Deep Learning for ECE EECE-580G

Introduction to Learning theory

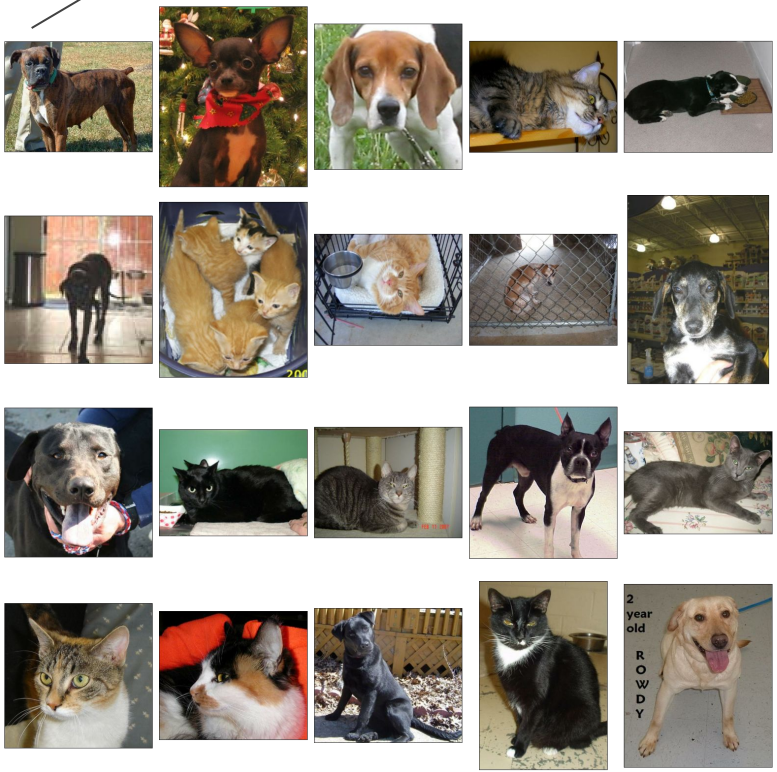
Notations

Notations

- m : samples or observations $x^{(i)}$ are collected from a subject of interest (e.g. for a pet classification task, samples are photos from multiple pets)
- We consider these observations to be labeled, i.e. each sample $x^{(i)}$ is assigned a label $y^{(i)}$ (e.g. for each pet photo we collected, we know the breed of the pet appearing in it)
- This setting is called "supervised learning"[1]
- $D = \left(x^{(i)}, y^{(i)}\right)_{i=i}^{m}$: is the set of the available tuples of observations and labels, this is the dataset at hand
- $X \rightarrow x^{(i)}$, and $Y \rightarrow y^{(i)}$: $x^{(i)}$ and $y^{(i)}$ are i.i.d. samples of the random variables X and Y
- $X \sim P_X$, and $Y \sim P_Y$,
- $(X, Y) \rightarrow D$ the dataset is an observation of the joint random variable (X, Y)
- $x^{(i)} \in \mathcal{X} = \mathbb{R}^n$: n is the dimensionality of the data (e.g. considering the pet photos being of size $H \times W$, $n = 3 \cdot H \cdot W$, the total number of pixels in one image represented in RGB space)
- $y^{(i)} \in \mathcal{Y} = \{0, 1\}$: we consider the simple case of a binary label (e.g. only two pet breeds) the task we are performing is a binary classification task[2]
- $D \in \mathcal{D} = \mathbb{R}^n \times \{0, 1\}$: the cartesian product of $\mathcal{X}$ and $\mathcal{Y}$

