Bias, Variance, and Regularization

Designing, Visualizing and Understanding Deep Neural Networks

CS W182/282A

Instructor: Sergey Levine
UC Berkeley

Will we get the right answer?

Empirical risk and true risk

zero-one loss: $\sum_i \delta(f_\theta(x_i) \neq y_i)$ 1 if wrong, 0 if right

Risk: probability you will get it wrong

expected value of our loss quantifies this

can be generalized to other losses

(e.g., NLL)

$\sim p(x)$

Risk $= E_{x\sim p(x),y\sim p(y|x)}[\mathcal{L}(x, y, \theta)]$ $y \sim p(y|x)$ how likely is it that $f_\theta(x)$ is wrong?

During training, we can't sample $x \sim p(x)$, we just have $\mathcal{D}$

Empirical risk $= \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(x_i, y_i, \theta) \approx E_{x\sim p(x),y\sim p(y|x)}[\mathcal{L}(x, y, \theta)]$

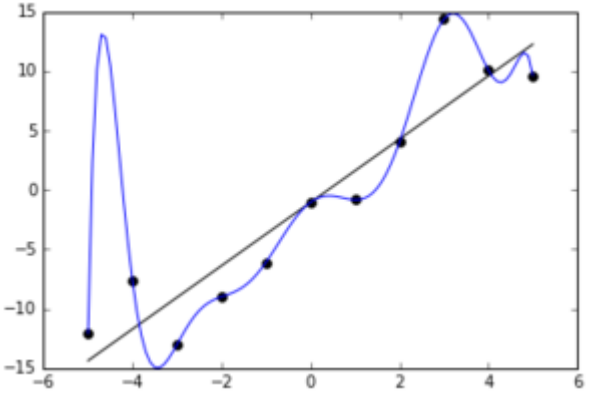
is this a **good** approximation?

Empirical risk minimization

$$\text{Empirical risk} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(x_i, y_i, \theta) \approx E_{x\sim p(x), y\sim p(y|x)}[\mathcal{L}(x, y, \theta)]$$

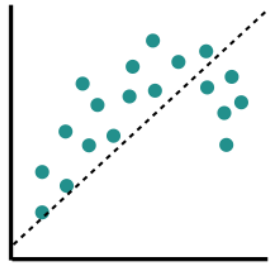
Supervised learning is (usually) *empirical* risk minimization

Is this the same as *true* risk minimization?



Overfitting: when the empirical risk is low, but the true risk is high

- can happen if the dataset is too small
- can happen if the model is too powerful (has too many parameters/capacity)



Underfitting: when the empirical risk is high, and the true risk is high

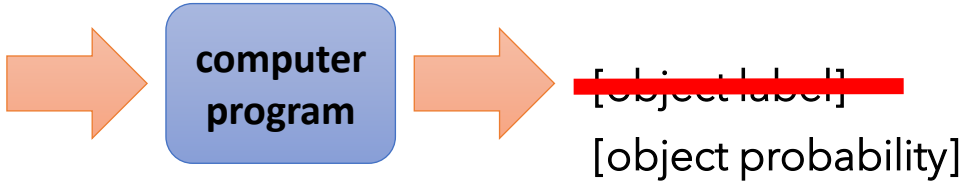
- can happen if the model is too weak (has too few parameters/capacity)
- can happen if your optimizer is not configured well (e.g., wrong learning rate)

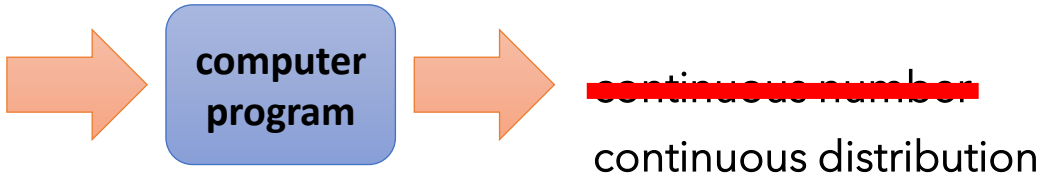
Let's analyze error!

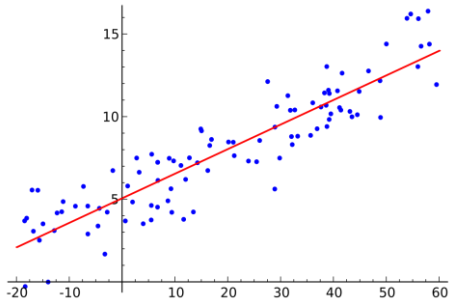
Last time, we discussed **classification**

This time, we'll focus on **regression**

All this stuff applies to classification too, it's just simpler to derive for regression







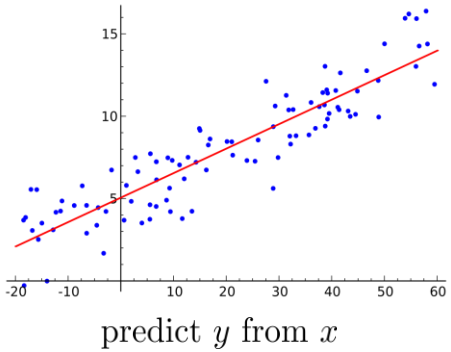
predict y from x

$$\log p_\theta(y|x) = \mathcal{N}(f_\theta(x), \Sigma_\theta(x)) = -\frac{1}{2}(f_\theta(x) - y)\Sigma_\theta(x)^{-1}(f_\theta(x) - y) - \frac{1}{2}\log|\Sigma_\theta(x)| + \text{const}$$

normal (Gaussian) distribution

$$\text{if } \Sigma_\theta(x) = \mathbf{I} \qquad = -\frac{1}{2}||f_\theta(x) - y||^2 + \text{const}$$

Let's analyze error!



$$\log p_\theta(y|x) = -\frac{1}{2}||f_\theta(x) - y||^2 + \text{const} \qquad \text{if } \Sigma_\theta(x) = \mathbf{I}$$

Also the same as the mean squared error (MSE) loss!

