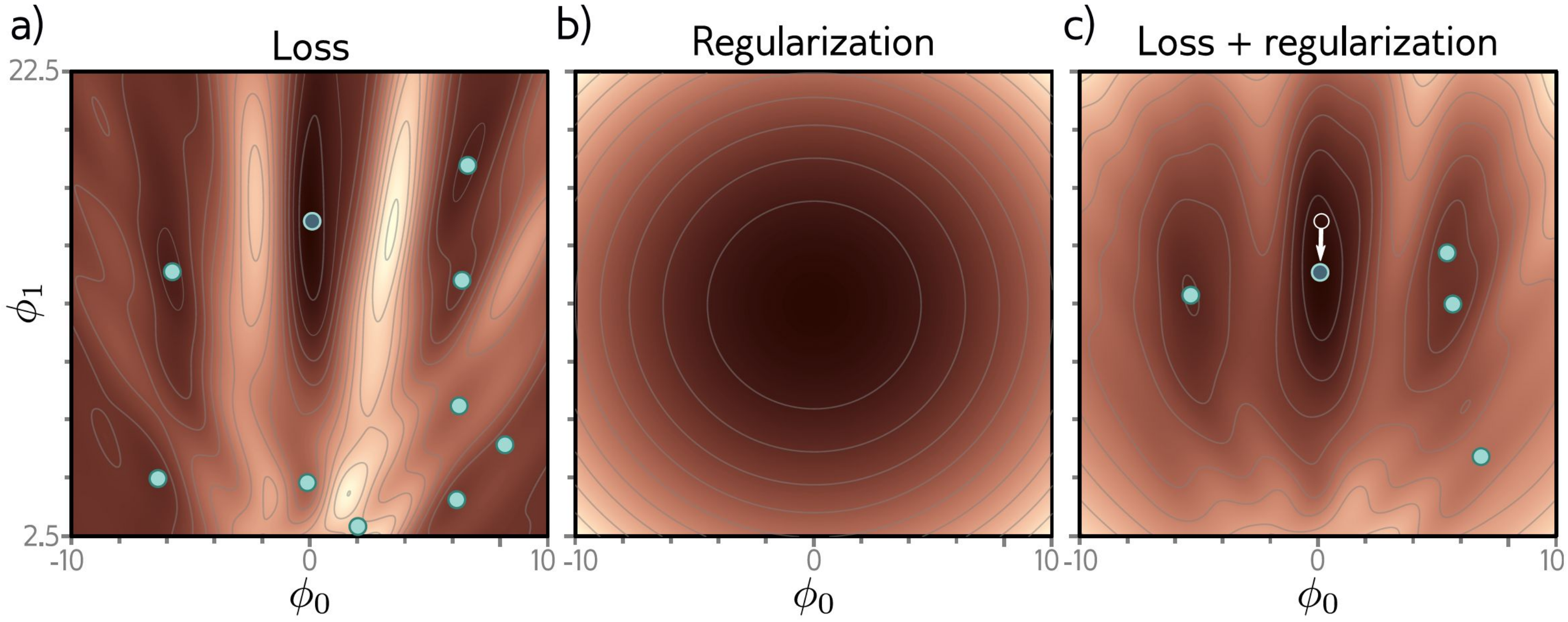
Explicit regularization



Example of a regularization function that prefers parameters close to 0.

Explicit regularization

Fewer local minima and the absolute minimum has moved.



denotes local minima

Probabilistic interpretation

- Maximum likelihood:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \phi) \right]$$

- Regularization is equivalent to adding a prior over parameters

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \phi) Pr(\phi) \right]$$

Maximum a posteriori or MAP criterion

… what you know about parameters *before* seeing the data

Equivalence

- Explicit regularization:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}] \right]$$

- Probabilistic interpretation:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\phi}) Pr(\boldsymbol{\phi}) \right]$$

- Converting to Negative Log Likelihood (e.g. $-\log(\cdot)$):

$$\lambda \cdot \mathrm{g}[\boldsymbol{\phi}] = -\log[Pr(\boldsymbol{\phi})]$$

L2 Regularization

- Most common regularizer is L2 regularization
- Favors smaller parameters (like in previous example)

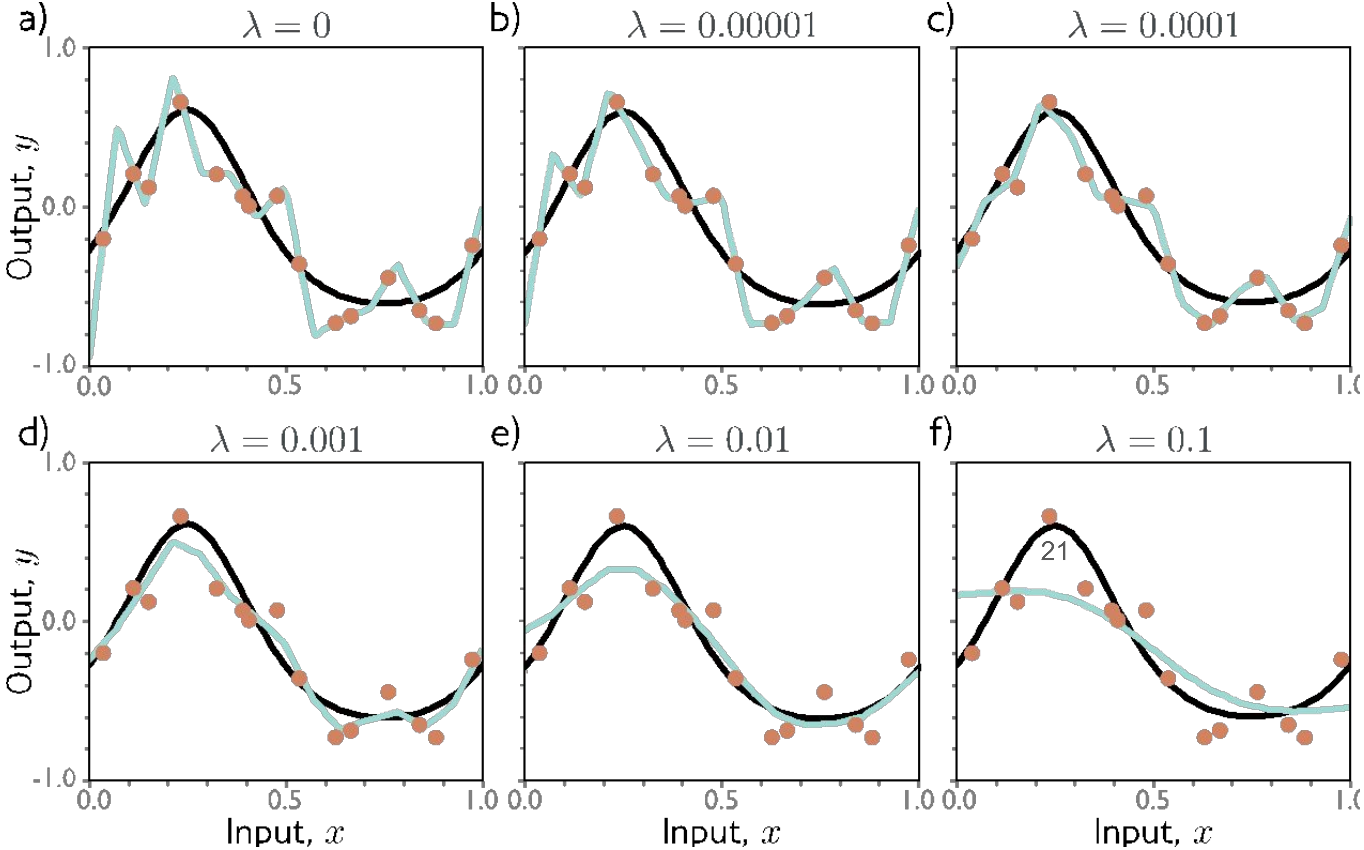
$$\hat{\phi} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\mathrm{L}[\boldsymbol{\phi}, \{\mathbf{x}_i, \mathbf{y}_i\}] + \lambda \sum_j \phi_j^2 \right]$$