Example

$x^{(0)}$

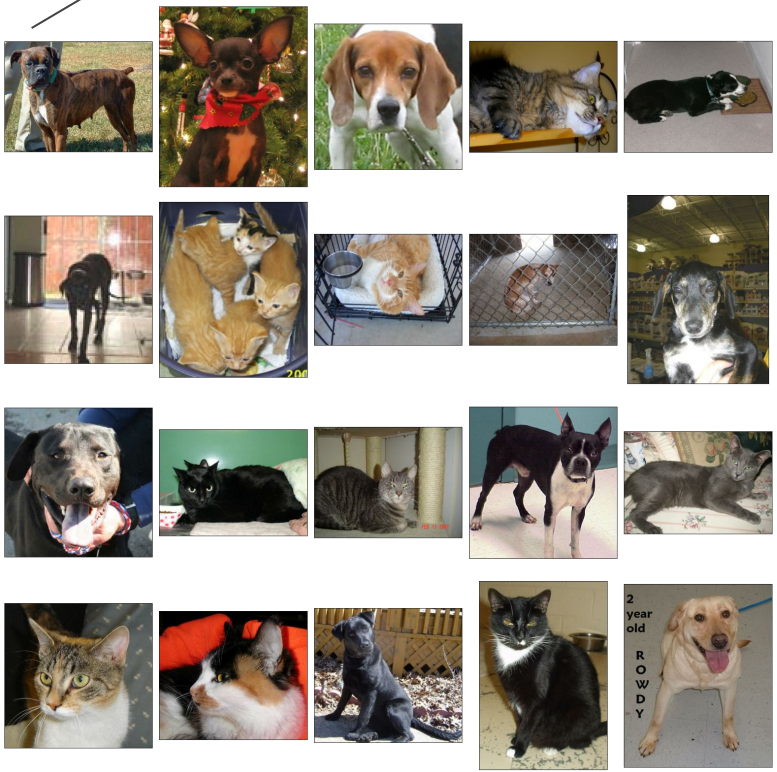


$y^{(0)}$

dog	dog	dog	cat	dog
dog	cat	cat	dog	dog
dog	cat	cat	dog	cat
cat	cat	dog	cat	dog

Example

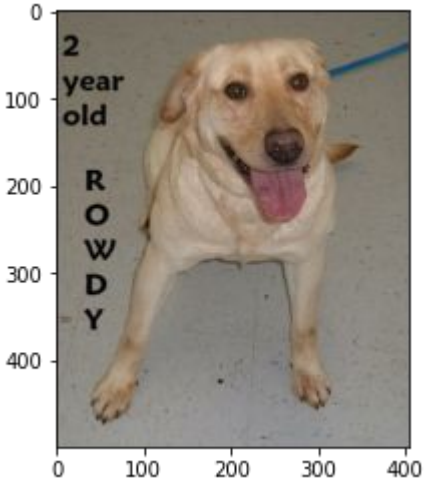
$x^{(0)}$



$y^{(0)}$

1	1	1	0	1
1	0	0	1	1
1	0	0	1	0
0	0	1	0	1

Example



=

```
array([[[108, 110, 113, ..., 110, 109
         [110, 111, 113, ..., 108, 108
         [112, 113, 114, ..., 108, 108
         ...,
array([[[102, 104, 107, ..., 106, 105, 105],
        [104, 105, 107, ..., 104, 104, 104],
        [106, 107, 108, ..., 104, 104, 104],
        ...,
array([[ 88,  90,  93, ...,  94,  93,  93],
       [ 90,  91,  93, ...,  92,  92,  92],
       [ 92,  93,  94, ...,  92,  92,  92],
       ...,
       [133, 137, 141, ..., 147, 144, 140],
       [135, 138, 142, ..., 148, 143, 138],
       [136, 139, 143, ..., 150, 148, 147]], dtype=uint8)
```

Notations

- We also consider that there exists a function f, that maps X to Y, i.e. $f(X) = Y$. This function is obviously unknown
- To link with the definition from the textbook:
  - Task T = binary classification task
  - Experiences E = the dataset D
  - Performance measures P = loss function and notion of risk which will be introduced later

Learning

A learning algorithm

$$\text{A learning algorithm}: \mathcal{D} \to \mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$$
$$D \mapsto \widehat{f},$$

$\mathcal{H}$ = The hypothesis space, we can't look at the space of all functions ...

$\widehat{f}$ = A function that "approximates" the ground truth function

i.e.