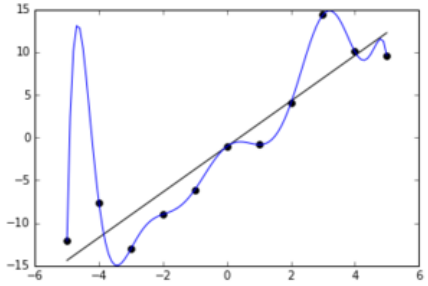
$$\mathcal{L}(\theta, x, y) = -\frac{1}{2}||f_\theta(x) - y||^2$$

a bit easier to analyze, but we can analyze other losses too

Overfitting: when the empirical risk is low, but the true risk is high

can happen if the dataset is too small

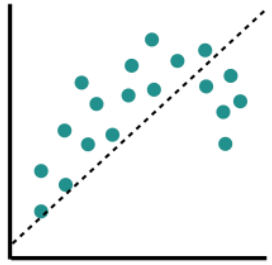
can happen if the model is too powerful (has too many parameters/capacity)



Underfitting: when the empirical risk is high, and the true risk is high

can happen if the model is too weak (has too few parameters/capacity)

can happen if your optimizer is not configured well (e.g., wrong learning rate)



Let's analyze error!

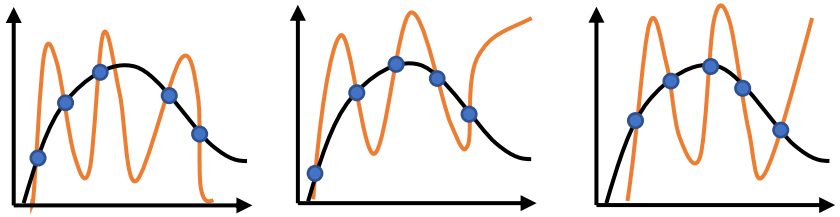
$$\mathcal{L}(\theta, x, y) = -\frac{1}{2}||f_\theta(x) - y||^2$$

Let's try to understand **overfitting** and **underfitting** more formally

Question: how does the error change for different **training sets**?

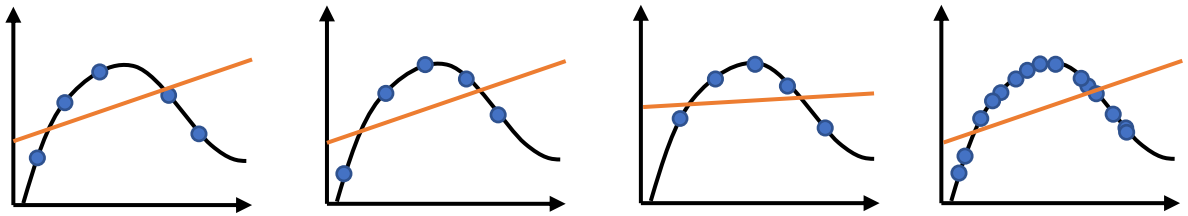
Why is this question important?

overfitting



- The training data is fitted well
- The true function is fitted poorly
- **The learned function looks different each time!**

underfitting



- The training data is fitted poorly
- The true function is fitted poorly
- **The learned function looks similar, even if we pool together all the datasets!**

Let's analyze error!

$$\mathcal{L}(\theta, x, y) = -\frac{1}{2}||f_\theta(x) - y||^2$$

What is the **expected** error, given a distribution over datasets?

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

$$\sim p(x)$$

$$p(\mathcal{D}) = \prod_{i=1}^{N} p(x_i)p(y_i|x_i)$$

$$y \sim p(y|x)$$

$f_\mathcal{D}(x) \quad f(x)$

$$E_{\mathcal{D}\sim p(\mathcal{D})}[||f_\theta(x) - y||^2] = \sum_{\mathcal{D}} p(\mathcal{D})||f_\mathcal{D}(x) - f(x)||^2$$

expected value of error w.r.t. data distribution

sum over all possible datasets

Let's analyze error!

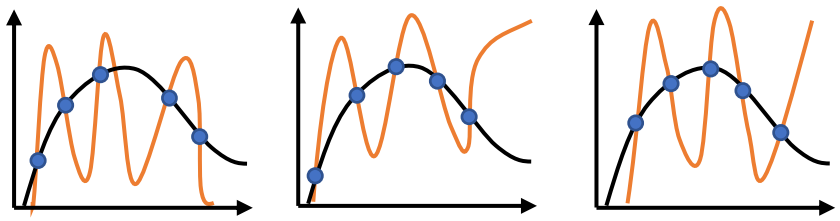
$$E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - f(x)||^2] = \sum_{\mathcal{D}} p(\mathcal{D})||f_{\mathcal{D}}(x) - f(x)||^2$$

Why do we care about this quantity?

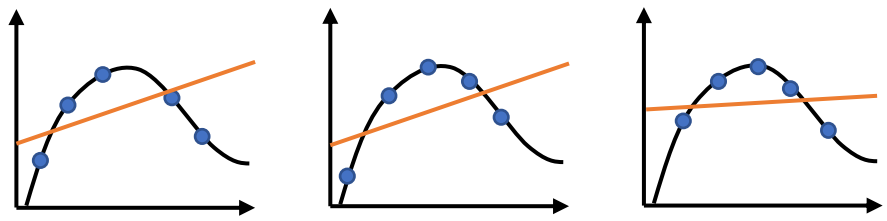
We want to understand how well our **algorithm** does independently of the **particular (random) choice of dataset**

This is very important if we want to **improve** our algorithm!

