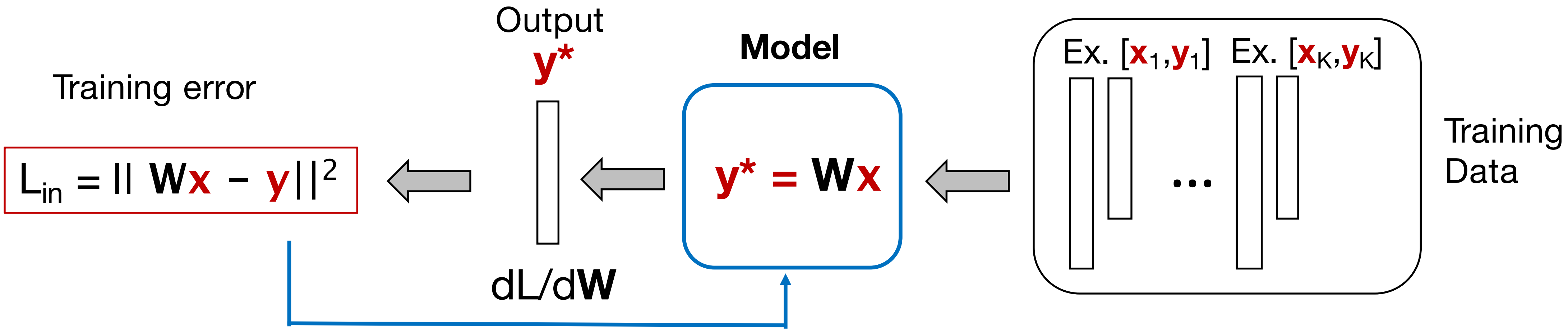
# Lecture 8: Theoretical basics of machine learning

Machine Learning and Imaging

BME 590L
Roarke Horstmeyer

# Last time: the linear classification model – what's not to like?



Training error

$L_{in} = || \mathbf{W}\mathbf{x} - \mathbf{y}||^2$

Output

$\mathbf{y}^*$

dL/d**W**

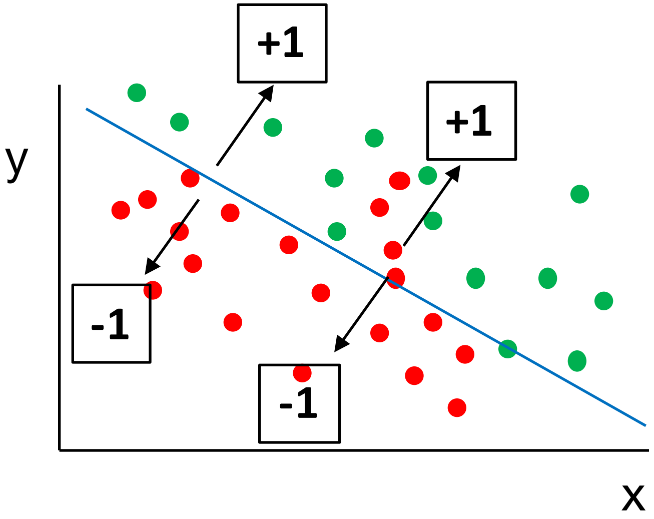
**Model**

$\mathbf{y}^* = \mathbf{W}\mathbf{x}$

Ex. $[\mathbf{x}_1, \mathbf{y}_1]$ Ex. $[\mathbf{x}_K, \mathbf{y}_K]$

…

Training Data

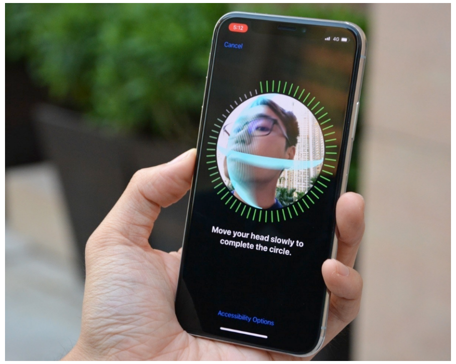


1. Can only separate data with lines (hyper-planes)…

2. We only allowed for binary labels (y = +/- 1)

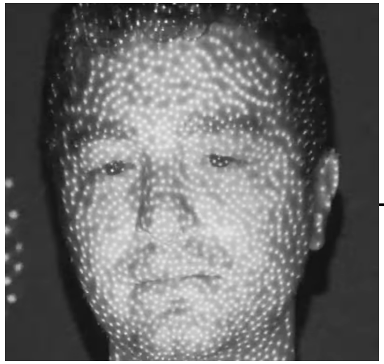
3. Error function $L_{in}$ inherently makes assumptions about statistical distribution of data

# Cost functions matter: a simple example



→

x



$$\rightarrow f(\textcolor{red}{x}, W) \rightarrow \begin{cases} +1 = \text{You} \\ -1 = \text{Bad guy} \end{cases}$$

What if you're a CIA agent?

$$L_{in} = \underbrace{\mathbf{100{,}000}\ \mathrm{ReLU}[f(x, W)-y]}_{\text{BIG penalty for intruder}} + \underbrace{\mathrm{ReLU}[y-f(x, W)]}_{\text{Don't mind about annoyance...}}$$

f(x, W)

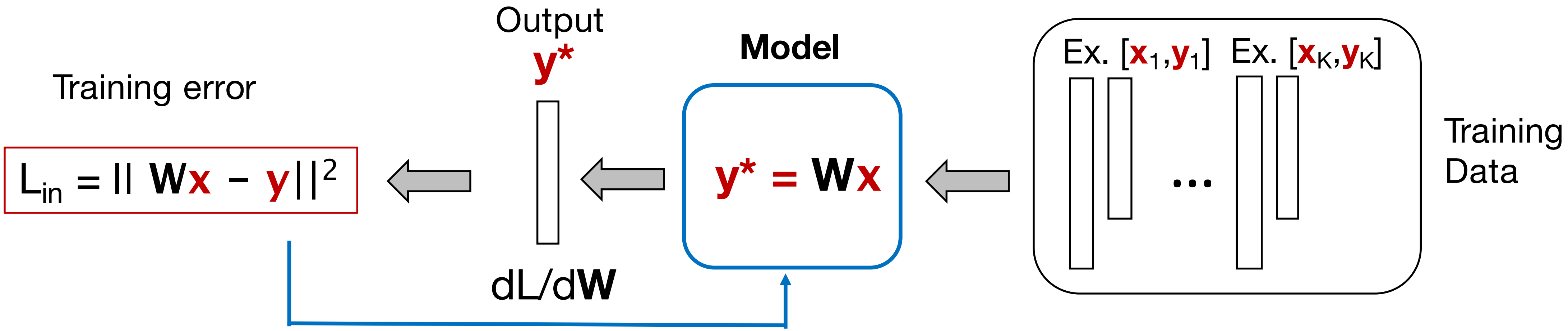
| y | +1 | -1 |
|---|---|---|
| +1 | No Error | False reject |
| -1 | False accept | No Error |

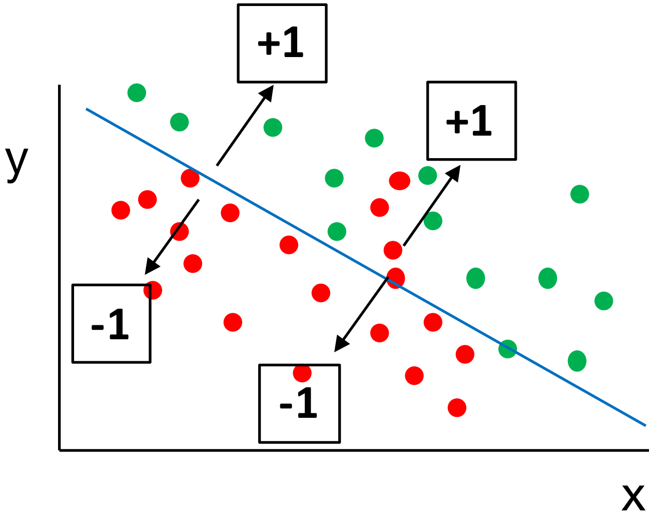
It's you, but you can't get in...

Letting an intruder in

https://www.cnet.com/how-to/apple-face-id-everything-you-need-to-know/

# Last time: the linear classification model – what's not to like?





1. Can only separate data with lines (hyper-planes)…

2. We only allowed for binary labels (y = +/- 1)

3. Error function $L_{in}$ inherently makes assumptions about statistical distribution of data

## Deriving cost function for logistic classification for probabilistic outputs

Maximize $P(y_1,y_2\ldots y_N \,|\, \mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N) = \prod_{n=1}^{N} \theta(y_n \mathbf{w}^T \mathbf{x}_n)$

Minimize $$-\frac{1}{N} \ln \left( \prod_{n=1}^{N} \theta(y_n \mathbf{w}^T \mathbf{x}) \right)$$

Minimize $$\frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{\theta(y_n \mathbf{w}^T \mathbf{x})} \right)$$

Use relationship $$\theta(a) = \frac{1}{1 + e^{-a}}$$

Minimize $$L_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}})$$

$$L_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbf{w}^T \mathbf{x})^2$$

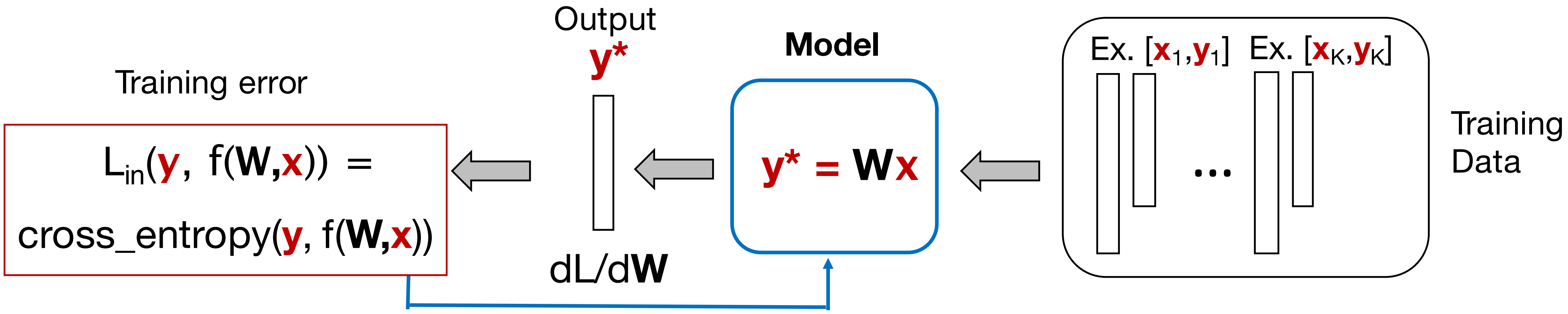
Cross entropy error for logistic classification

Requires iterative solution to minimize

Mean-square error for linear classification

Closed form solution available

# The linear classification model – what's not to like?



Training error

$L_{in}(\mathbf{y}, f(\mathbf{W},\mathbf{x})) =$

cross_entropy($\mathbf{y}$, f($\mathbf{W},\mathbf{x}$))

Output

$\mathbf{y}^*$

dL/d**W**

**Model**

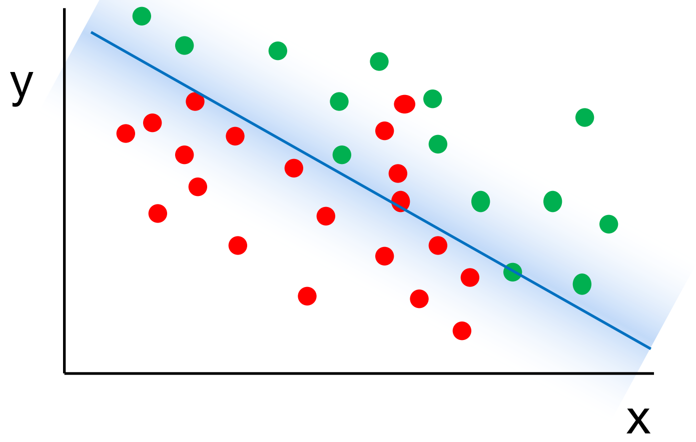
$\mathbf{y}^* = \mathbf{W}\mathbf{x}$

Ex. [$\mathbf{x}_1, \mathbf{y}_1$] Ex. [$\mathbf{x}_K, \mathbf{y}_K$]

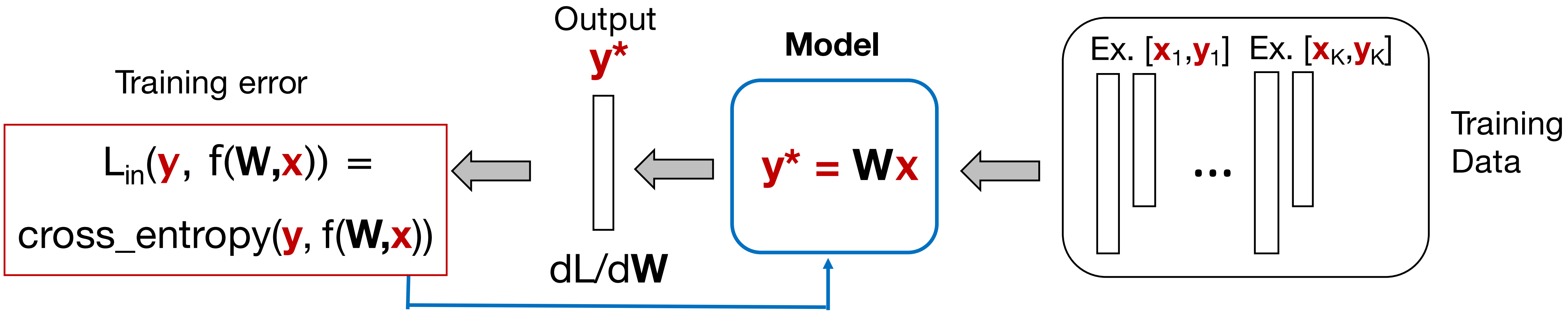
...

