Deep Learning

05 - Model Training. Measuring Performance and Regularization

SAPIENZA
UNIVERSITÀ DI ROMA

Fabrizio Silvestri

Recapping

Recapping

- We have, more or less deeply (pun intended!), described neural networks:
  - Models
  - Loss functions, and
  - Training algorithms.
- How do we know that we have done a good job?
  - I.e., how to measure the performance of the trained models.
- With sufficient capacity (i.e., number of hidden units), a neural network model will often perform perfectly on the training data.
  - However, this does not necessarily mean it will generalize well to new test data.

Measuring Performance

(Chapter 8)

SAPIENZA
UNIVERSITÀ DI ROMA

Training a simple model

- We explore model performance using the MNIST-1D dataset
  - This consists of ten classes $y \in \{0, 1, \ldots, 9\}$, representing the digits 0–9. The data are derived from 1D templates for each of the digits.
- Each data example x is created by randomly transforming one of these templates and adding noise.
  - The full training dataset $\{x_i, y_i\}$ consists of $I = 4{,}000$ training examples,
  - each consisting of $D_i = 40$ dimensions representing the horizontal offset at 40 positions.
  - The ten classes are drawn uniformly during data generation, so there are ~400 examples of each class.
- We use a network with $D_i = 40$ inputs and $D_o = 10$ outputs which are passed through a softmax function to produce class probabilities.
  - The network has two hidden layers with $D = 100$ hidden units each.
  - It is trained using stochastic gradient descent with batch size 100 and learning rate 0.1 for 6,000 steps (150 epochs) with a multiclass cross-entropy loss.

Training a simple model

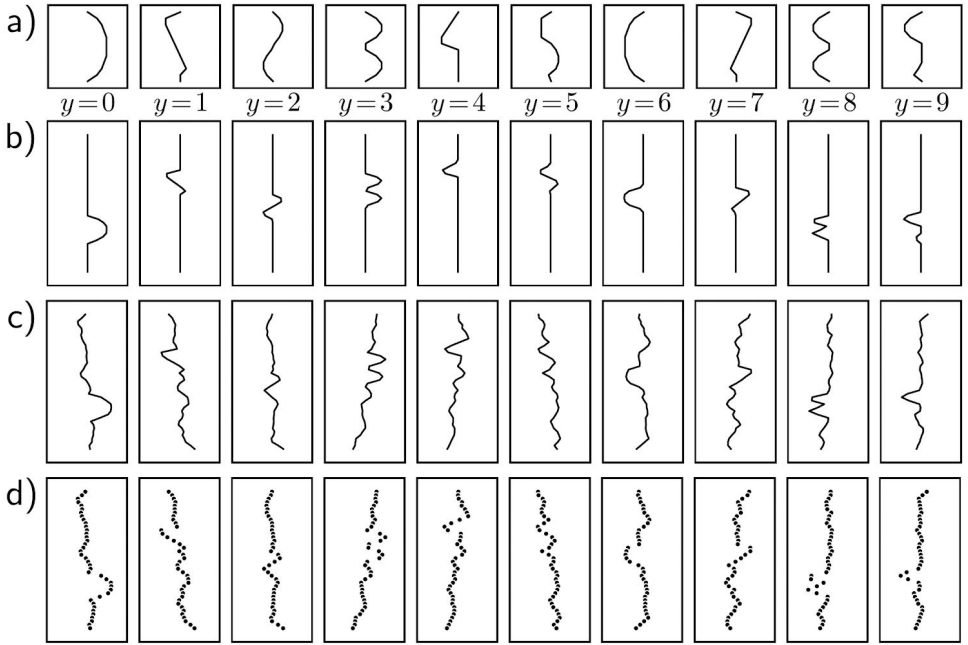
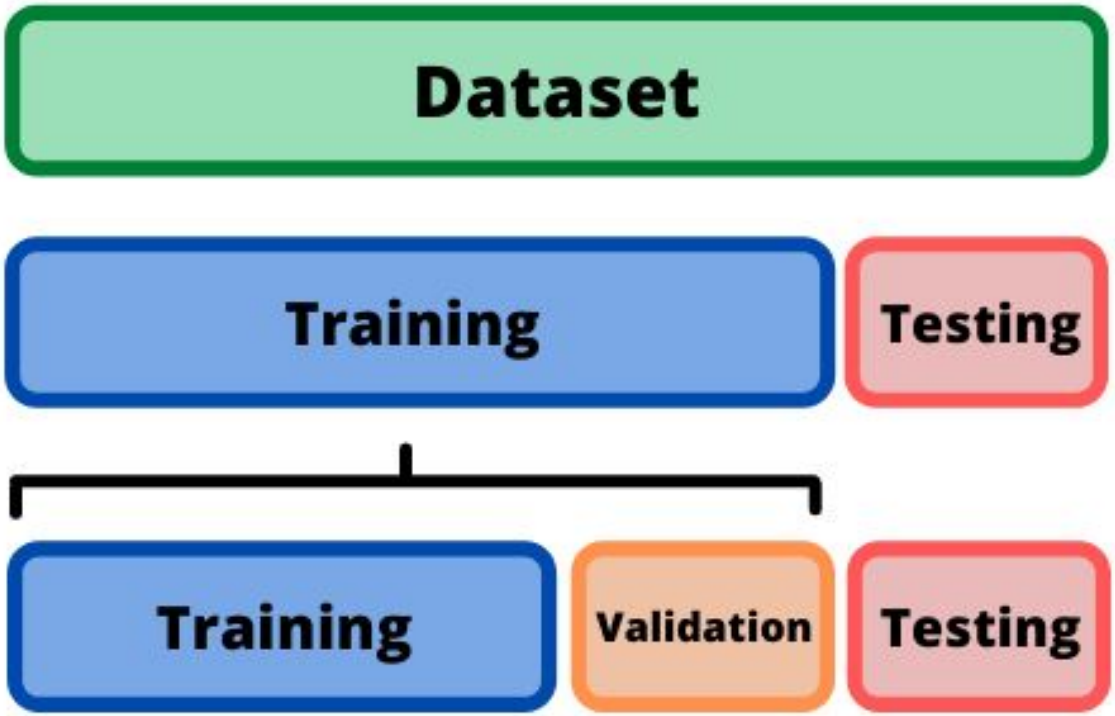


Figure 8.1 MNIST-1D. a) Templates for 10 classes $y \in \{0, \ldots, 9\}$, based on digits 0–9. b) Training examples $\mathbf{x}$ are created by randomly transforming a template and c) adding noise. d) The horizontal offset of the transformed template is then sampled at 40 vertical positions. Adapted from (Greydanus, 2020)

Testing the model



DANGER

Never, ever, test on the training or train on the test set

Colab time: MNIST 1D Performance

https://colab.research.google.com/drive/1mHr2uHM2tx6B4-rMECL_4hOuXG3e3Mnx

Solution: MNIST 1D Performance

https://colab.research.google.com/drive/181rI-0KWi3RekXssgVRNjCeolZly5f1d

Learning Curve

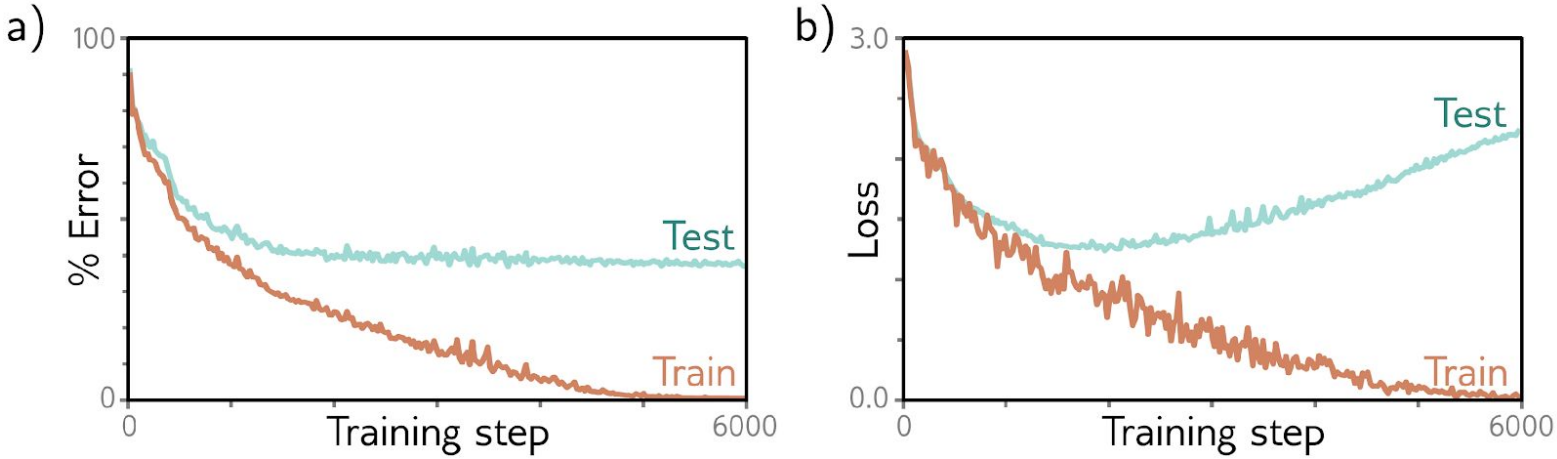
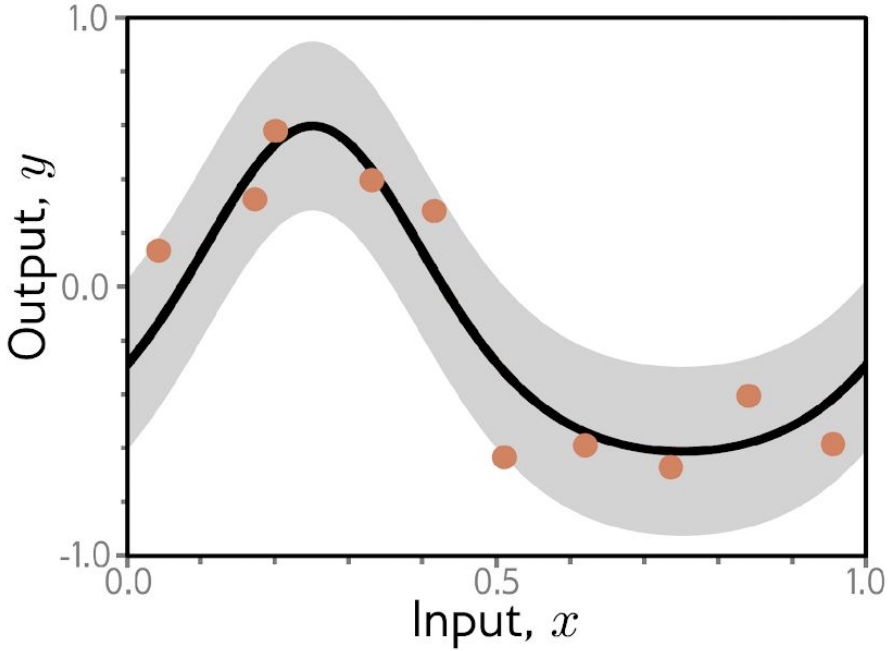


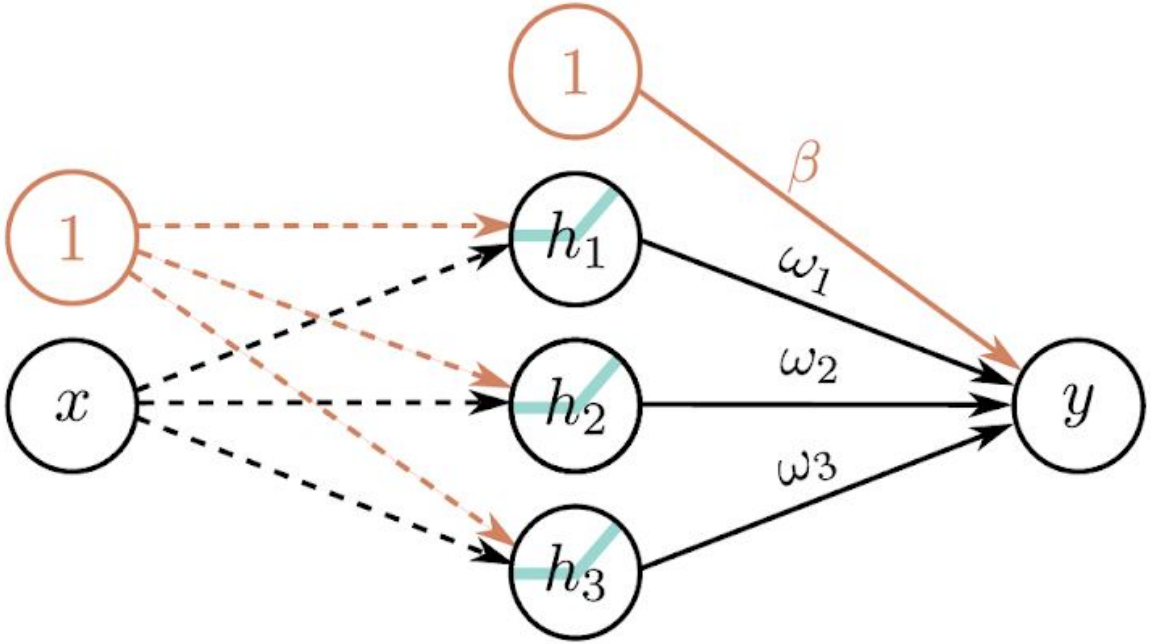
Figure 8.2 MNIST-1D results. a) Percent classification error as a function of the training step. The training set errors decrease to zero, but the test errors do not drop below $\sim 40\%$. This model doesn't generalize well to new test data. b) Loss as a function of the training step. The training loss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

Sources of Errors: Example Dataset

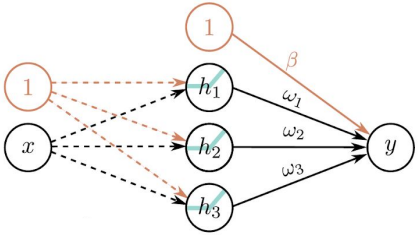
Figure 8.3 Regression function. Solid black line shows the ground truth function. To generate I training examples $\{x_i, y_i\}$, the input space $x \in [0, 1]$ is divided into I equal segments and one sample x_i is drawn from a uniform distribution within each segment. The corresponding value y_i is created by evaluating the function at x_i and adding Gaussian noise (gray region shows ± 2 standard deviations). The test data are generated in the same way.

