The problem of overfitting

(larger)?

MLP Two layers, ReLU
3 hidden units

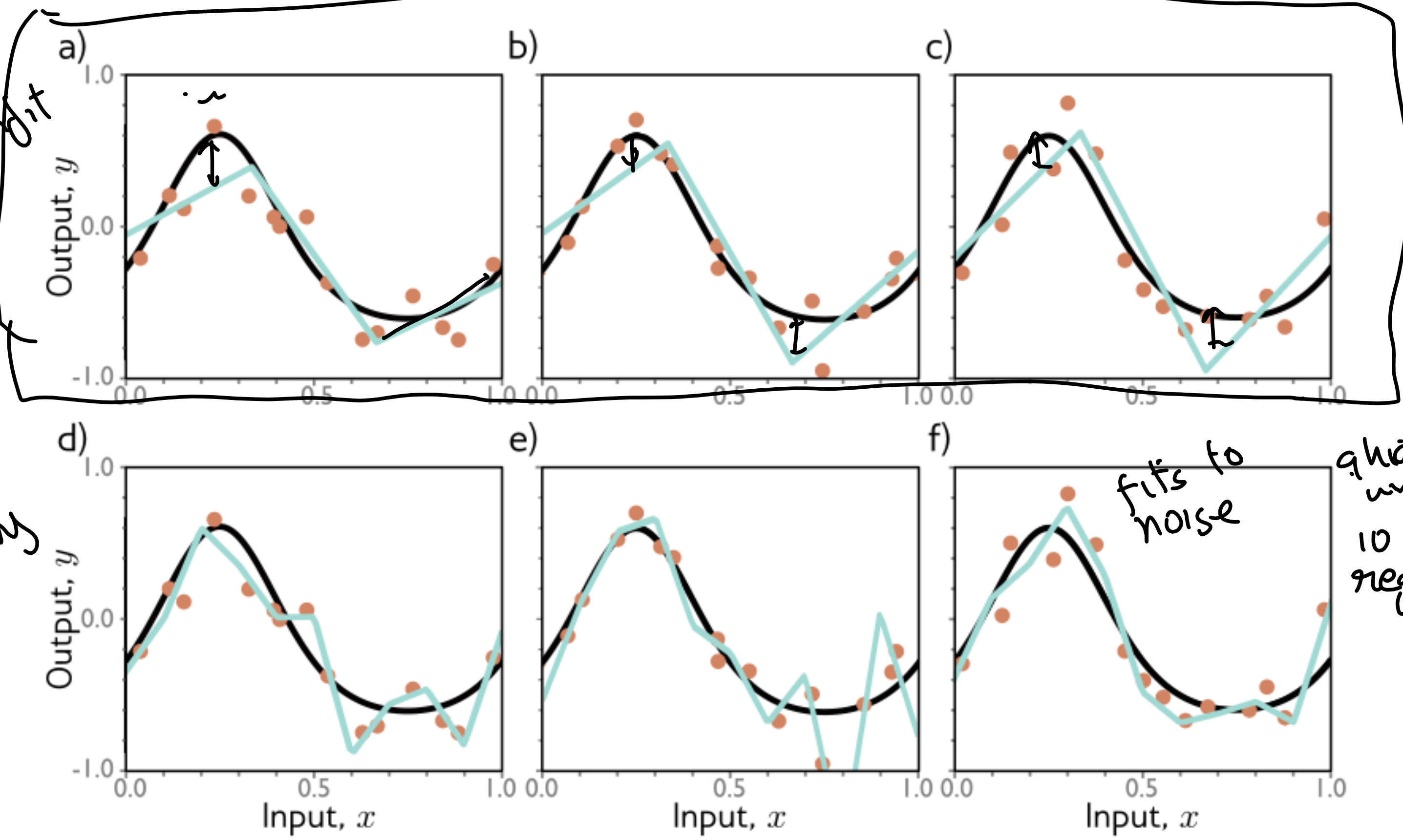


Figure 8.8 Overfitting. a–c) A model with three regions is fit to three different datasets of fifteen points each. The result is very similar in all three cases (i.e., the variance is low). d–f) A model with ten regions is fit to the same datasets. The additional flexibility does not necessarily produce better predictions. While these three models each describe the training data better, they are not necessarily closer to the true underlying function (black curve). Instead, they overfit the data and describe the noise, and the variance (difference between fitted curves) is larger.