- Also called Tikhonov regularization, ridge regression
- In neural networks, usually just for weights

Why does L2 regularization help?

- Discourages fitting excessively to the training data (overfitting)
- Encourages smoothness between data points

L2 regularization (simple net from last lecture)



PyTorch Explicit L2 Regularizer

SGD

CLASS torch.optim.SGD(*params, lr=0.001, momentum=0, dampening=0, weight_decay=0, nesterov=False, *, maximize=False, foreach=None, differentiable=False*) [SOURCE]

Implements stochastic gradient descent (optionally with momentum).

Parameters

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float, optional*) – learning rate (default: 1e-3)
- **momentum** (*float, optional*) – momentum factor (default: 0)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)

https://pytorch.org/docs/stable/generated/torch.optim.SGD.html

ADAM

CLASS torch.optim.Adam(*params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False, *, foreach=None, maximize=False, capturable=False, differentiable=False, fused=None*) [SOURCE]

Implements Adam algorithm.

Parameters

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float, Tensor, optional*) – learning rate (default: 1e-3). A tensor LR is not yet supported for all our implementations. Please use a float LR if you are not also specifying fused=True or capturable=True.
- **betas** (*Tuple[float, float], optional*) – coefficients used for computing running averages of gradient and its square (default: (0.9, 0.999))
- **eps** (*float, optional*) – term added to the denominator to improve numerical stability (default: 1e-8)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)

https://pytorch.org/docs/stable/generated/torch.optim.Adam.html

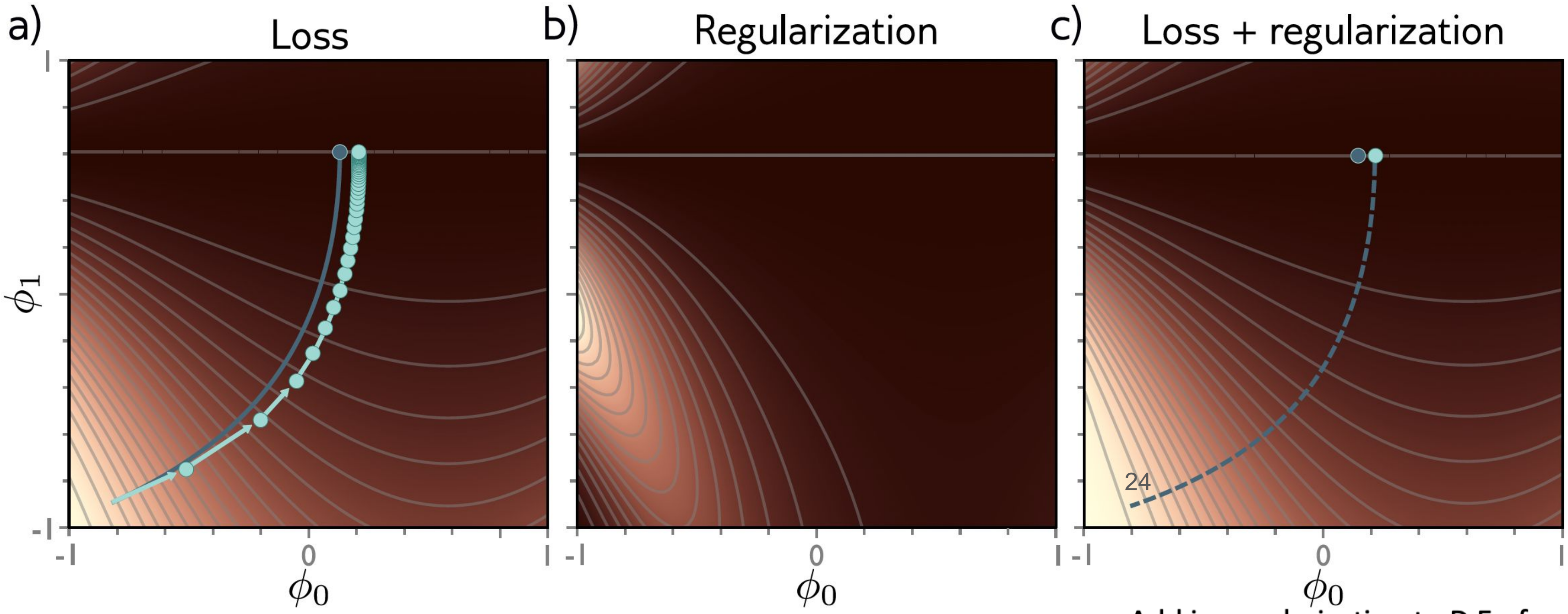
Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Implicit regularization

Going to infinitesimal (continuous) step, change in ϕ governed by:

$$\frac{\partial \phi}{\partial t} = -\frac{\partial L}{\partial \phi}$$



Gradient descent approximates a differential equation (infinitesimal step size)

Approximate implicit regularization added to continuous gradient descent

Add in regularization to D.E of $\sim\left\|\partial L/\partial \phi\right\|^2$ and differential equation converges to same place

See derivation at end of UDL Ch. 9

Implicit regularization

- Gradient descent disfavors areas where gradients are steep

$$\tilde{\mathrm{L}}_{GD}[\phi] = \mathrm{L}[\phi] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \phi}\right\|^2$$

See derivation at end of UDL Ch. 9

Implicit regularization

- Gradient descent disfavors areas where gradients are steep

$$\tilde{\mathrm{L}}_{GD}[\phi] = \mathrm{L}[\phi] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \phi}\right\|^2$$

- SGD likes all batches to have similar gradients

$$\begin{aligned}\tilde{\mathrm{L}}_{SGD}[\phi] &= \tilde{\mathrm{L}}_{GD}[\phi] + \frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \phi} - \frac{\partial L}{\partial \phi}\right\|^2 \\ &= \mathrm{L}[\phi] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \phi}\right\|^2 + \underbrace{\frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \phi} - \frac{\partial L}{\partial \phi}\right\|^2}\end{aligned}$$