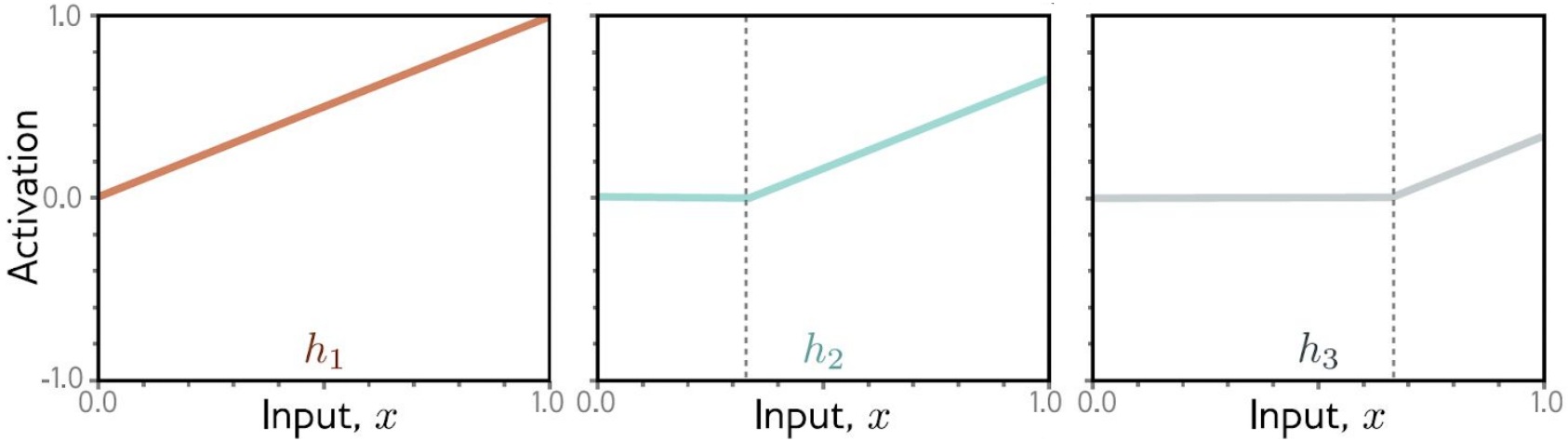


Sources of Errors: The Model



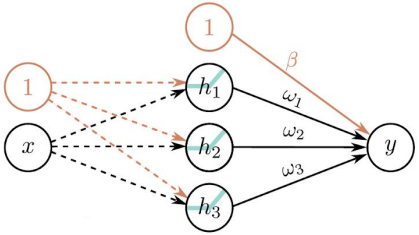
Sources of Errors: The Model

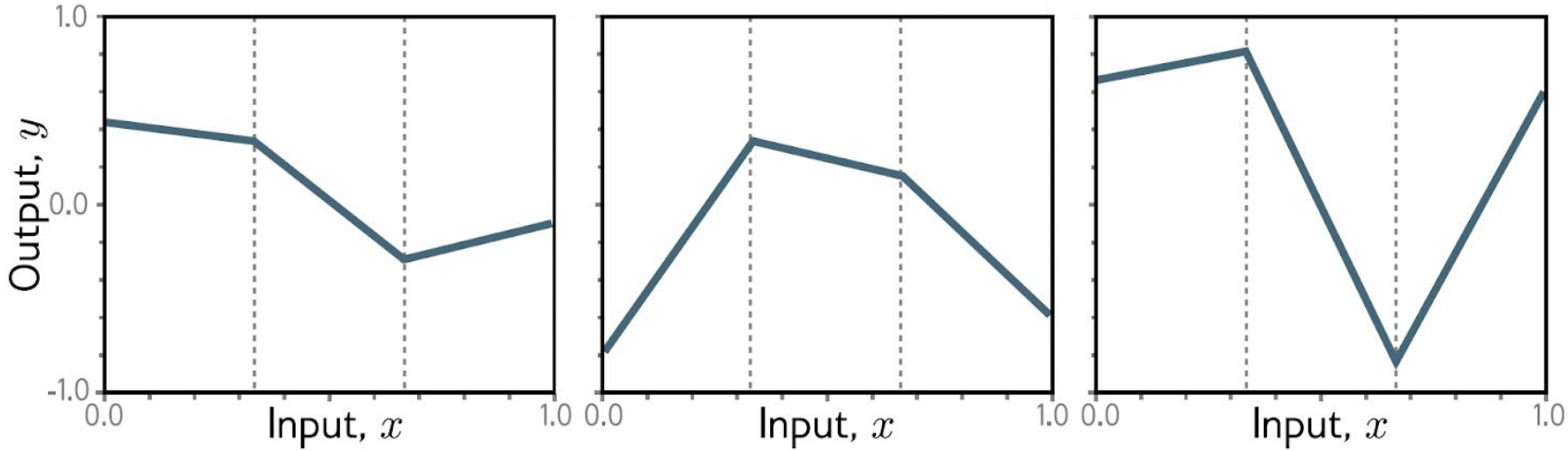




The weights and biases between the input and hidden layer are fixed (dashed arrows). They are chosen so that the hidden unit activations have slope one, and their joints are equally spaced across the interval, with joints at x = 0, x = 1/3, and x = 2/3, respectively.

Sources of Errors: The Model

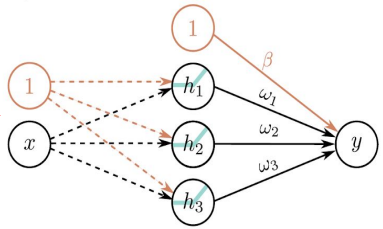




Modifying the remaining parameters $\phi = \{\beta, \omega_1, \omega_2, \omega_3\}$ one can create any piecewise linear function over $x \in [0, 1]$ with joints at 1/3 and 2/3. Here are three example functions with different values of the parameters ϕ.

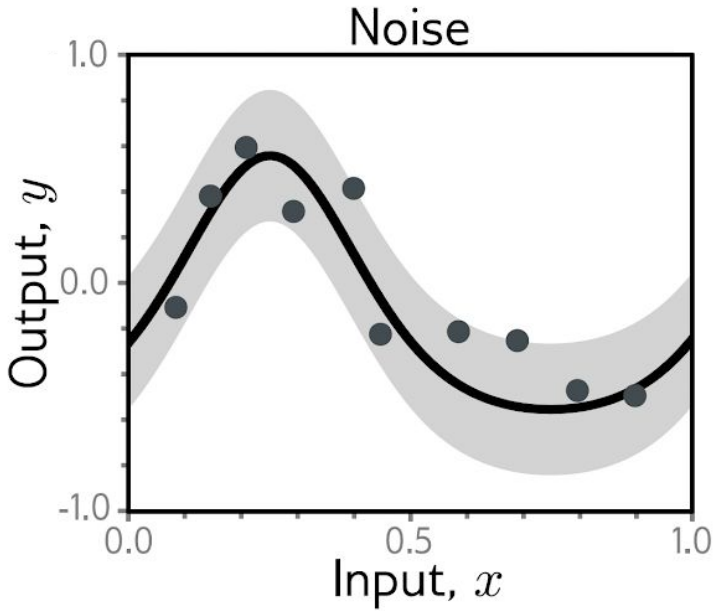
SAPIENZA
UNIVERSITÀ DI ROMA

Exercise Time



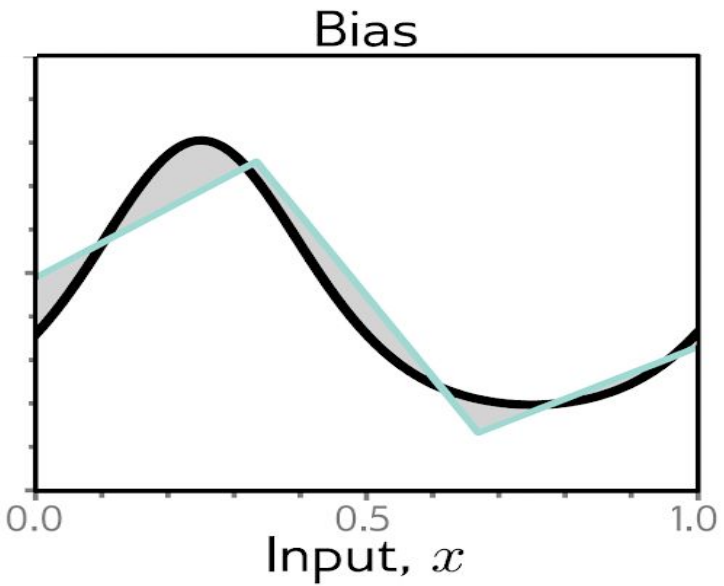
Problem 8.3 Given a training dataset consisting of I input/output pairs $\{x_i, y_i\}$, show how the parameters $\{\beta, \omega_1, \omega_2, \omega_3\}$ for the model in figure 8.4a using the least squares loss function can be found in closed form.

Noise



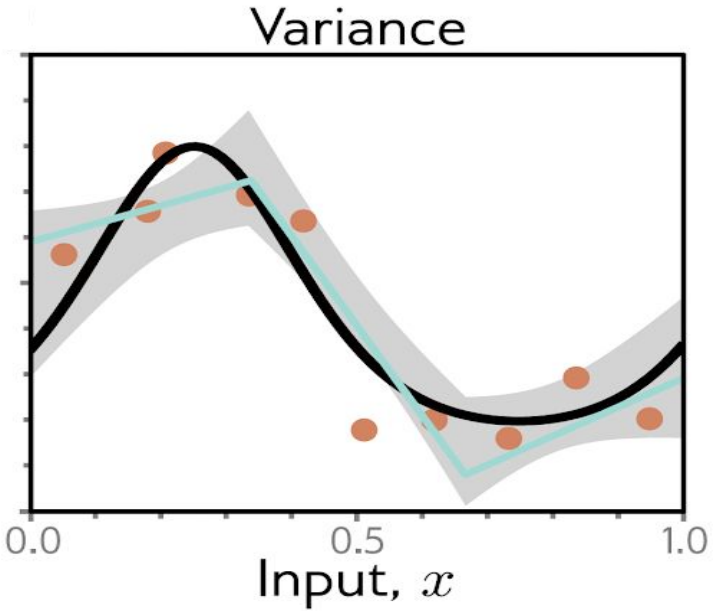
- The data generation process includes the addition of noise, so there are multiple possible valid outputs y for each input x.
- This source of error is insurmountable for the test data.
  - Note that it does not necessarily limit the training performance; e.g., the exact same input twice is a rare event.
- Noise may arise from:
  - genuine stochastic element to the data generation process
  - data are mislabeled
  - further explanatory variables that were not observed.
  - In rare cases, noise may be absent.

Bias



- A second potential source of error may occur because the model is not flexible enough to fit the true function perfectly.
  - For example, the three-region neural network
- model cannot exactly describe the quasi-sinusoidal function, even when the parameters are chosen optimally.

Noise



- We have limited training examples, and there is no way to distinguish systematic changes in the underlying function from noise in the underlying data.
- When we fit a model, we do not get the closest possible approximation to the true underlying function.
  - Indeed, for different training datasets, the result will be slightly different each time.
- In practice, there might also be additional variance due to the stochastic learning algorithm, which does not necessarily converge to the same solution each time.

Tying things together with Maths

$$\mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_y[L[x]]\Big] = \underbrace{\mathbb{E}_{\mathcal{D}}\Big[\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)^2\Big]}_{\text{variance}} + \underbrace{\big(\mathrm{f}_\mu[x] - \mu[x]\big)^2}_{\text{bias}} + \underbrace{\sigma^2}_{\text{noise}}.$$

- In the case of Linear Regression, we can easily express the expected value of the loss function L[x] with respect to the output y on a data distribution D as the sum of variance, bias, and noise.
- Decreasing the variance corresponds to an increase in the bias and vice versa. Noise it's not depending on the training data but it's intrinsic to the data generation process.

How to Prove that

We now make the notions of noise, bias, and variance mathematically precise. Consider a 1D regression problem where the data generation process has additive noise with variance σ^2 (e.g., figure 8.3); we can observe different outputs y for the same input x, so for each x, there is a distribution $Pr(y|x)$ with expected value (mean) $\mu[x]$:

$$\mu[x] = \mathbb{E}_y[y[x]] = \int y[x]Pr(y|x)dy, \tag{8.1}$$

How to Prove that

We now make the notions of noise, bias, and variance mathematically precise. Consider a 1D regression problem where the data generation process has additive noise with variance σ^2 (e.g., figure 8.3); we can observe different outputs y for the same input x, so for each x, there is a distribution $Pr(y|x)$ with expected value (mean) $\mu[x]$:

$$\mu[x] = \mathbb{E}_y[y[x]] = \int y[x] Pr(y|x) dy, \tag{8.1}$$

and fixed noise $\sigma^2 = \mathbb{E}_y\left[(\mu[x] - y[x])^2\right]$. Here we have used the notation $y[x]$ to specify that we are considering the output y at a given input position x.

How to Prove that

Now consider a least squares loss between the model prediction $\mathrm{f}[x, \boldsymbol{\phi}]$ at position x and the observed value $y[x]$ at that position:

$$\begin{aligned} L[x] &= \big(\mathrm{f}[x, \boldsymbol{\phi}] - y[x]\big)^2 \\ &= \Big(\big(\mathrm{f}[x, \boldsymbol{\phi}] - \mu[x]\big) + \big(\mu[x] - y[x]\big)\Big)^2 \\ &= \big(\mathrm{f}[x, \boldsymbol{\phi}] - \mu[x]\big)^2 + 2\big(\mathrm{f}[x, \boldsymbol{\phi}]) - \mu[x]\big)\big(\mu[x] - y[x]\big) + \big(\mu[x] - y[x]\big)^2, \end{aligned} \tag{8.2}$$

where we have both added and subtracted the mean $\mu[x]$ of the underlying function in the second line and have expanded out the squared term in the third line.