Training Data

Probabilistic mapping to y



# The linear classification model – what's not to like?



Training error

$L_{in}(\mathbf{y}, f(\mathbf{W},\mathbf{x})) =$

$\text{cross\_entropy}(\mathbf{y}, f(\mathbf{W},\mathbf{x}))$

Output

$\mathbf{y^*}$

$dL/d\mathbf{W}$

**Model**

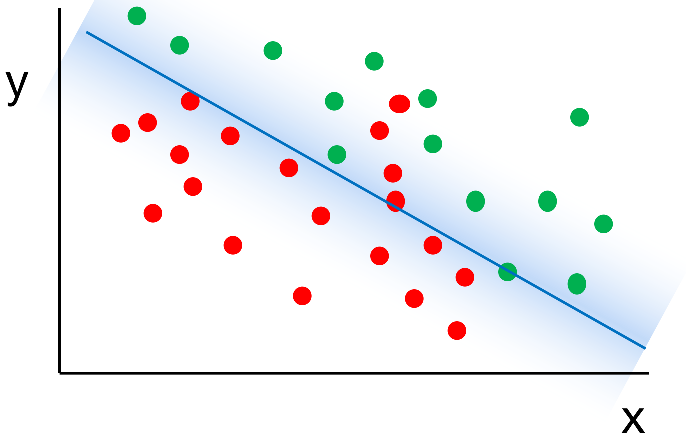
$\mathbf{y^*} = \mathbf{Wx}$

Ex. $[\mathbf{x}_1,\mathbf{y}_1]$ Ex. $[\mathbf{x}_K,\mathbf{y}_K]$

…

Training Data

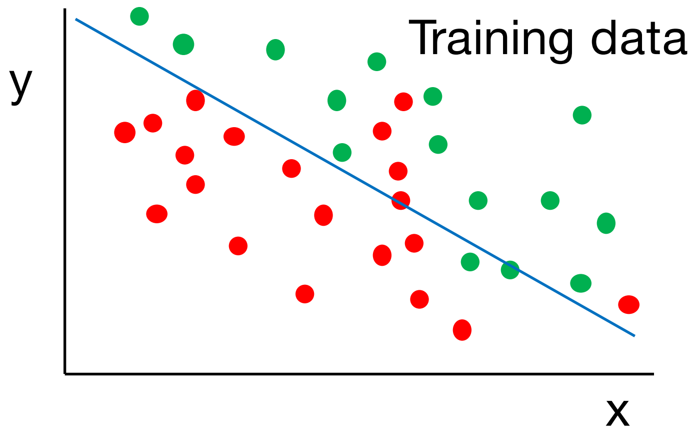
Probabilistic mapping to y



1. Can only separate data with lines (hyper-planes)…

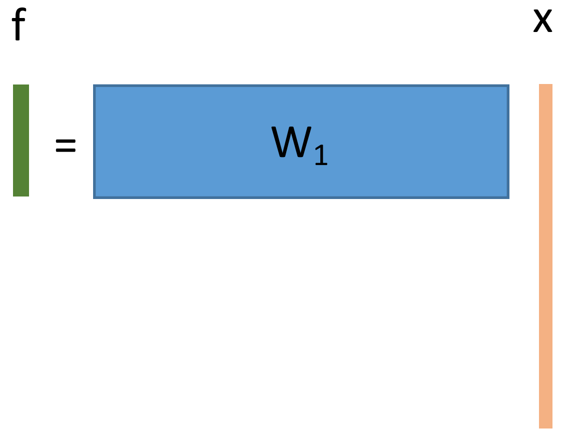
2. We only allowed for binary labels (y = +/- 1)

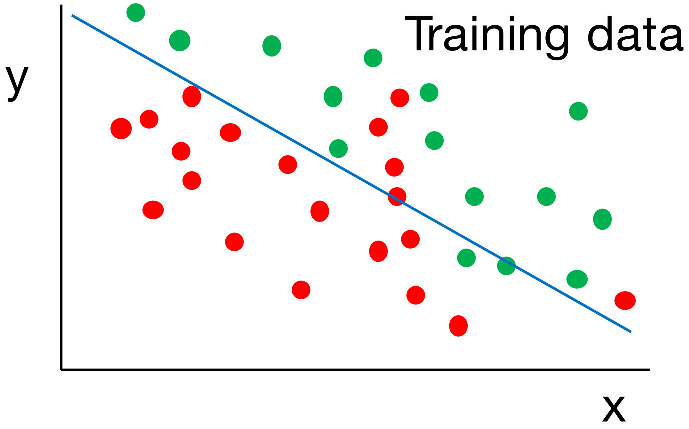
3. Error function $L_{in}$ inherently makes assumptions about statistical distribution of data



$$f = W_1 x$$

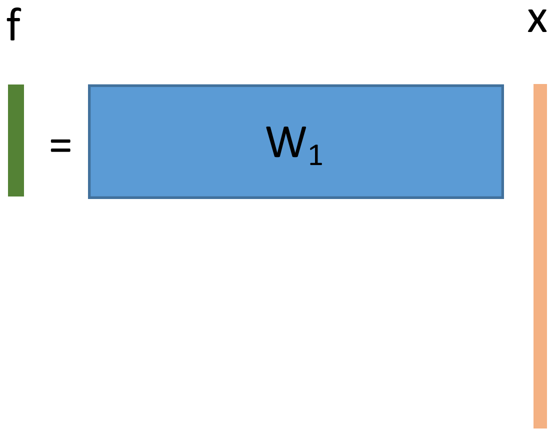
Learned *f:* not flexible





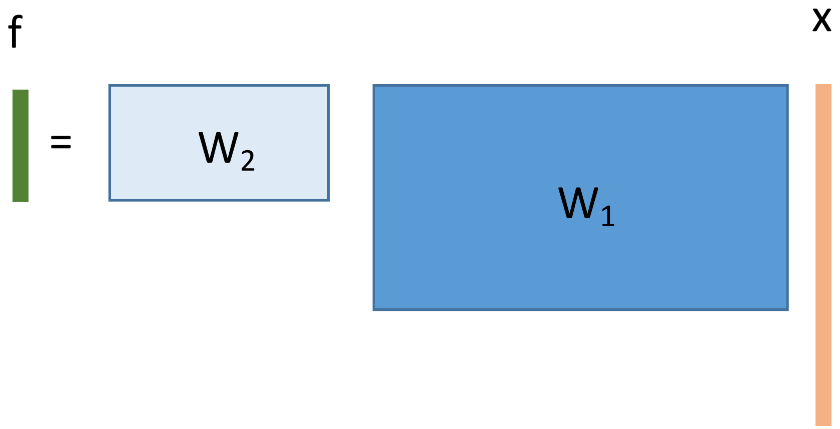
$$f = W_1 x$$

Learned *f:* not flexible



Can we add flexibility by multiplying with another weight matrix?

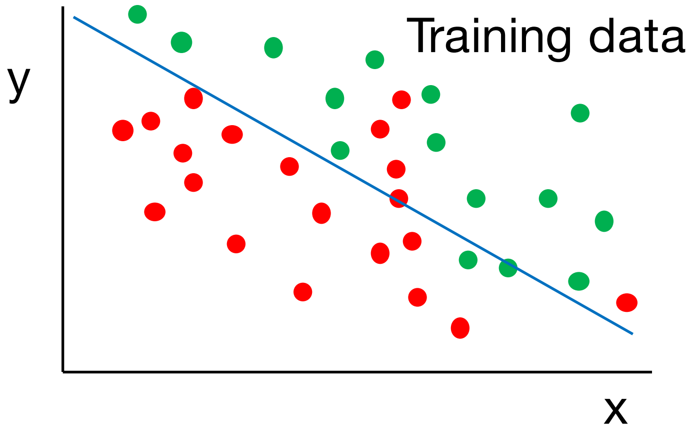
$$\begin{cases} f_1 = W_1 x + b_1 \\ f_2 = W_2 f_1 + b_2 \end{cases}$$