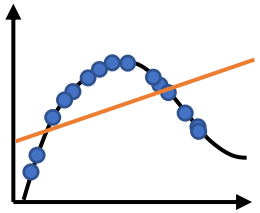
overfitting



underfitting



Bias-variance tradeoff



$$E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - f(x)||^2] \qquad \text{let } \bar{f}(x) = E_{\mathcal{D}\sim p(\mathcal{D})}[f_{\mathcal{D}}(x)]$$

$$= E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - \bar{f}(x) + \bar{f}(x) - f(x)||^2]$$

$$= E_{\mathcal{D}\sim p(\mathcal{D})}[||(f_{\mathcal{D}}(x) - \bar{f}(x)) + (\bar{f}(x) - f(x))||^2]$$

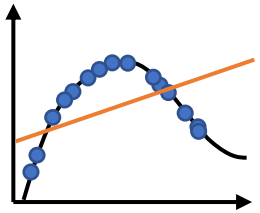
$$= E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - \bar{f}(x)||^2] + E_{\mathcal{D}\sim p(\mathcal{D})}[||\bar{f}(x) - f(x)||^2]$$

$$E_{\mathcal{D}\sim p(\mathcal{D})}[2(f_{\mathcal{D}}(x) - \bar{f}(x))^T(\bar{f}(x) - f(x))]$$

$$0$$

$$2E[(f_{\mathcal{D}}(x) - \bar{f}(x))]^T(\bar{f}(x) - f(x))$$

Bias-variance tradeoff

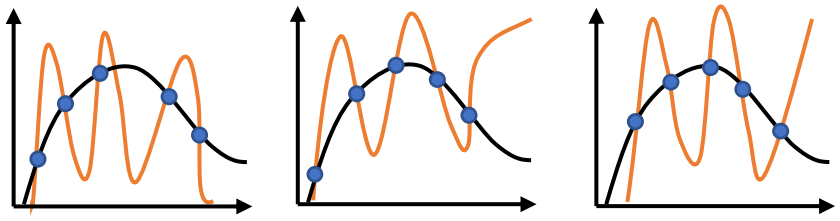


$$E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - f(x)||^2] \qquad \text{let } \bar{f}(x) = E_{\mathcal{D}\sim p(\mathcal{D})}[f_{\mathcal{D}}(x)]$$

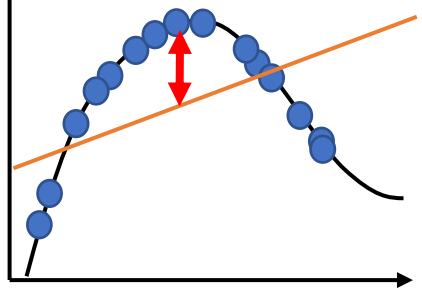
$$= \underbrace{E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - \bar{f}(x)||^2]}_{\text{Variance}} + \underbrace{E_{\mathcal{D}\sim p(\mathcal{D})}[||\bar{f}(x) - f(x)||^2]}_{||\bar{f}(x) - f(x)||^2}$$

$$\text{Bias}^2$$

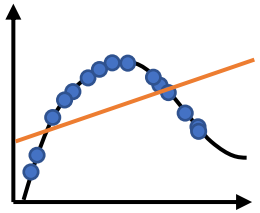
Regardless of what the true function is, how much does our prediction change with dataset?



This error doesn't go away no matter how much data we have!



Bias-variance tradeoff



$$E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - f(x)||^2] \qquad \text{let } \bar{f}(x) = E_{\mathcal{D}\sim p(\mathcal{D})}[f_{\mathcal{D}}(x)]$$

$$= E_{\mathcal{D}\sim p(\mathcal{D})}[||f_{\mathcal{D}}(x) - \bar{f}(x)||^2] + E_{\mathcal{D}\sim p(\mathcal{D})}[||\bar{f}(x) - f(x)||^2]$$

$$= \text{Variance} + \text{Bias}^2$$

If **variance** is too high, we have too little data/too complex a function class/etc. => this is **overfitting**

If **bias** is too high, we have an insufficiently complex function class => this is **underfitting**

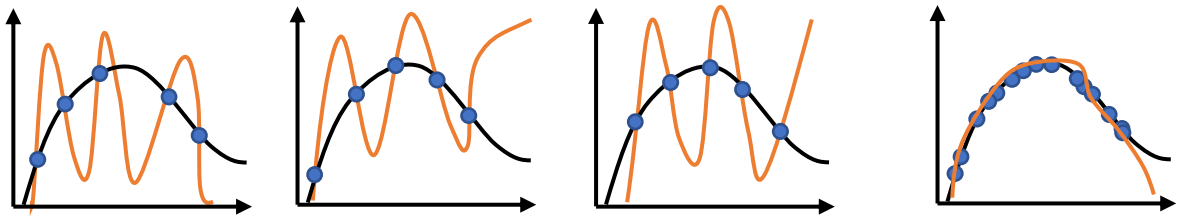
How do we **regulate** the bias-variance tradeoff?

Regularization

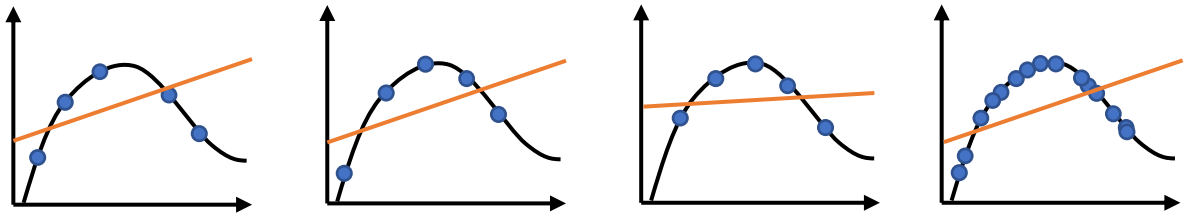
How to regulate bias/variance?

Get more data

addresses **variance**



has no effect on **bias**



Change your model class e.g., 12th degree polynomials to linear functions

Can we "smoothly" restrict the model class?