How to Prove that

The underlying function is stochastic, so this loss depends on the particular $y[x]$ we observe. The expected loss is:

$$\begin{aligned}
\mathbb{E}_y\left[L[x]\right] &= \mathbb{E}_y\left[\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-y[x]\right)+\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-\mathbb{E}_y\left[y[x]\right]\right)+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\cdot 0+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+\sigma^2, \qquad (8.3)
\end{aligned}$$

How to Prove that

The expected model output $\mathrm{f}_\mu[x]$ with respect to all possible datasets $\mathcal{D}$ is hence:

$$\mathrm{f}_\mu[x] = \mathbb{E}_{\mathcal{D}}\Big[\mathrm{f}\big[x, \boldsymbol{\phi}[\mathcal{D}]\big]\Big]. \tag{8.4}$$

Returning to the first term of equation 8.3, we add and subtract $\mathrm{f}_\mu[x]$ and expand:

$$\begin{aligned}
&\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mu[x]\big)^2 \qquad (8.5)\\
&\quad = \Big(\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big) + \big(\mathrm{f}_\mu[x] - \mu[x]\big)\Big)^2 \\
&\quad = \big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)^2 + 2\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)\big(\mathrm{f}_\mu[x] - \mu[x]\big) + \big(\mathrm{f}_\mu[x] - \mu[x]\big)^2.
\end{aligned}$$

We then take the expectation with respect to the training dataset $\mathcal{D}$:

$$\mathbb{E}_{\mathcal{D}}\Big[\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mu[x]\big)^2\Big] = \mathbb{E}_{\mathcal{D}}\Big[\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)^2\Big] + \big(\mathrm{f}_\mu[x] - \mu[x]\big)^2, \tag{8.6}$$

SAPIENZA
UNIVERSITÀ DI ROMA

How to Prove that

- You then replace with the following

$$\mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_y[L[x]]\Big] = \underbrace{\mathbb{E}_{\mathcal{D}}\Big[\big(\mathrm{f}[x,\boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)^2\Big]}_{\text{variance}} + \underbrace{\big(\mathrm{f}_\mu[x] - \mu[x]\big)^2}_{\text{bias}} + \underbrace{\sigma^2}_{\text{noise}}.$$

$$\begin{aligned}
\mathbb{E}_y\big[L[x]\big] &= \mathbb{E}_y\Big[\big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)^2 + 2\big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)\big(\mu[x] - y[x]\big) + \big(\mu[x] - y[x]\big)^2\Big] \\
&= \big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)^2 + 2\big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)\big(\mu[x] - \mathbb{E}_y\left[y[x]\right]\big) + \mathbb{E}_y\left[\big(\mu[x] - y[x]\big)^2\right] \\
&= \big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)^2 + 2\big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)\cdot 0 + \mathbb{E}_y\Big[\big(\mu[x] - y[x]\big)^2\Big] \\
&= \big(\mathrm{f}[x,\boldsymbol{\phi}] - \mu[x]\big)^2 + \sigma^2,
\end{aligned}$$

Reducing Errors: Noise

Reducing Errors: Variance

- Recall that the variance results from limited noisy training data.
- Fitting the model to two different training sets results in slightly different parameters.
- It follows we can reduce the variance by increasing the quantity of training data.
- This averages out the inherent noise and ensures that the input space is well sampled.

Reducing Error

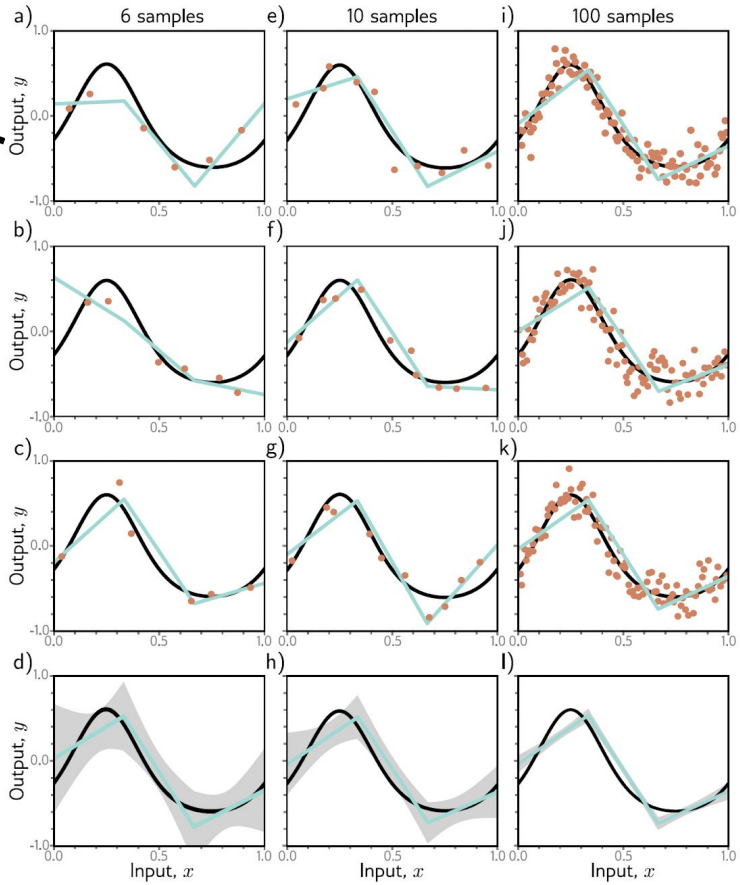


Figure 8.6 Reducing variance by increasing training data. a–c) The three-region model fitted to three different randomly sampled datasets of six points. The fitted model is quite different each time. d) We repeat this experiment many times and plot the mean model predictions (cyan line) and the variance of the model predictions (gray area shows two standard deviations). e–h) We do the same experiment, but this time with datasets of size ten. The variance of the predictions is reduced. i–l) We repeat this experiment with datasets of size 100. Now the fitted model is always similar, and the variance is small.

Reducing Errors: Bias

- The bias term results from the inability of the model to describe the true underlying function.
- This suggests that we can reduce this error by making the model more flexible.
  - This is usually done by increasing the model capacity.
    - For neural networks, this means adding more hidden units and/or hidden layers.
  - In the simplified model, adding capacity corresponds to adding more hidden units so that the interval [0, 1] is divided into more linear regions.

Reduc

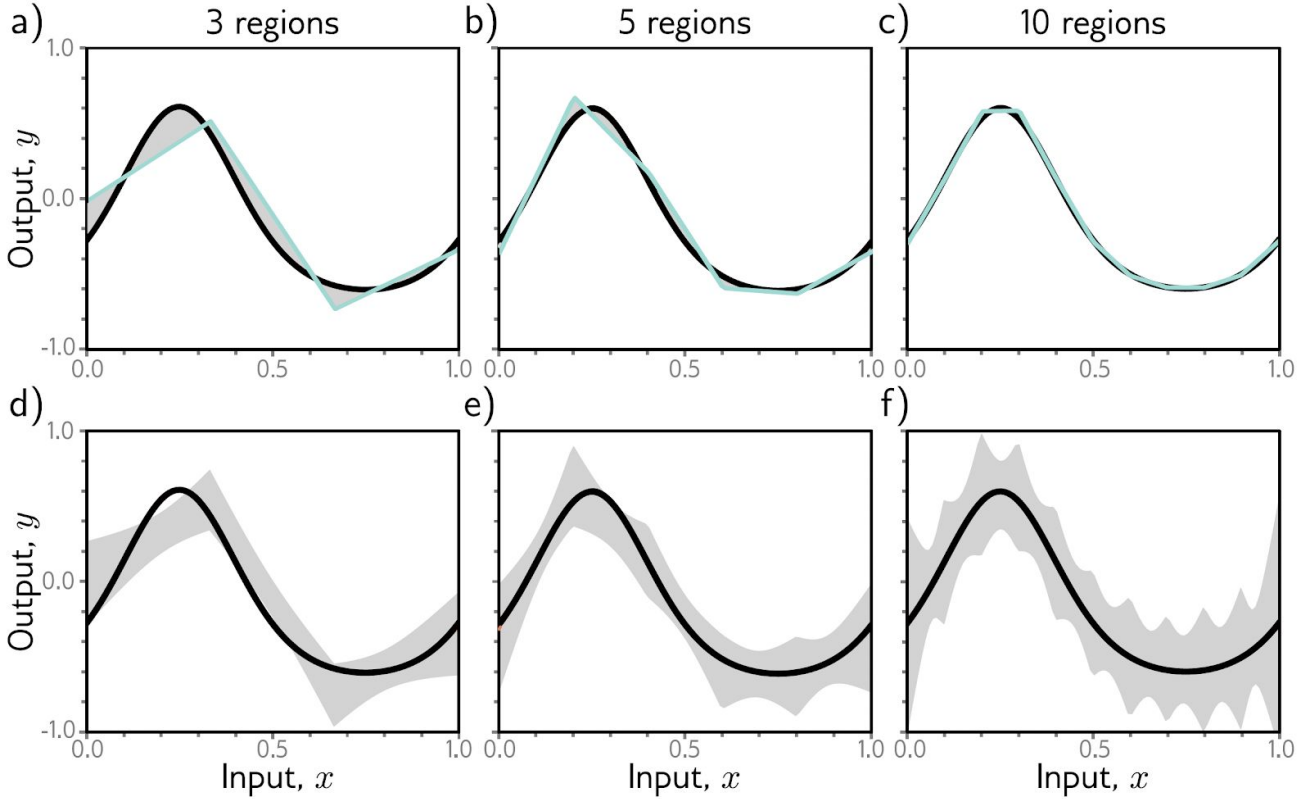


Figure 8.7 Bias and variance as a function of model capacity. a–c) As we increase the number of hidden units of the toy model, the number of linear regions increases, and the model becomes able to fit the true function closely; the bias (gray region) decreases. d–f) Unfortunately, increasing the model capacity has the side-effect of increasing the variance term (gray region). This is known as the bias-variance trade-off.

Bias-Variance Trade Off

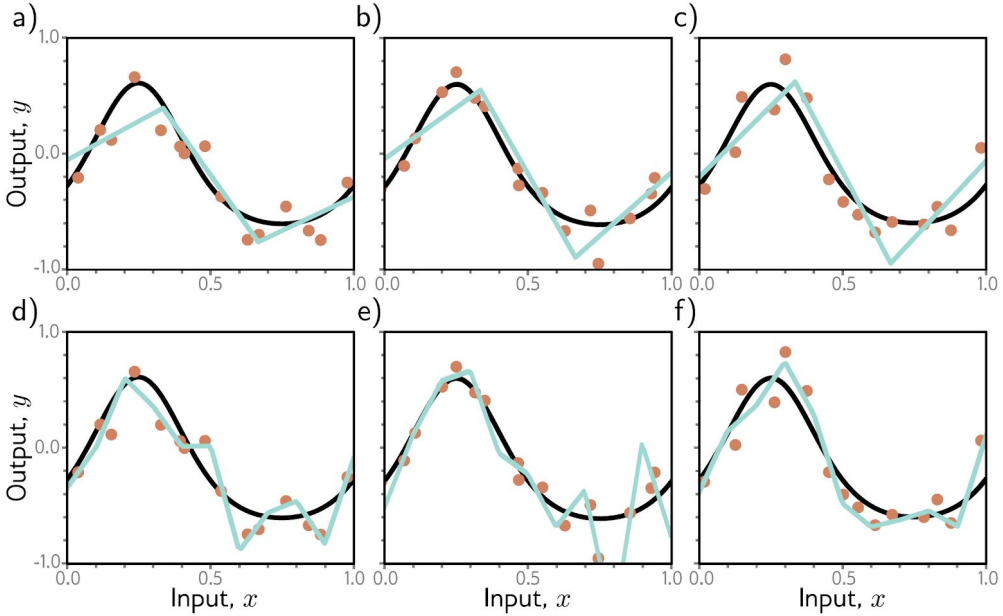
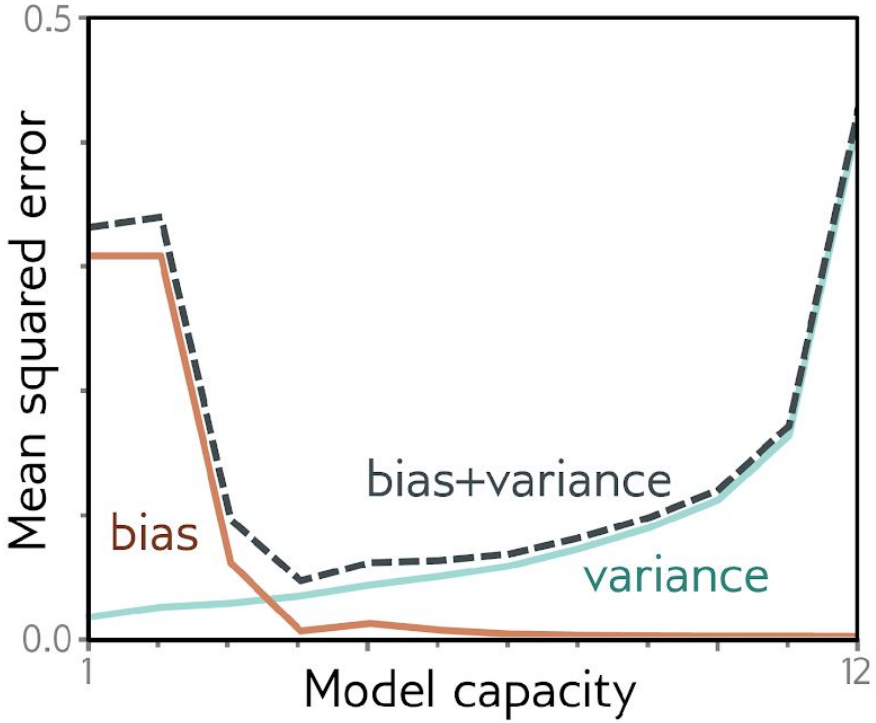


Figure 8.8 Overfitting. a–c) A model with three regions is fit to three different datasets of fifteen points each. The result is similar in all three cases (i.e., the variance is low). d–f) A model with ten regions is fit to the same datasets. The additional flexibility does not necessarily produce better 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.

Bias-Variance Trade Off

Figure 8.9 Bias-variance trade-off. The bias and variance terms from equation 8.7 are plotted as a function of the model capacity (number of hidden units / linear regions) in the simplified model using training data from figure 8.8. As the capacity increases, the bias (solid orange line) decreases, but the variance (solid cyan line) increases. The sum of these two terms (dashed gray line) is minimized when the capacity is four.