Validation

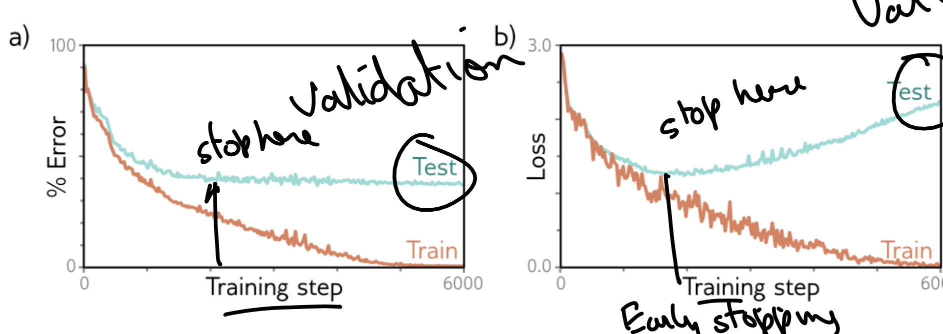
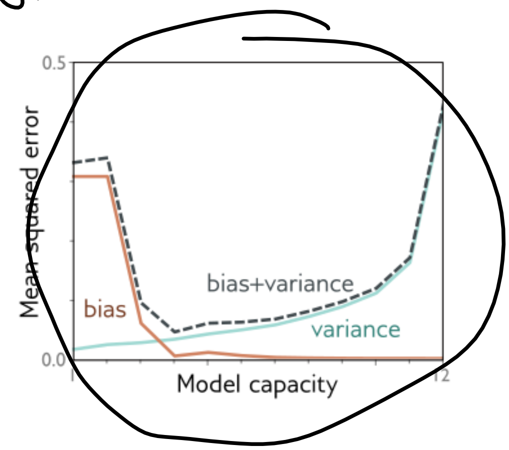
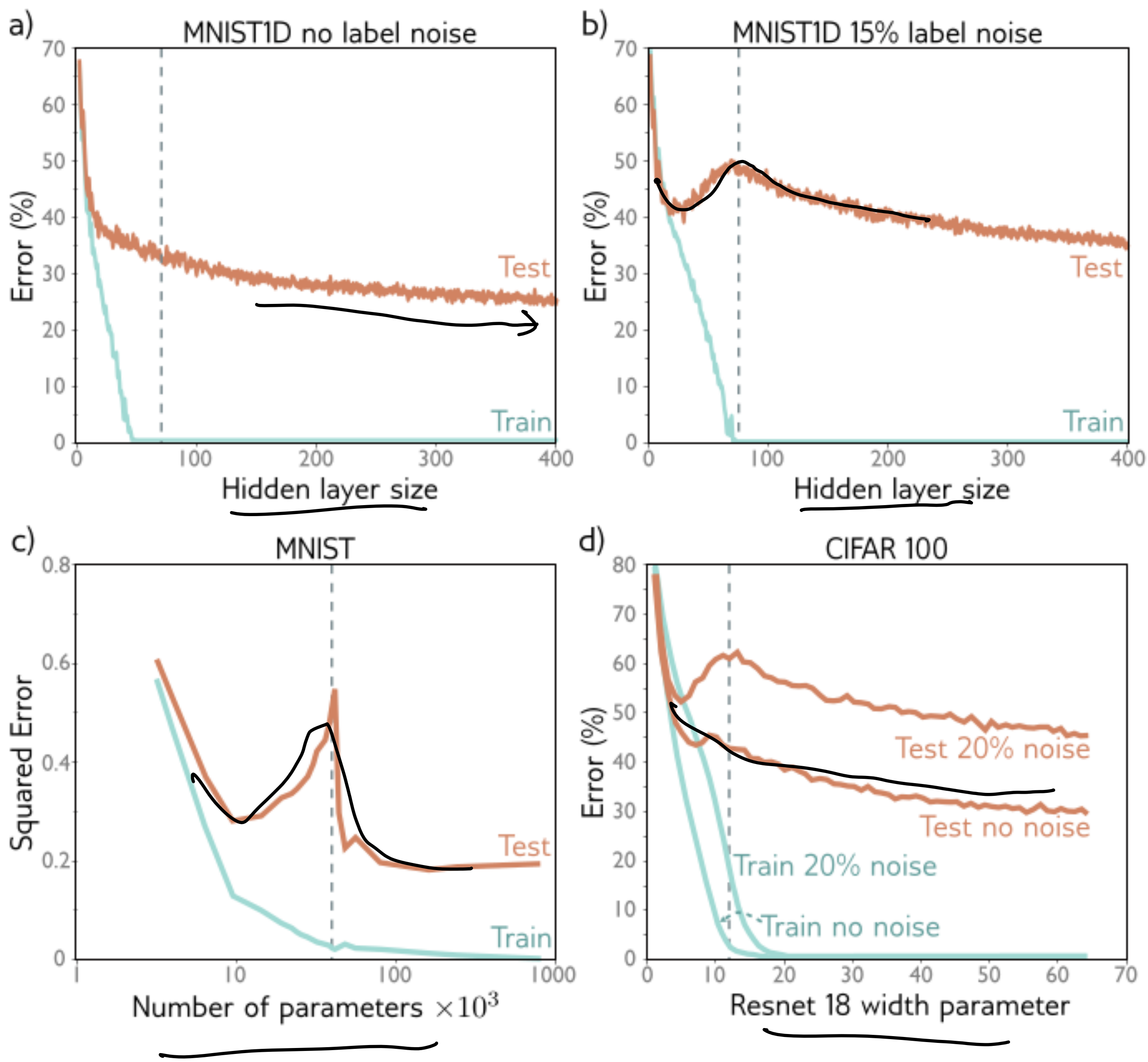