$$f_2 = W_2(W_1 x + b_1) + b_2$$

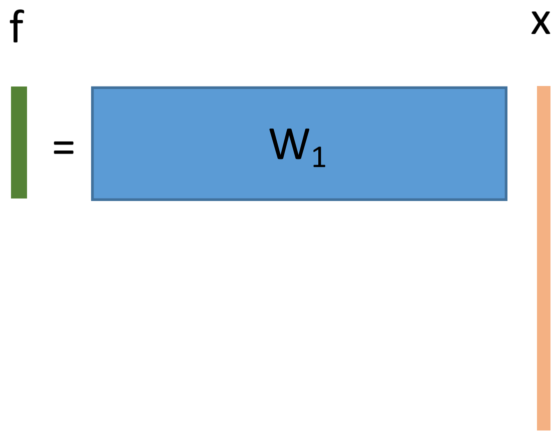
$$f_2 = W' x + b'$$

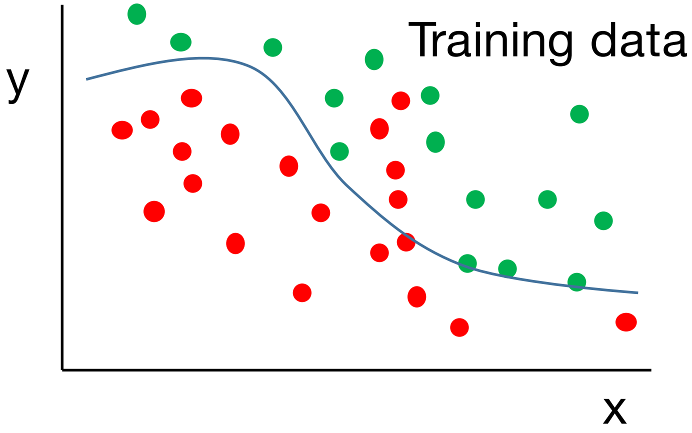
Unfortunately not…



$$f = W_1 x$$

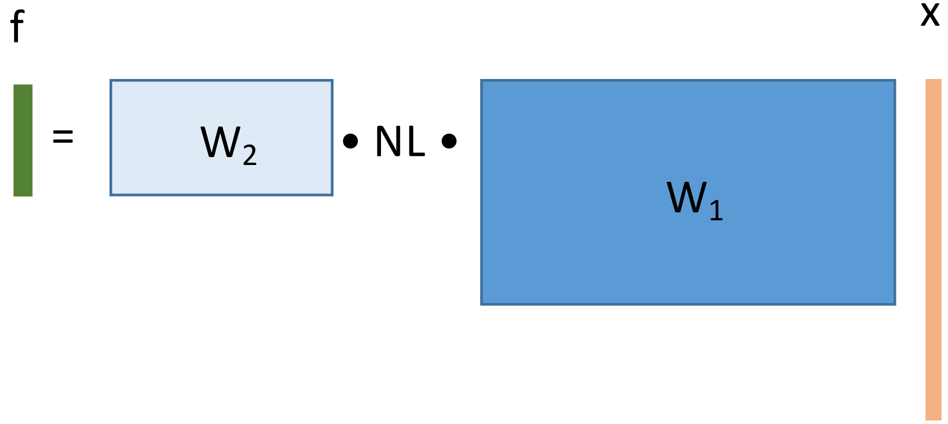
Learned *f:* not flexible

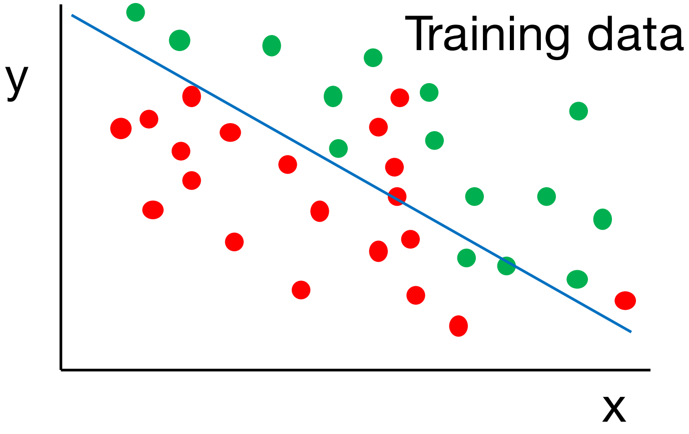




$$f = W_2 \max(W_1 x, 0)$$

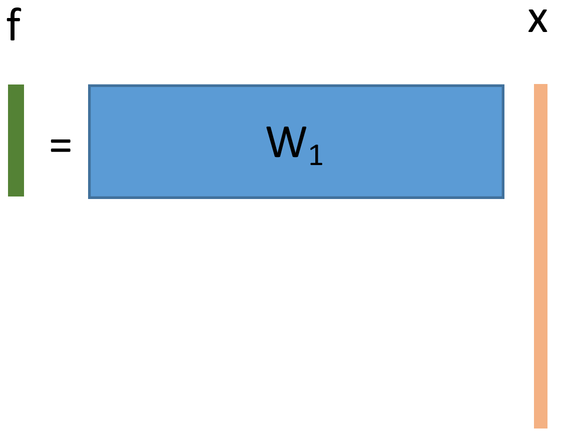
Learned *f:* a bit flexible

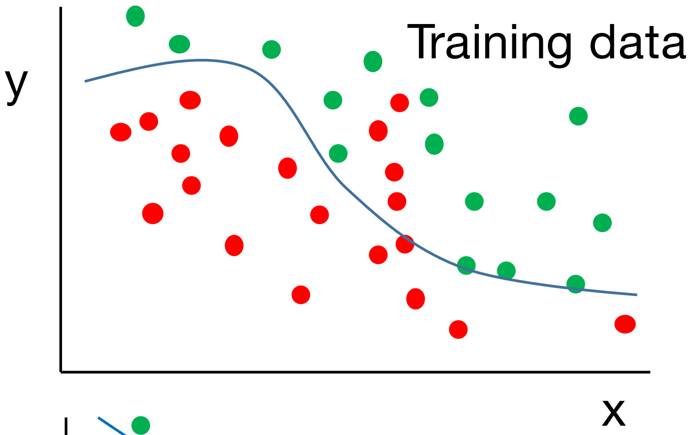




$$f = W_1 x$$

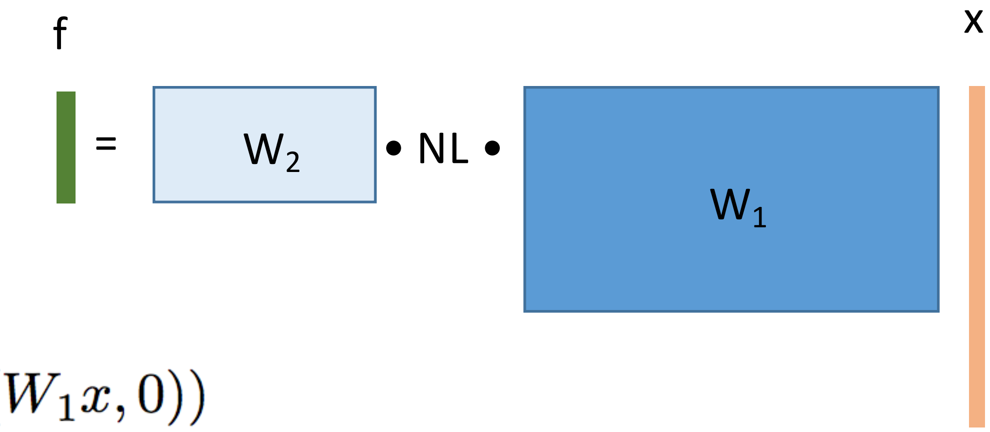
Learned *f:* not flexible

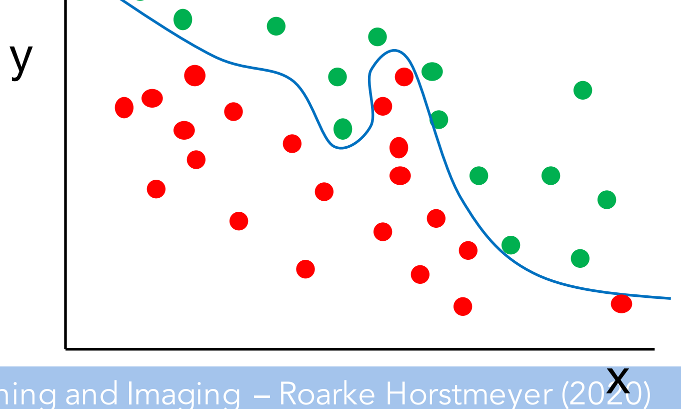




$$f = W_2 \max(W_1 x, 0)$$

Learned *f:* a bit flexible





$$f = W_3 \max(0, W_2 \max(W_1 x, 0))$$

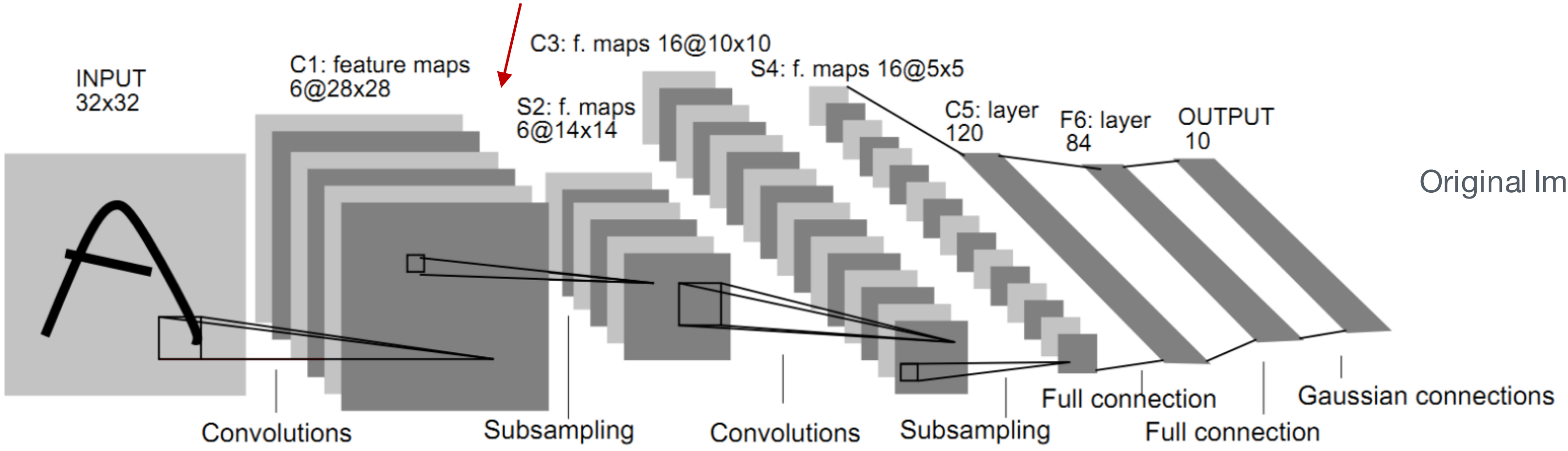
Learned *f:* more flexible

Does it generalize???

↓ We can keep adding these “layers”…

# Getting us to Convolutional Neural Networks

After, apply non-linearity and sub-sampling



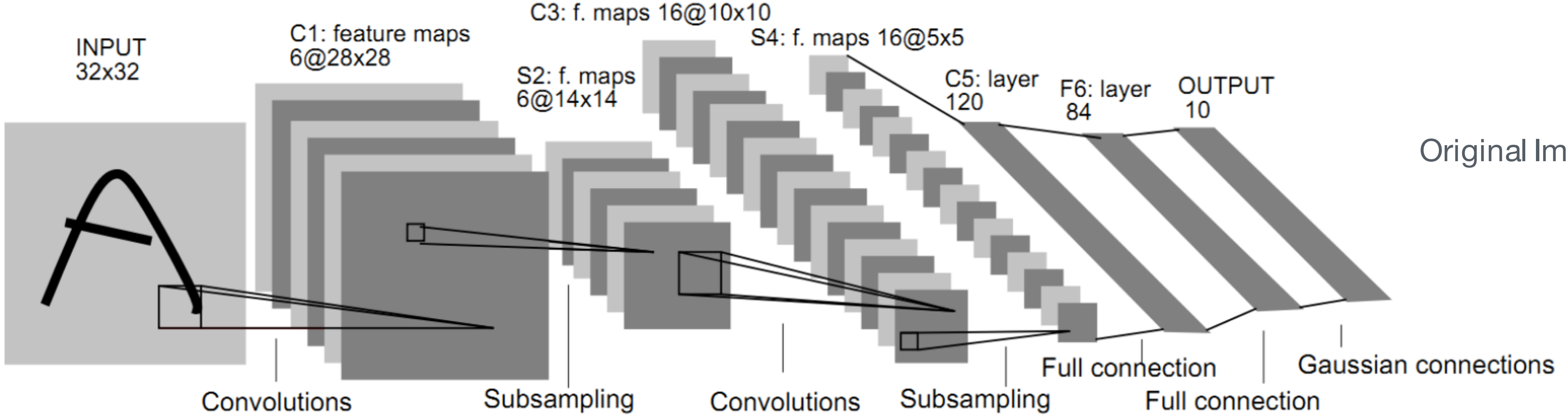
Original Image published in [LeCun et al., 1998]

Each matrix W is a convolution matrix

Repeat a few times

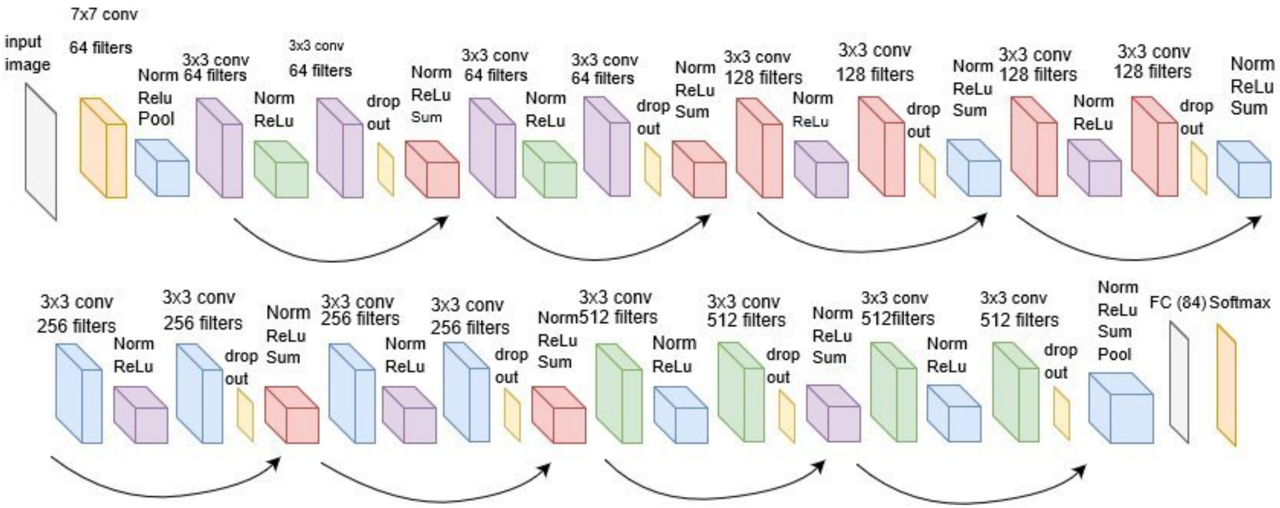
At the end, use a full W for a final matrix multiplication

# Getting us to Convolutional Neural Networks



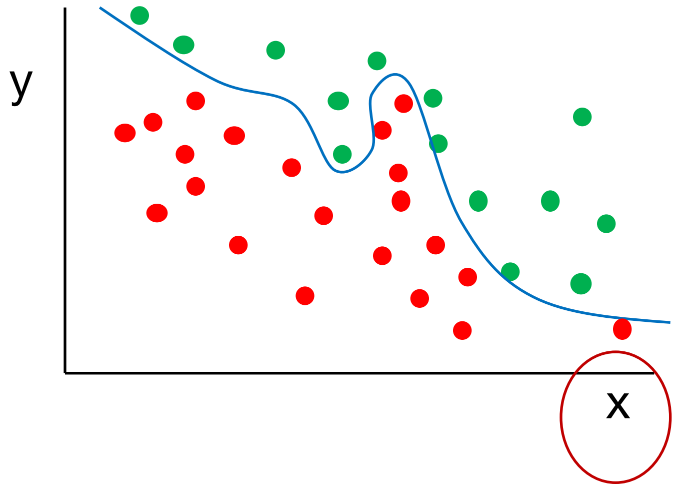
Original Image published in [LeCun et al., 1998]

In practice, this process is repeated many times:



# Aside #1 before convolutional neural network details

Q: Can we try to avoid making these learning models too complicated?