Colab time: Bias-Variance Trade Off

https://colab.research.google.com/drive/13GFH-HS-CWpHvoT6xH1qsutjLAUEnQt-

A Strange Effect: Double Descent

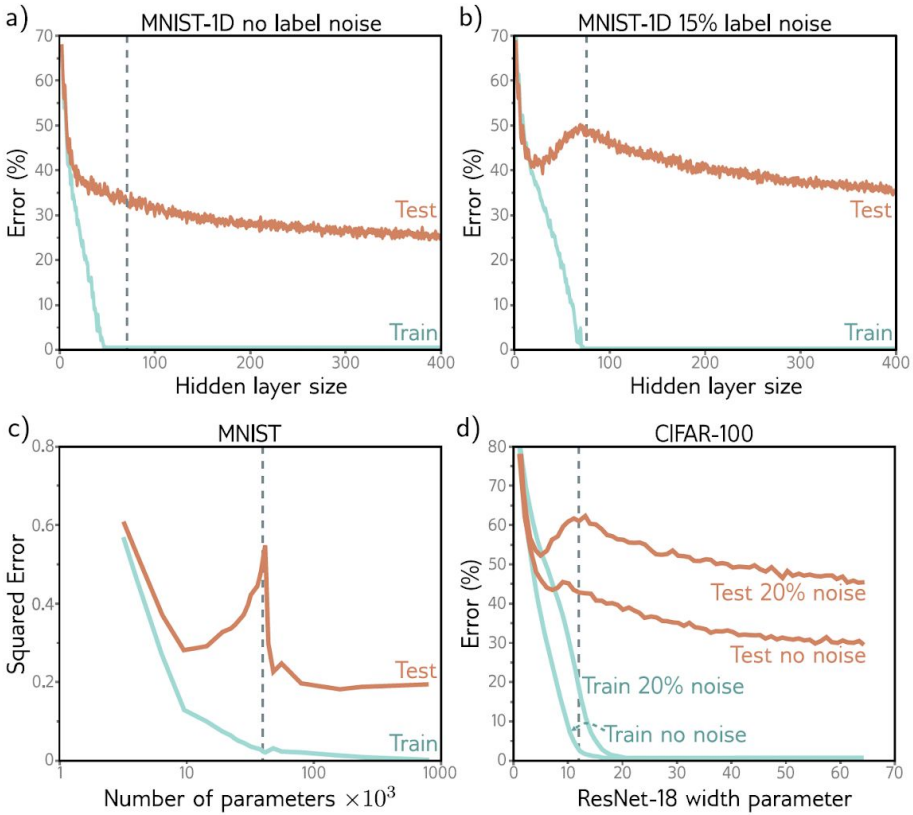
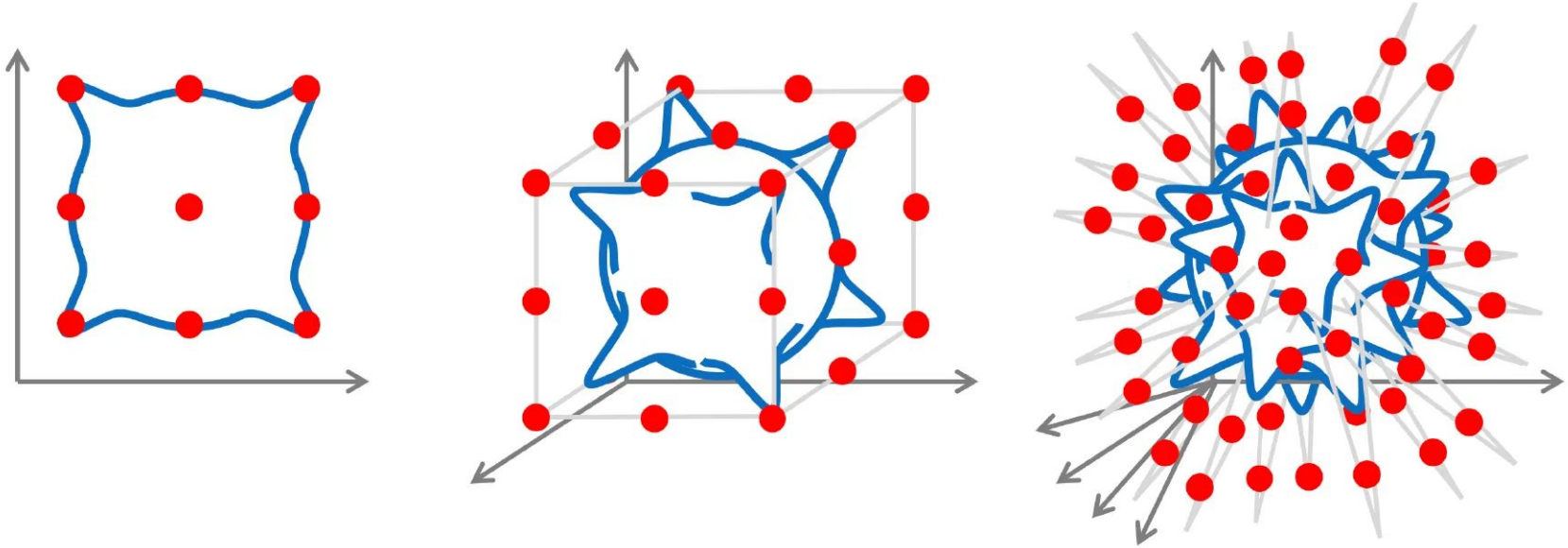


Figure 8.10 Double descent. a) Training and test loss on MNIST-1D for a two-hidden layer network as we increase the number of hidden units (and hence parameters) in each layer. The training loss decreases to zero as the number of parameters approaches the number of training examples (vertical dashed line). The test error does not show the expected bias-variance trade-off but continues to decrease even after the model has memorized the dataset. b) The same experiment is repeated with noisier training data. Again, the training error reduces to zero, although it now takes almost as many parameters as training points to memorize the dataset. The test error shows the predicted bias/variance trade-off; it decreases as the capacity increases but then increases again as we near the point where the training data is exactly memorized. However, it subsequently decreases again and ultimately reaches a better performance level. This is known as double descent. Depending on the loss, the model, and the amount of noise in the data, the double descent pattern can be seen to a greater or lesser degree across many datasets. c) Results on MNIST (without label noise) with shallow neural network from Belkin et al. (2019). d) Results on CIFAR-100 with ResNet18 network (see chapter 11) from Nakkiran et al. (2021). See original papers for details.

Colab time: Double Descent

https://colab.research.google.com/drive/1ct-iLf06-Yhehvz1a5PNtwcvJv1LGVtS

The Curse of High Dimensionality



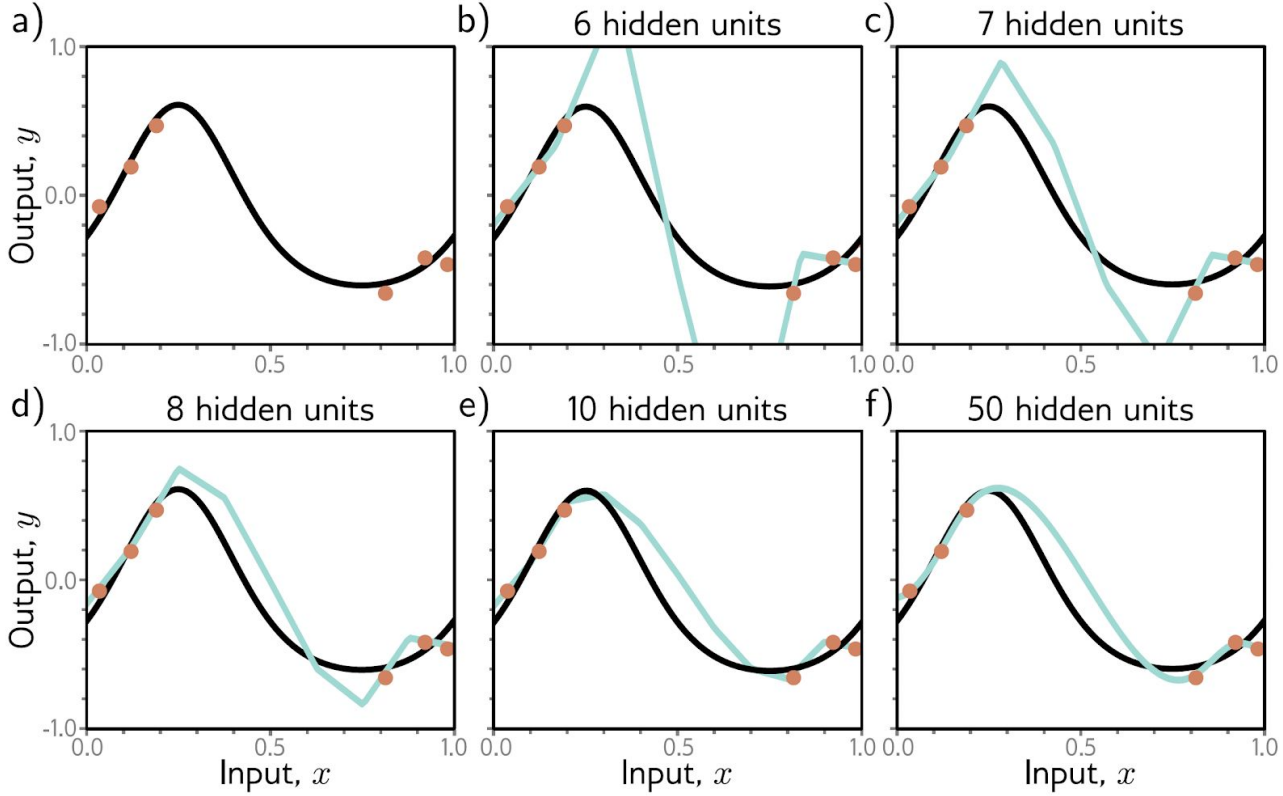
From the Geometric Deep Learning protobook.

Colab time: High Dimensional Spaces

https://colab.research.google.com/drive/1Tvj2NajinaiaJE7_iaMxiAjVnF9UgLpk

Double Descent and the Cu

Figure 8.11 Increasing capacity (hidden units) allows smoother interpolation between sparse data points. a) Consider this situation where the training data (orange circles) are sparse; there is a large region in the center with no data examples to constrain the model to mimic the true function (black curve). b) If we fit a model with just enough capacity to fit the training data (cyan curve), then it has to contort itself to pass through the training data, and the output predictions will not be smooth. c–f) However, as we add more hidden units, the model has the *ability* to interpolate between the points more smoothly (smoothest possible curve plotted in each case). However, unlike in this figure, it is not obliged to.



The MNIST-1D dataset has 40 dimensions, and we trained with 10,000 examples. If this seems like plenty of data, consider what would happen if we quantized each input dimension into 10 bins. There would be 10^{40} bins in total, constrained by only 10^5 examples. Even with this coarse quantization, there will only be one data point in every 10^{35} bins! The tendency of the volume of high-dimensional space to overwhelm the number of training points is termed the curse of dimensionality.

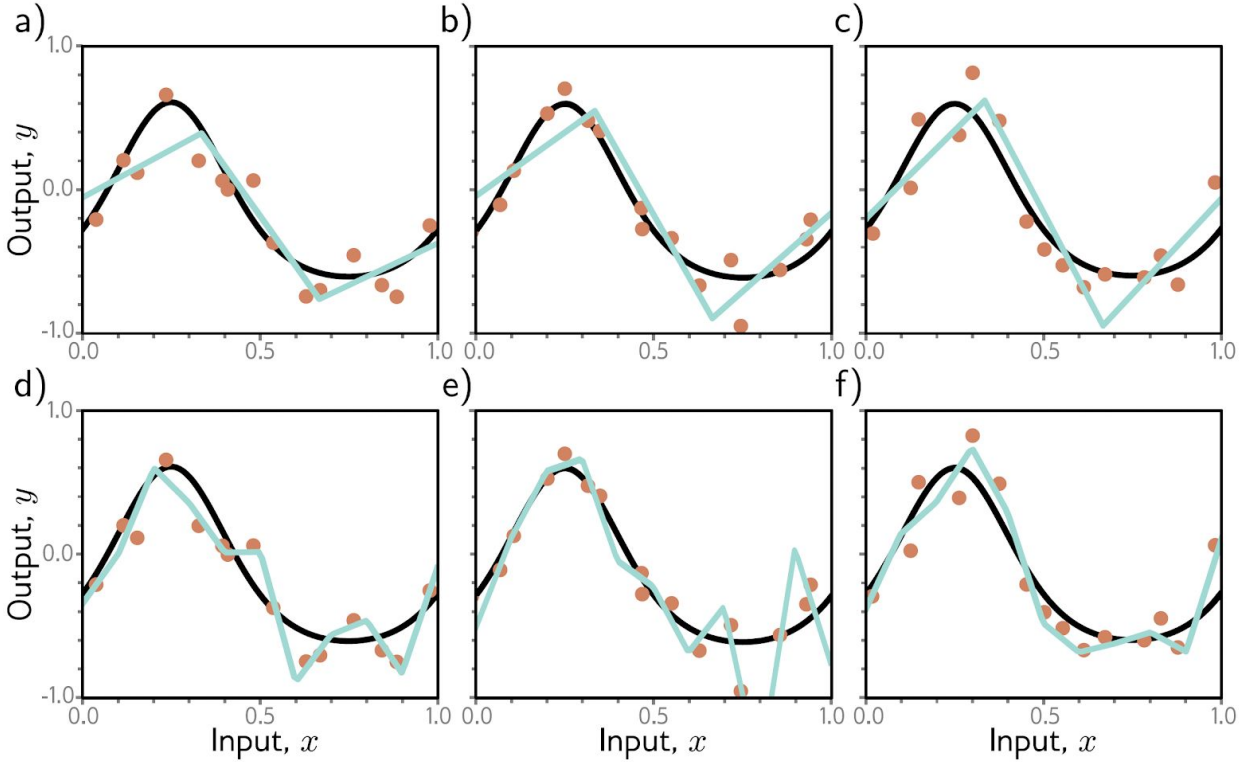
Exercise Time

$$\text{Vol}[r] = \frac{r^D \pi^{D/2}}{\Gamma[D/2+1]}$$

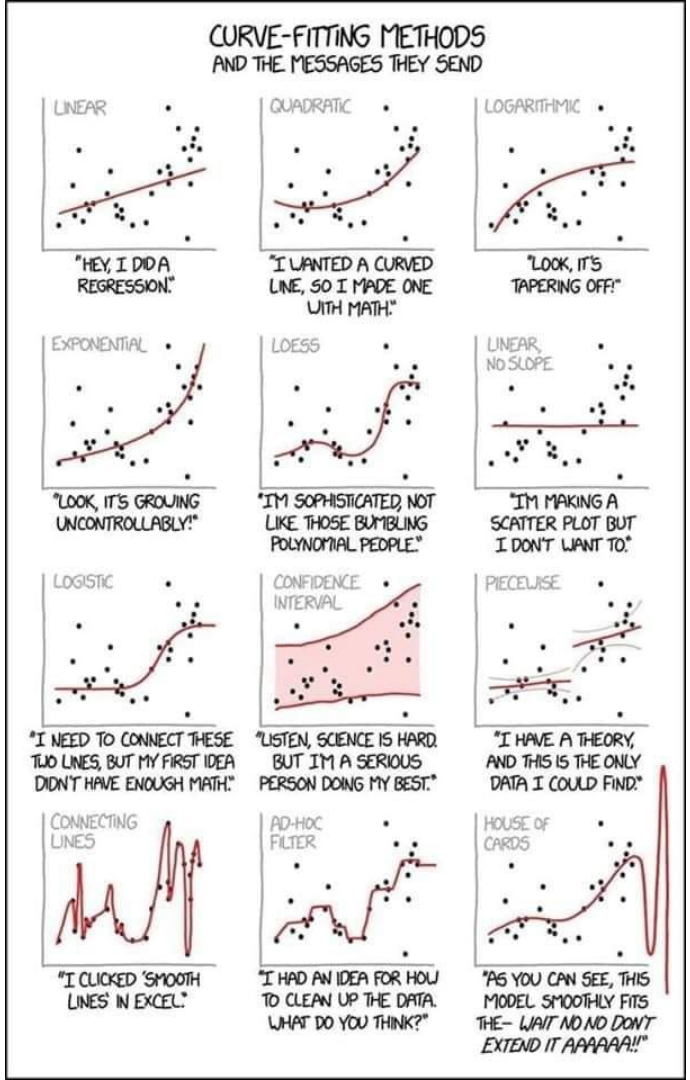
Problem 8.8 Consider a hypersphere of radius $r = 1$. Show the proportion of the total volume in the 1% of the radius closest to the surface of the hypersphere becomes one as the dimension increases.