Where $L = \frac{1}{I}\sum_{i=1}^{I}\ell_i[\mathbf{x}_i, y_i]$ and $L_b = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}_b}\ell_i[\mathbf{x}_i, y_i].$

Want the batch variance to be small, rather than some batches fitting well and others not well…

Implicit regularization

- Gradient descent disfavors areas where gradients are steep

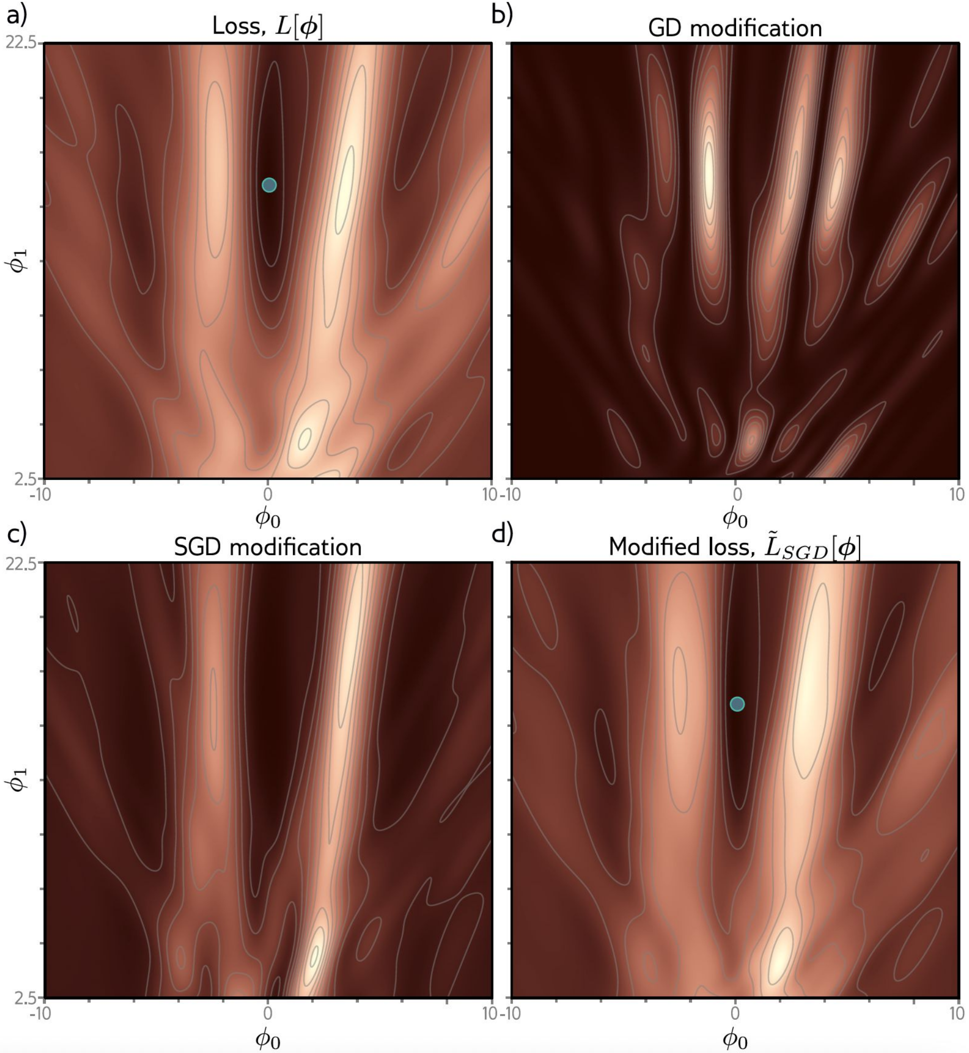
$$\tilde{\mathrm{L}}_{GD}[\boldsymbol{\phi}] = \mathrm{L}[\boldsymbol{\phi}] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2$$

- SGD likes all batches to have similar gradients

$$\begin{aligned}\tilde{\mathrm{L}}_{SGD}[\boldsymbol{\phi}] &= \tilde{\mathrm{L}}_{GD}[\boldsymbol{\phi}] + \frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \boldsymbol{\phi}} - \frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2 \\ &= \mathrm{L}[\boldsymbol{\phi}] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2 + \frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \boldsymbol{\phi}} - \frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2\end{aligned}$$

- Depends on learning rate – perhaps why larger learning rates generalize better.

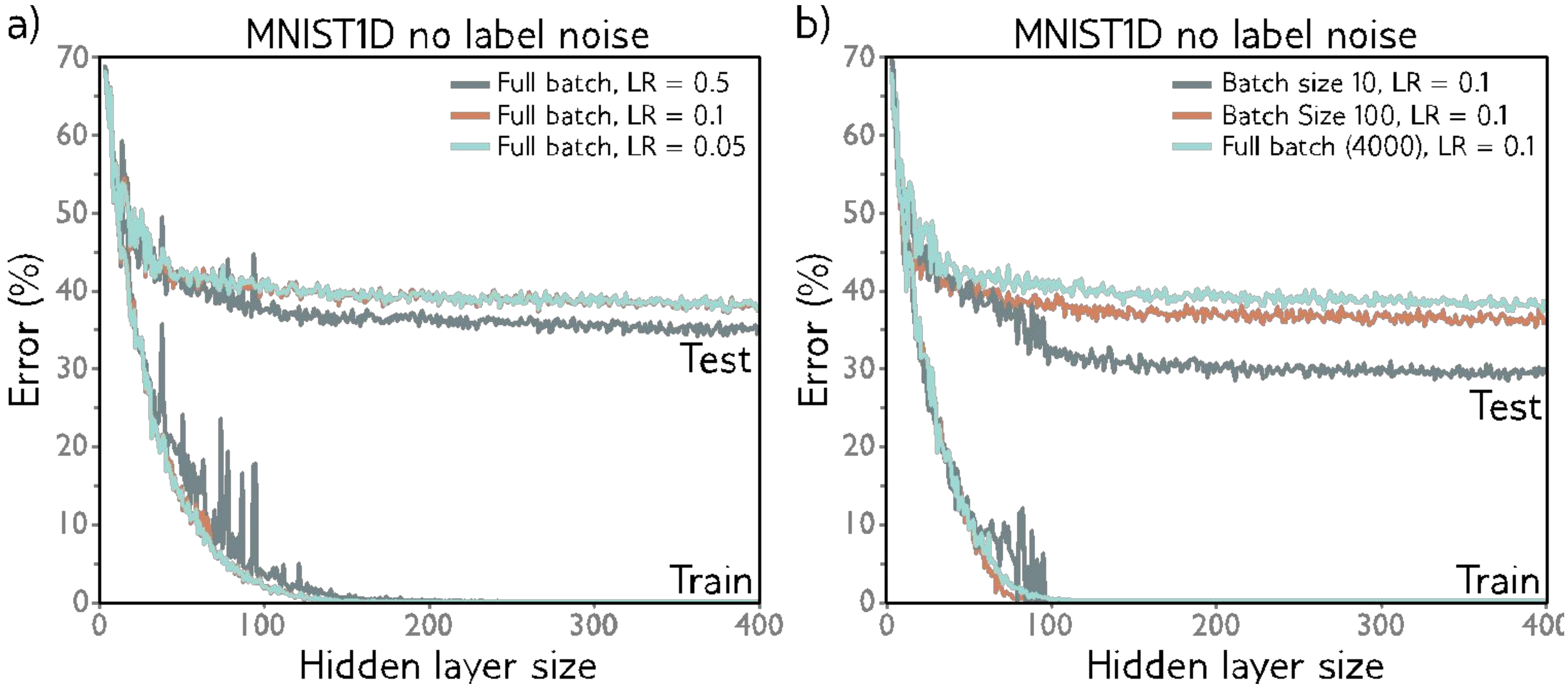
Original Gabor
Model
Loss



$$\frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2$$

$$\frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \boldsymbol{\phi}}-\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2$$

$$\tilde{\mathrm{L}}_{SGD}[\boldsymbol{\phi}] = \mathrm{L}[\boldsymbol{\phi}] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2 + \frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \boldsymbol{\phi}}-\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2$$