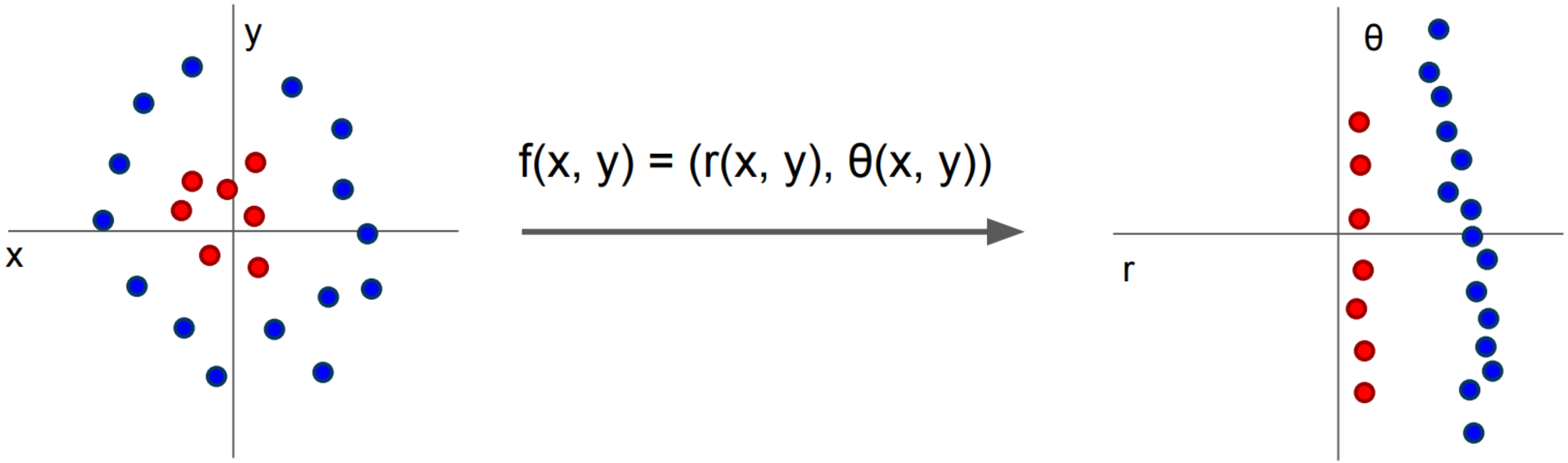


Learned *f:* more flexible

**Does it generalize???**

A: Yes, by transforming the data coordinates *before* classification

# Image Features: Motivation

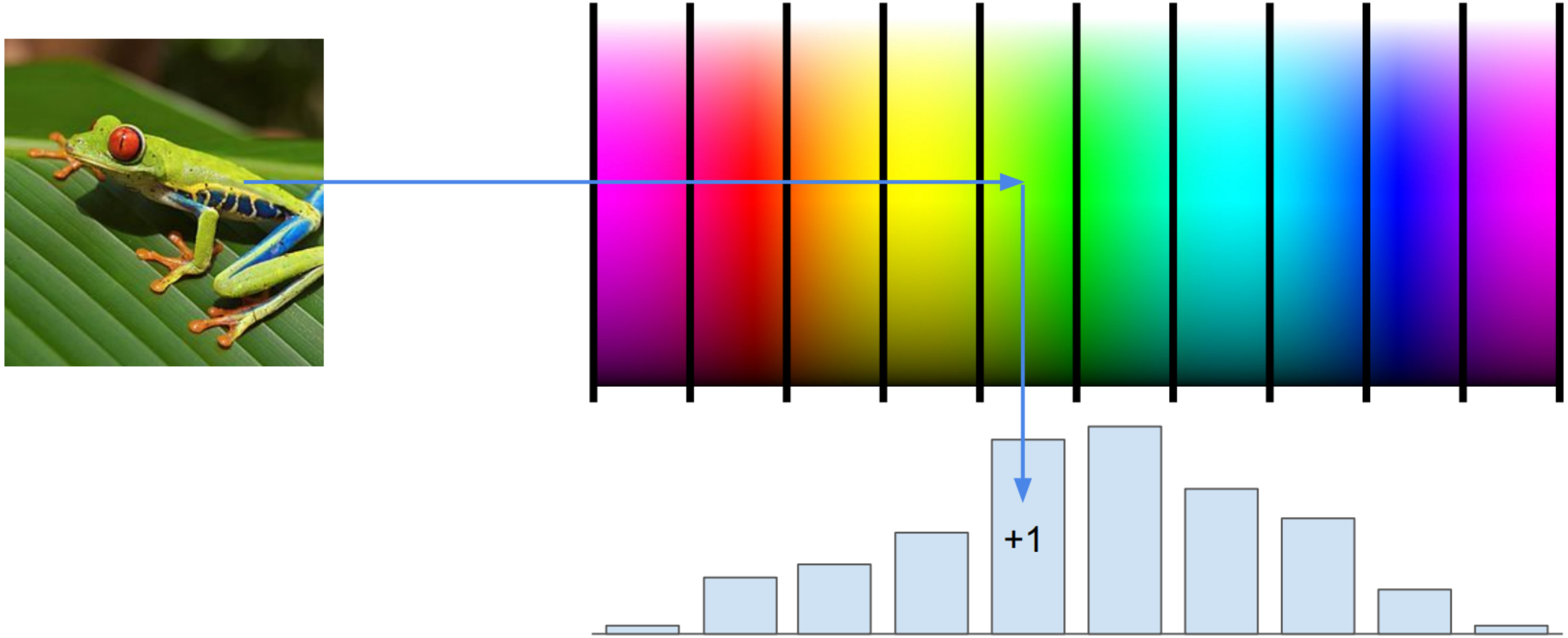


Cannot separate red and blue points with linear classifier

After applying feature transform, points can be separated by linear classifier

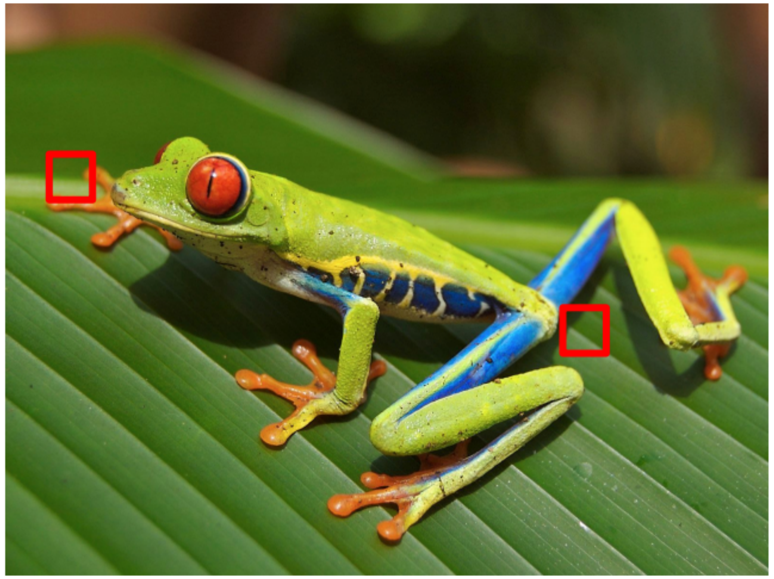
From Stanford CS231: http://cs231n.stanford.edu/

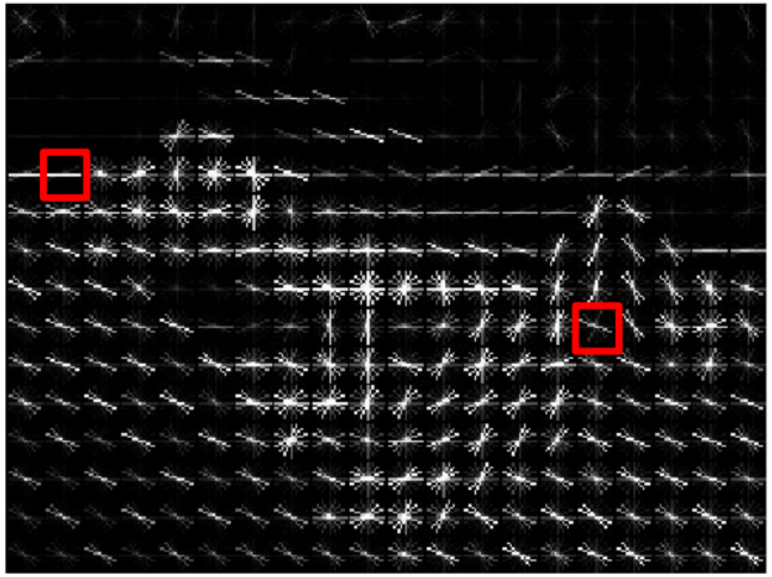
# Example: Color Histogram



From Stanford CS231: http://cs231n.stanford.edu/

# Example: Histogram of Oriented Gradients (HoG)





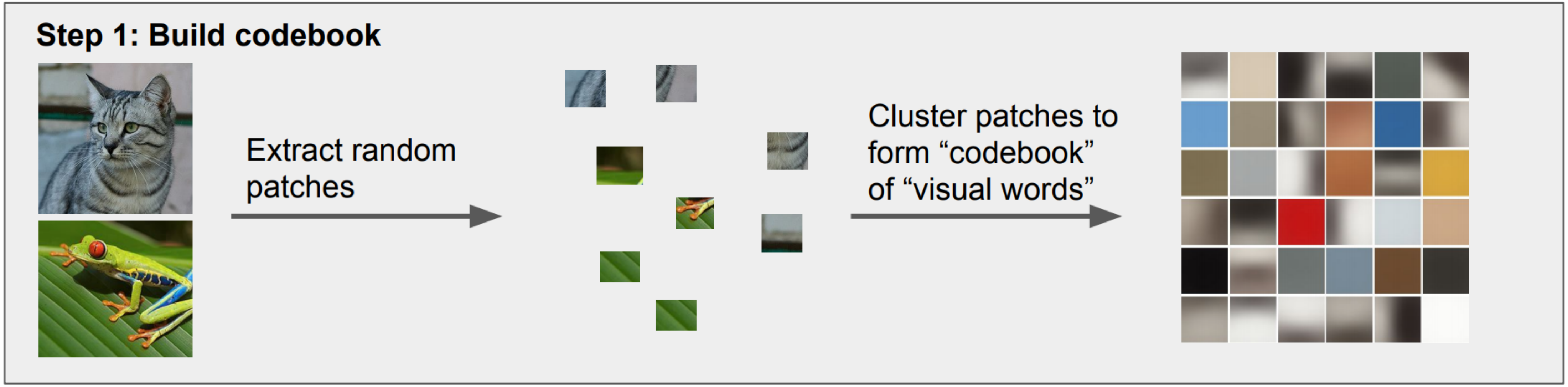
Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins

Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, “Object recognition from local scale-invariant features”, ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

From Stanford CS231: [http://cs231n.stanford.edu/](http://cs231n.stanford.edu/)

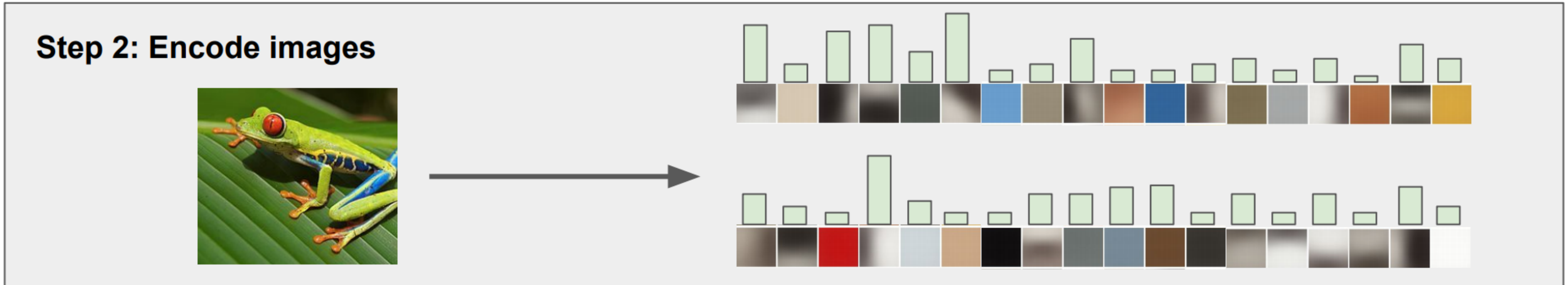
# Example: Bag of Words



**Step 1: Build codebook**

Extract random patches

Cluster patches to form “codebook” of “visual words”

