Regularization

January 30th, 2025

Deep Learning (CS60010)

**Slides adapted from http://udlbook.com*

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

Regularization

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

Explicit regularization

- Standard loss function:

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

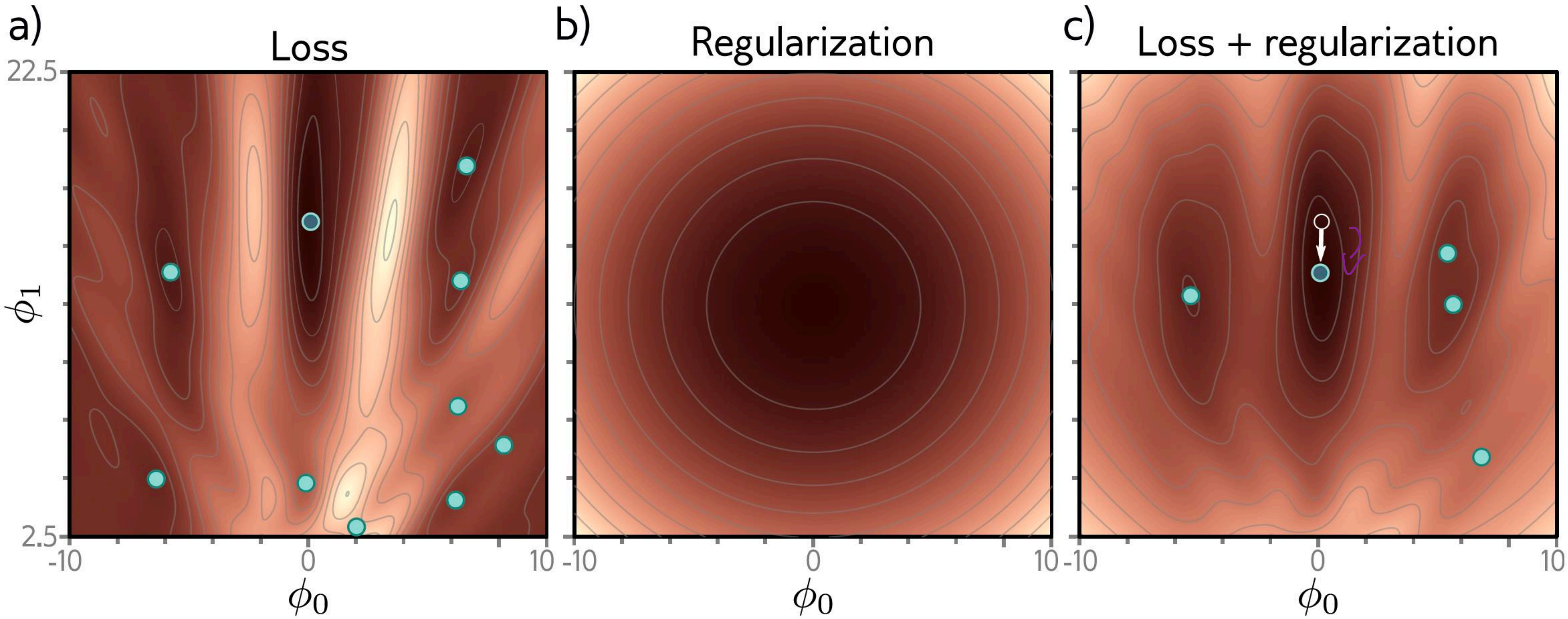
$\lambda \sum \phi_i^2$

- Regularization adds an extra term

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

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

Explicit regularization



Probabilistic interpretation

- Maximum likelihood:

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

- Regularization is equivalent to a adding a prior over parameters

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

… 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:

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

- Mapping:

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

L2 Regularization

- Can only use very general terms
- Most common is L2 regularization
- Favors smaller parameters