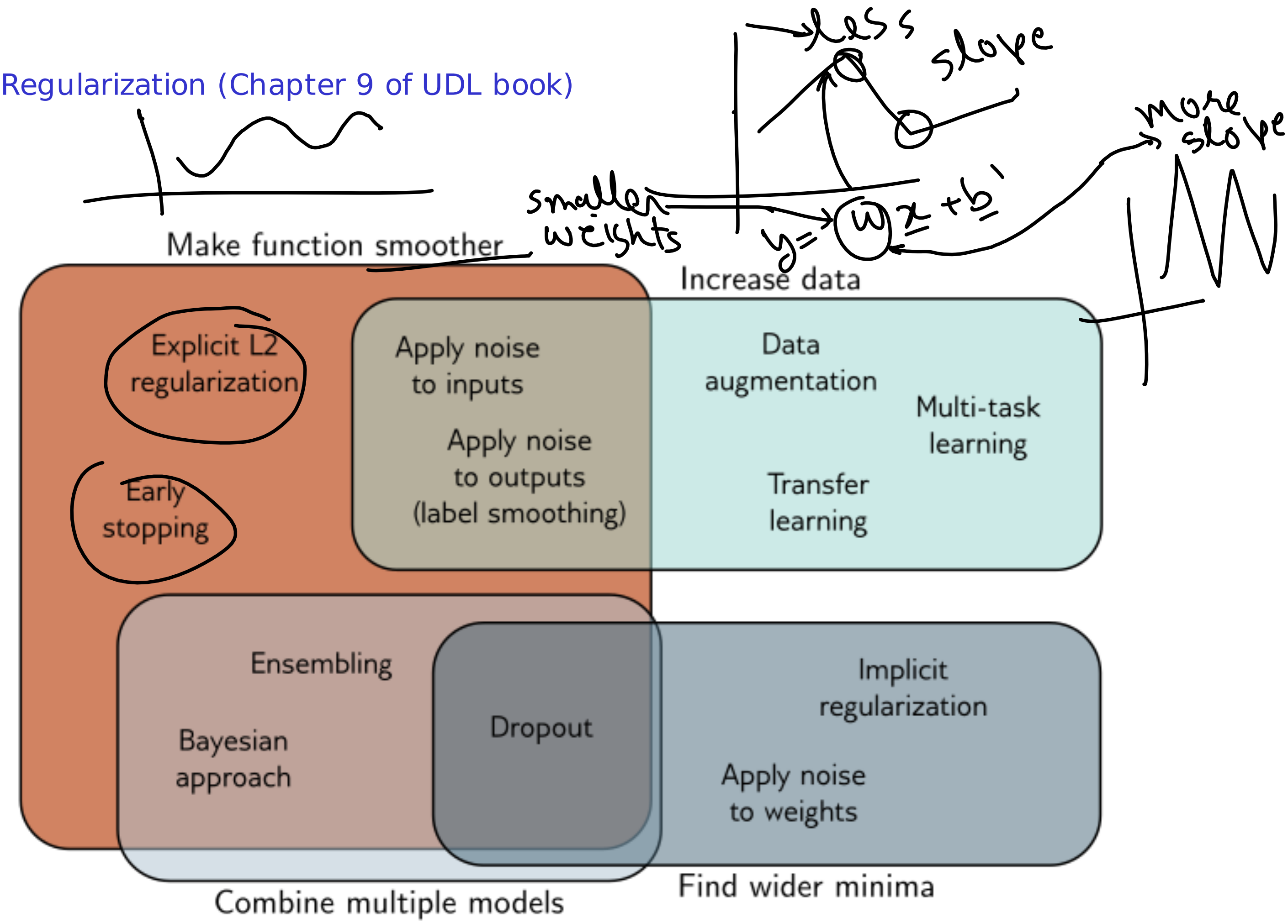
Figure 8.2 MNIST-1D results. a) Percent classification error as a function of the training step. The training set errors decrease to zero, but the test errors do not decrease below ∼ 40%. This model doesn't generalize well to new test data. b) Loss as a function of the training step. The training loss decreases steadily towards zero. The test loss decreases at first but then increases as the model becomes increasingly confident about its (wrong) predictions.



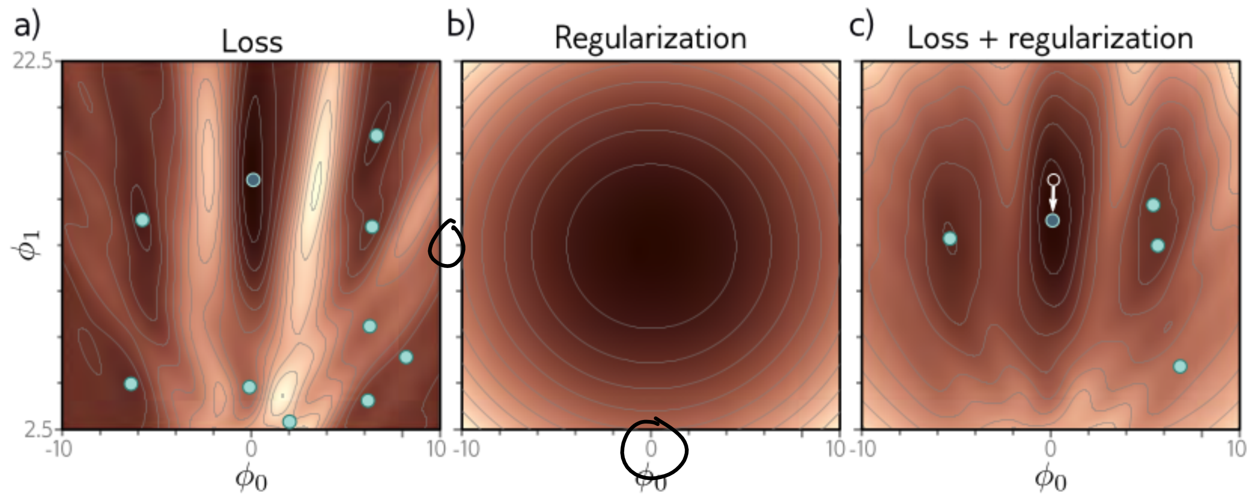
Aside: Model capacity in neural networks is weird though