$\forall (x, y) \leftarrow (X, Y), \widehat{f}(x) \approx f(x)$

1-Nearest-Neighbor

$$\widehat{f}(x) \mapsto y^{(j)}, j = \text{argmin}_i \left\| x - x^{(i)} \right\|$$

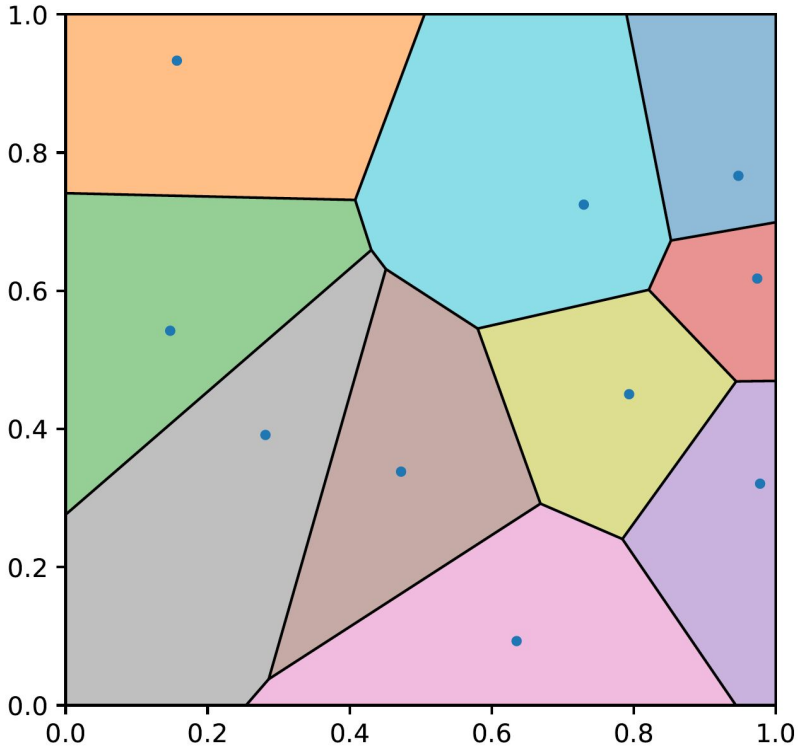
- $\| . \|$ can be any distance but we will use the Euclidean distance
- Maps x to the label of its closest neighbor in the dataset
- $\mathcal{H}$?

1-Nearest-Neighbor

$$\widehat{f}(x) \mapsto y^{(j)}, j = \text{argmin}_i \left\| x - x^{(i)} \right\|$$

- $\| . \|$ can be any distance but we will use the Euclidean distance
- Maps x to the label of its closest neighbor in the dataset
- $\mathcal{H}$ can be described using m parameters, the number of data points
- = the representational capacity c

1-Nearest-Neighbor



Empirical Risk Minimization

Risk - loss function $l\left(\widehat{f}(x), y\right)$

- We need to quantify “goodness” of fit of the approximation
  - 0-1 loss
  - Weighted 0-1 loss with a cost matrix
  - “Smooth” surrogate losses (later in the course)
  - ...

Empirical risk - Generalization risk

$$\widetilde{L}(h) = \frac{1}{m}\sum_{i=1}^{m} l(h(x^{(i)}), y^{(i)})$$

- We don't know the distributions of (X, Y)
- We are interested in the "generalization risk"

$$L(h) = E\left[l(h(X), Y)\right]$$

- Luckily when samples are i.i.d.

$$E\left[\widetilde{L}(h)\right] = E\left[\frac{1}{m}\sum_{i=1}^{m} l(h(x^{(i)}), y^{(i)})\right] = E\left[l(h(X), Y)\right] = L(h)$$