Generally, performance is

- best for larger learning rates
- best with smaller batches

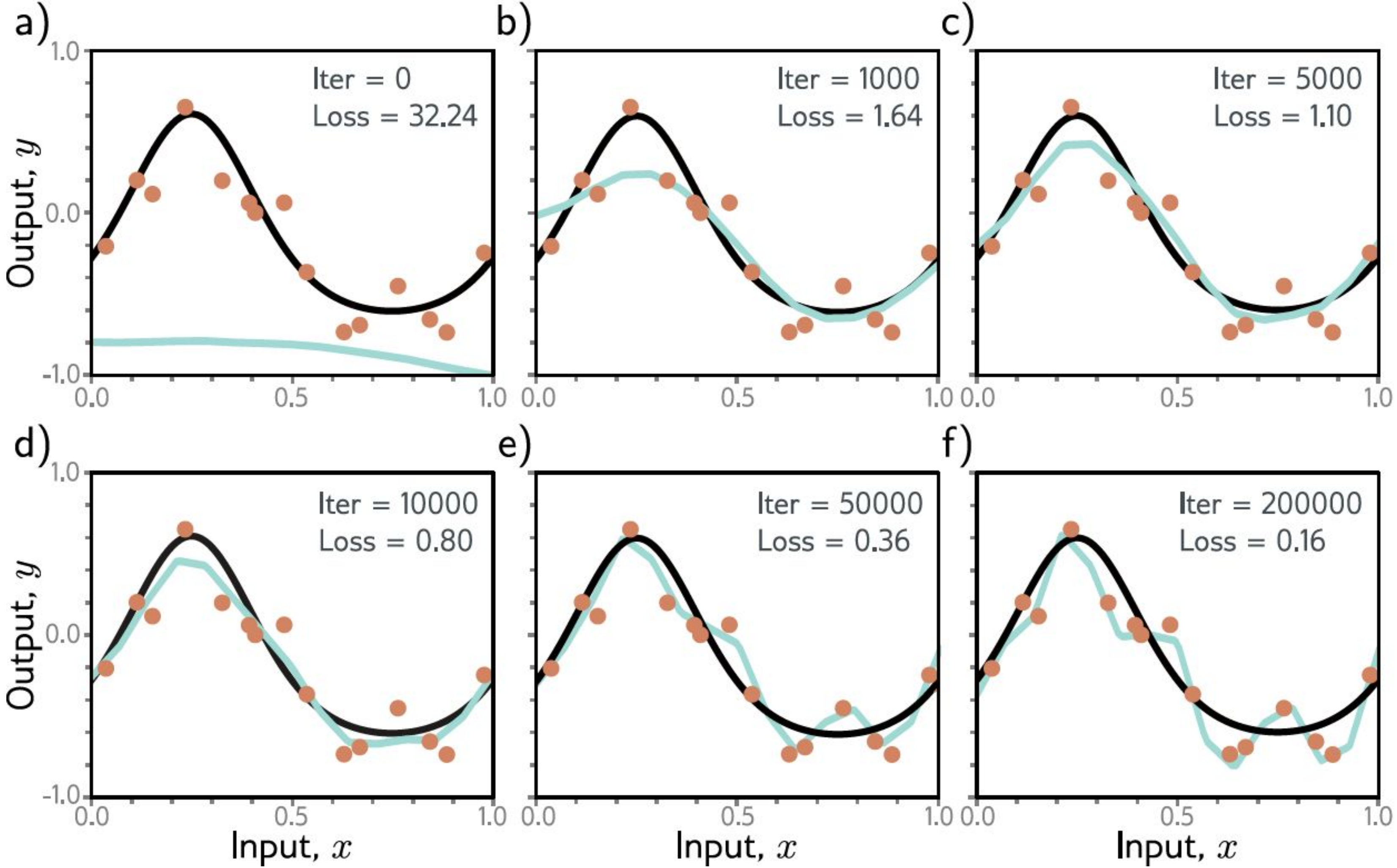
Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Early stopping

- If we stop training early, weights don’t have time to overfit to noise
- Weights start small, don’t have time to get large
- Reduces effective model complexity
- Known as early stopping
- Don’t have to re-train with different hyper-parameters – just ”checkpoint” regularly and pick the model with lowest validation loss

Old heuristic. Somewhat recent paper (2015) that you just skimmed.