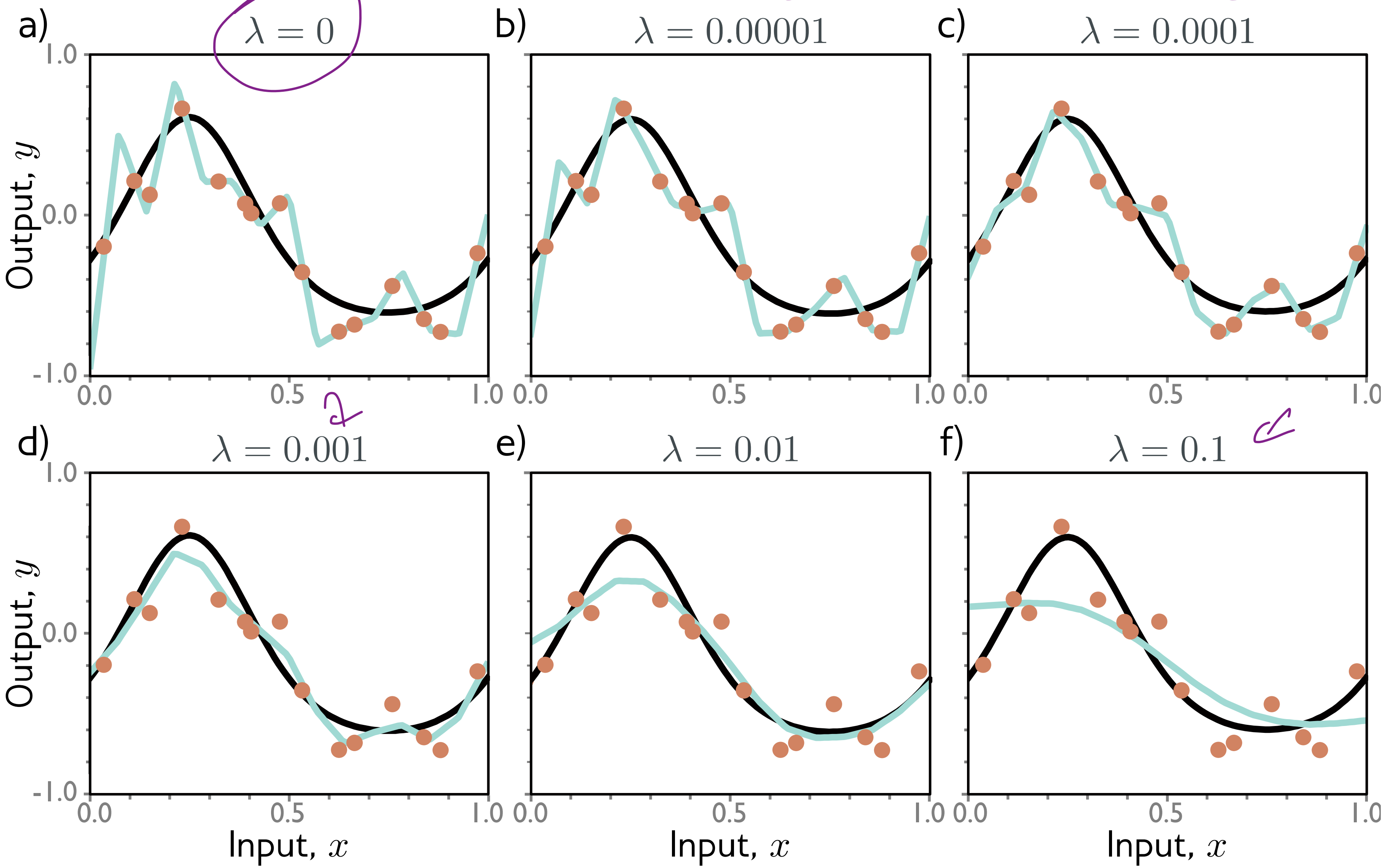
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\mathrm{argmin}} \left[\mathrm{L}[\boldsymbol{\phi}, \{\mathbf{x}_i, \mathbf{y}_i\}] + \lambda \sum_j \phi_j^2 \right]$$

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

Why does L2 regularization help?