Empirical risk minimizer

$$\widetilde{L}(h) = \frac{1}{m}\sum_{i=1}^{m} l(h(x^{(i)}), y^{(i)})$$

- So let's pick the approximation function as:

$$\widehat{f} = \text{argmin}_{h \in \mathcal{H}} \widetilde{L}(h)$$

- The "empirical risk minimizer"
- How does the empirical risk of empirical risk minimizer link to its generalization risk

$$E\left[\widetilde{L}(\widehat{f})\right] \stackrel{?}{=} L(\widehat{f})$$

Empirical risk minimizer

$$\widetilde{L}(h) = \frac{1}{m}\sum_{i=1}^{m} l(h(x^{(i)}), y^{(i)})$$

- So let's pick the approximation function as:

$$\widehat{f} = \text{argmin}_{h \in \mathcal{H}} \widetilde{L}(h)$$

- The "empirical risk minimizer"
- How does the empirical risk of empirical risk minimizer link to its generalization risk

$$E\left[\widetilde{L}(\widehat{f})\right] \neq L(\widehat{f})$$

- Why?

Empirical risk minimizer

- $l(\widehat{f}(x^{(i)}), y^{(i)})$ Are not necessarily i.i.d anymore
- $\widehat{f}$ Is dependent on the entire dataset! $\widehat{f} = \text{argmin}_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l(h(x^{(i)}), y^{(i)})$
- In general, the empirical risk of empirical risk minimizer is a **biased estimator** of its generalization risk

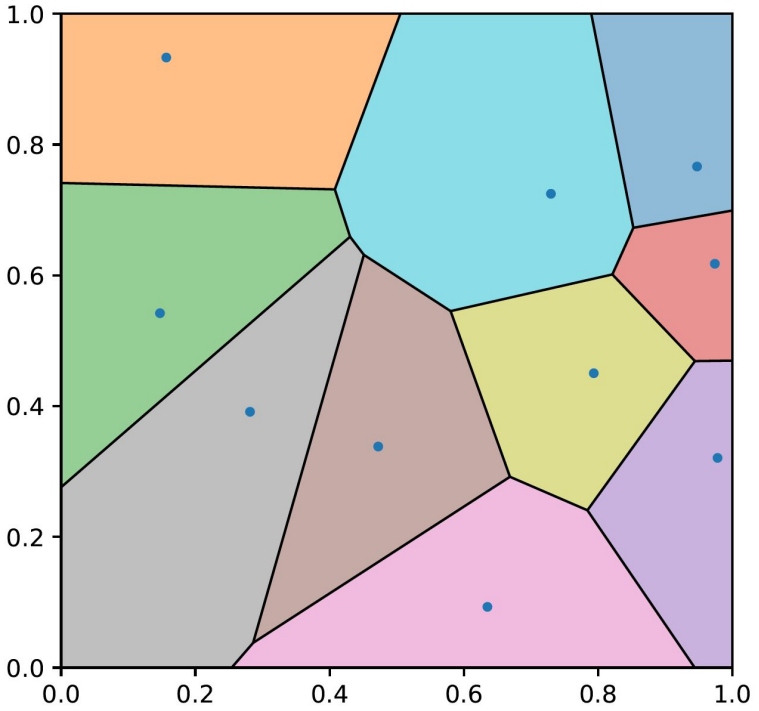
$$E\left[\widetilde{L}(\widehat{f})\right] \neq L(\widehat{f})$$

Example - 1-Nearest-Neighbor

- $\widetilde{L}(\widehat{f}) = 0$

  Because the 1-Nearest Neighbor to each datapoint is itself!

- But in almost all cases, $L(\widehat{f}) \neq 0$



Risk decomposition

Risk decomposition

$$\widehat{f} = \operatorname{argmin}_{h \in \mathcal{H}} \widetilde{L}(h)$$ Empirical risk minimizer

$$f^* = \operatorname{argmin}_{h \in \mathcal{H}} L(h)$$ Generalization risk minimizer

- f^* Is unknown and impossible to compute, we can't compute the expectations...