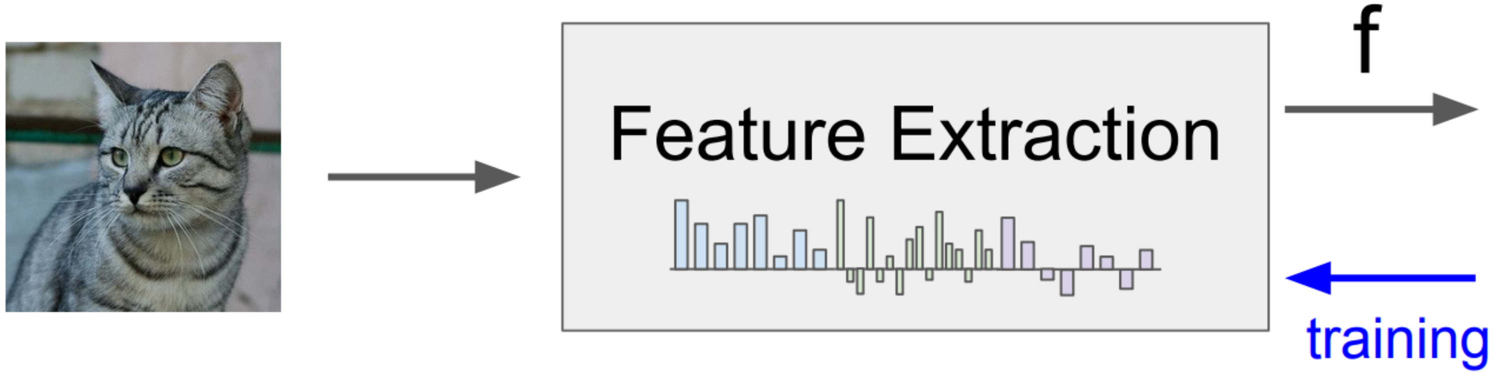


**Step 2: Encode images**

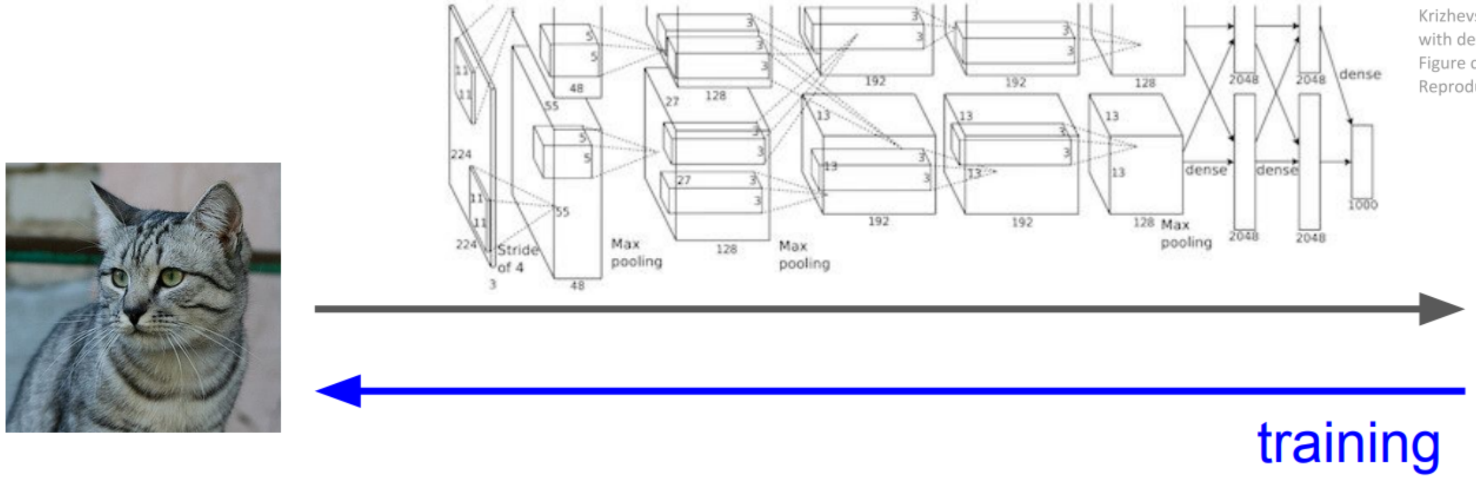
Fei-Fei and Perona, “A bayesian hierarchical model for learning natural scene categories”, CVPR 2005

From Stanford CS231: http://cs231n.stanford.edu/

# Image features vs ConvNets



**10** numbers giving scores for classes

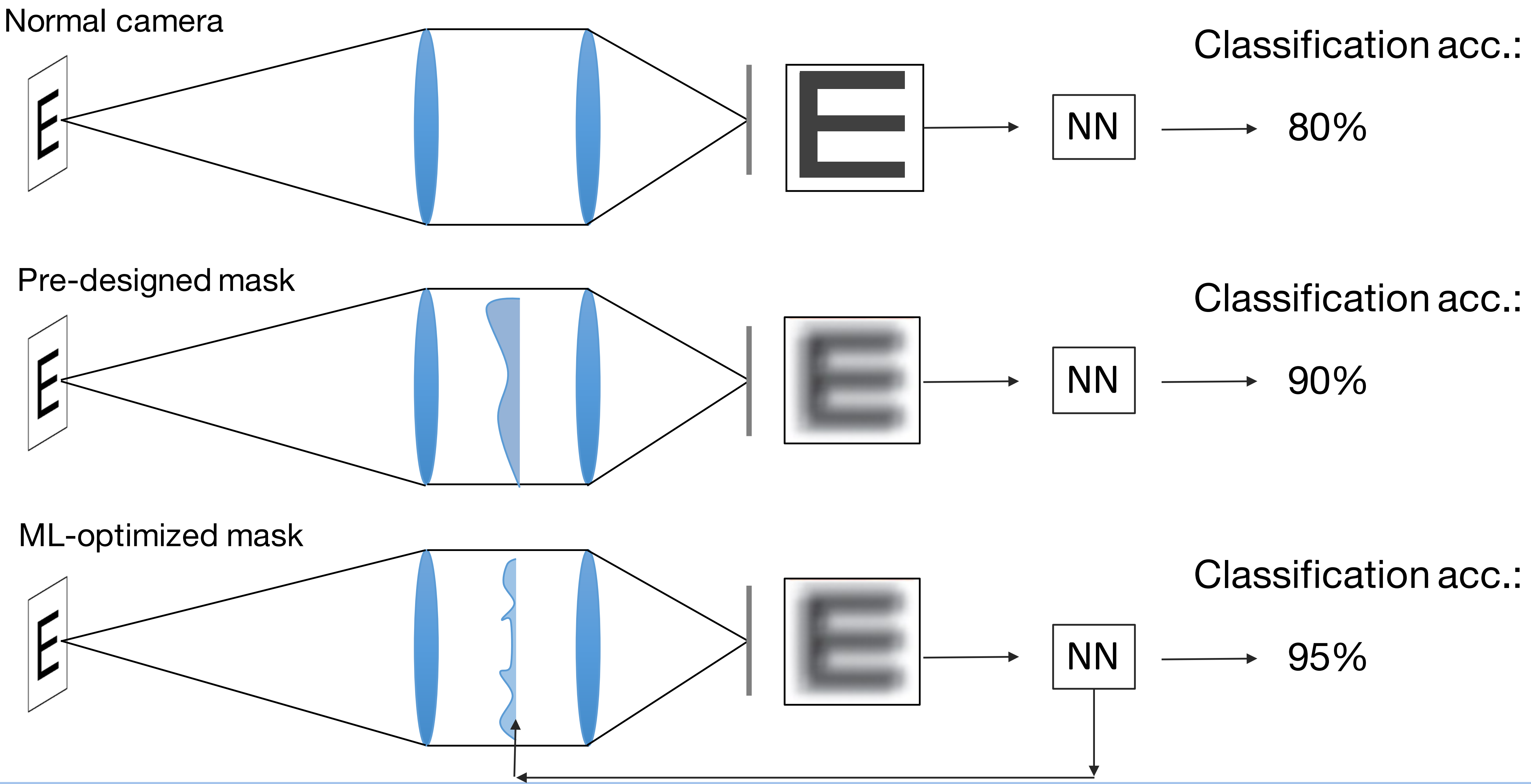


Krizhevsky, Sutskever, and Hinton, “Imagenet classification with deep convolutional neural networks”, NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

**10** numbers giving scores for classes

From Stanford CS231: http://cs231n.stanford.edu/

# Hand-crafted versus learned features also applies to imaging



## Statistical Machine Learning in 30 minutes

Two competing goals in machine learning:

1. Can we make sure the in-sample error $L_{in}(y, f(x,W))$ is small enough?
   - Appropriate cost function
   - “complex enough” model

# Statistical Machine Learning in 30 minutes

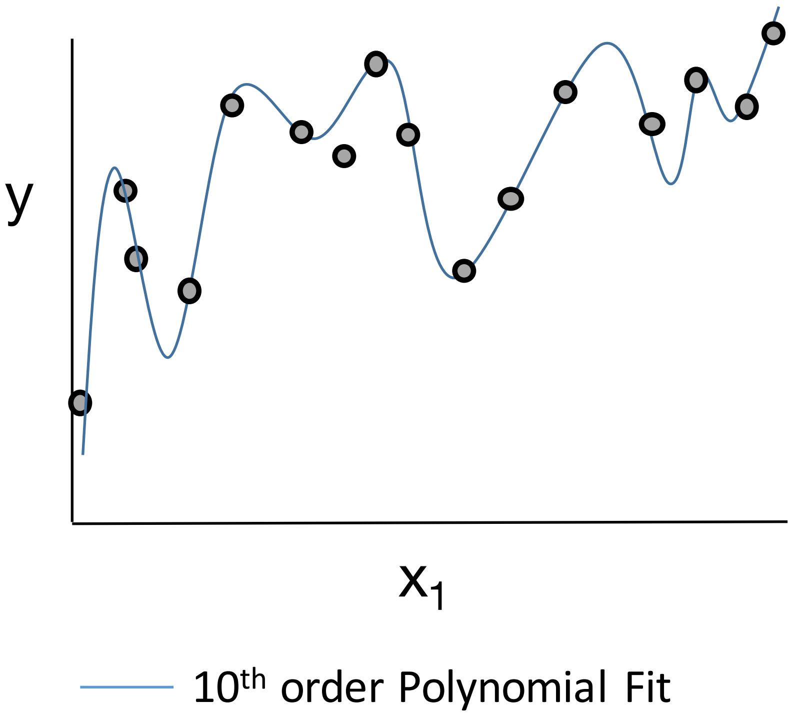
Two competing goals in machine learning:

1. Can we make sure the in-sample error $L_{in}(y, f(x,W))$ is small enough?
   - Appropriate cost function
   - “complex enough” model

2. Can we make sure that $L_{out}(y, f(x,W))$ is close enough to $L_{in}(y, f(x,W))$?
   - Probabilistic analysis says yes!
   - $|L_{in} - L_{out}|$ bounded from above
   - Bound grows with model capacity (bad)
   - Bound shrinks with # of training examples (good)

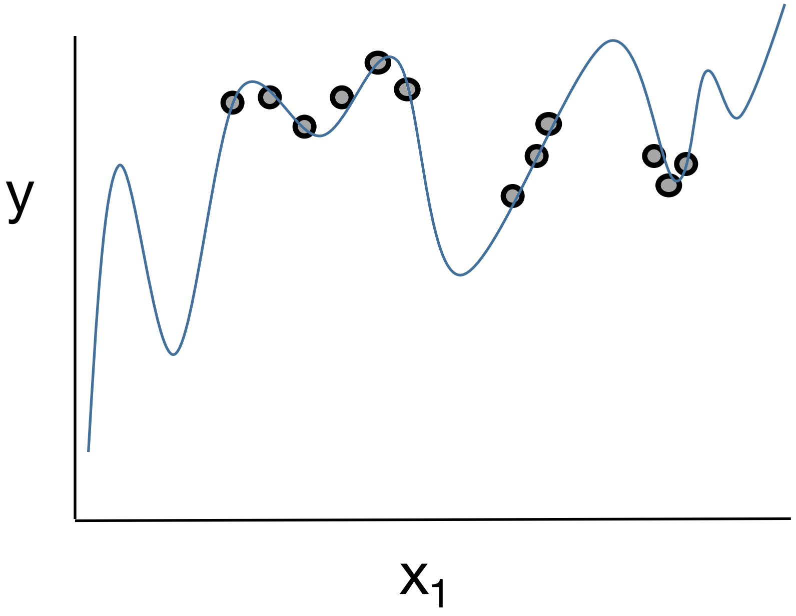
# Model overfitting versus underfitting – a thought exercise