- Discourages slavish adherence to the data (overfitting)
- Encourages smoothness between datapoints

L2 regularization

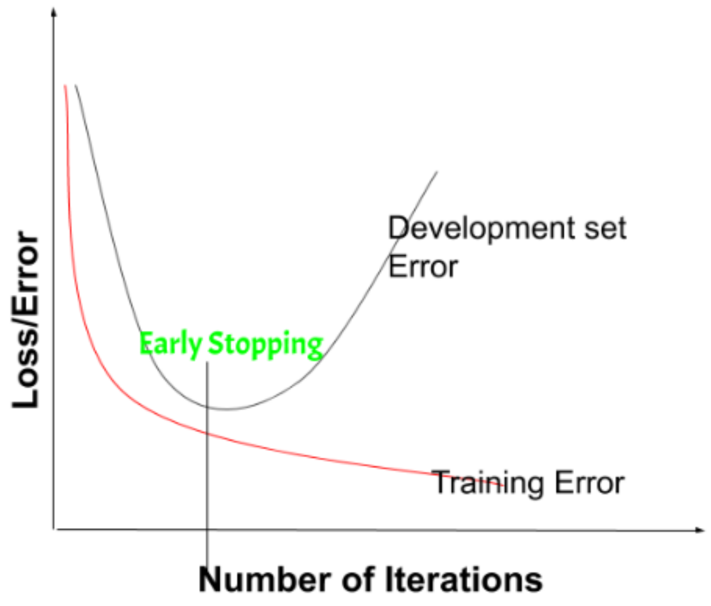


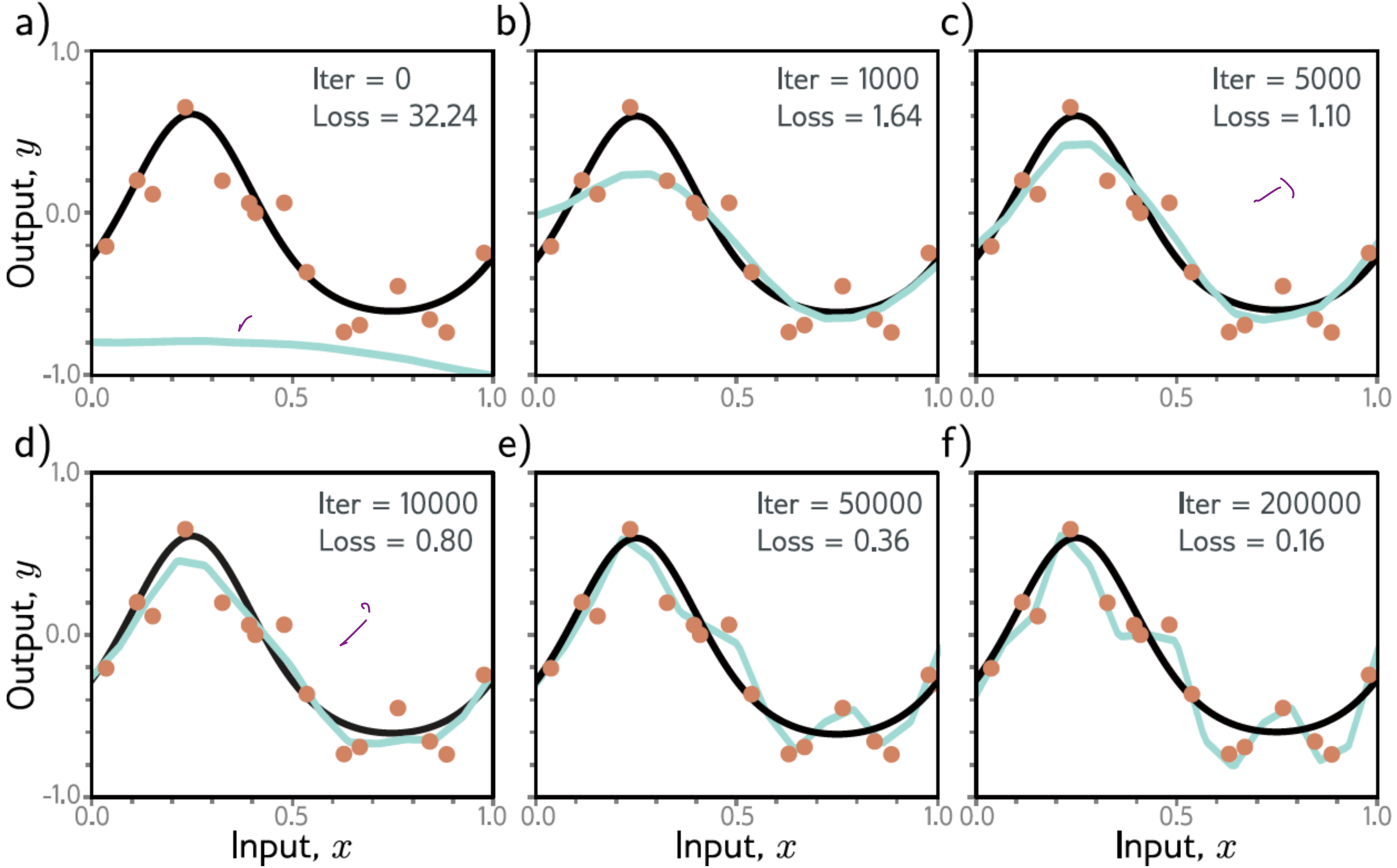
Regularization

- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Bayesian approaches
- 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