Overfitting

Figure 8.8 Overfitting. a–c) A model with three regions is fit to three different datasets of fifteen points each. The result is 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.



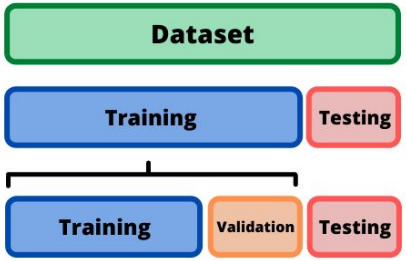
Overfitting

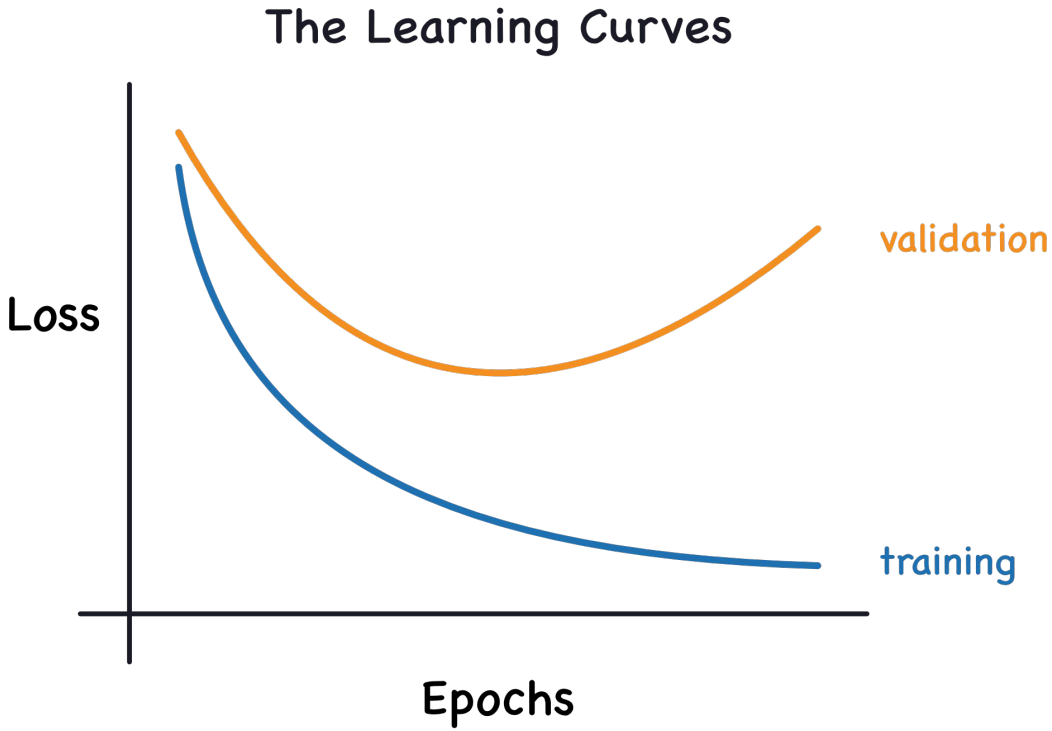


Credits XKCD: https://xkcd.com/2048/

SAPIENZA
UNIVERSITÀ DI ROMA

Overfitting





SAPIENZA
UNIVERSITÀ DI ROMA

Regularization

(Chapter 9)

SAPIENZA
UNIVERSITÀ DI ROMA

Explicit Regularization

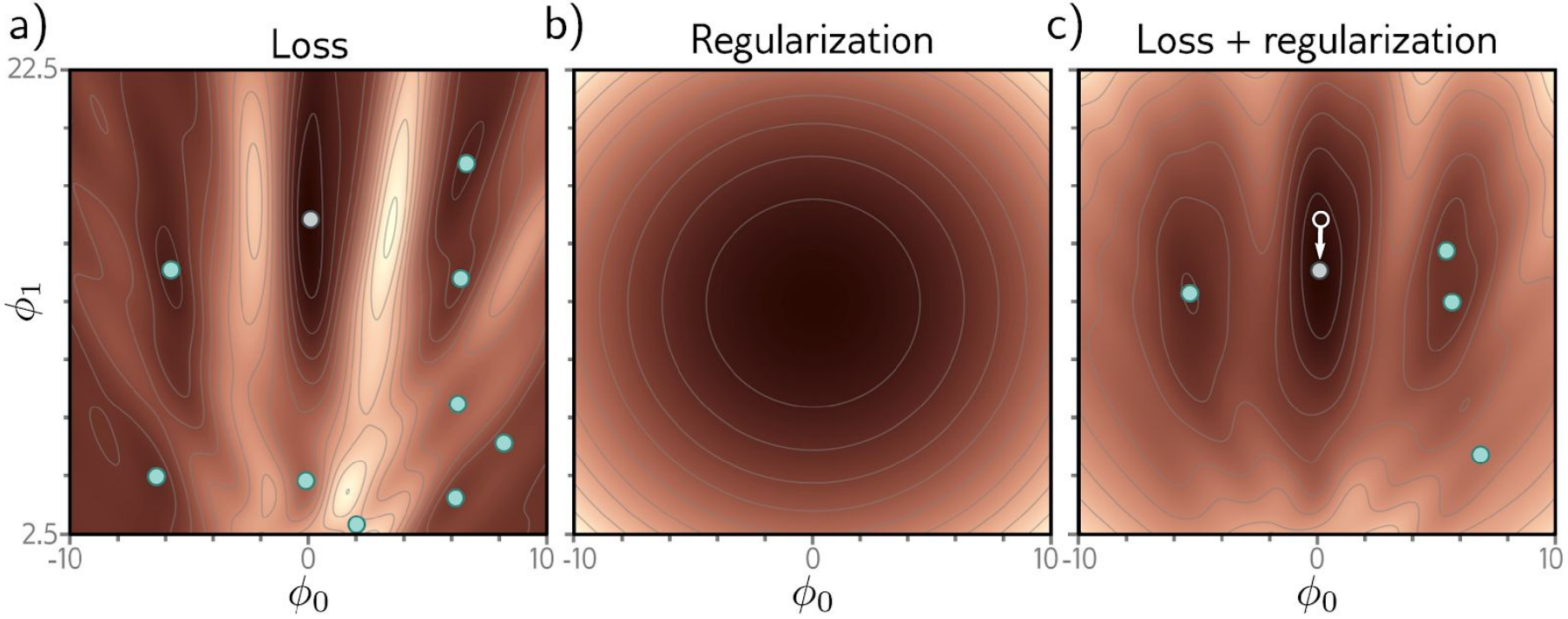
$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[\mathrm{L}[\boldsymbol{\phi}]\Big] \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i,\mathbf{y}_i]\right]\end{aligned}$$

VS

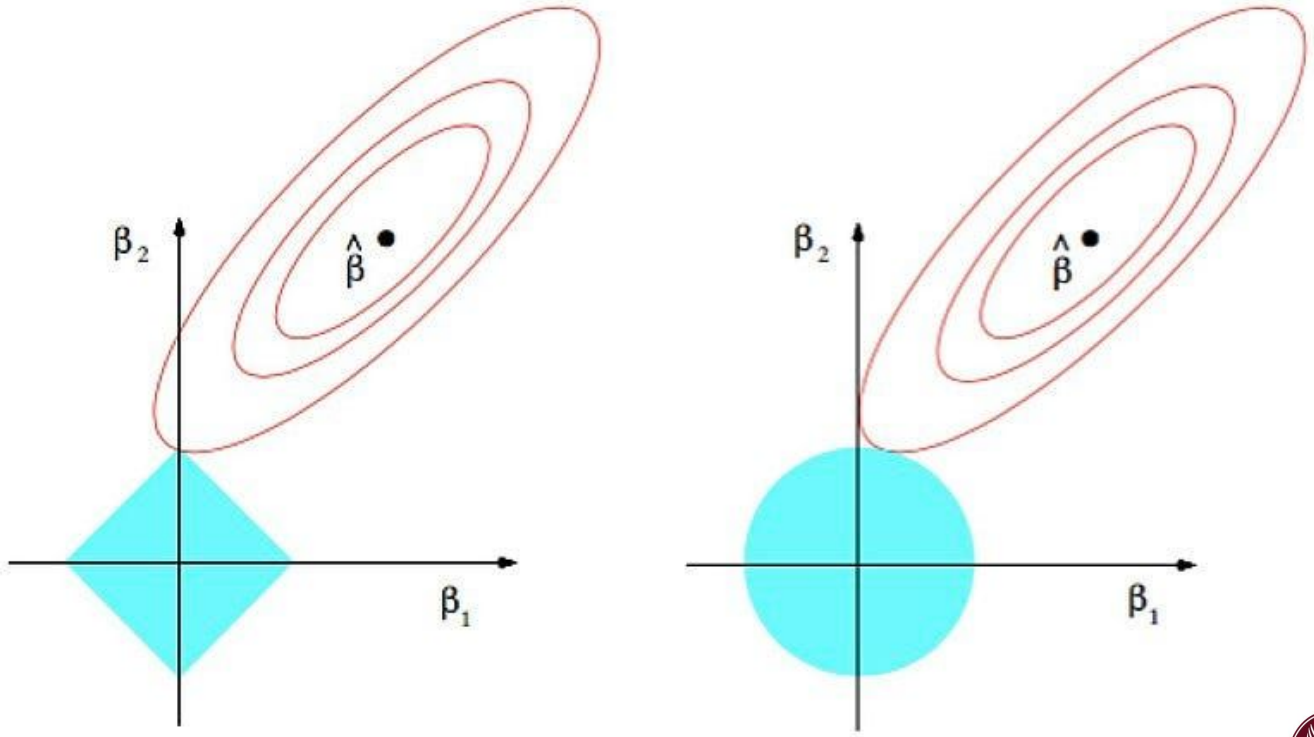
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i,\mathbf{y}_i] + \lambda\cdot \mathrm{g}[\boldsymbol{\phi}]\right]$$

g[ϕ] is a function that returns a scalar that takes a larger value when the parameters are less preferred

Explicit Regularization



Explicit Regularization



Courtesy of: https://towardsdatascience.com/understanding-l1-and-l2-regularization-93918a5ac8d0

Explicit Regularization

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

VS

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

g[ϕ] is a function that returns a scalar that takes a larger value when the parameters are less preferred

Reminder: MLE Criterion

- Observations (x_i, y_i) are:
  - Independent
  - Identically distributed
- Observations i.i.d.

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

$$Pr(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_I|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_I) = \prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i)$$

Regularization as a Prior

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

- Exercise: Compute the negative log likelihood and negate it.

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

L_2 Regularization

Weights and not biases

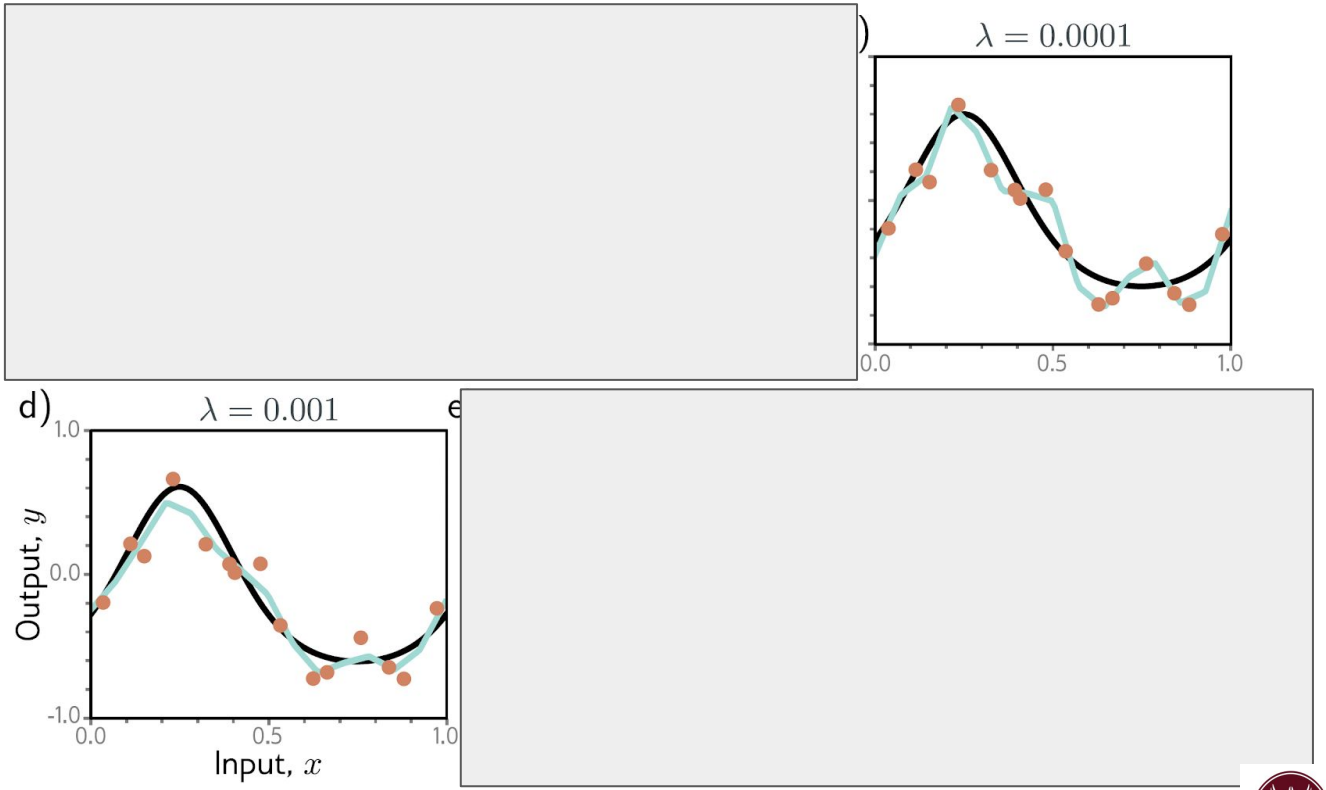
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \sum_{j} \phi_j^2 \right]$$

Problem 9.1 Consider a model where the prior distribution over the parameters is a normal distribution with mean zero and variance $\sigma_{\boldsymbol{\phi}}^2$ so that

$$Pr(\boldsymbol{\phi}) = \prod_{j=1}^{J} \text{Norm}_{\phi_j}[0, \sigma_{\boldsymbol{\phi}}^2], \tag{9.21}$$

where j indexes the model parameters. We now maximize $\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\phi}) Pr(\boldsymbol{\phi})$. Show that the associated loss function of this model is equivalent to L2 regularization.