Can we construct a "continuous knob" for complexity?

Regularization

Regularization: something we add to the loss function to reduce variance

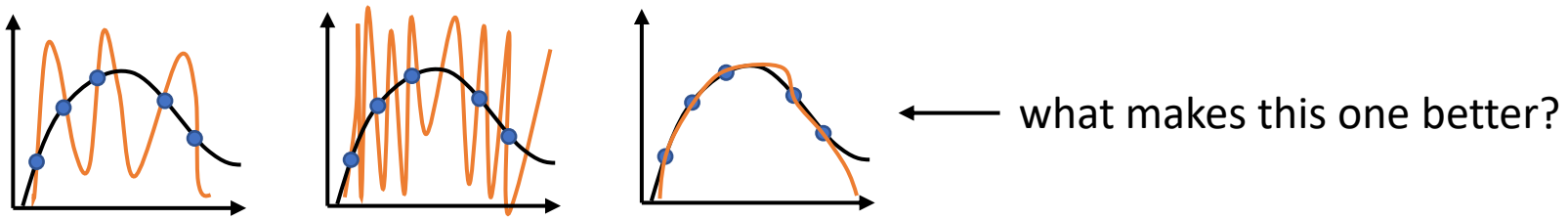
Bayesian interpretation: could be regarded as a prior on parameters **(but this is not the only interpretation!)**

High level intuition:

When we have **high variance**, it's because the data doesn't give enough information to identify parameters

If there is not enough information in the **data**, can we give **more information** through the loss function?

If we provide enough **information** to **disambiguate** between (almost) equally good models, we can pick the best one



all of these solutions have zero training error

The Bayesian perspective

Regularization: something we add to the loss function to reduce variance

Bayesian interpretation: could be regarded as a prior on parameters **(but this is not the only interpretation!)**

Question: Given $\mathcal{D}$, what is the most likely θ?

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

$$p(\theta|\mathcal{D}) = \frac{p(\theta, \mathcal{D})}{p(\mathcal{D})} \propto p(\theta, \mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$$

what is this part?

$p(\theta)$ – how *likely* θ is before we've seen $\mathcal{D}$

this is called the *prior*

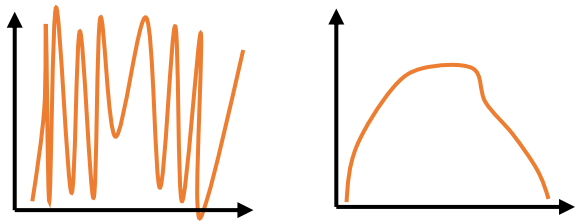
we've seen this part before!

$$p(\mathcal{D}|\theta) = \prod_i p(x_i)p_\theta(y_i|x_i)$$

remember: this is just shorthand for

$$p(y_i|x_i, \theta)$$

Can we pick a prior that makes the *smoother* function more likely?



The Bayesian perspective

Regularization: something we add to the loss function to reduce variance

Bayesian interpretation: could be regarded as a prior on parameters **(but this is not the only interpretation!)**

Question: Given $\mathcal{D}$, what is the most likely θ? $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$

$$p(\theta|\mathcal{D}) = \frac{p(\theta, \mathcal{D})}{p(\mathcal{D})} \propto p(\theta, \mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$$

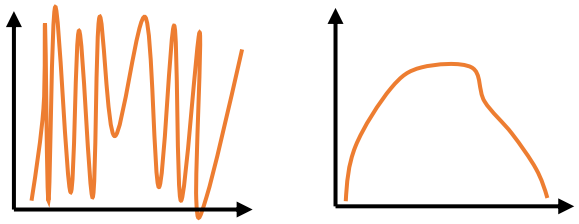
New loss function:

we **choose** this bit

$$\mathcal{L}(\theta) = -\left(\sum_{i=1}^{N} \log p(y_i|x_i, \theta)\right) - \log p(\theta)$$

Example: regularized linear regression

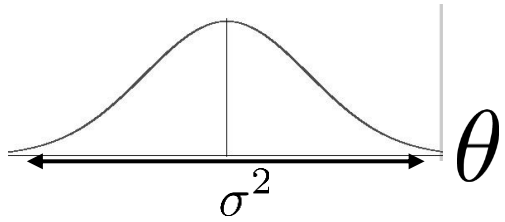
Can we pick a prior that makes the *smoother* function more likely?



$$\mathcal{L}(\theta) = -\left(\sum_{i=1}^{N} \log p(y_i|x_i,\theta)\right) - \log p(\theta)$$

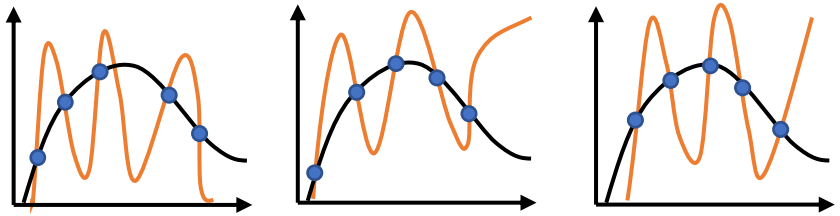
what kind of distribution assigns higher probabilities to **small** numbers?

Simple idea: $p(\theta) = \mathcal{N}(0, \sigma^2)$

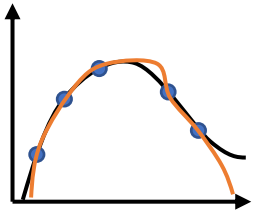


example: linear regression with polynomial features

$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \ldots$$



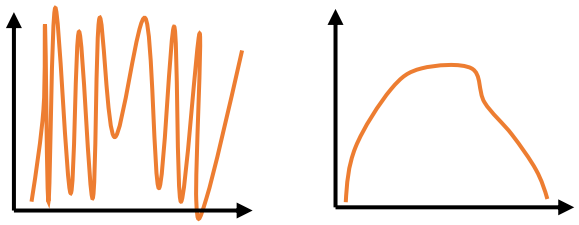
this kind of thing typically requires large coefficients



if we only allow **small** coefficients, best fit might be more like this

Example: regularized linear regression

Can we pick a prior that makes the *smoother* function more likely?



$$\mathcal{L}(\theta) = -\left(\sum_{i=1}^{N} \log p(y_i|x_i,\theta)\right) + \log p(\theta)$$

what kind of distribution assigns higher probabilities to **small** numbers?