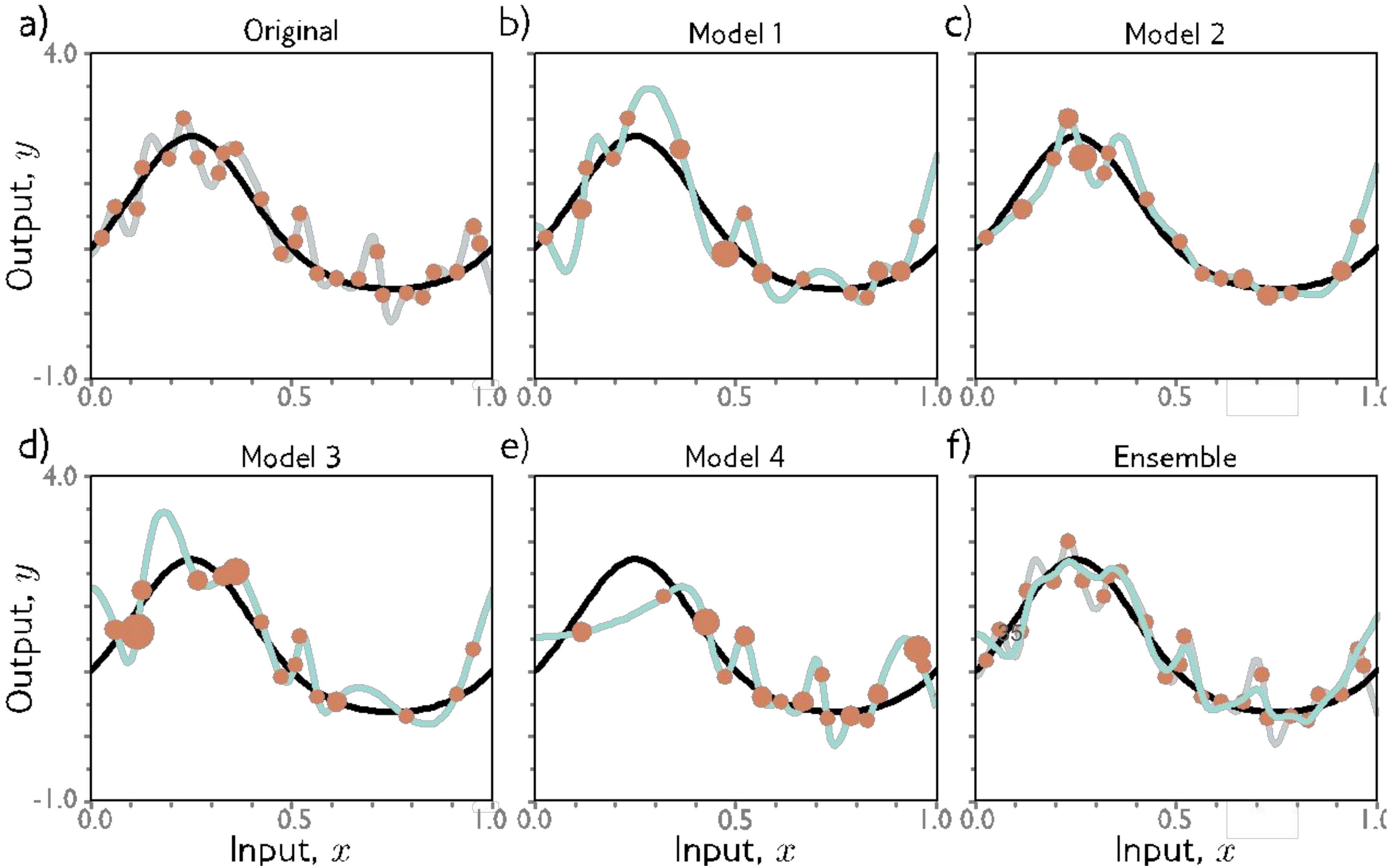
a)
Iter = 0
Loss = 32.24
b)
Iter = 1000
Loss = 1.64
c)
Iter = 5000
Loss = 1.10
d)
Iter = 10000
Loss = 0.80
e)
Iter = 50000
Loss = 0.36
f)
Iter = 200000
Loss = 0.16
Output, y
Input, x
1.0
0.0
-1.0
0.0
0.5
1.0

Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Ensembling

- Average together several models – an ensemble
- Can take mean or median
    - Before softmax for classification
- Simply different initializations or even different models
    - Numer.ai hedge fund outsources model building to data scientists around the globe and combines them into a “meta model” that drives their trading.
- Or train with different subsets of the data resampled with replacements -- bagging
    - “Bootstrap aggregation”. Old technique (1994) training many related predictors from different bootstrap samples.

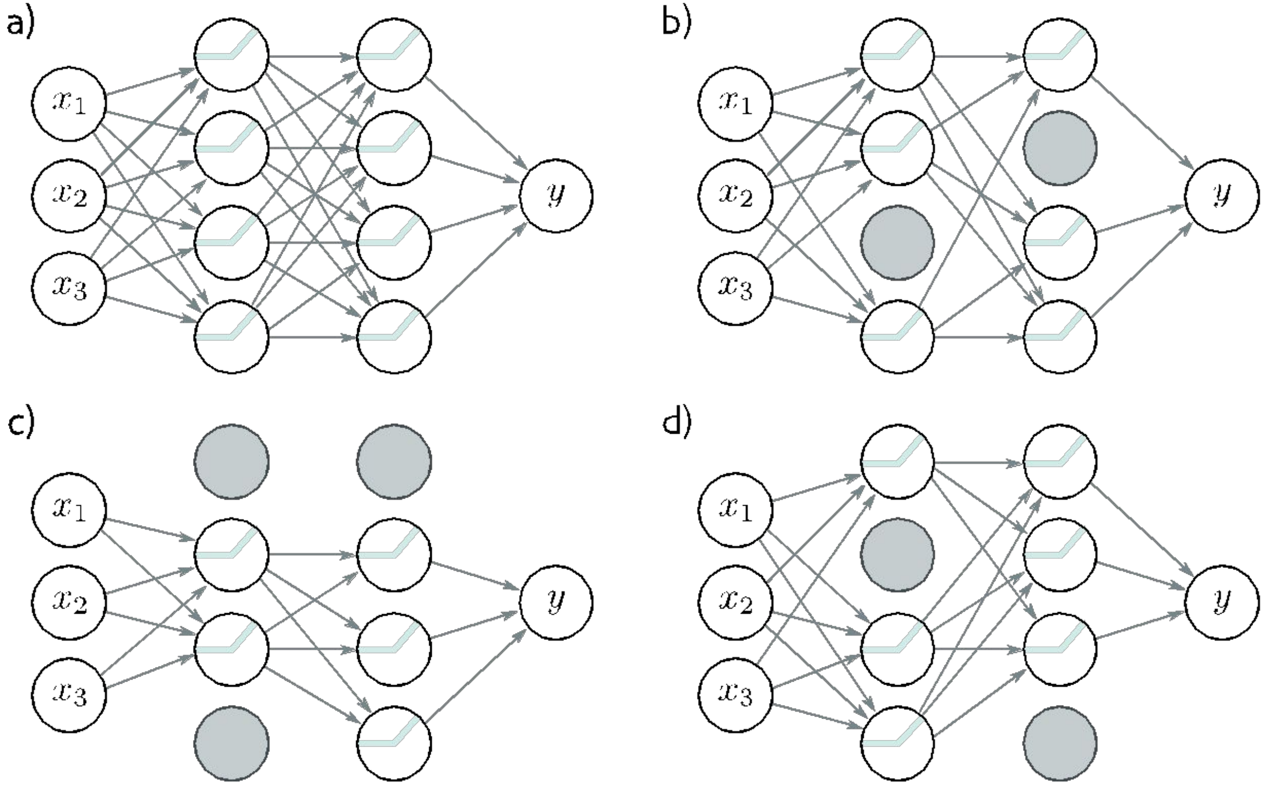

a)
Original
b)
Model 1
c)
Model 2
d)
Model 3
e)
Model 4
f)
Ensemble
4.0
-1.0
0.0
0.5
1.0
Output, y
Input, x