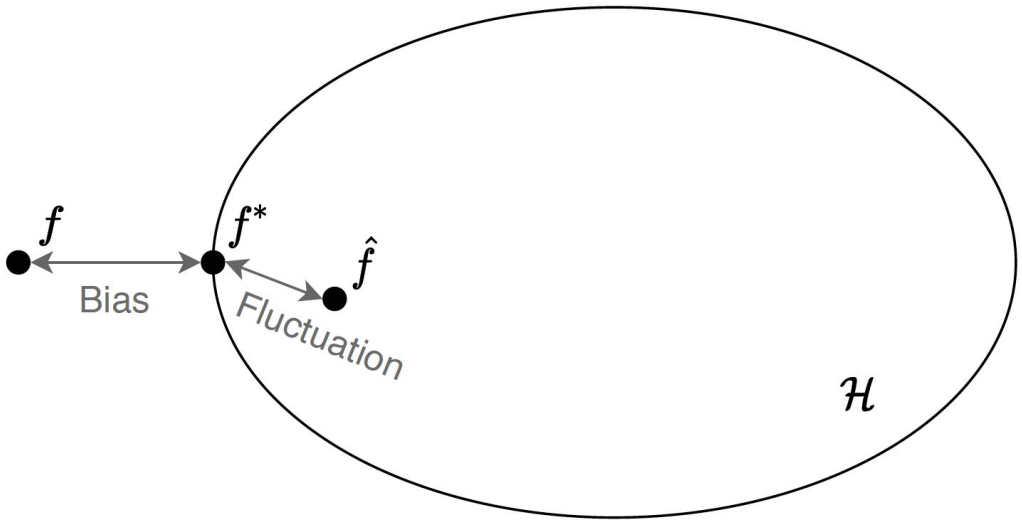
- Is $L(\widehat{f})$ close enough to $L(f^*)$?

Risk decomposition

$$L(f^*) \leq L(\widehat{f}) \leq \underbrace{L(f^*)}_{\text{Bias}} + \underbrace{2 \max_{h \in \mathcal{H}} \left| L(h) - \widetilde{L}(h) \right|}_{\text{Fluctuation (variance)}}$$

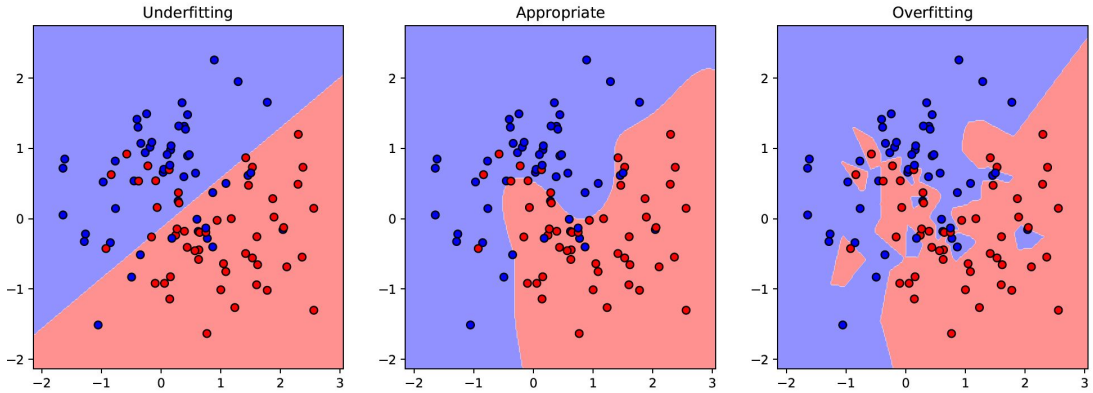
- Proof in handout

Risk decomposition



Bias variance trade-off

- Larger $\mathcal{H} \longrightarrow$ **smaller bias**, but **higher fluctuations**
- The choice of $\mathcal{H}$ is not trivial, needs to be validated by the data too
- This is called the **bias-variance trade-off**