Simple idea: $p(\theta) = \mathcal{N}(0, \sigma^2)$

$$\log p(\theta) = \sum_{i=1}^{D} -\frac{1}{2}\frac{\theta_i^2}{\sigma^2} \underbrace{- \log \sigma - \frac{1}{2}\log 2\pi}_{\text{doesn't influence } \theta}$$

$$= -\lambda||\theta||^2 + \text{const}$$

$$\lambda = \frac{1}{2\sigma^2}$$ "hyperparameter"

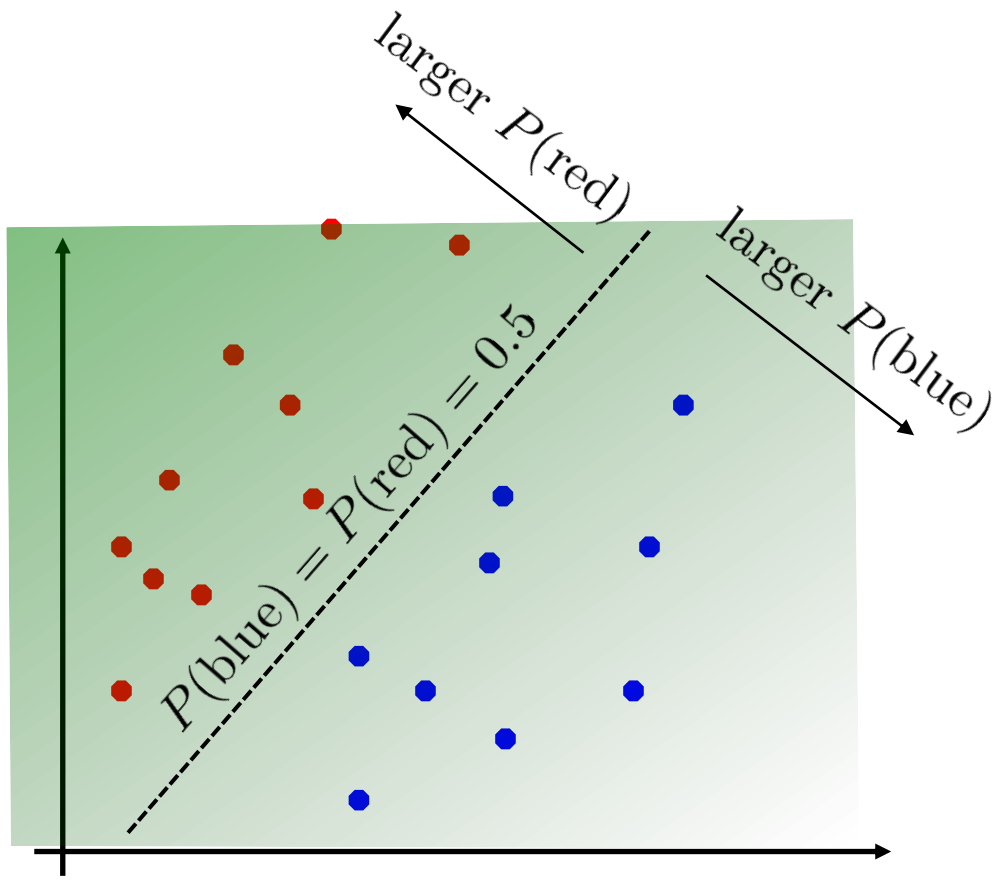
$$\mathcal{L}(\theta) = \left(\sum_{i=1}^{N} ||f_\theta(x_i) - y_i||^2\right) + \lambda||\theta||^2$$

if $\Sigma_\theta(x) = \mathbf{I}$

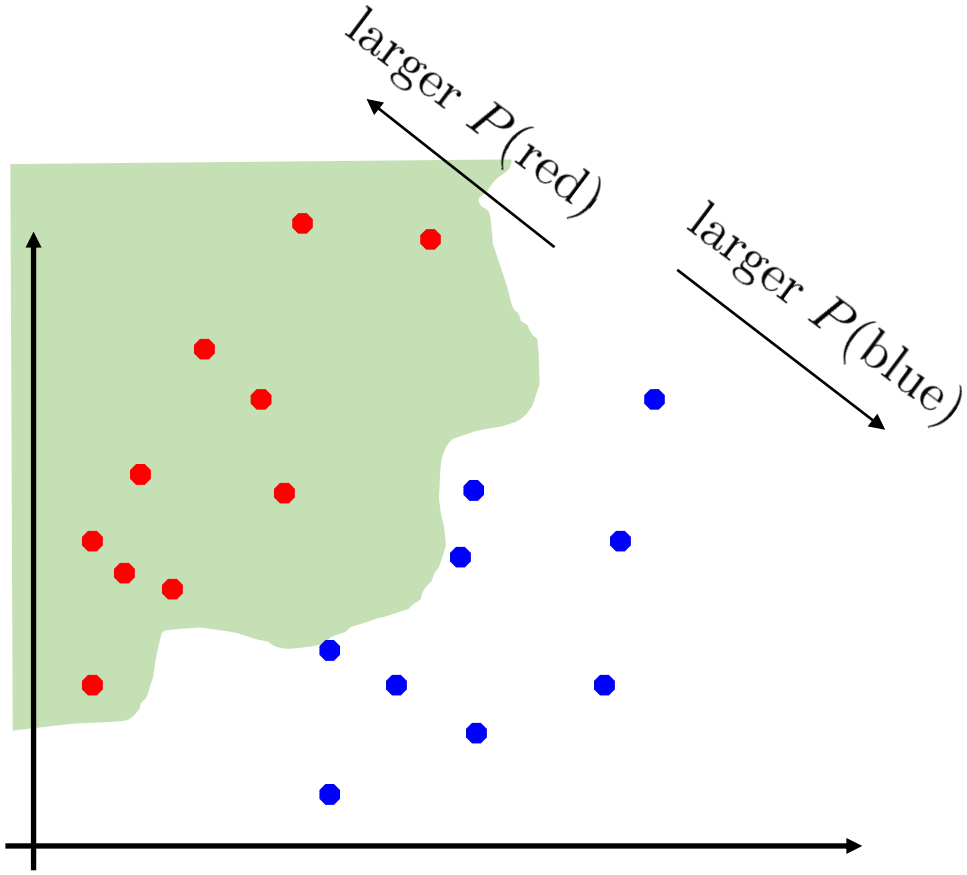
(but we don't care, we'll just select it directly)