Let's fit these "training" data points:



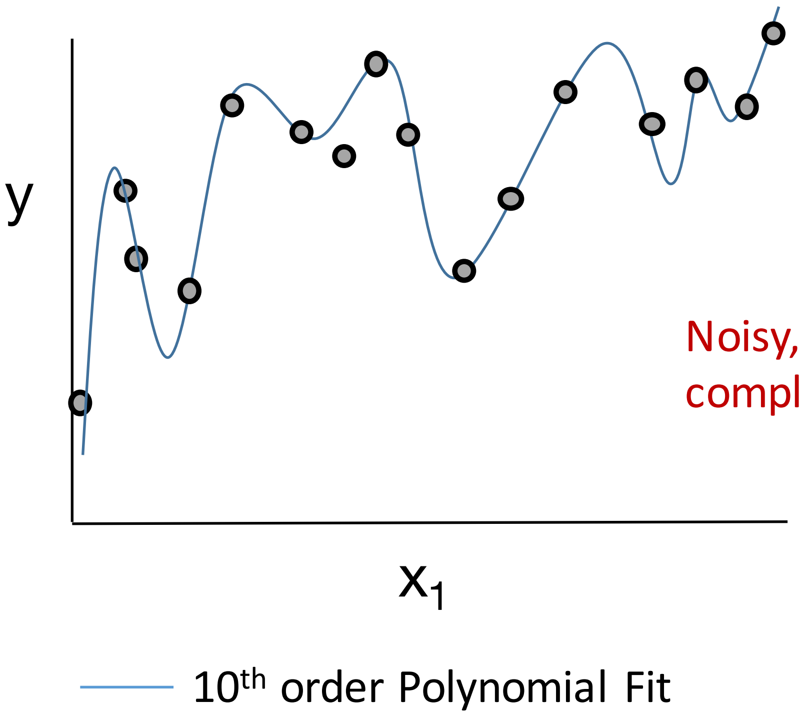
And then here's our testing dataset – good?

Perfect! But lucky….



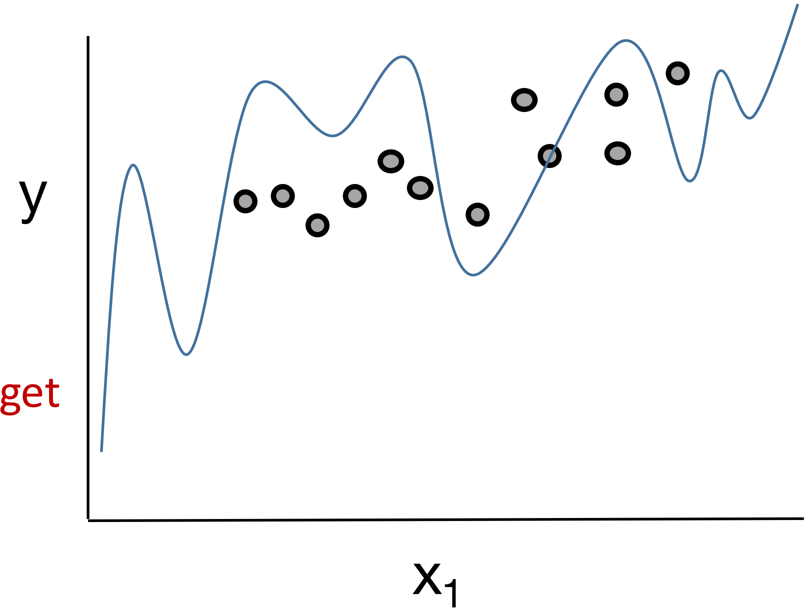
# Model overfitting versus underfitting – a thought exercise

Let’s fit these “training” data points:



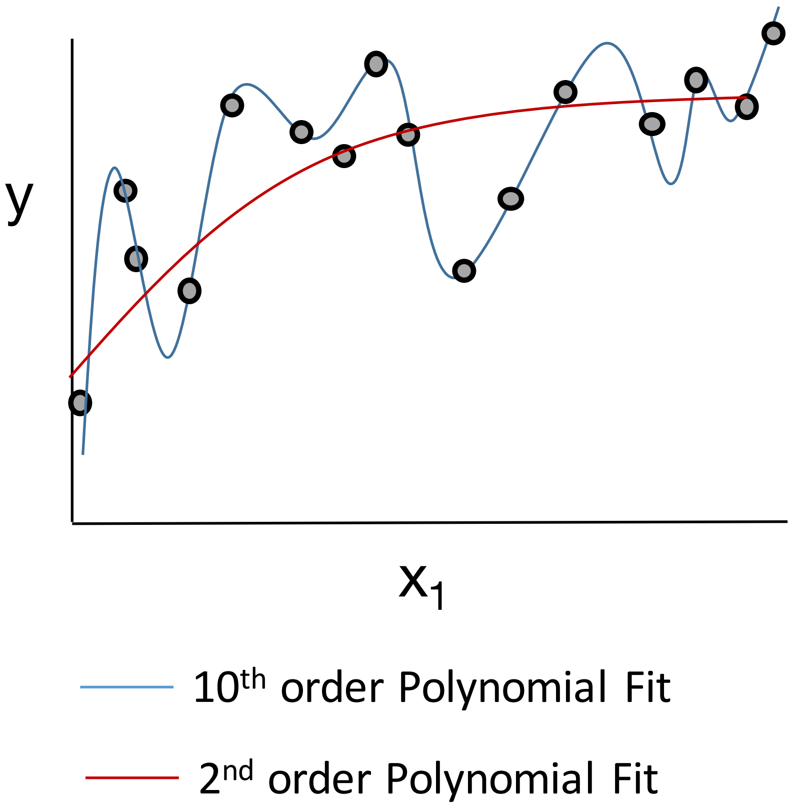
What if our test dataset was this :

Noisy, low complexity target

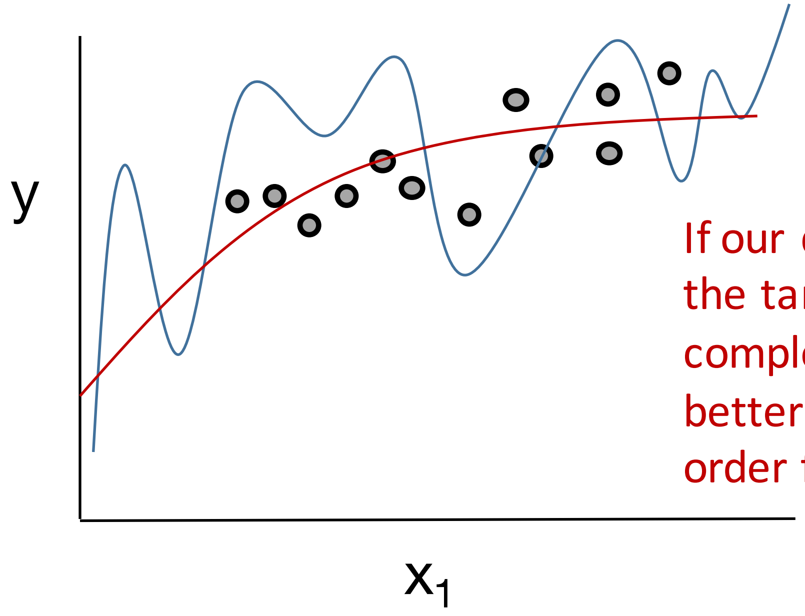


# Model overfitting versus underfitting – a thought exercise

Let's fit these "training" data points:

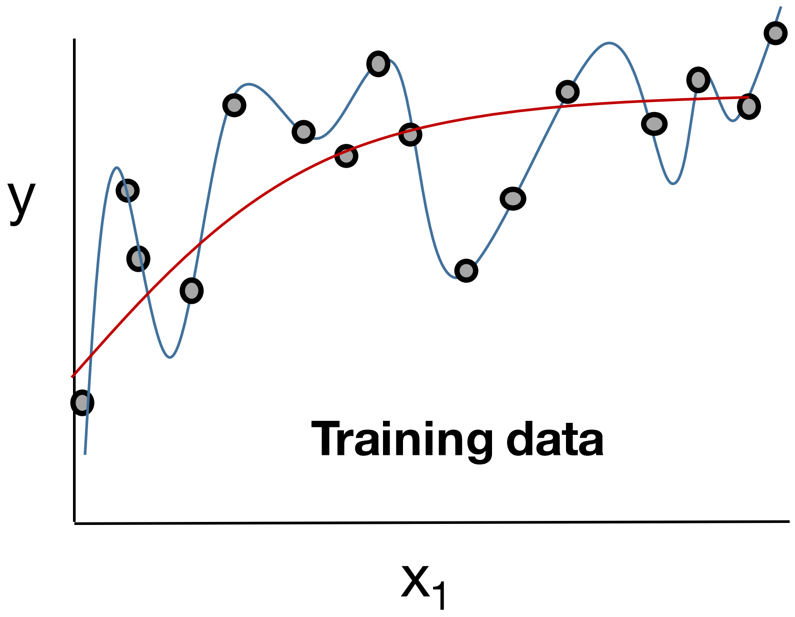


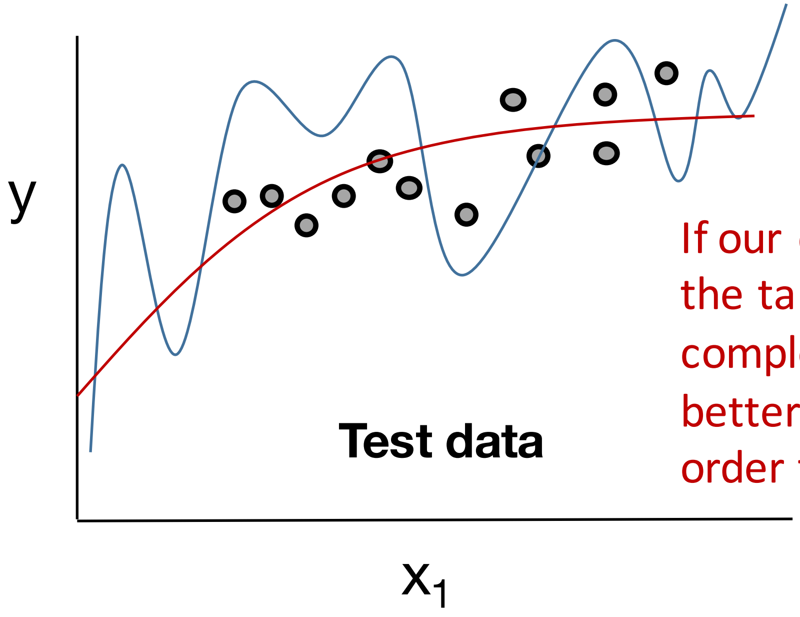
What if our test dataset was this :



If our data was noisy and the target followed a low-complexity model, we'd be better off with a second order fit!

# Model overfitting versus underfitting – a thought exercise



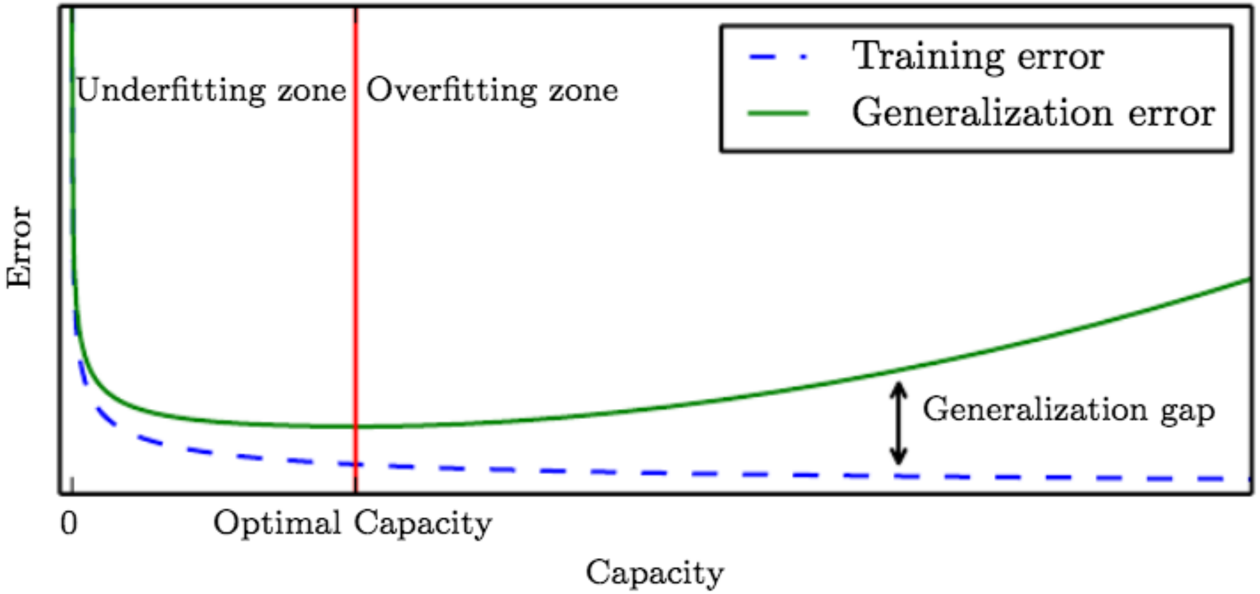


If our data was noisy and the target followed a low-complexity model, we'd be better off with a second order fit!