Training/validation/testing sets

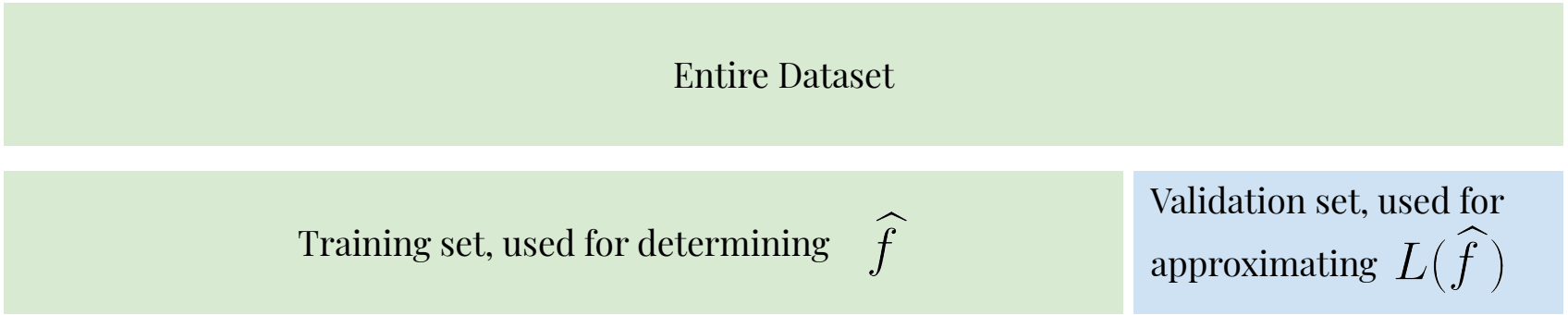
The need for a Validation set

- Remember that $E\left[\widetilde{L}(\widehat{f})\right] \neq L(\widehat{f})$
- But what if we computed $\widetilde{L}(\widehat{f})$ using a different set of data, not used to find $\widehat{f}$
- I.e. we split D into D^{Train} and $D^{\text{Validation}}$

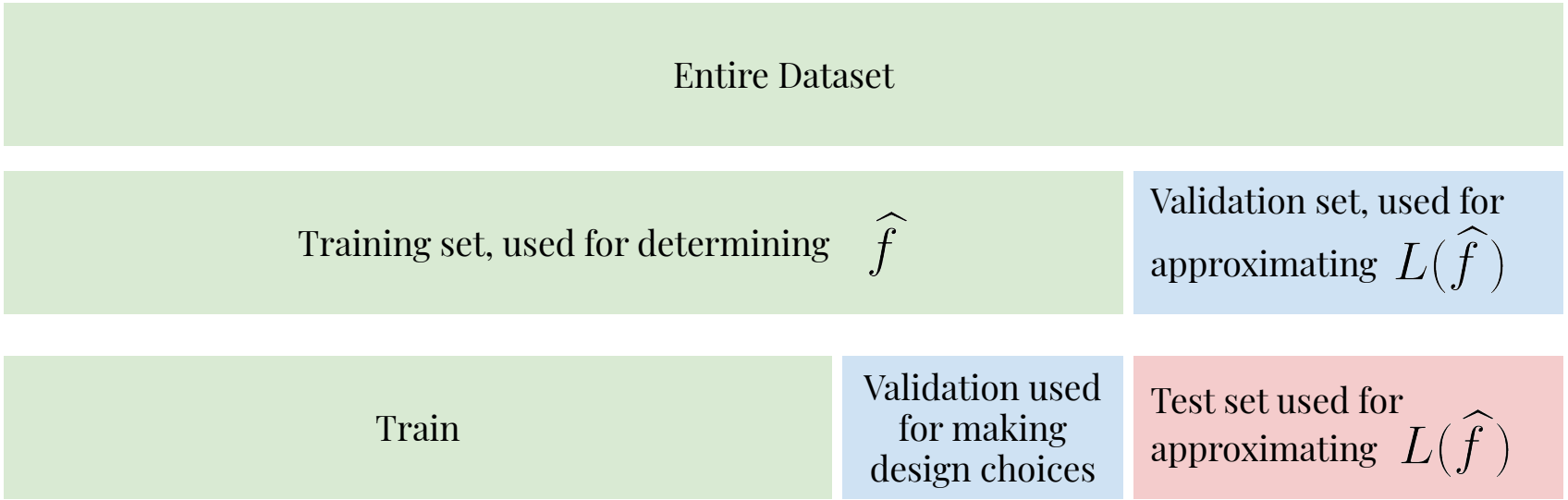
$$\widetilde{L}^{\text{Validation}}(\widehat{f}) = \frac{1}{m} \sum_{\left(x^{(i)}, y^{(i)}\right) \in D^{\text{Validation}}} l(\widehat{f}(x^{(i)}), y^{(i)})$$