Example: regularized logistic regression

what we wanted

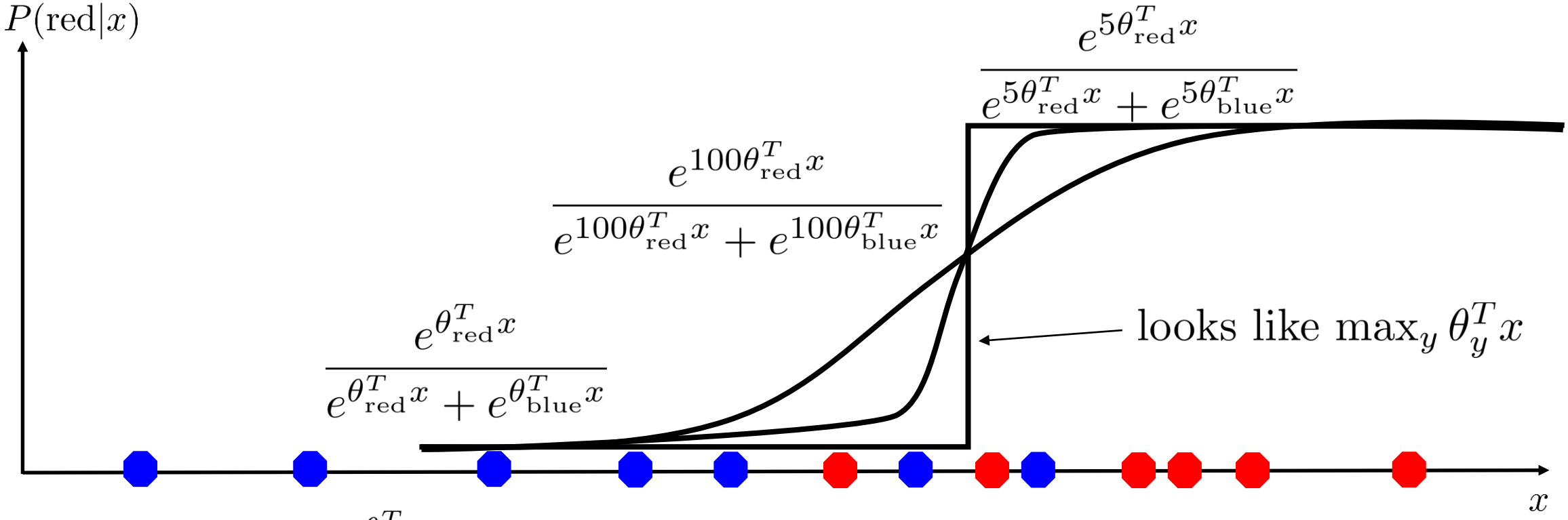


what we got



technically every point is classified correctly

Example: regularized logistic regression

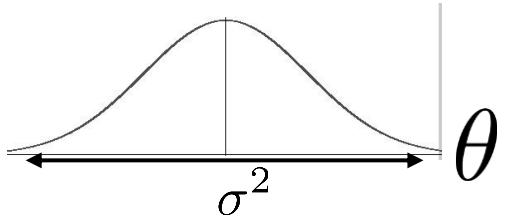


$$P(\text{red}|x) = \frac{e^{\theta_{\text{red}}^T x}}{e^{\theta_{\text{red}}^T x} + e^{\theta_{\text{blue}}^T x}}$$

Larger θ values $\Rightarrow$ sharper probabilities

Solution: prefer **smaller** θ!

Simple idea: $p(\theta) = \mathcal{N}(0, \sigma^2)$



Example: regularized logistic regression

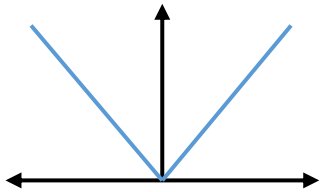
$$\mathcal{L}(\theta) = -\left(\sum_{i=1}^{N} \log p(y_i|x_i,\theta)\right) + \lambda||\theta||^2$$

same prior, but now for a **classification** problem

this is sometimes called **weight decay**

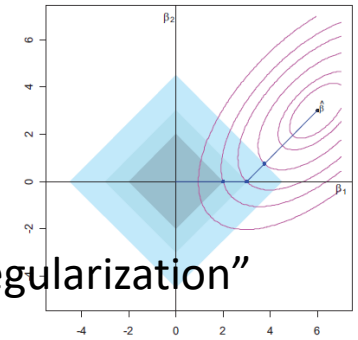
Other examples of regularizers (we'll discuss some of these later):

$$\lambda \sum_{i=1}^{D} |\theta_i|$$

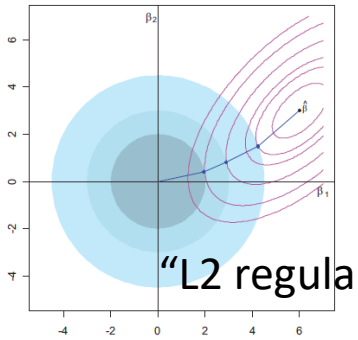


creates a preference for zeroing out dimensions!