**Model capacity**: ability to fit a wide range of functions

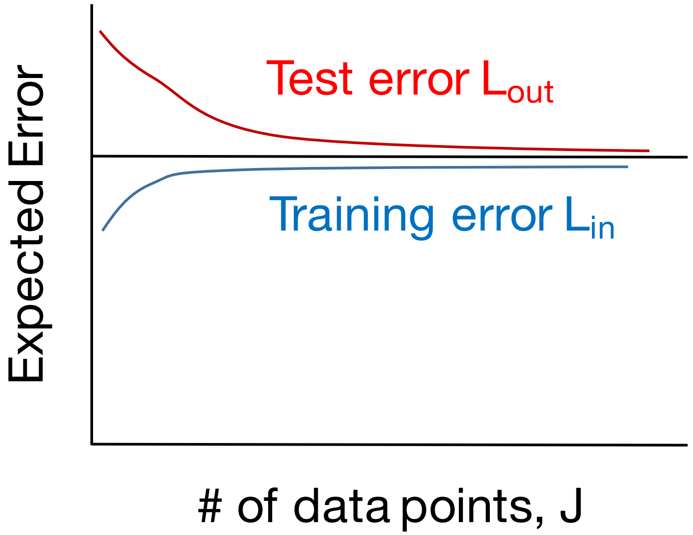
Control capacity through model's hypothesis space (set of functions model can take)

Hard to know ahead of time!

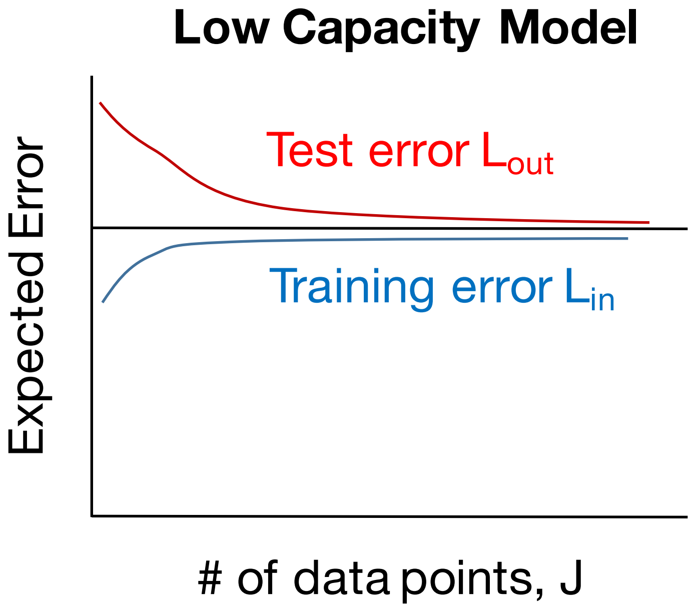


*Deep Learning*, I. Goodfellow et al., Fig. 5.3

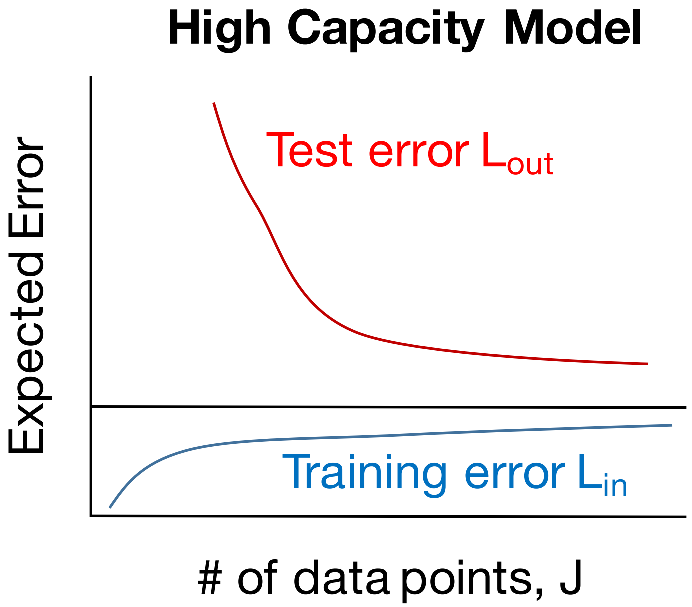
## Low Capacity Model



Relatively high error



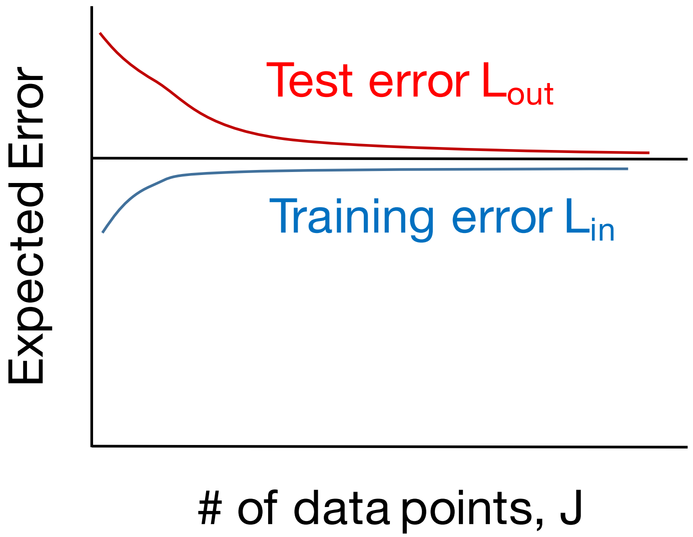
Relatively high error



But harder to get test error down…

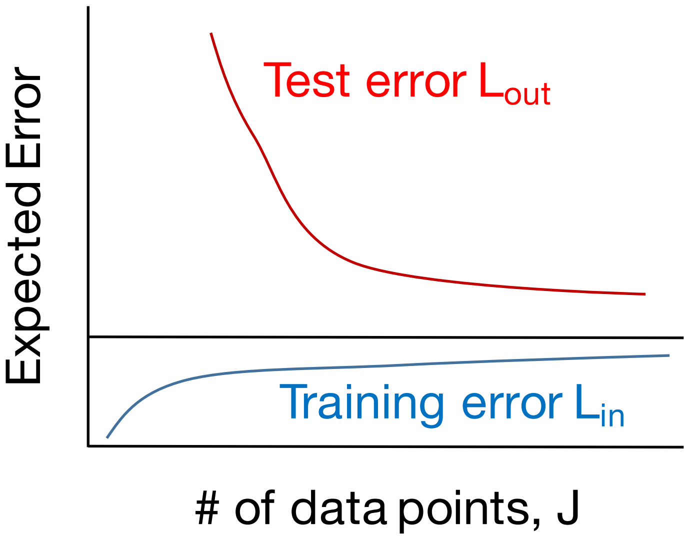
Relatively low error

### Low Capacity Model



Relatively high error

### High Capacity Model



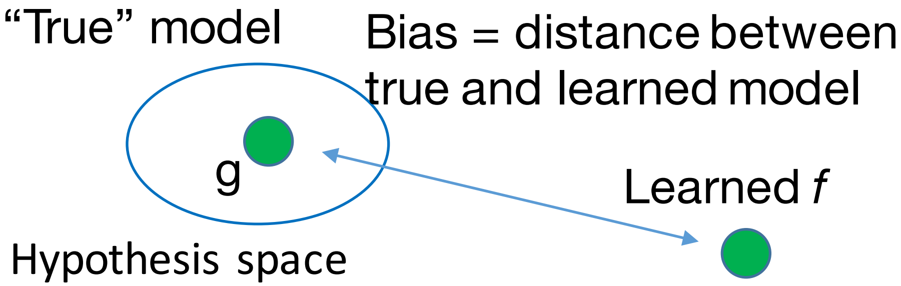
But harder to get test error down…

Relatively low error

**Take away concepts:**

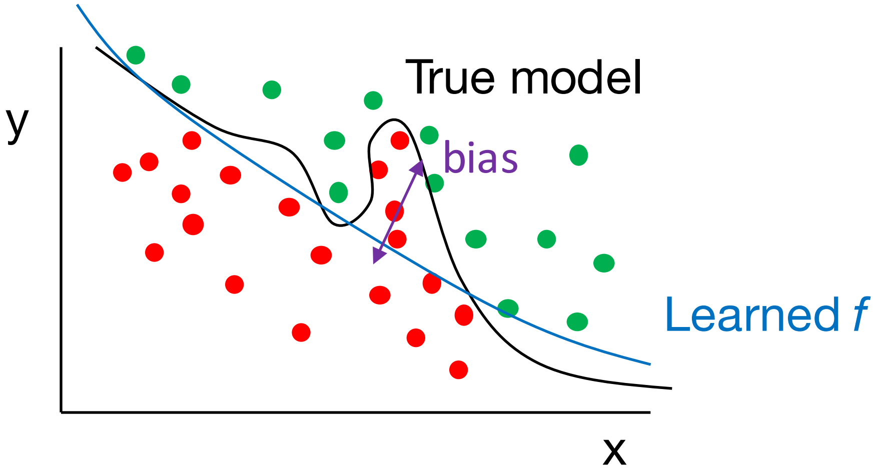
- Can’t ever really expect test error to be less than training error
- Complicated models tend to appear to “do better” during training, before trying test data
- When the model gets complicated and you don’t have enough data, challenging to get test error down

# Model bias versus variance

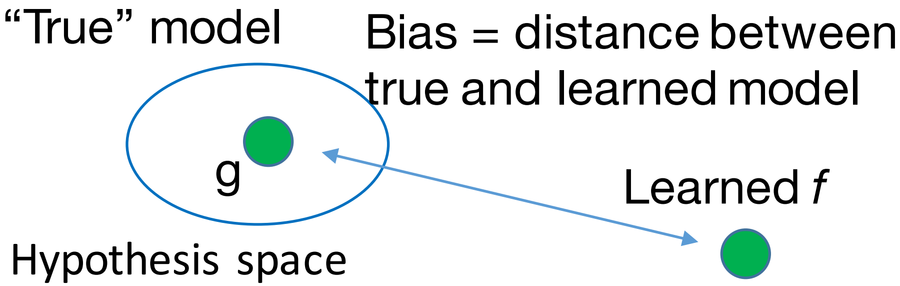


“True” model

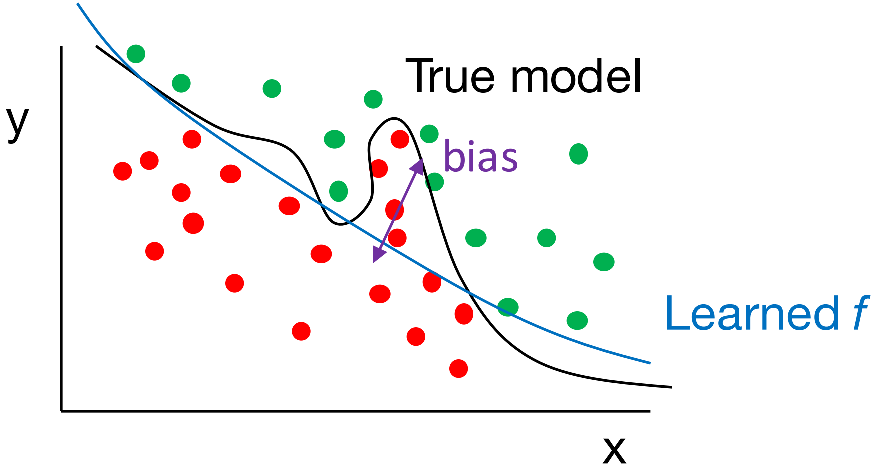
Bias = distance between true and learned model