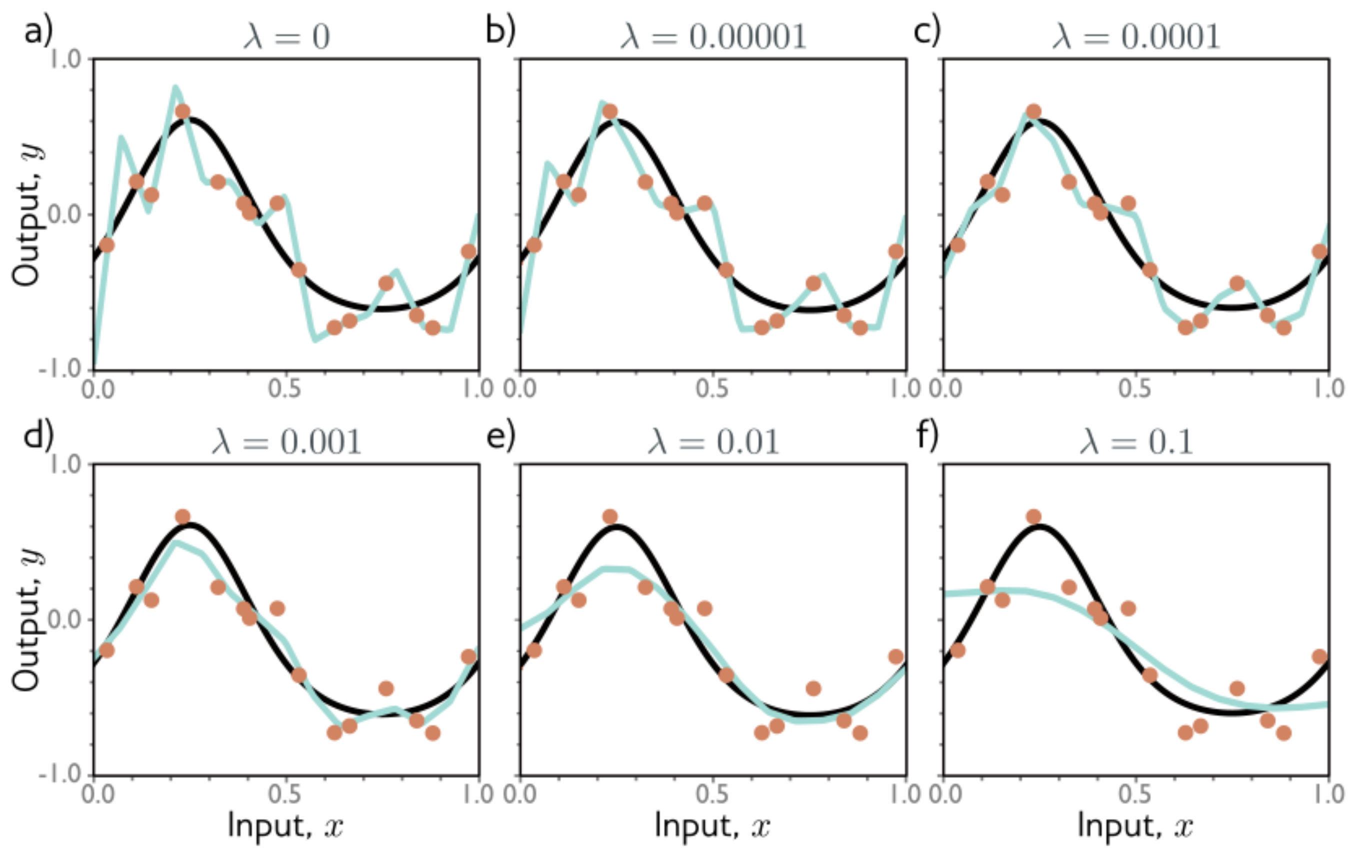
Regularization (Chapter 9 of UDL book)



Explicit regularization



L2 Regularization



L2-regularization

Mean squared loss for regression

Ex ①

$$L(D;W) = \sum_{i=1}^{D} \| y_i - f(x_i;W) \|_2^2$$

$$f(x_i;W) = \underline{w}_2^T a(W_1 \underline{x} + \underline{b}_1)$$

$$W = \{\underline{w}_2, W_1, \underline{b}_1\}$$

For L2-regularization

$$L(D_i;W) = \underbrace{\sum_{i=1}^{D} \| y_i - f(\underline{x}_i;W) \|_2^2}_{\text{fit to data}} + \underbrace{\overset{\text{positive scalar}}{\lambda} \overset{\text{L2-norm}}{\| W \|_2^2}}_{\text{regularization term}}$$

keep the weights small

$$W^* = \arg\min_{W} L(D_i;W)$$

L2-norm ? = vector magnitude

$$\|\underline{w}_2\|_2 = \sqrt{w_{21}^2 + w_{22}^2 + \dots + w_{2n}^2} = (w_{21}^2 + w_{22}^2 + \dots + w_{2n})^{1/2}$$

$$\underline{w}_2 = \begin{bmatrix} w_{21} \\ w_{22} \\ \vdots \\ w_{2n} \end{bmatrix}$$

L1-norm

$$\|\underline{w}_2\|_1 = (|w_{21}| + |w_{22}| + \dots + |w_{2n}|)$$

Lp-norm

$$\|w_2\|_p = ((w_{21})^p + w_{22}^p + \dots + w_{2n}^p)^{1/p}$$

For Ex 1

$$w = \{ \underline{w}_2, W_1, \underline{b}_1 \}$$

$$\frac{\partial \lambda \|\underline{w}\|_2^2}{\partial \underline{w}} = \frac{\partial \lambda \underline{w}^T \underline{w}}{\partial \underline{w}} = 2\lambda \underline{w}$$

$$L(D; w) = \underbrace{\| \quad \quad \|}_{\text{Data term}} + \lambda \underbrace{\|w\|_2^2}$$

In neural networks, regularization is restricted to weights not biases and it is called <u>weight decay</u>

Regularization term

$$\lambda \|w\|_2^2 = \lambda \|\underline{w}_2\|_2^2 + \lambda \|\text{flatten}(W_1)\|_2^2$$

$$= \lambda \left\| \begin{matrix} \underline{w}_2 \\ \text{flatten}(W_1) \end{matrix} \right\|_2^2$$