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

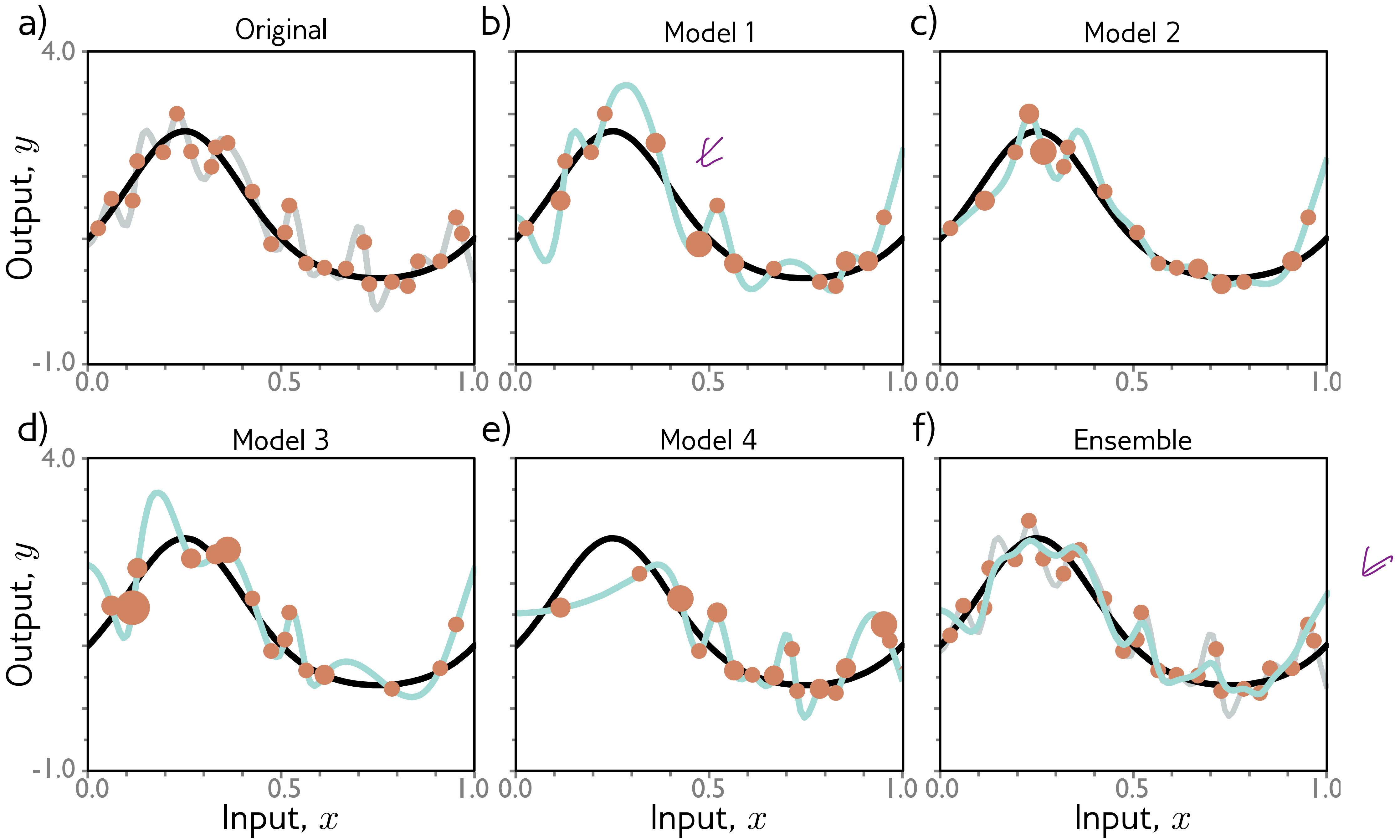
- Explicit regularization
- Implicit regularization
- Early stopping
- Ensembling
- Dropout
- Adding noise
- Bayesian approaches
- Transfer learning, multi-task learning, self-supervised learning
- Data augmentation