- Is unbiased now!

The need for a Validation set



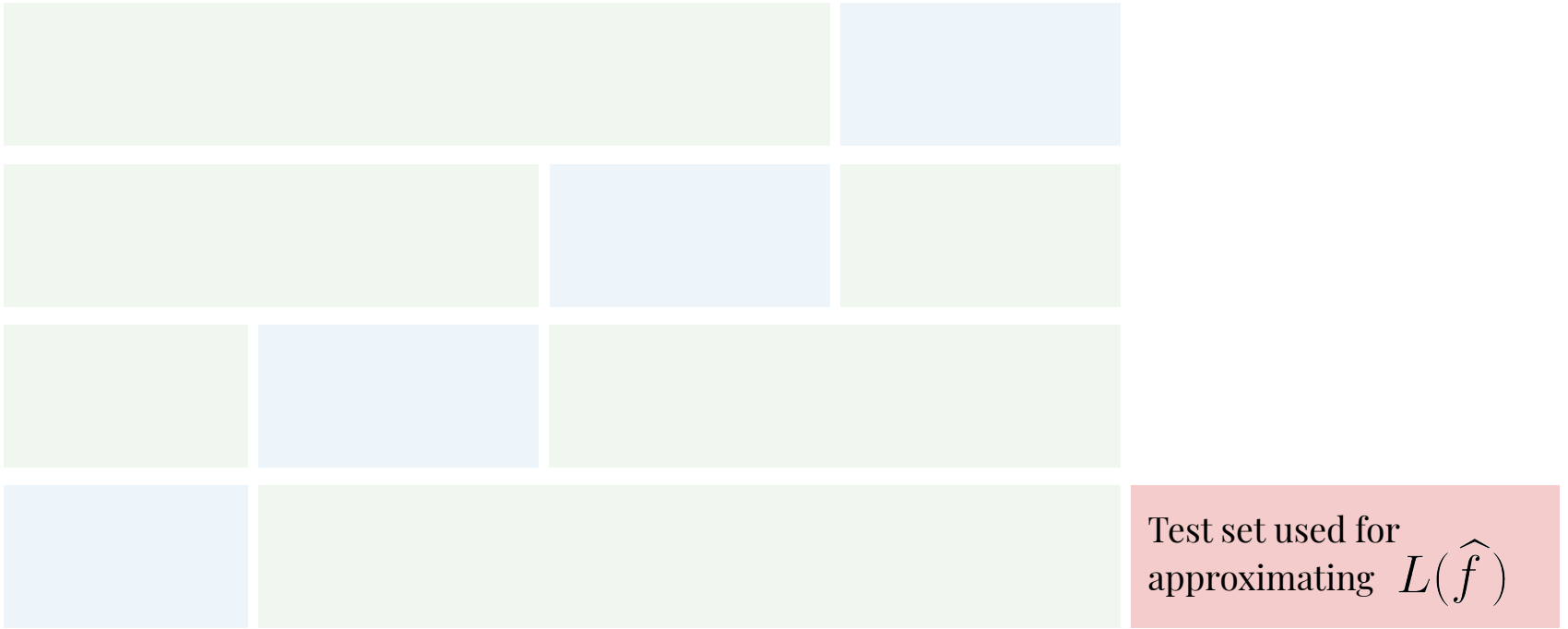
The need for a Validation set



The need for a Validation set

Used for
design choices

The need for a Validation set



End