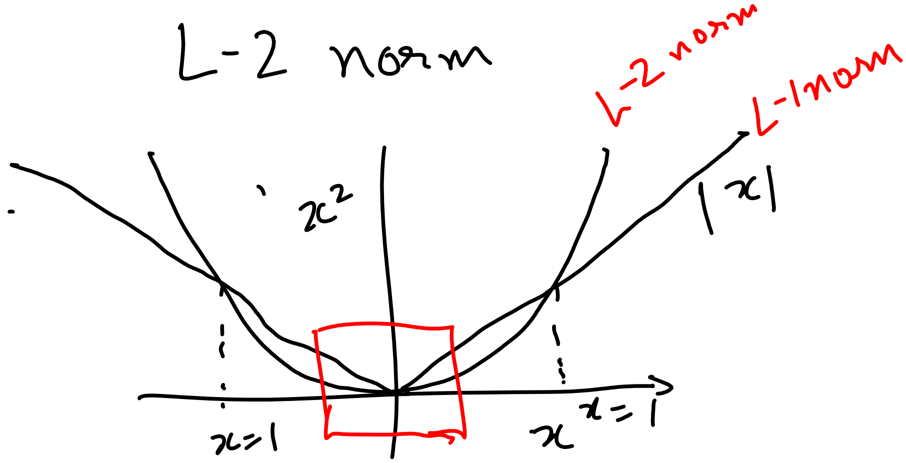
L-1 vs L-2

Regularization

$x = 10^{-3}$

$|x| = 10^{-3}$

$x^2 = \underline{10^{-6}}$ much smaller



Regularizing with L-1 norm, then smaller values are made even smaller

L-1 regularization leads to $\boxed{\text{sparse weights}}$

large weights are okay but smaller weights become zero

without regularization

$$\underline{W} = \begin{bmatrix} 1.2 \\ 10.45 \\ 10^{-6} \\ \vdots \end{bmatrix} \xrightarrow{L2} \begin{bmatrix} 0.8 \\ 5.2 \\ 10^{-4} \\ \vdots \end{bmatrix}$$

$$\underline{W} \xrightarrow{L-1} \begin{bmatrix} 0.9 \\ 8.4 \\ \boxed{0} \end{bmatrix}$$

$w_{t+1} = w_t - \alpha \nabla_{w_t} \ell(D: w_t)$

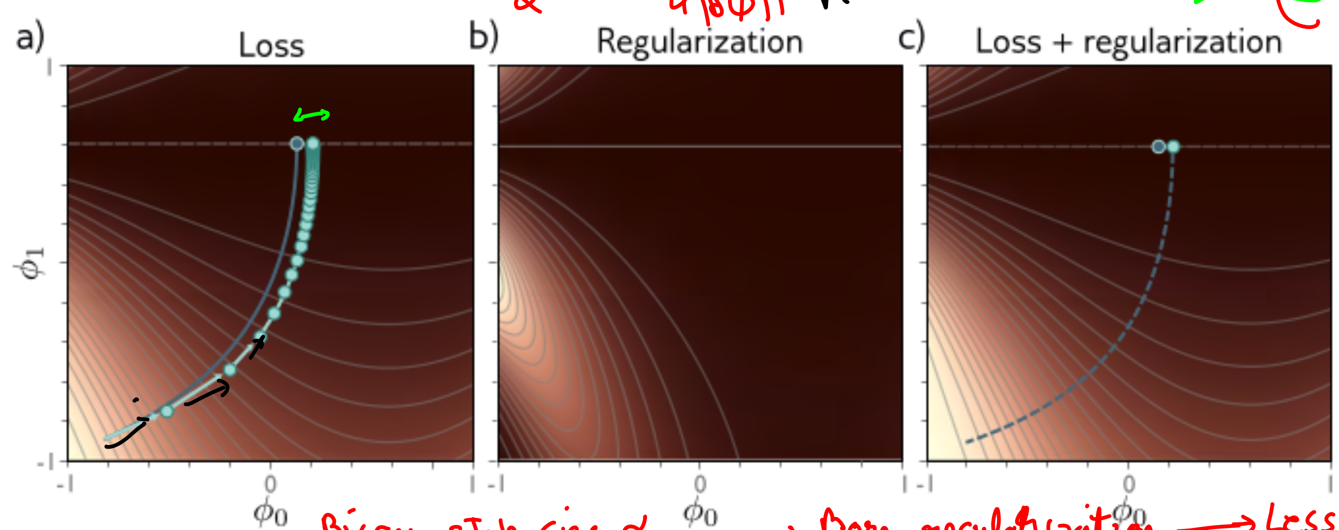
finite

Theoretically $w(t+h) = w(t) - h \nabla_w \ell(D; w(t))$

$\alpha \rightarrow \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \phi}\right\|^2$ $\lim_{h \to 0}$

Implicit regularization in gradient descent

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

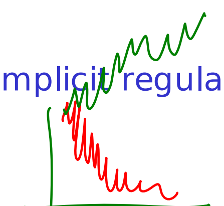


Bigger step size α Learning rate → More regularization → Less overfitting

Figure 9.3 Implicit regularization in gradient descent. a) Loss function with family of global minima on horizontal line $\phi_1 = 0.61$. Dashed blue line shows continuous gradient descent path starting in bottom left. Cyan trajectory shows discrete gradient descent with step size 0.1 (first few steps shown explicitly as arrows). The finite step size causes the paths to diverge and to reach a different final position. b) This disparity can be approximated by adding a regularization term to the continuous gradient descent loss function that penalizes the squared gradient magnitude . c) After adding this term, the continuous gradient descent path converges to the same place that the discrete one did on the original function.

Smaller batch sizes → More regularization → Less overfitting

Implicit regularization in stochastic gradient descent



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

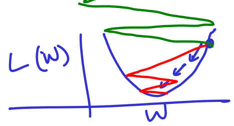
$$= \underbrace{L[\phi] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \phi}\right\|^2}_{GD} + \underbrace{\frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \phi} - \frac{\partial L}{\partial \phi}\right\|^2}_{\text{variance of batch loss}}.$$