Regularization

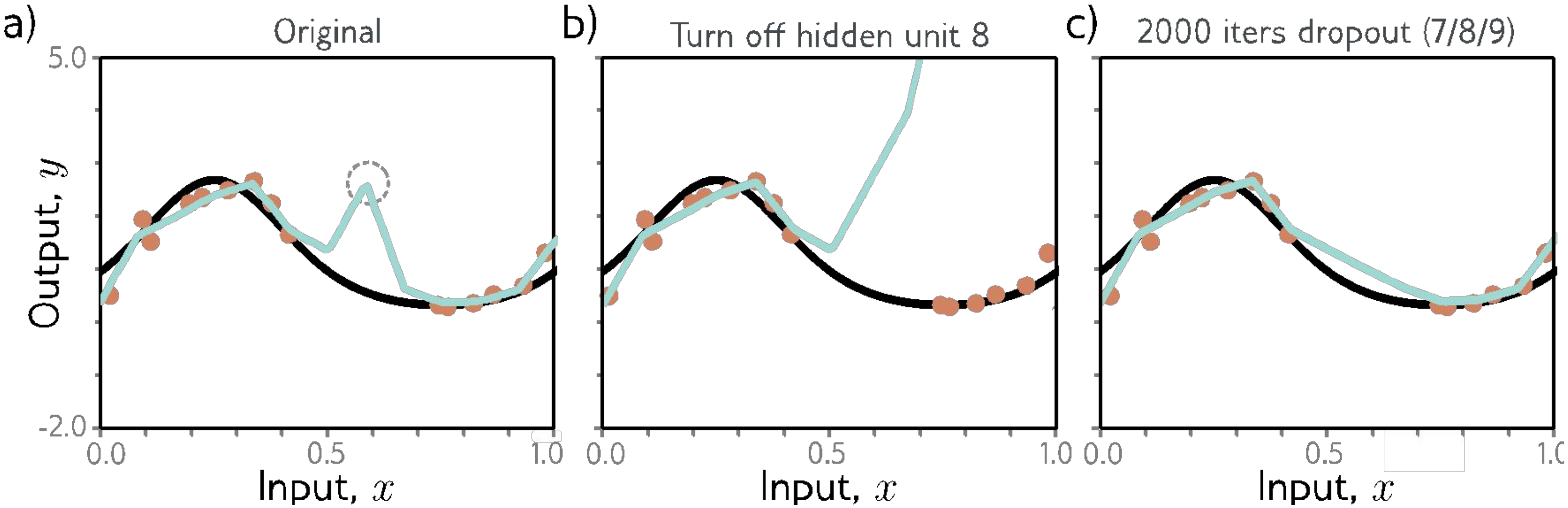
- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Dropout

Randomly clamp ~50% of hidden units to 0 on each iteration.



Dropout



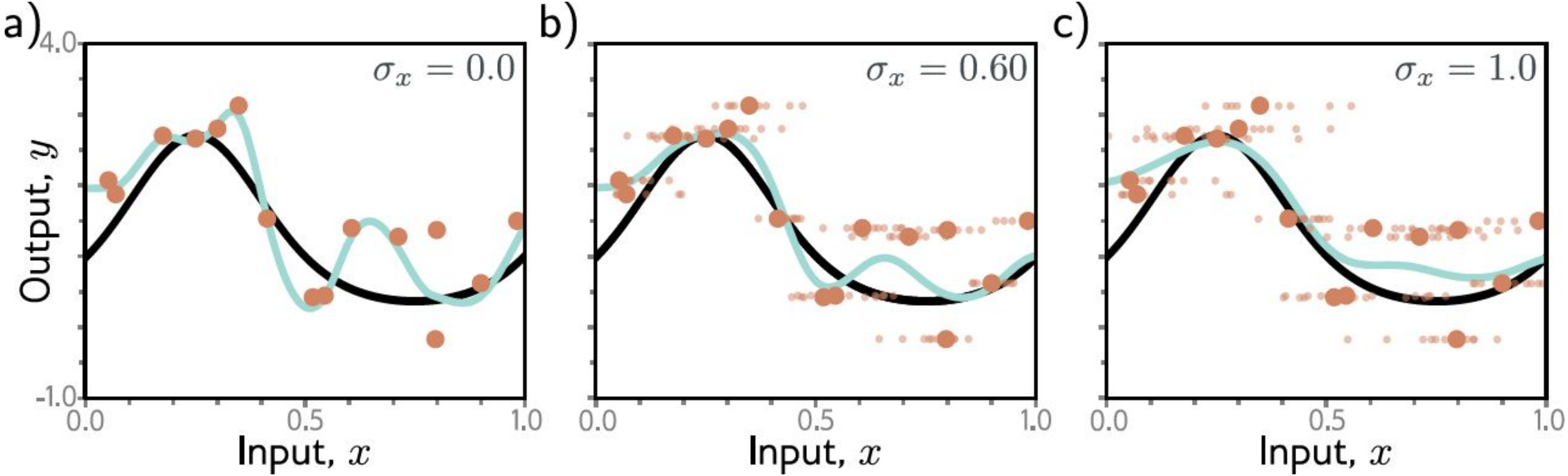
- Makes the network less dependent on any given hidden unit.
- Prevents situations where subsequent hidden units correct for excessive swings from earlier hidden units
- Can eliminate kinks in function that are far from data and don't contribute to training loss
- Must use *weight scaling inference rule* – multiple weights by (1 – dropout probability)

Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Adding noise

Adding noise to input with different variances.



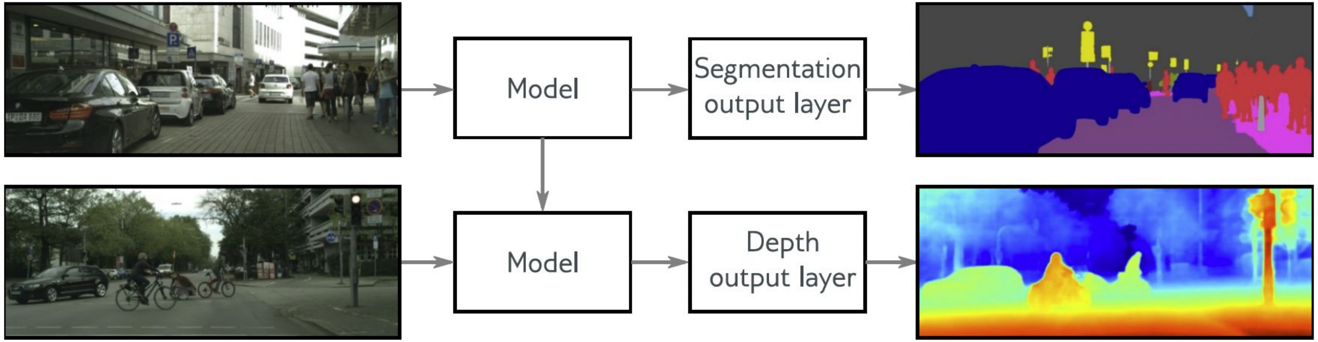
- to inputs – induces weight regularization (see Exercise 9.3 in UDL)
- to weights – makes robust to small weight perturbations
- to outputs (labels) – reduces “overconfident” probability for target class

Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Transfer Learning (Pre Training)