L_2 Regularization



Colab time: L2 Regularization

https://colab.research.google.com/drive/12iEQw6Eh44JqunDn7-TkXLs7dAnoz5Hu

Effect of Regularization on Overfitting

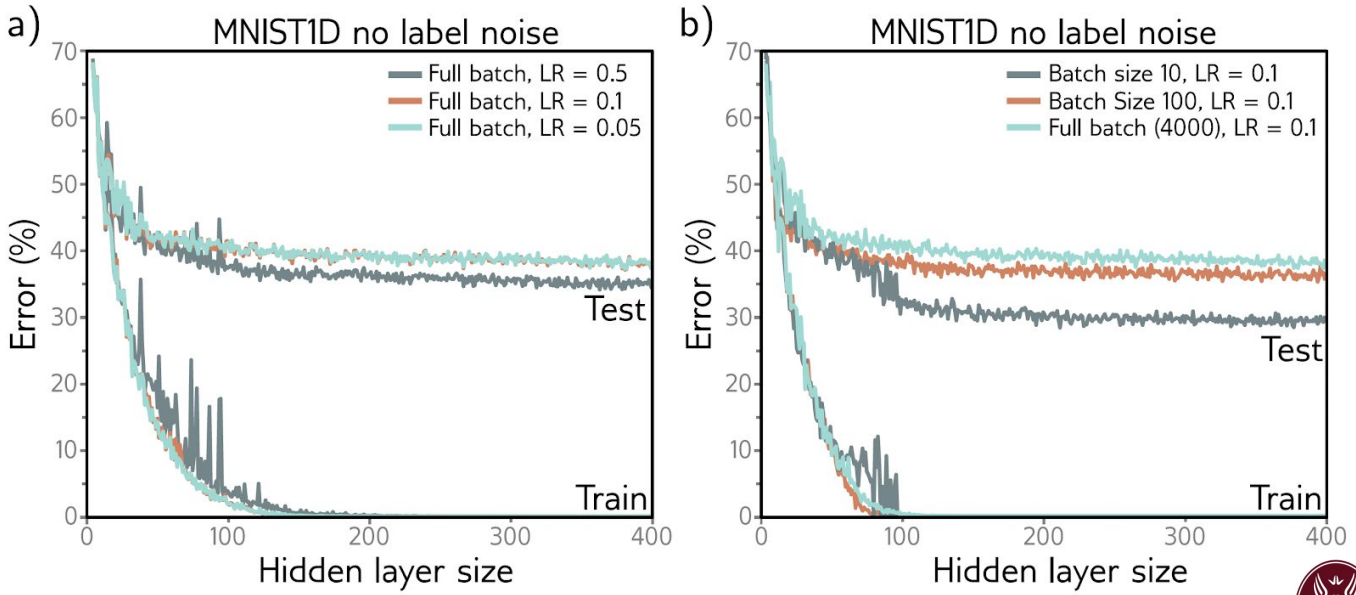
- If the network is overfitting, then adding the regularization term means that the network must trade off slavish adherence to the data against the desire to be smooth.
- One way to think about this is that the **error due to variance reduces** (the model no longer needs to pass through every data point) at the cost of **increased bias** (the model can only describe smooth functions).

Effect of Regularization on Overparametrization

- When the network is overparameterized, some of the extra model capacity describes areas with no training data.
- Here, the regularization term will **favor functions that smoothly interpolate between the nearby points**.
    - This is reasonable behavior in the absence of knowledge about the true function.

Regularization due to (Stochastic) Gradient Descent

- Outside the scope of this course.
- In case you are interested have a look at sections 9.2.1 and 9.2.2 in the book



Early Stopping

- Stop the training procedure before it has fully converged.
  - After K steps
- This can reduce overfitting if the model has already captured the coarse shape of the underlying function but has not yet had time to overfit to the noise.
  - Intuitively, since the weights are initialized to small values, they simply don't have time to become large, so early stopping has a similar effect to explicit L2 regularization.
- The model is trained once, the performance on the validation set is monitored every T iterations, and the associated models are stored.
  - The stored model where the validation performance was best is selected.

Early Stopping

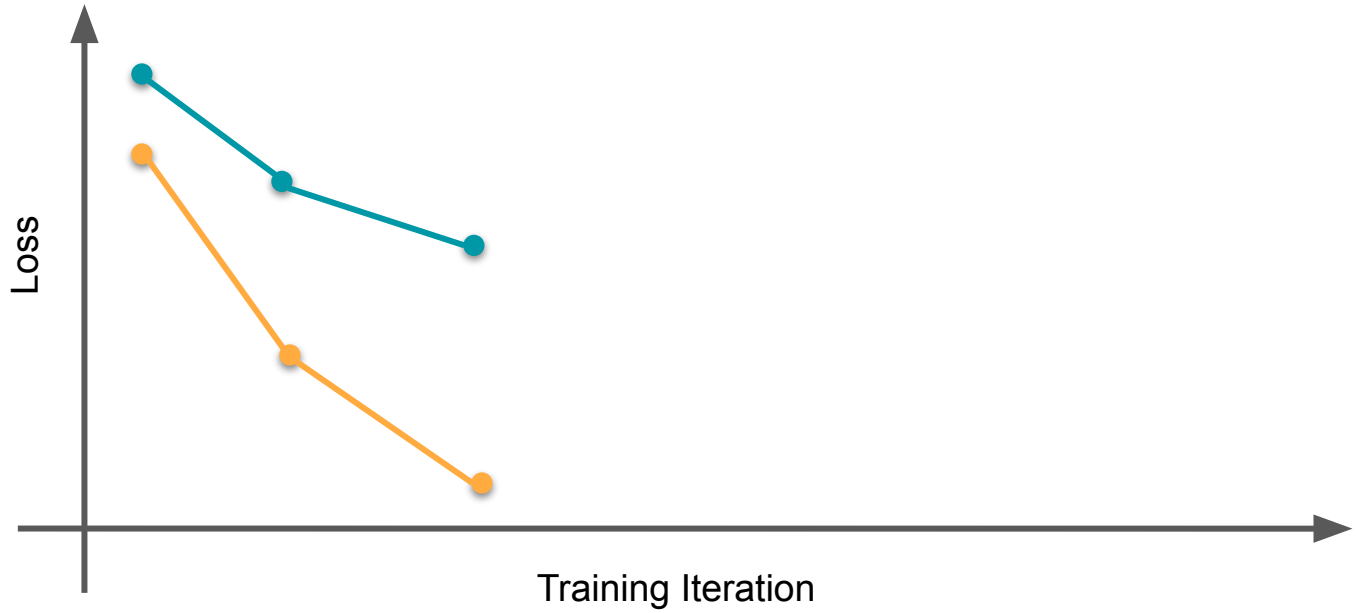
- Stop training as we detect overfitting