Batch loss

Full data loss

total batches in the Dataset

variance of batch loss will be greater for smaller batch sizes



- smallest lr
- bigger lr
- so big that loss diverging

Effect of batch size and learning rate

Thumb rule: pick the biggest learning rate for which your loss does not diverge

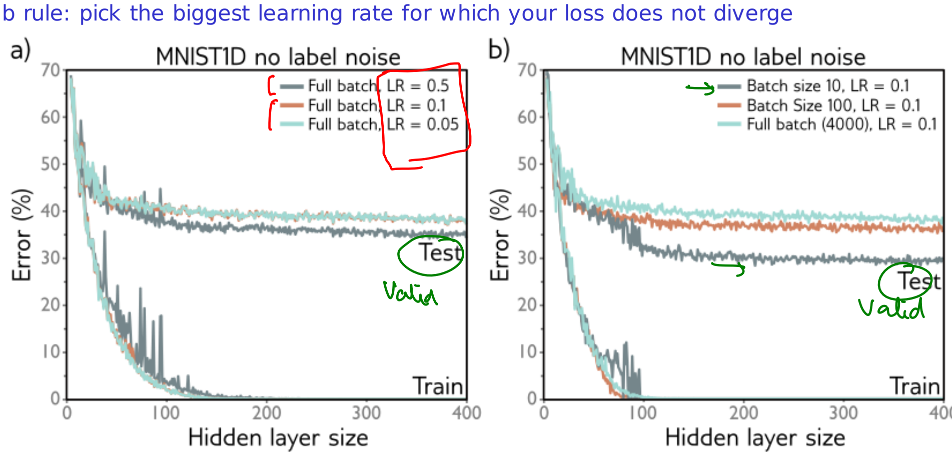
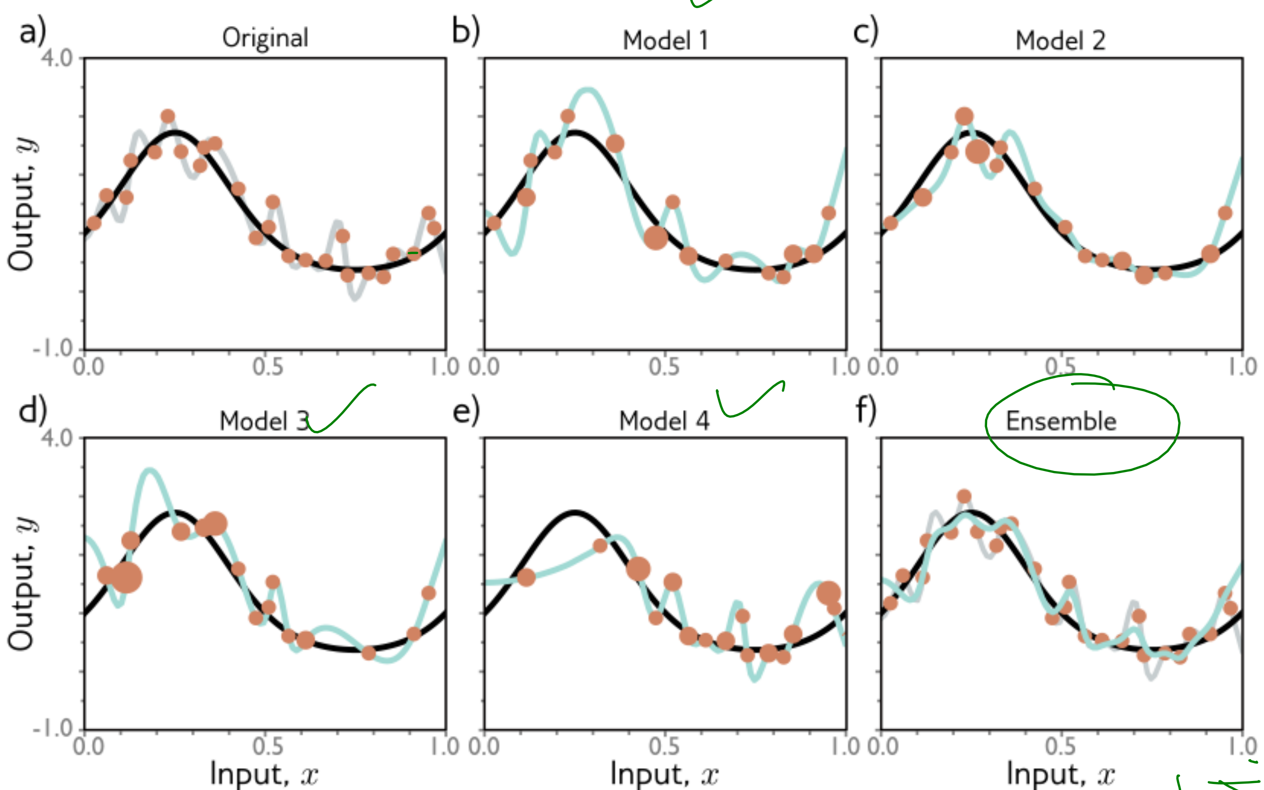


Figure 9.5 Effect of learning rate and batch size for 4000 training and 4000 test examples from MNIST-1D (see figure 8.1) for a neural network with two hidden layers. a) Performance is better for large learning rates than for intermediate or small ones. In each case, the number of iterations is 6000× the learning rate so each solution has the opportunity to move the same distance. b) Performance is superior for smaller batch sizes. In each case the number of iterations was chosen so that the training data were memorized at roughly the same model capacity.

Ensemble methods



f_1

f_2

f_3

$\frac{1}{4}\sum_i f_1+f_2+f_3+f_4$

Figure 9.7 Ensemble methods. a) Fitting a single model (gray curve) to the entire dataset (orange points). b-e) Four models created by re-sampling the data with replacement (bagging) four times (size of orange point indicates number of times the data point was re-sampled). f) When we average the predictions of this ensemble, the result (cyan curve) is smoother than the result from panel (a) for the full dataset (gray curve) and will probably generalize better.