(1) Train the model for segmentation



Assume we have lots of segmentation training data

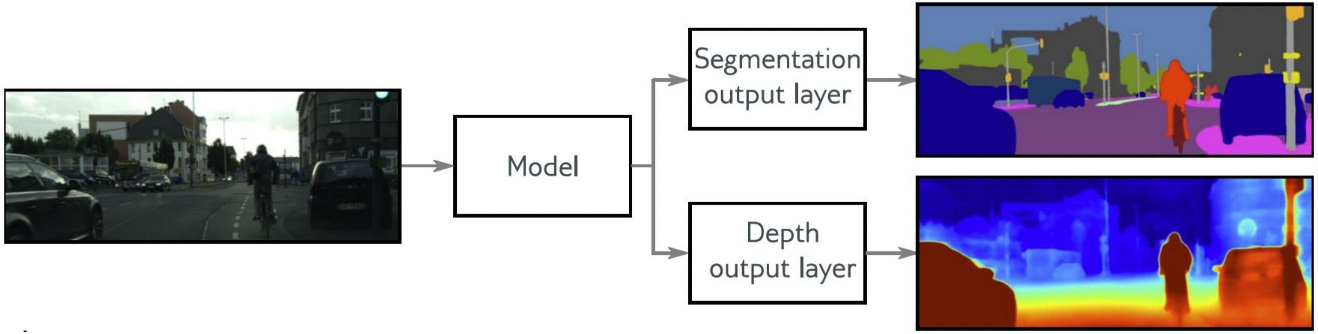
Assume we limited depth training data

(2) Replace the final layers to match the new task and

(3) Either:

a) Freeze the rest of the layers and train the final layers
b) Fine tune the entire model

Multi-Task Learning



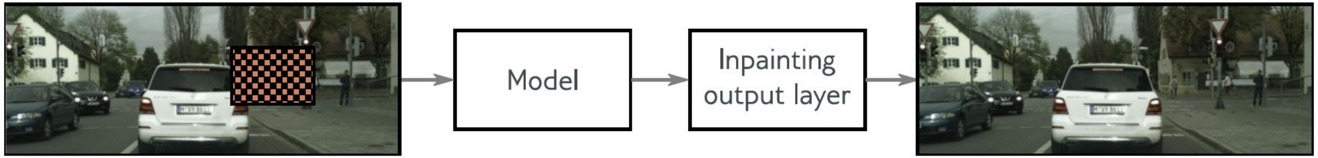
- Train the model for 2 or more tasks simultaneously
  - Weighted combo of loss fncs

$$L_{total} = \alpha \cdot L_{segmentaiton} + \beta \cdot L_{depth}$$

- Less likely to overfit to training data of one task

- Can be harder to get training to converge. Might have to vary the individual task loss weightings, α and β.

Self-Supervised Learning



The animal didn't cross the $\boxed{}$ *because it was too tired.*

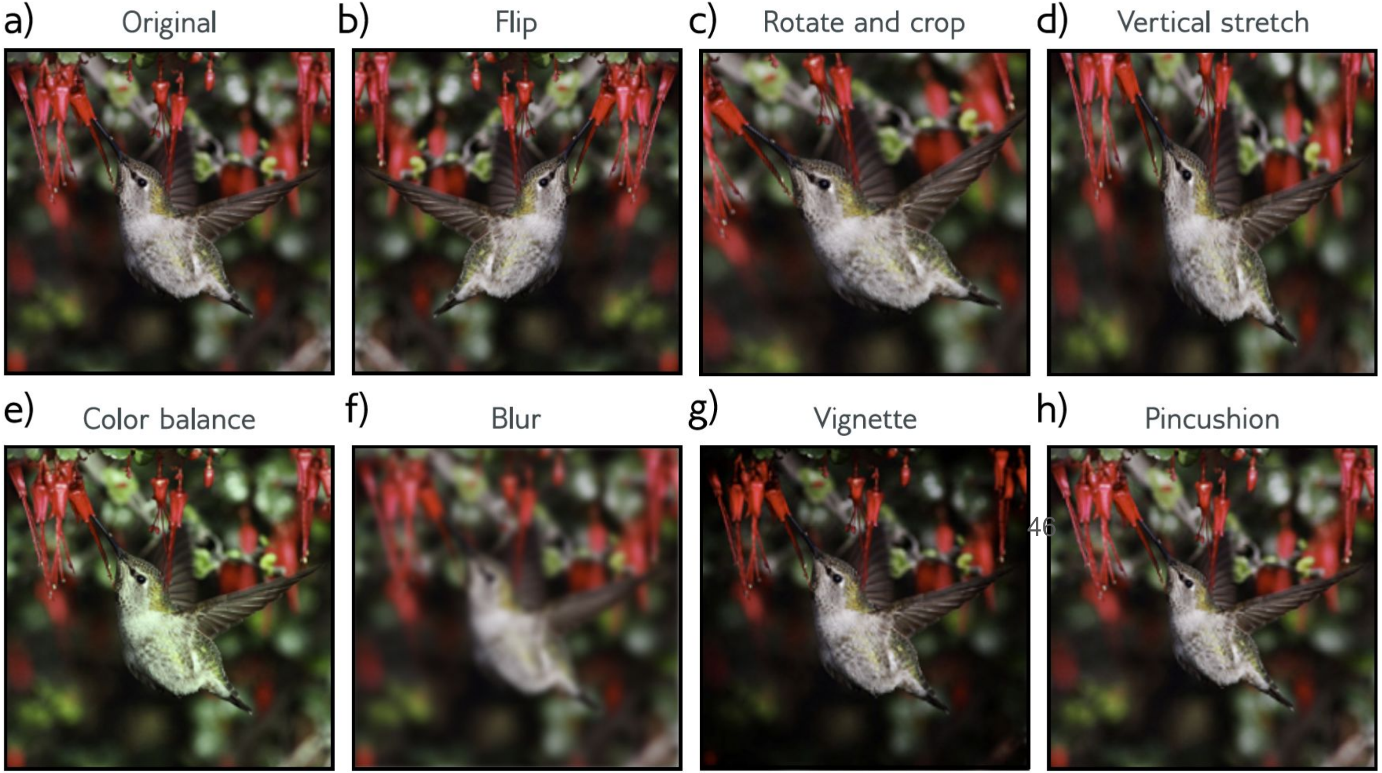
- Mask out part of the training data
- Train model to try to infer missing data
    - masked data is the target
- □ Model learns characteristics of the data