$$\lambda \sum_{i=1}^{D} |\theta_i| \qquad \lambda \sum_{i=1}^{D} \theta_i^2$$



"L1 regularization"



"L2 regularization"

Dropout: a special type of regularizer for neural networks

Gradient penalty: a special type of regularizer for GANs

…lots of other choices

Other perspectives

Regularization: something we add to the loss function to reduce variance

Bayesian perspective: the regularizer is prior knowledge about parameters

Numerical perspective: the regularizer makes underdetermined problems well-determined

Optimization perspective: the regularizer makes the loss landscape easier to search

paradoxically, regularizers can sometimes reduce **underfitting** if it was due to poor optimization!

especially common with GANs

In machine learning, any "heuristic" term added to the loss that doesn't depend on data is generally called a regularizer

$$= -\lambda||\theta||^2 + \text{const}$$

"hyperparameter"

Regularizers introduce **hyperparameters** that we have to select in order for them to work well

Training sets and test sets

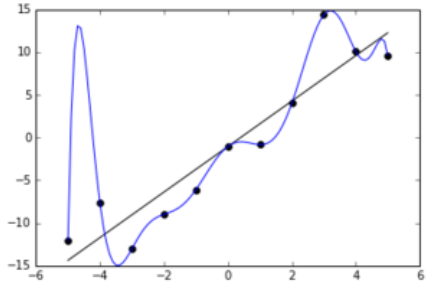
Some questions…

How do we **know** if we are overfitting or underfitting?

How do we select which algorithm to use?

How do we select **hyperparameters**?

One idea: choose whatever makes the **loss** low



$$\mathcal{L}(\theta) = 0$$

Can't diagnose **overfitting** by looking at the training loss!

The machine learning workflow

the dataset

use this for training

$$\mathcal{L}(\theta, \mathcal{D}_{\text{train}})$$

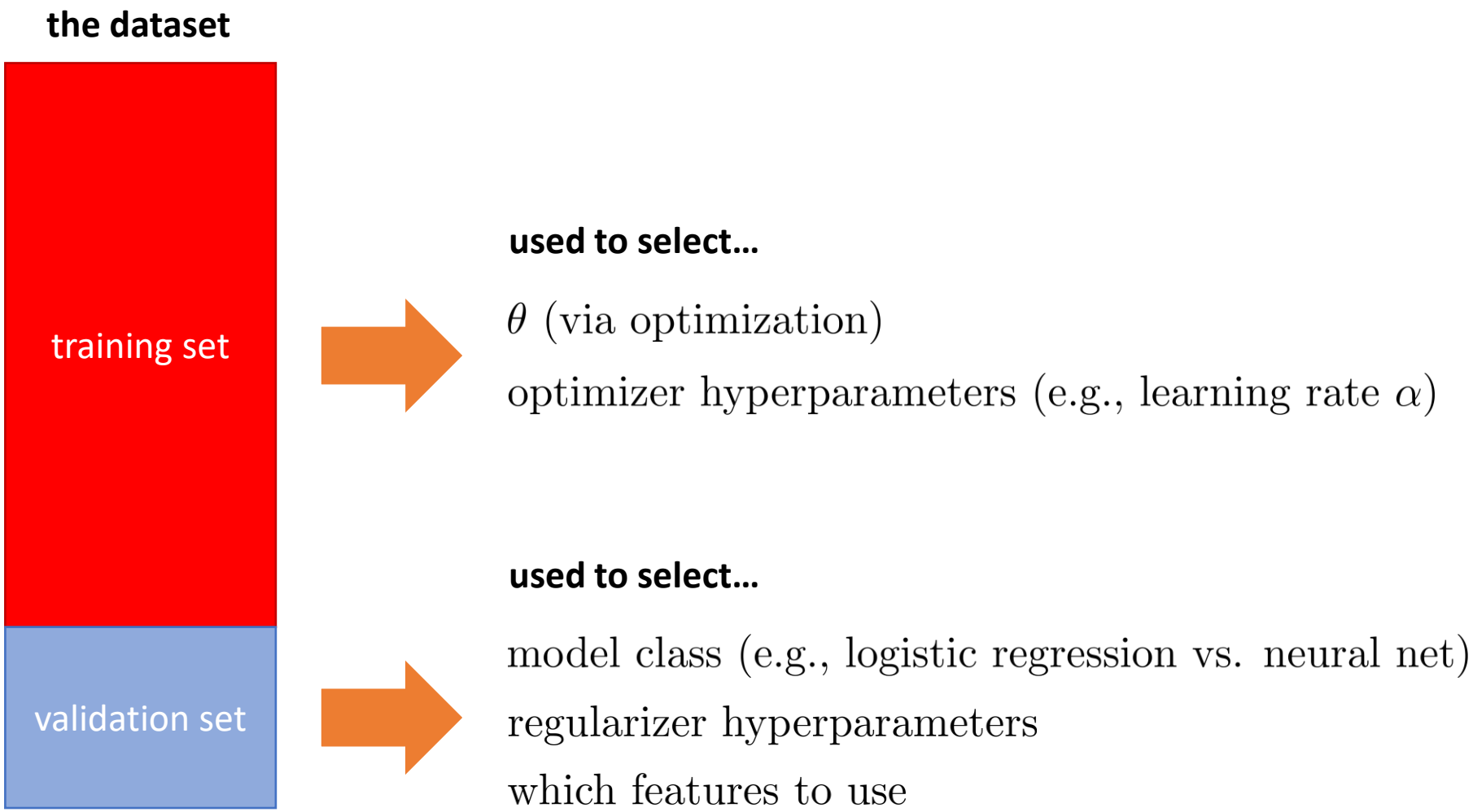
1. Train θ with $\mathcal{L}(\theta, \mathcal{D}_{\text{train}})$
   if $\mathcal{L}(\theta, \mathcal{D}_{\text{train}})$ not low enough
   you are **underfitting**
   decrease regularization
   improve your optimizer
2. Look at $\mathcal{L}(\theta, \mathcal{D}_{\text{val}})$
   if $\mathcal{L}(\theta, \mathcal{D}_{\text{val}}) >> \mathcal{L}(\theta, \mathcal{D}_{\text{train}})$
   you are **overfitting**
   increase regularization