Dropout

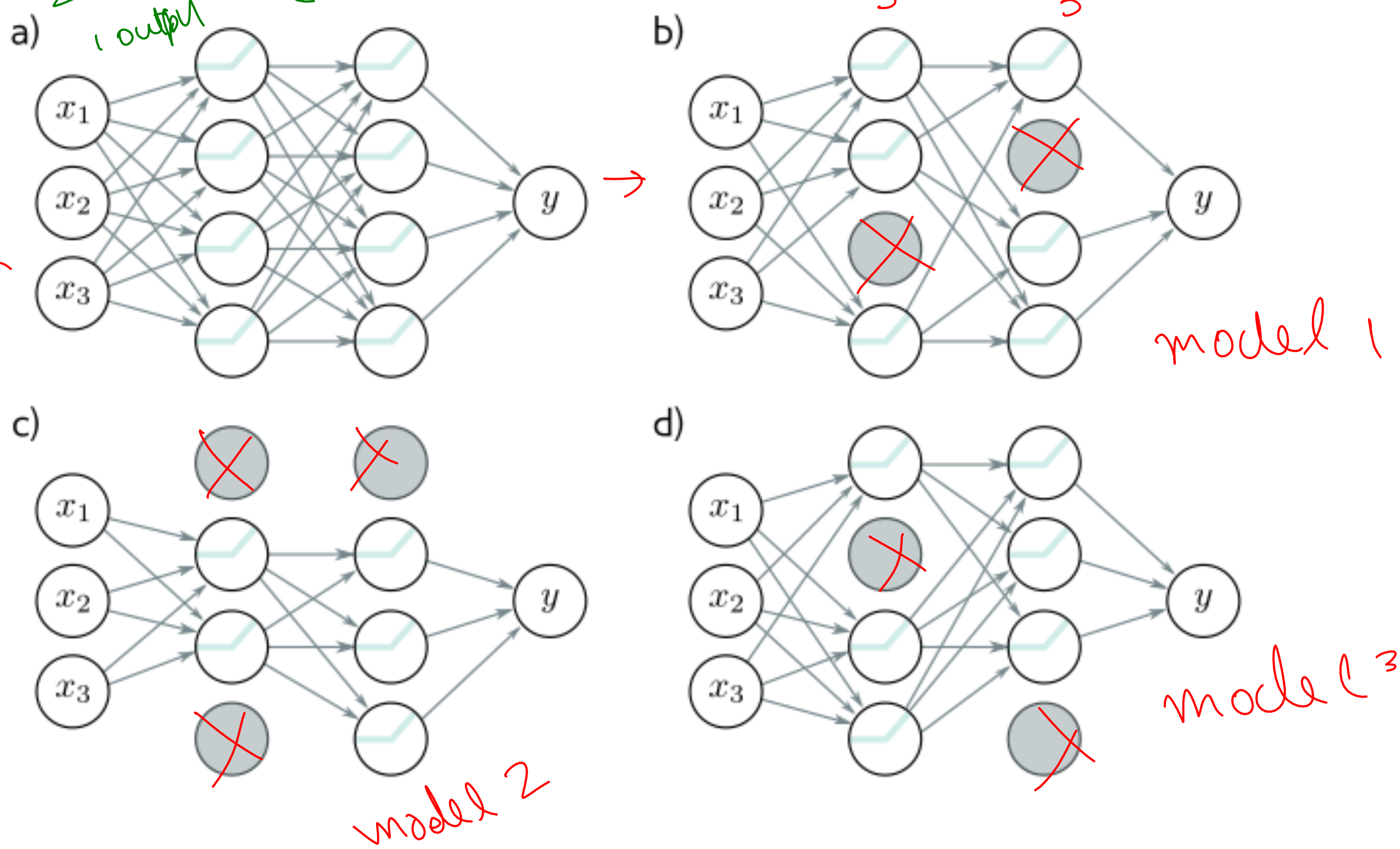
3 inputs

3 Layer MLP

2 ReLU, (4,4) hidden units

1 output

Training a random smaller network than the original

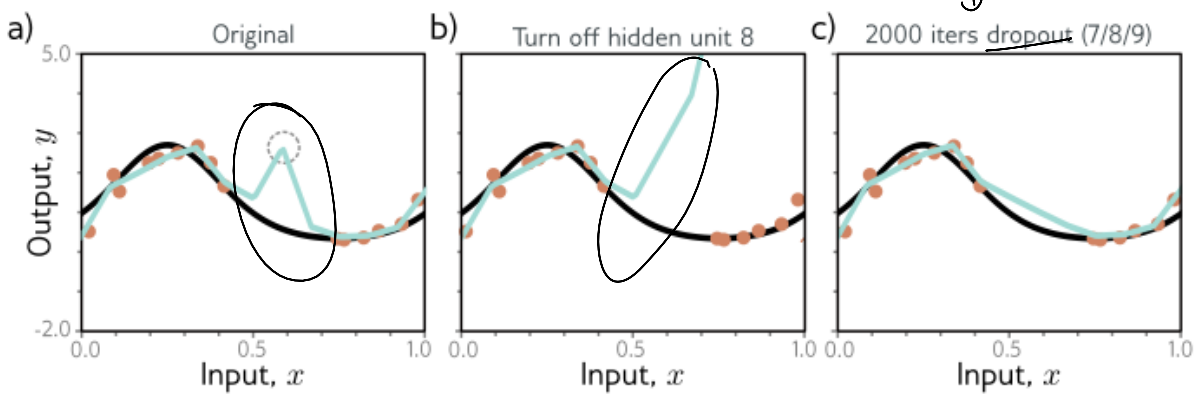


Being the average of mode 1,2,3

Dropout layers → Create an implicit ensemble of subnetwork → Regularize → reduce overfitting