Early Stopping

- Stop training as we detect overfitting

Early Stopping

- Stop training as we detect overfitting



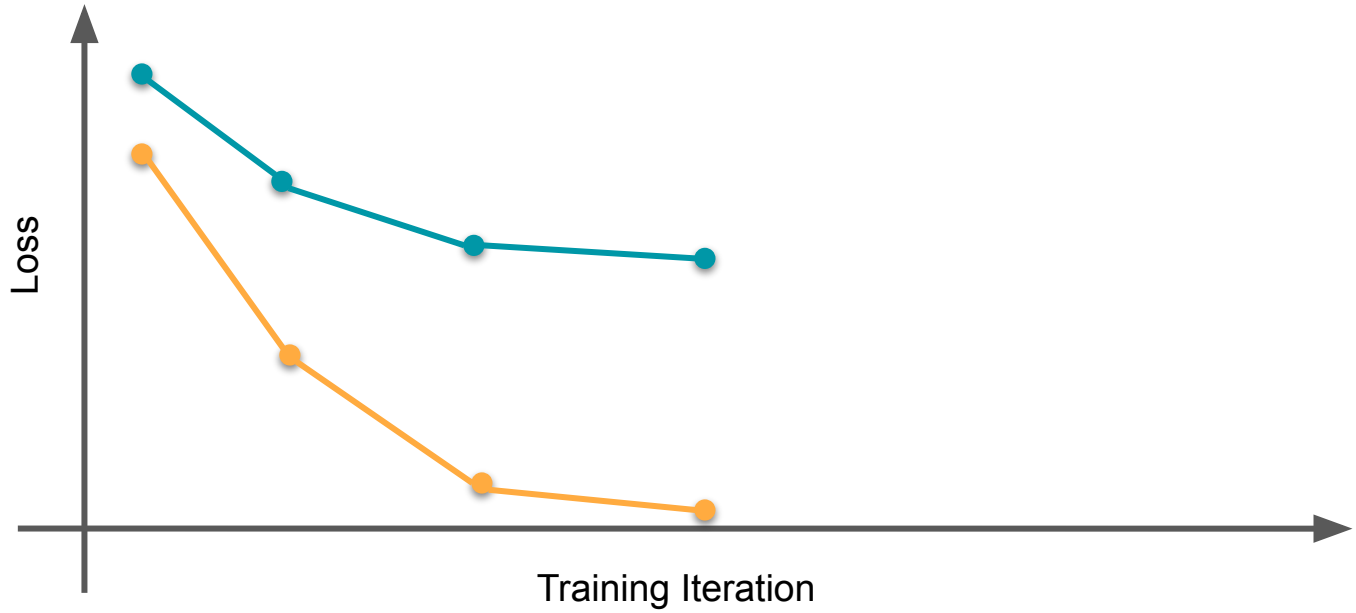
Legend

Test

Train

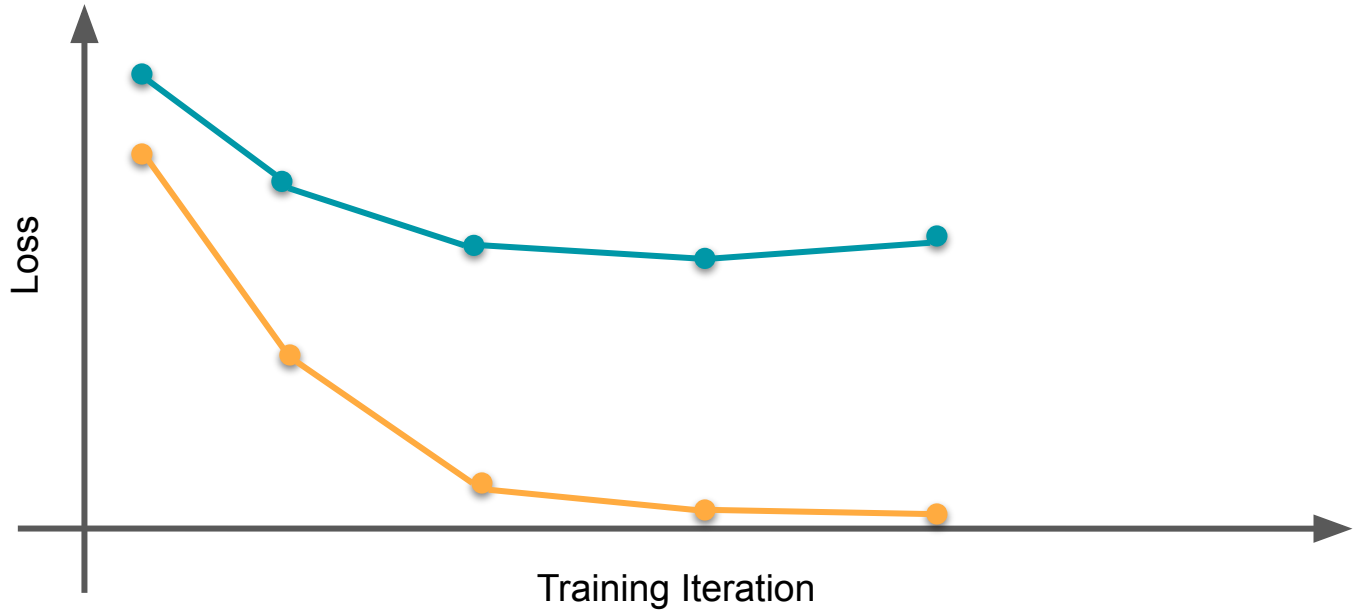
Early Stopping

- Stop training as we detect overfitting



Early Stopping

- Stop training as we detect overfitting



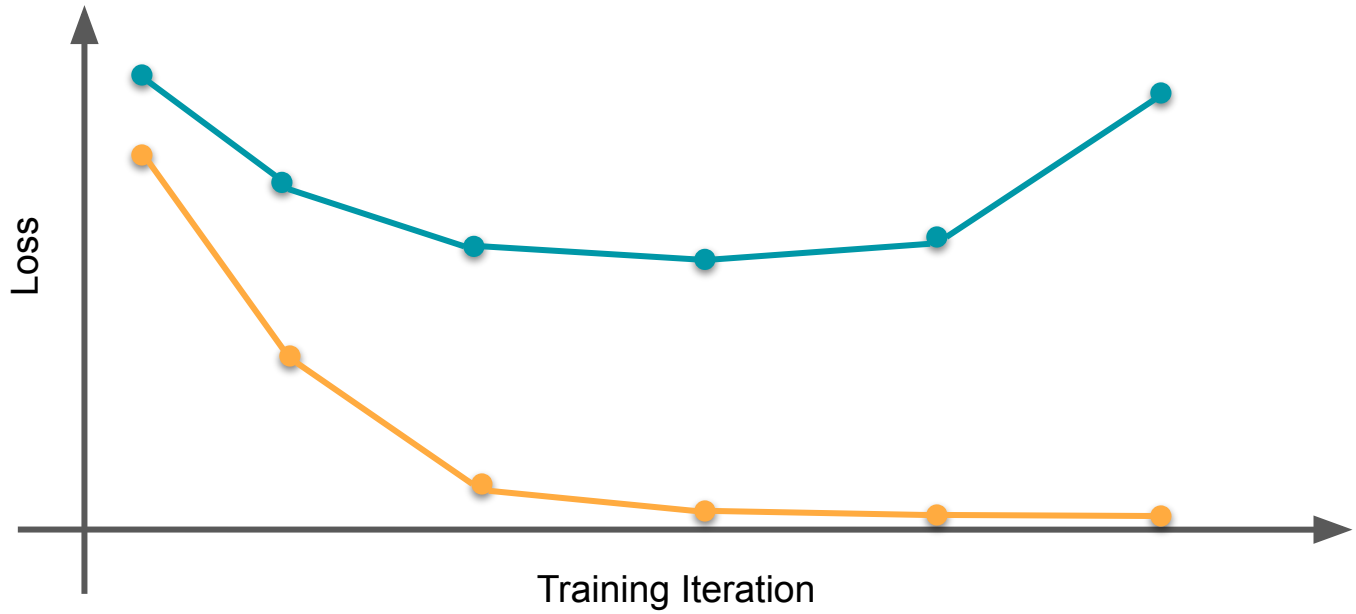
Legend

Test

Train

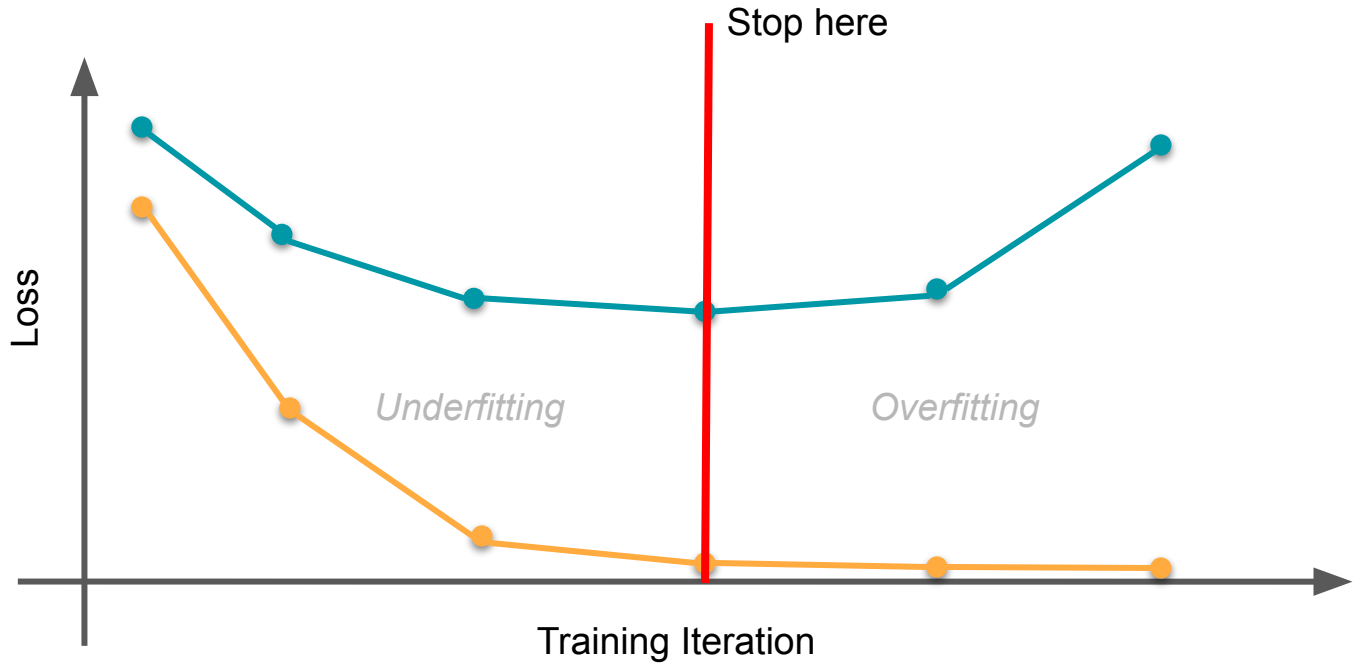
Early Stopping

- Stop training as we detect overfitting



Early Stopping

- Stop training as we detect overfitting

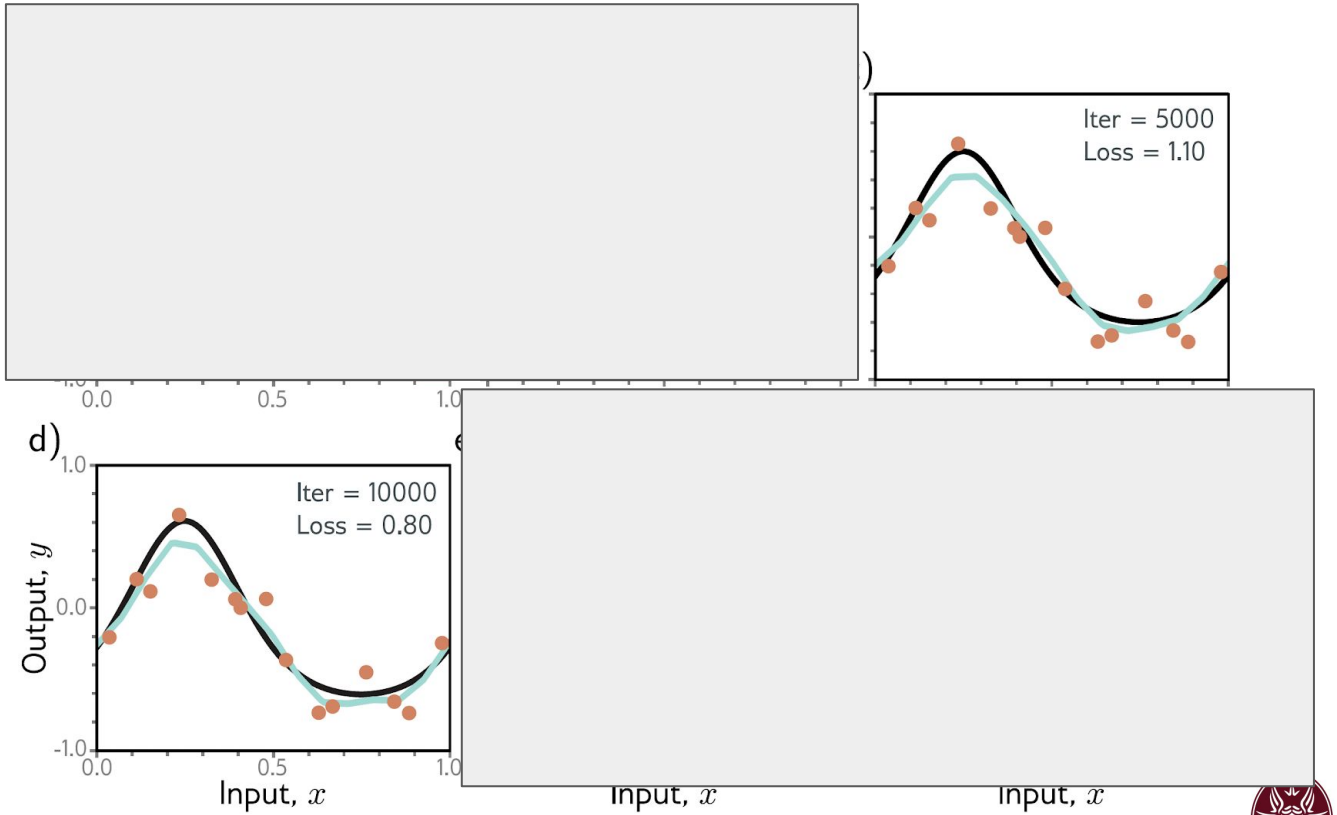


Legend

Test

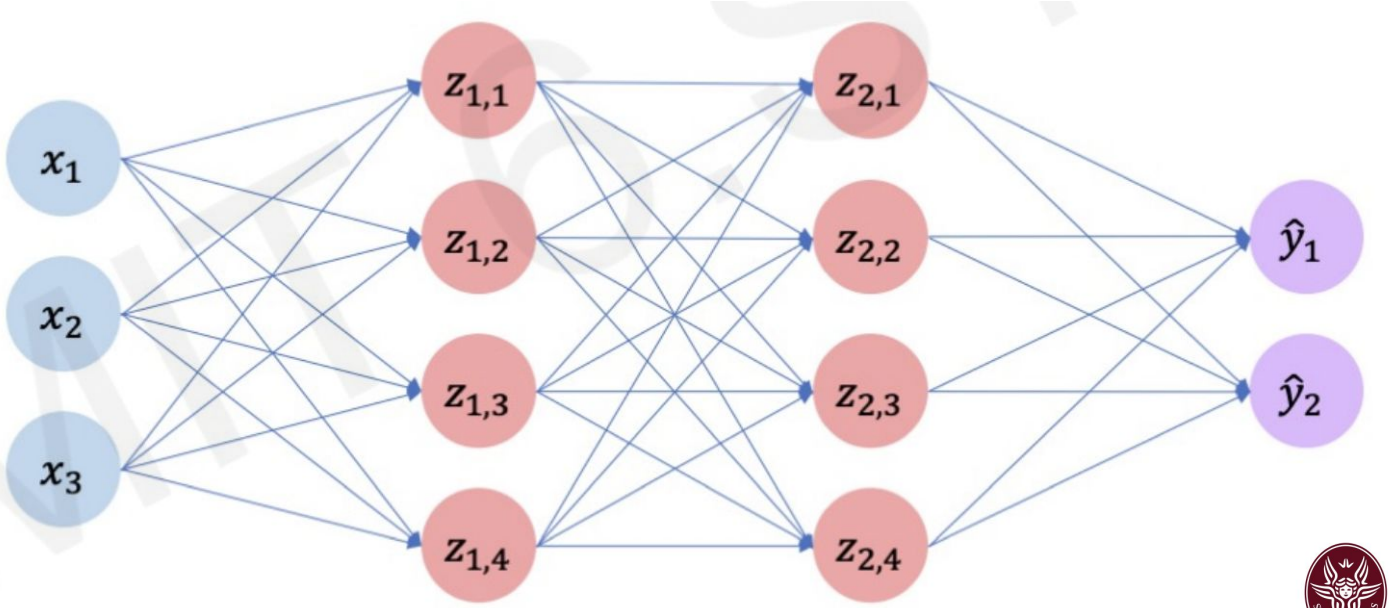
Train

Early Stopping



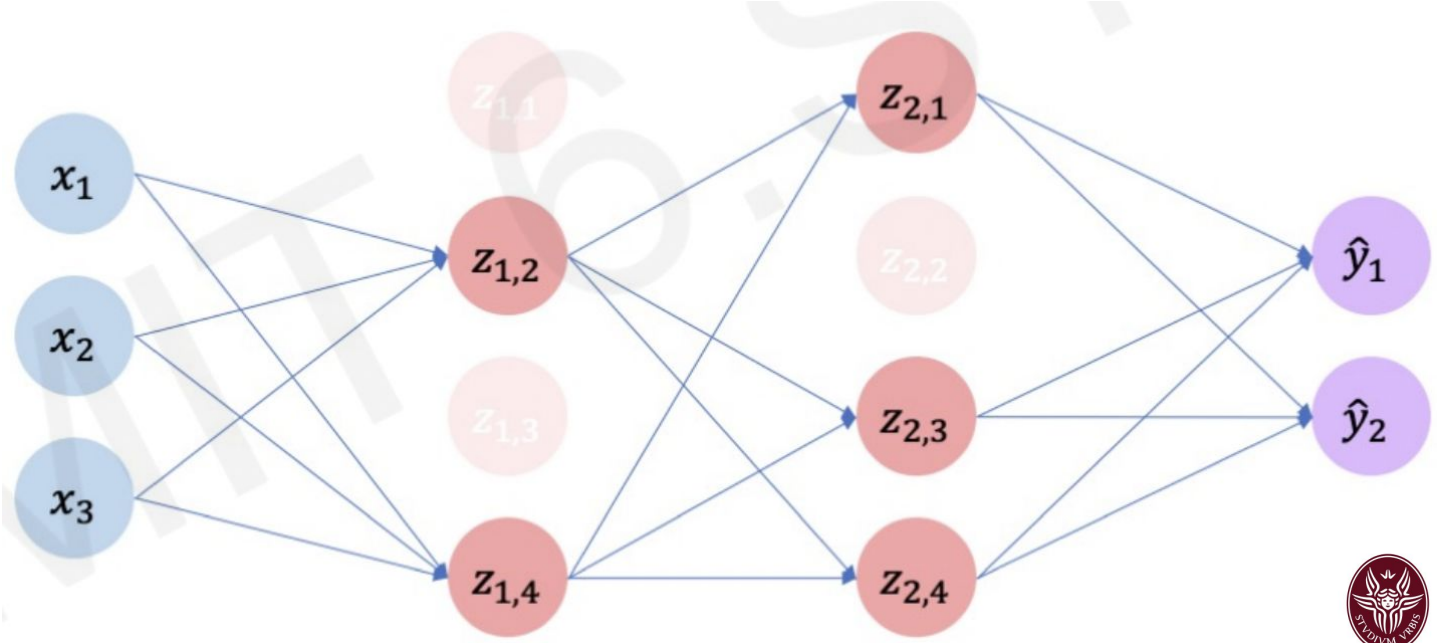
Dropout

- During training set some weights to zero
  - u.a.r. w/ probability p (typically .5)



Dropout

- During training set some weights to zero
  - u.a.r. w/ probability p (typically .5)