reserve this for…

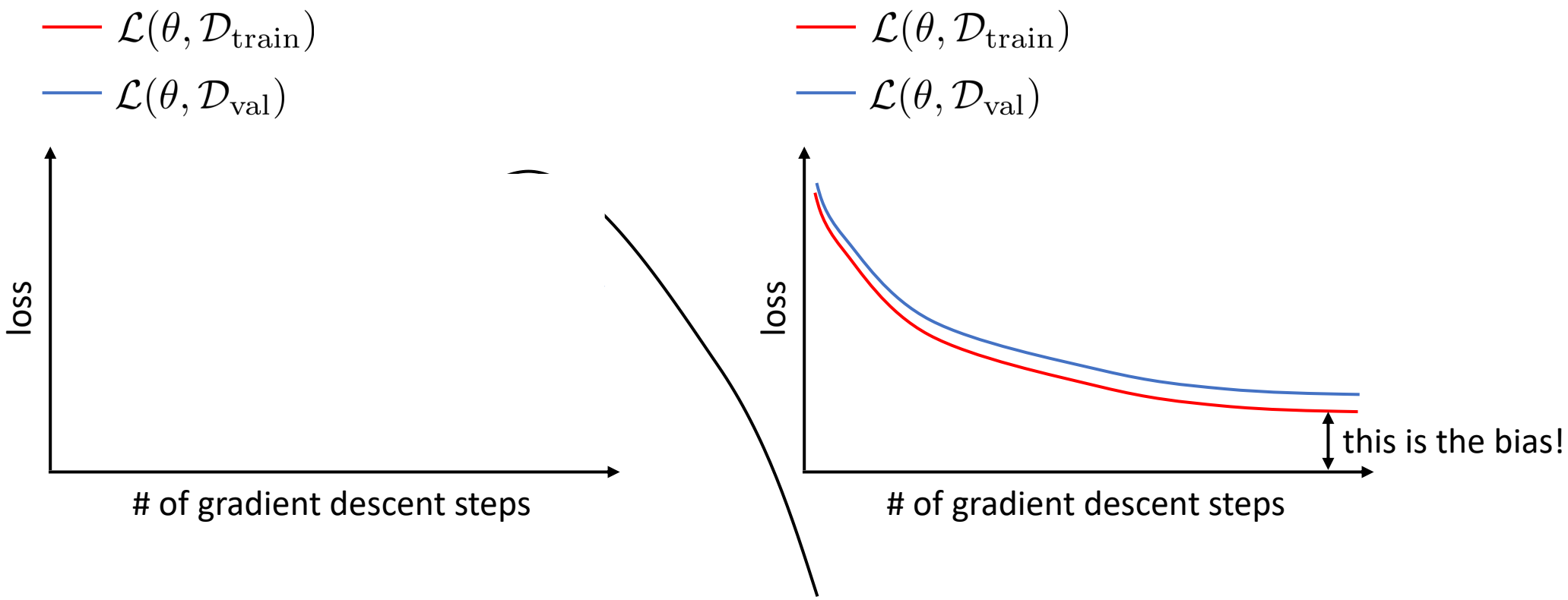
…selecting hyperparameters
…adding/removing features
…tweaking your model class

$$\mathcal{L}(\theta, \mathcal{D}_{\text{val}})$$

The machine learning workflow



Learning curves



Question: can we stop here?

How do we know when to stop?

The final exam

1. Train θ with $\mathcal{L}(\theta, \mathcal{D}_{\text{train}})$
   if $\mathcal{L}(\theta, \mathcal{D}_{\text{train}})$ not low enough
   you are **underfitting**
   decrease regularization
   improve your optimizer
2. Look at $\mathcal{L}(\theta, \mathcal{D}_{\text{val}})$
   if $\mathcal{L}(\theta, \mathcal{D}_{\text{val}}) >> \mathcal{L}(\theta, \mathcal{D}_{\text{train}})$
   you are **overfitting**
   increase regularization

We followed the recipe, **now what?**

How good is our final classifier?

$\mathcal{L}(\theta, \mathcal{D}_{\text{val}})$?

That's **no good** – we already used the validation set to pick hyperparameters!

What if we reserve **another** set for a **final exam** (a kind of… validation validation set!)

The machine learning workflow

the dataset

training set

validation set

test set

used to select…

θ (via optimization)

optimizer hyperparameters (e.g., learning rate α)

used to select…

model class (e.g., logistic regression vs. neural net)

regularizer hyperparameters

which features to use

Used **only** to report final performance

Summary and takeaways

- Where do errors come from?
  - Variance: too much capacity, not enough information in the data to find the right parameters
  - Bias: too little capacity, not enough representational power to represent the true function
  - Error = Variance + Bias^2
  - Overfitting = too much variance
  - Underfitting = too much bias
- How can we trade off bias and variance?
  - Select your model class carefully
  - Select your features carefully
  - Regularization: stuff we add to the loss to reduce variance
- How do we select hyperparameters?
  - Training/validation split
  - Training set is for optimization (learning)
  - Validation set is for selecting hyperparameters
  - Test set is for **reporting final results and nothing else!**