- Then apply transfer learning

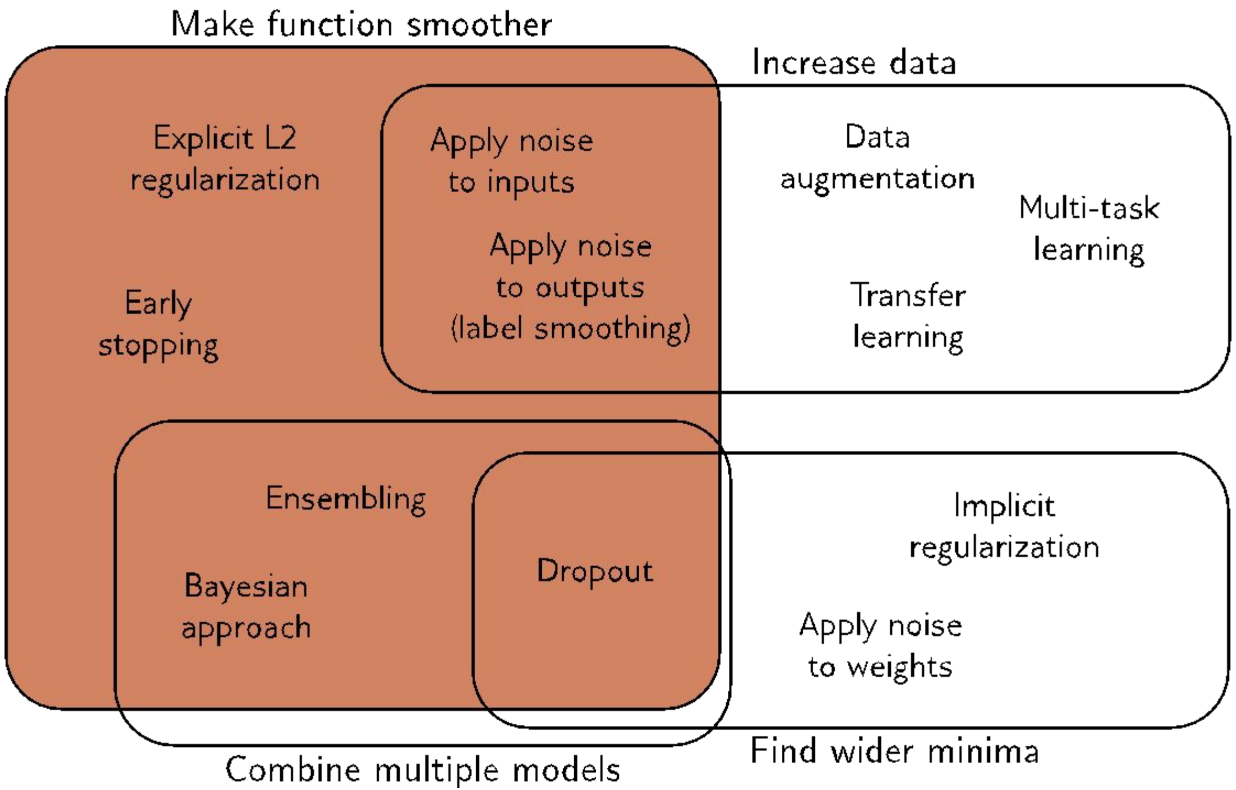
Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Data augmentation

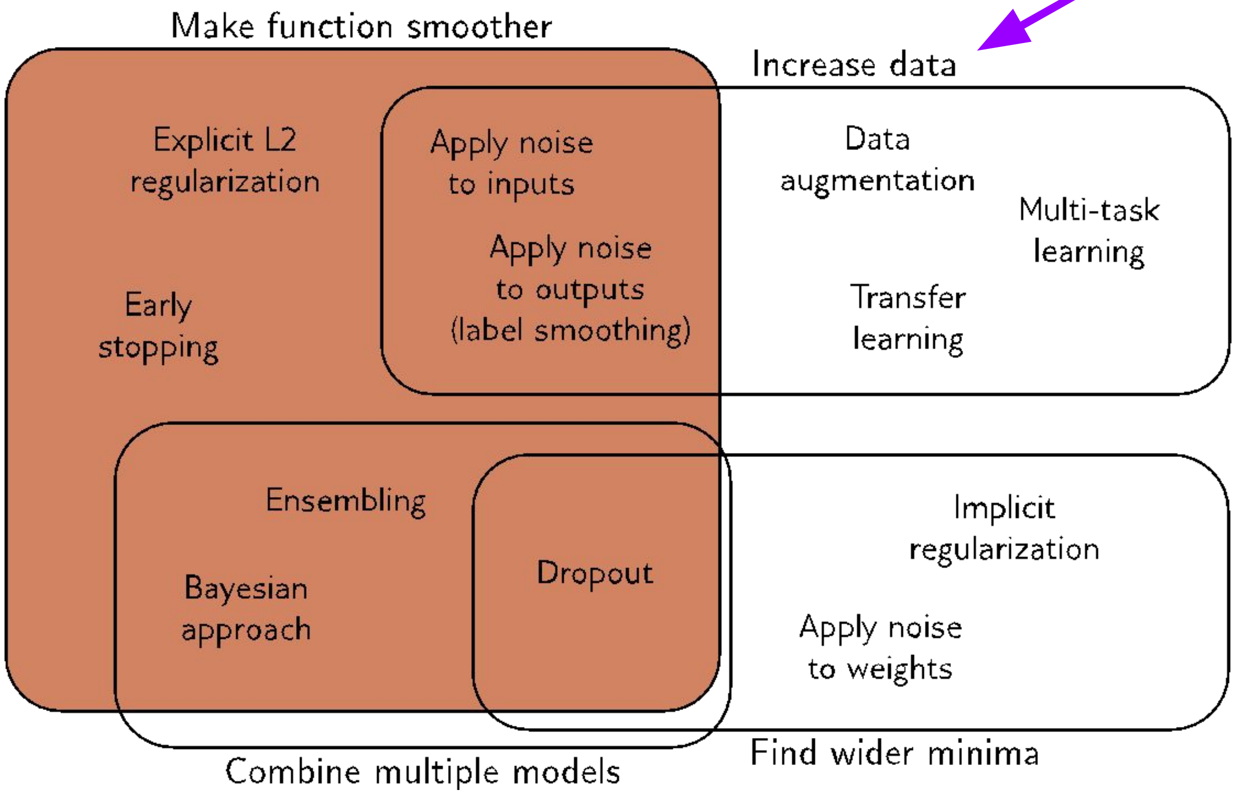


Regularization overview



Regularization overview

This was our fix for variance last week.



Bonus Techniques (not in the book)

- Input/Output standardization
- Layer normalization

Both of these address question from last time -

> “How do we control the input value distribution?”

Input/Output Normalization

Usually easier to train when inputs and outputs are in a small range close to zero.

- Idealized cases
  - 0/1 values
  - [0,1] values
  - Mean 0, standard deviation 1
  - Imagine if your network had to deal with 10^100 or 10^100+1 for yes or no?
- Standardization preprocessing
  - Subtract training data mean
  - Divide by training data standard deviation
- Whitening transformation
  - Linear transformation of data with known covariance matrix to remove correlations too

Layer Normalization

Normalize pre-activations before bias to mean 0, standard deviation 1

$$\mathbf{f}_k = \beta_k + \boxed{\Omega_k \mathbf{h}_k}$$

- Normalization not learned - calculated per instance.
- This directly controls the magnitudes in the forward pass.
- No longer a “vanilla neural network”, but who cares?

https://arxiv.org/abs/1607.06450 (2016, the year after He initialization)

~~Batch Normalization~~

Predecessor to layer normalization.

- Normalization calculated within mini batch of SGD.
- Dependent on what else was in the batch.
  - Supposed be chosen randomly, but will be noisy.
  - More noisy for small batches.
  - Does not work with batch size 1???

- Weird effects if you tried testing by target outcome.
  - Came up for one of last semester’s final projects.

Next Week

Last topics before midterm -

- Convolutional networks
  - Specialized for spatial structure, mostly used for images
- Residual networks
  - Another technique for training really deep networks

If we have time…

Notebook walkthrough

Feedback?