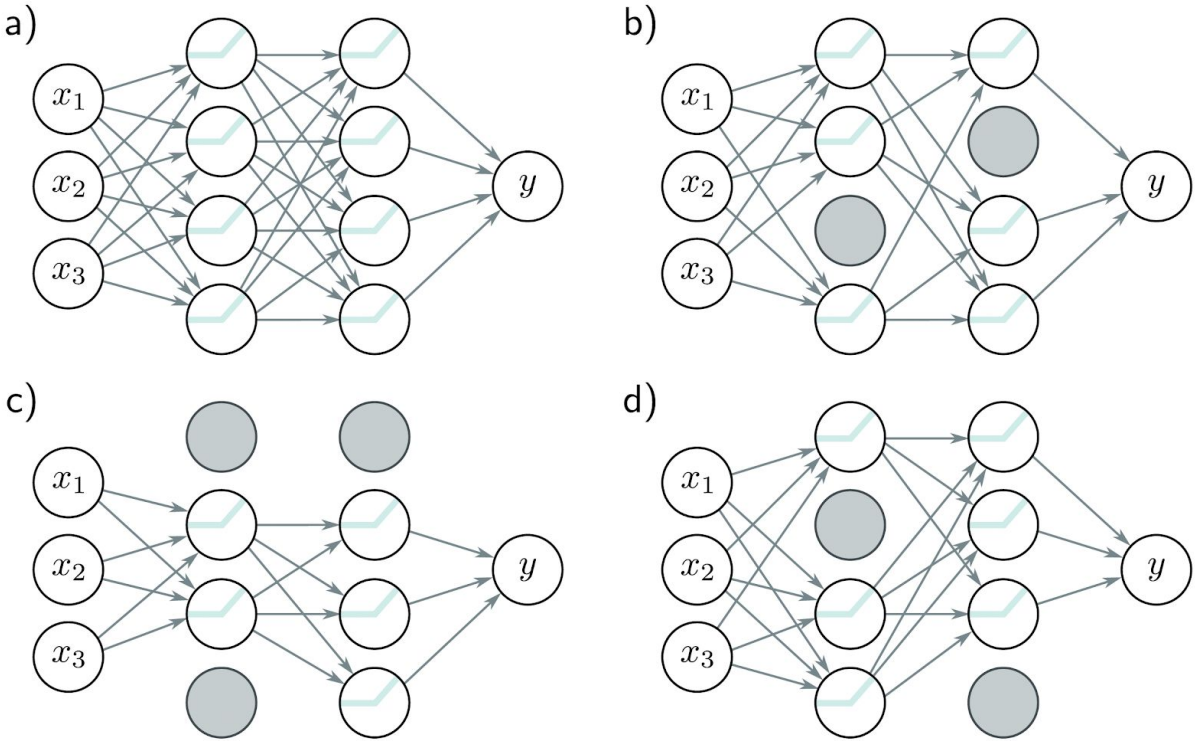
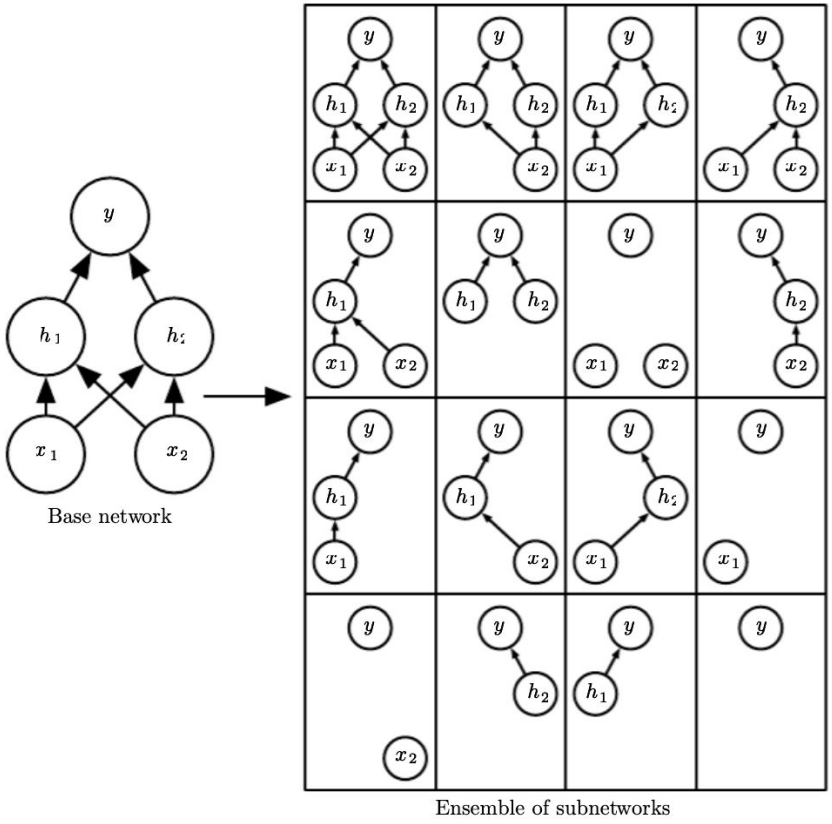


Figure 9.8 Dropout. a) Original network. b–d) At each training iteration, a random subset of hidden units is clamped to zero (gray nodes). The result is that the incoming and outgoing weights from these units have no effect, so we are training with a slightly different network each time.

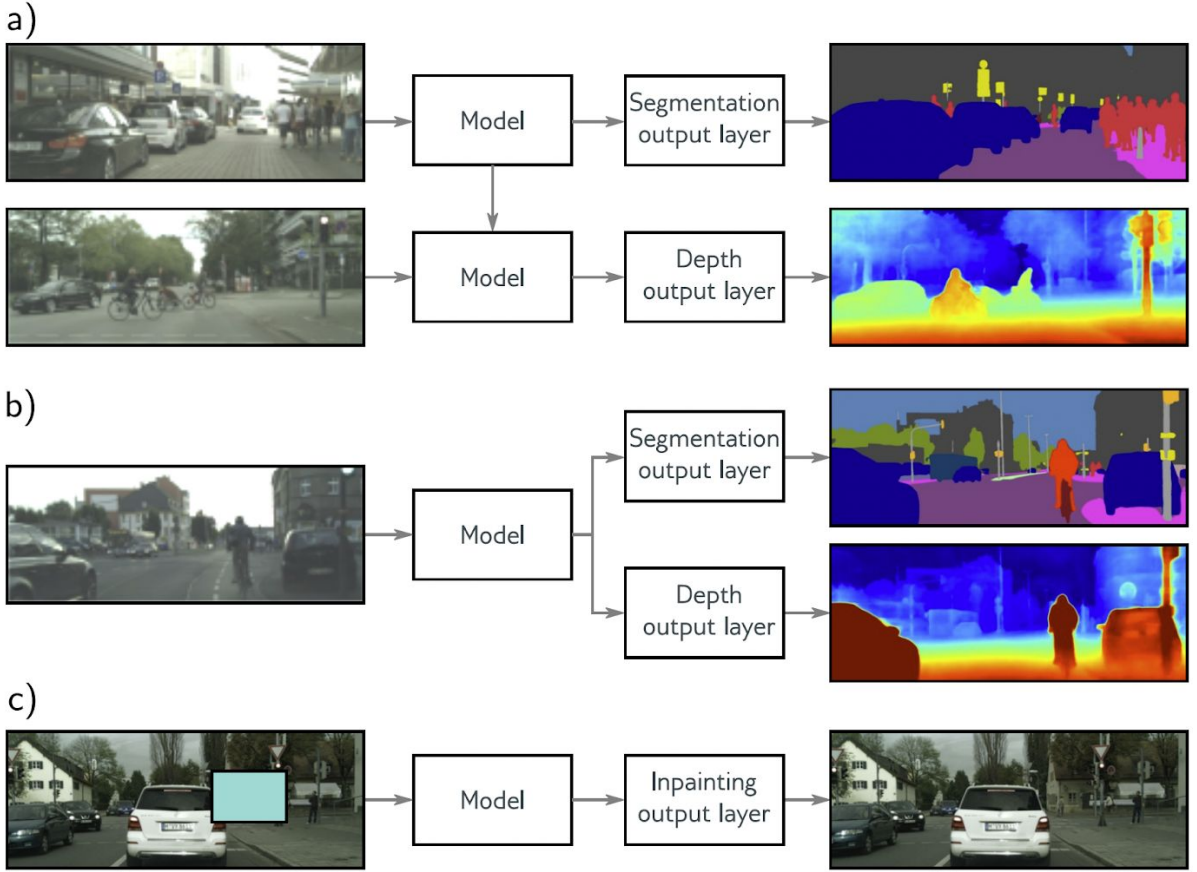
Dropout as Ensemble of Models



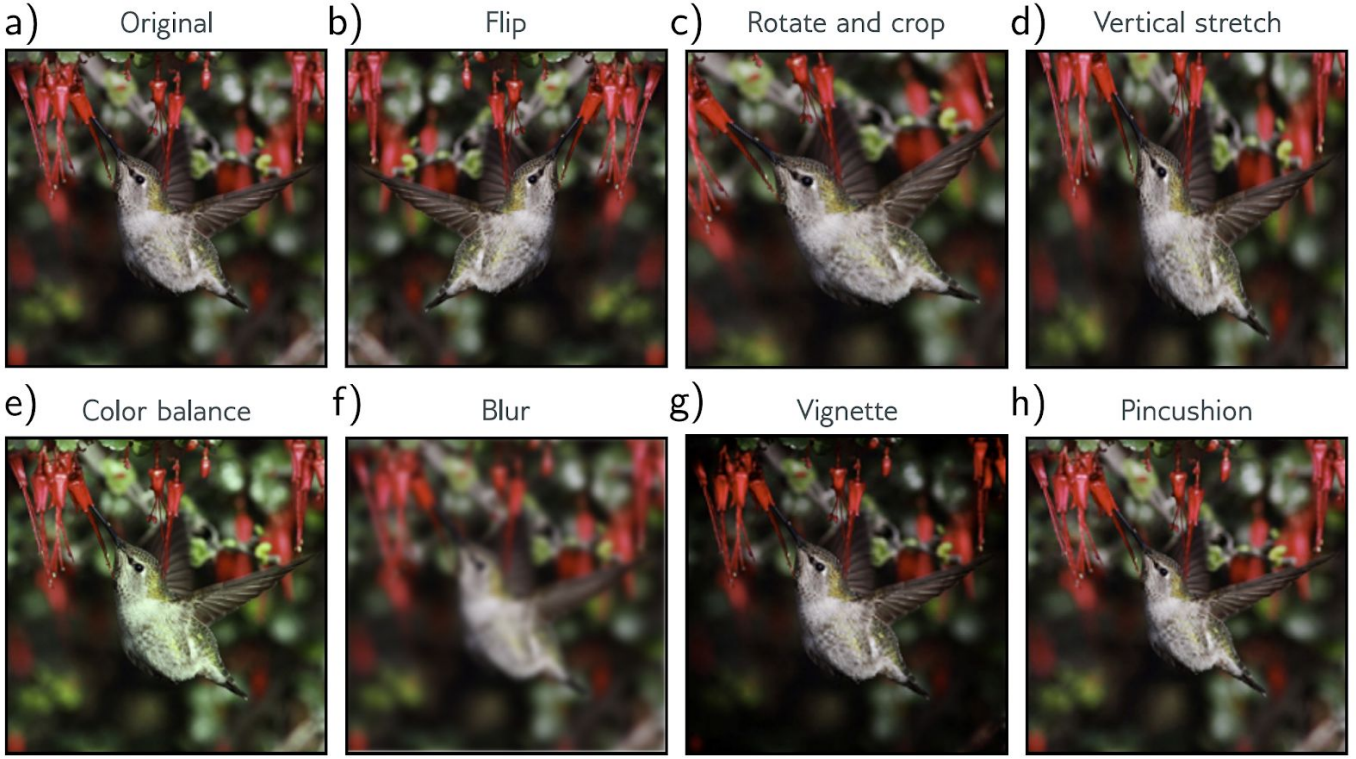
Dropout

- Only one caveat:
  - When extremely few labeled training examples are available, dropout is less effective.

Data Augmentation



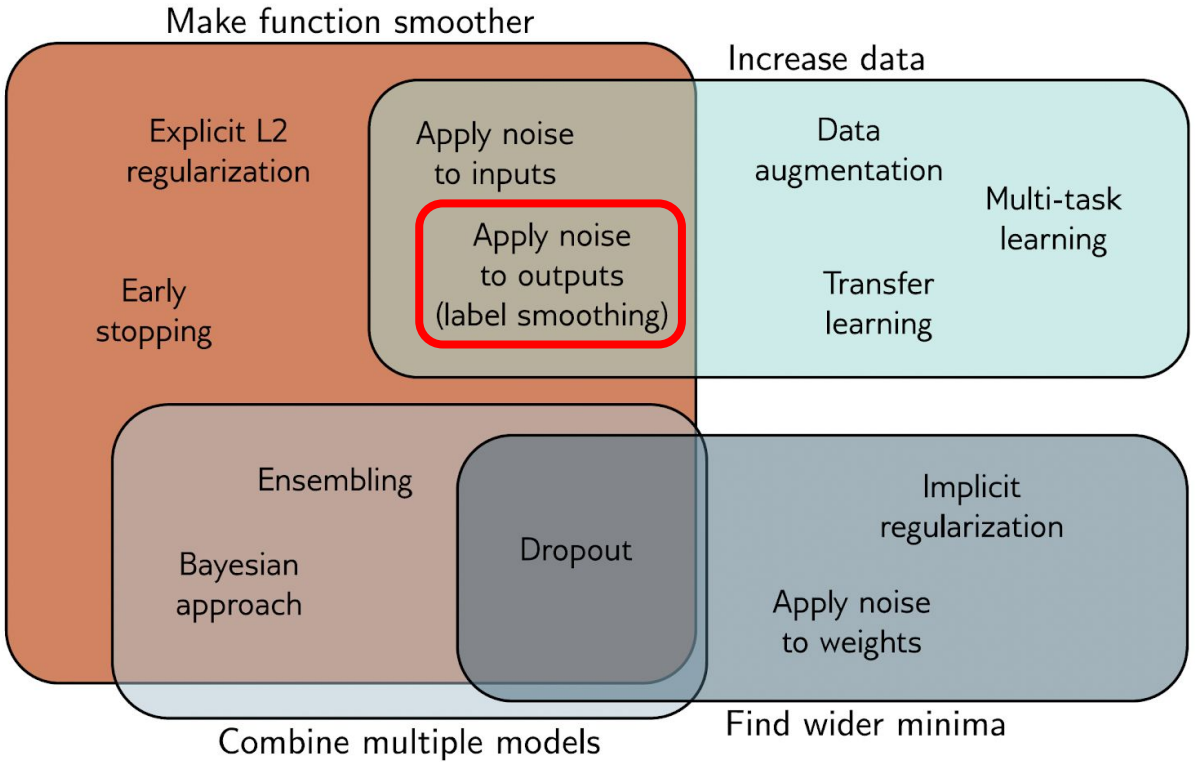
Data Augmentation



Colab time: Data Augmentation

https://colab.research.google.com/drive/1zHGGs7k9nJL1RH9zxhBdVYmGh-Xt2C-h

Regularization Methods: Summary



Exercise

$$\text{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^{K} \exp[z_{k'}]}$$

Problem 9.4 Derive the loss function for multi-class classification when we use label smoothing so that the target probability distribution has 0.9 at the correct class and the remaining probability mass of 0.1 is divided between the remaining $D_o - 1$ classes.

Deep Learning

End of Lecture

05 - Model Training. Measuring Performance and Regularization

SAPIENZA
UNIVERSITÀ DI ROMA

Fabrizio Silvestri