Effect of dropout

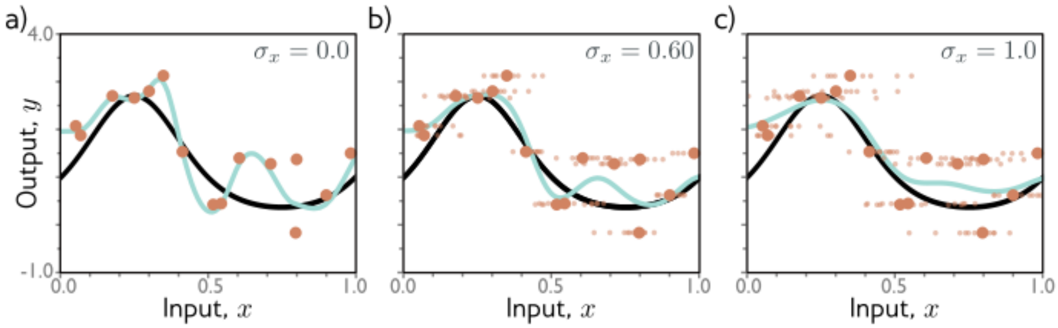
Dropout reduces dependence of
a network on specific hidden units.
Because any hidden unit can
be turned off anytime.



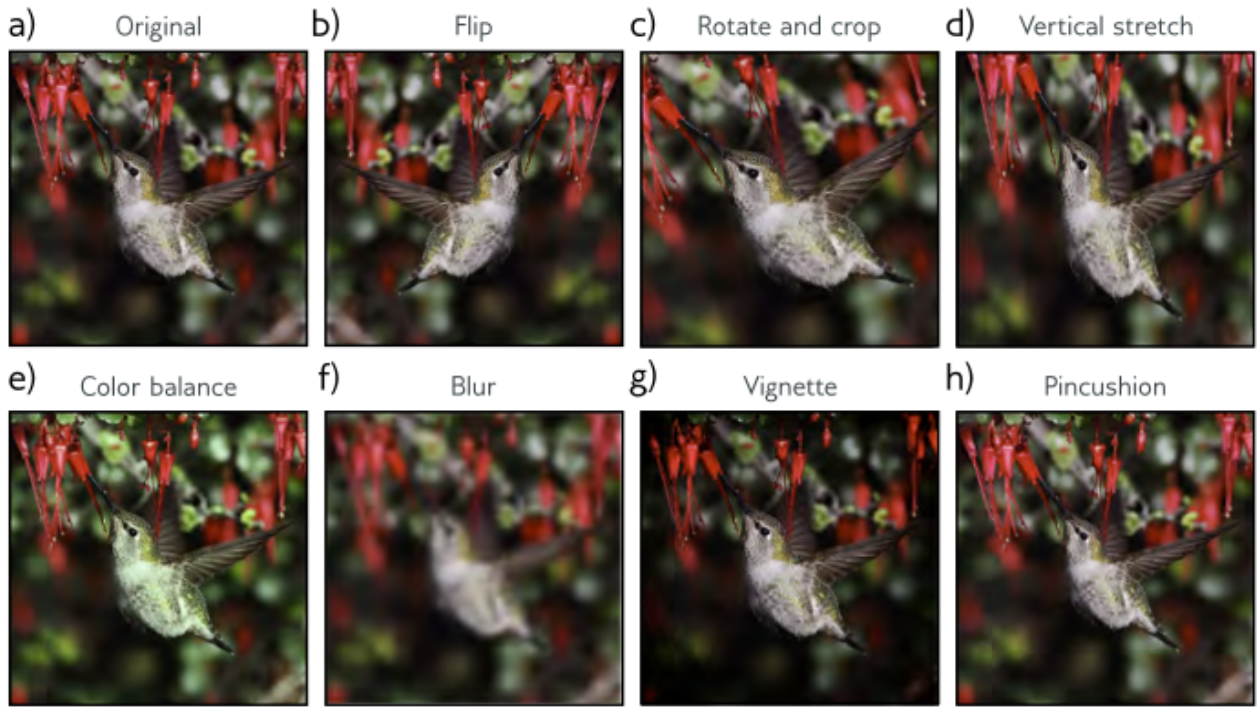
Adding noise to each batch

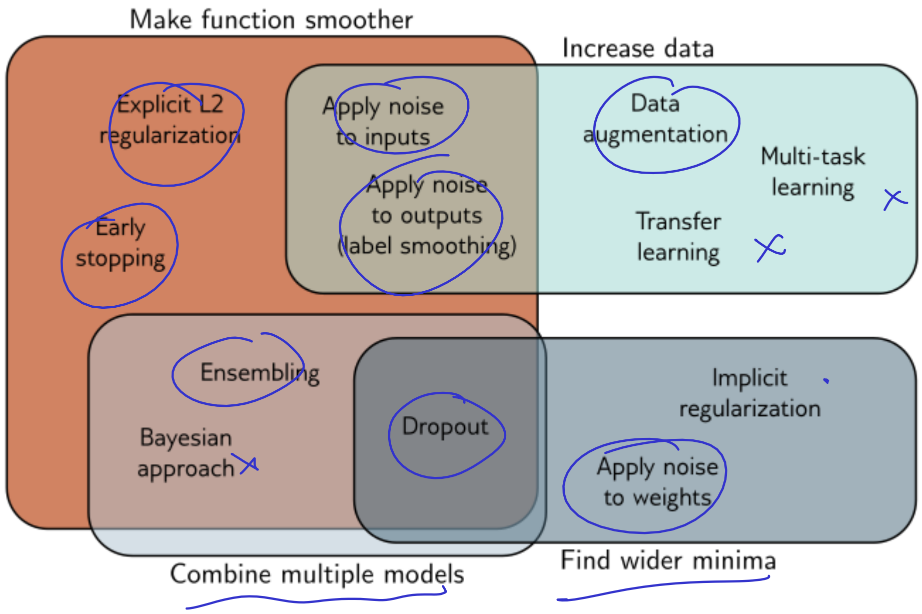
You can add noise to the inputs, outputs or the weights during training time.

The training process will make your learned function to be robust to these noises making it "smoother"



Data augmentation





(Batch size learning)