Ensembling

- Average together several models – an ensemble
- Can take mean or median of the model outputs (for regression)
- Mean of the pre-softmax activations (for classification)
  - Or the most frequent predicted class

How to train multiple models?

- Different initializations / different models
- Different subsets of the data resampled with replacements -- bagging

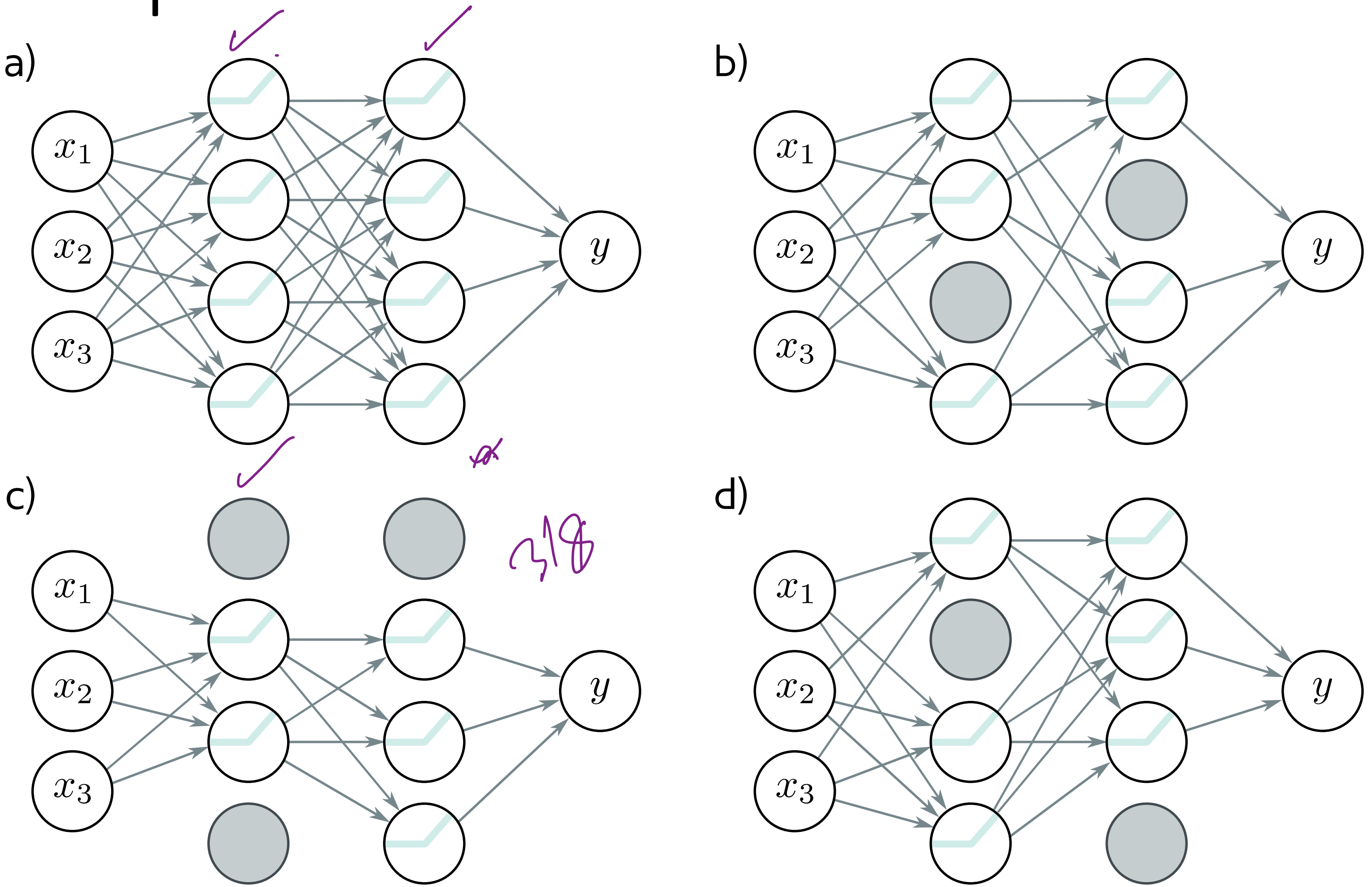


a)
Original
b)
Model 1
c)
Model 2
d)
Model 3
e)
Model 4
f)
Ensemble
4.0
-1.0
0.0
0.5
1.0
Output, y
Input, x

Regularization

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

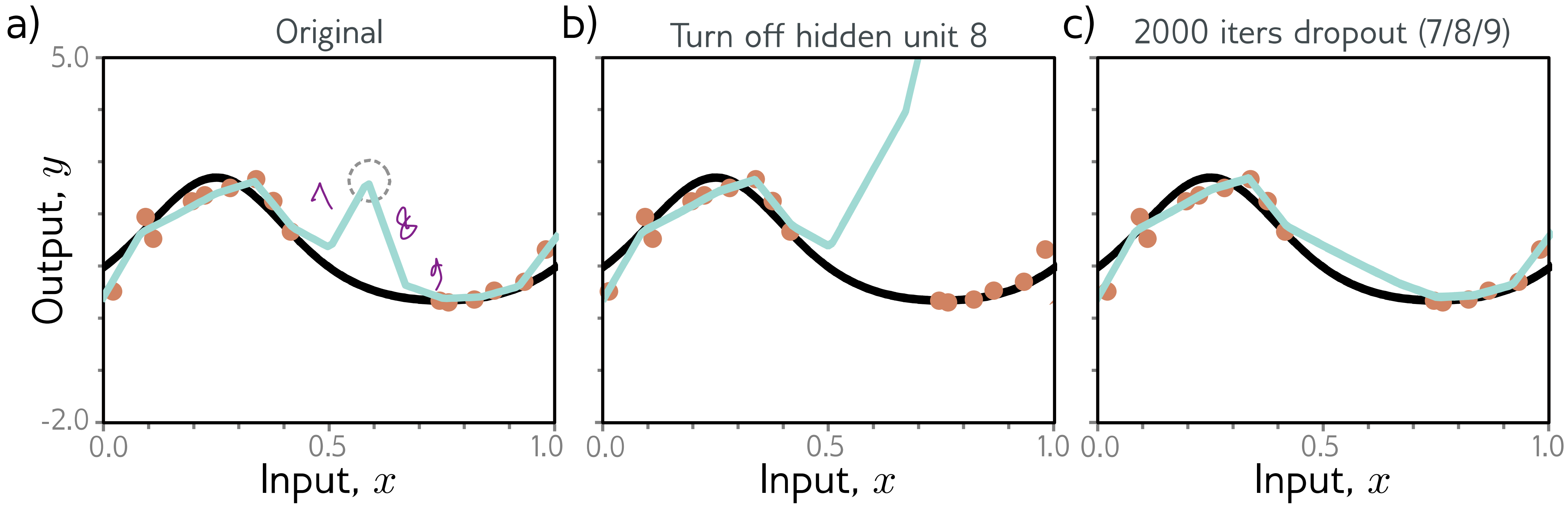
Dropout



At each training iteration, a certain subset of hidden units is clamped to zero (gray nodes). Thus, both the incoming and outgoing weights from these units have no effect.

Dropout

$p = 0.8$ (retaining)



Can eliminate kinks in function that are far from data and don't contribute to training loss

Dropout: At Inference time

- We can run the model as usual with all hidden units active.
- *What is the issue?*
  - The network now has more hidden units than it was trained with at
- What is the remedy?
  - Multiply the weights by (1-p) to compensate, p is the dropout probability

0.2

- Ensembling
  - Run the network multiple times with different random subset of units clamped to zero, and combine the results