# Model bias versus variance



Bias = distance between true and learned model

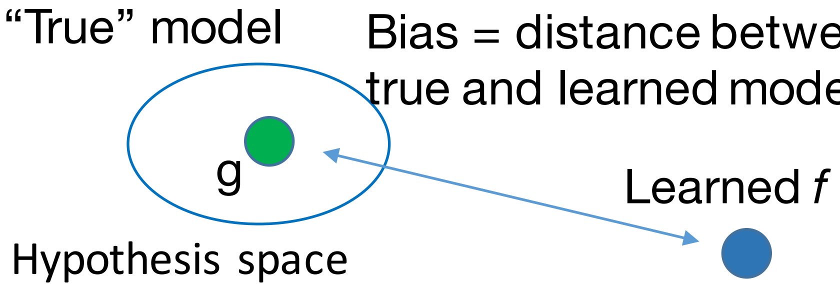


Models that tend to be "a bit too simple" are biased away from "true" model

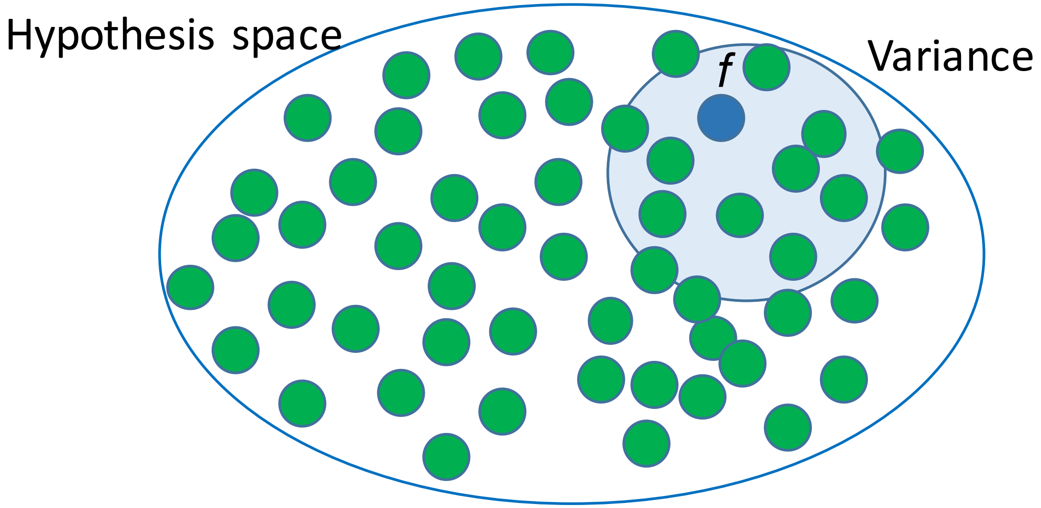
$$\text{Bias} = (g(\mathbf{x}) - f(\mathbf{x}))^2$$

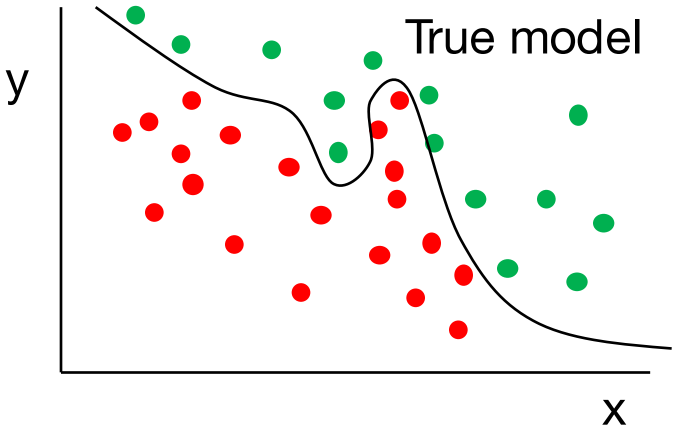
Measures how far our learning model f is biased away from target function g (for perfect training data classification)

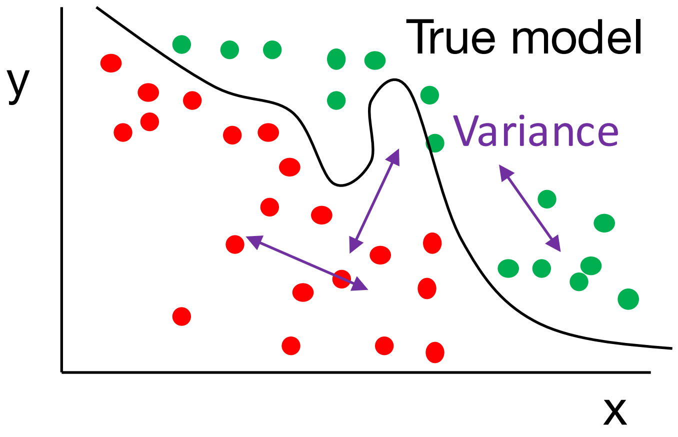
# Model bias versus variance



Bias = distance between true and learned model



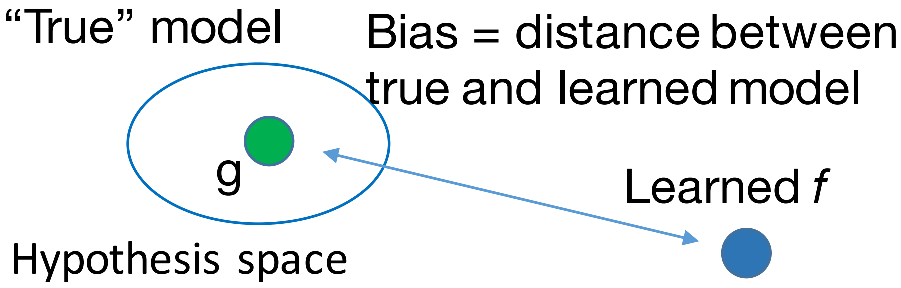




Variance = Var[g(**x**)]

More complicated datasets exhibit lots of variance between training and test set

# Model bias versus variance

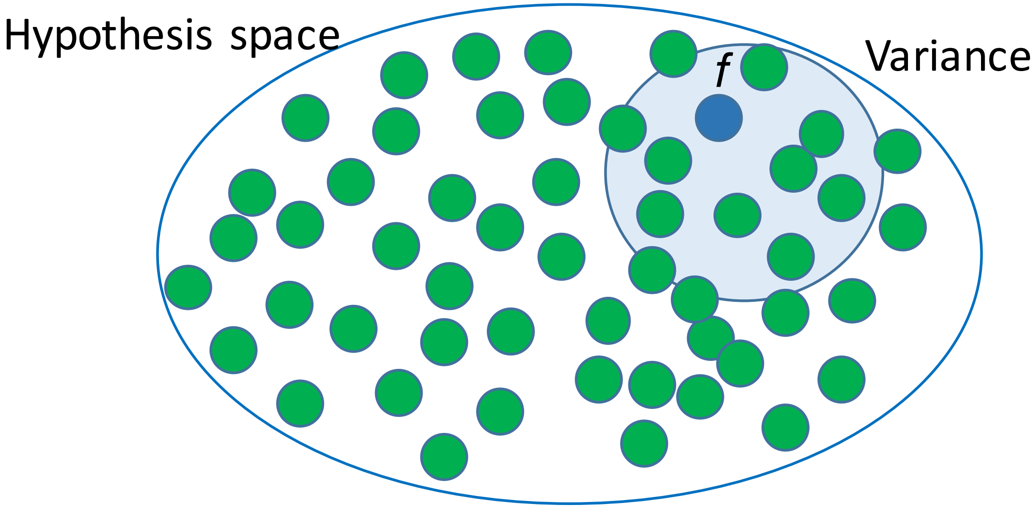


“True” model

Bias = distance between true and learned model

Learned *f*

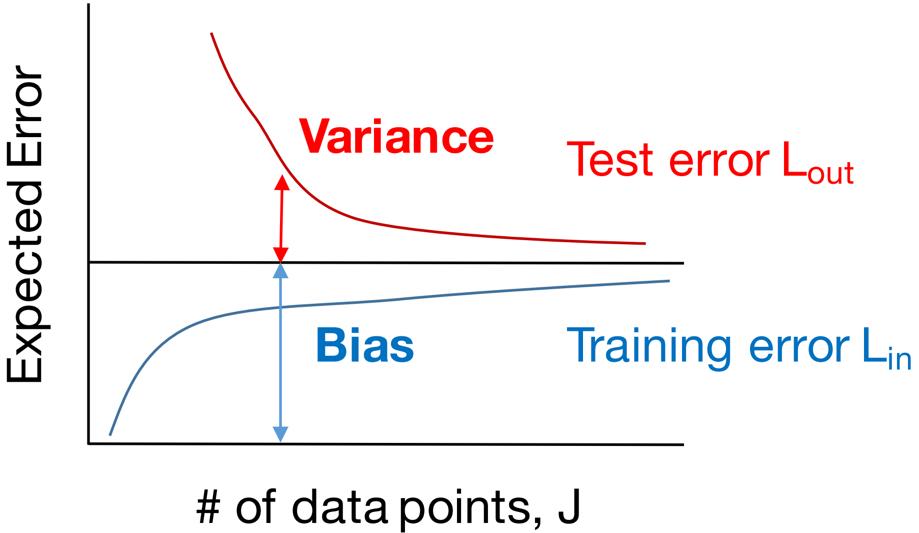
Hypothesis space



Hypothesis space

Variance

**Error vs. # data points**



Test Error is sum of model bias and variance!

Goal is to find a model *f* that balances between these two quantities for a given dataset

# How to formally define capacity and complexity?

- Short answer: it's complicated…
- Related to something called the *VC Dimension*
  - Can provide theoretical bounds on performance
  - Dimensional bounds rather than scalar bounds…
- I decided not to go into it, but please let me know if you'd like me to!

# Conclusions from statistical machine learning

- Conclusion: you want a model that is complex enough to capture variations within high-dimensional space, but not too complex such that it overfits the data

- Want a model with a high capacity, but can still *generalize* to data outside training set
  - More data -> less overfitting, complex target -> more overfitting

- For simple models, we can measure complexity via degrees of freedom, the VC bound and so-on to help us nail down ideal models that can generalize well

## Conclusions from statistical machine learning

- Conclusion: you want a model that is complex enough to capture variations within high-dimensional space, but not too complex such that it overfits the data

- Want a model with a high capacity, but can still *generalize* to data outside training set
  - More data -> less overfitting, complex target -> more overfitting

- For simple models, we can measure complexity via degrees of freedom, the VC bound and so-on to help us nail down ideal models that can generalize well

- **For DL models**: this will get too hard…here's a few counter-intuitive properties:

1. A fixed DL *architecture* exhibits data-dependent complexities
   - e.g., "good" DL networks achieve 0 training error on images with random labels, so cannot generalize at all in this case, and are too complex

2. DL networks with more hidden units leads to *better* generalization (the main finding of the last few years). So deeper models tend to be less complex, actually…

3. Complexity depends upon loss function and optimization method…

## Important to remember: “No Free Lunch Theorem”

- *“Averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.”*
- The most sophisticated DL algorithm has save average performance (averaged over all possible tasks) as the simplest.

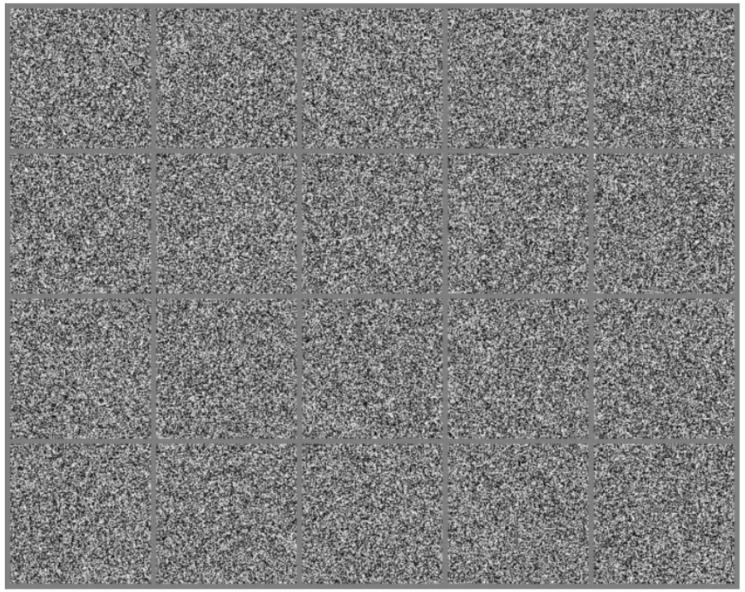
# Important to remember: “No Free Lunch Theorem”

- *“Averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.”*

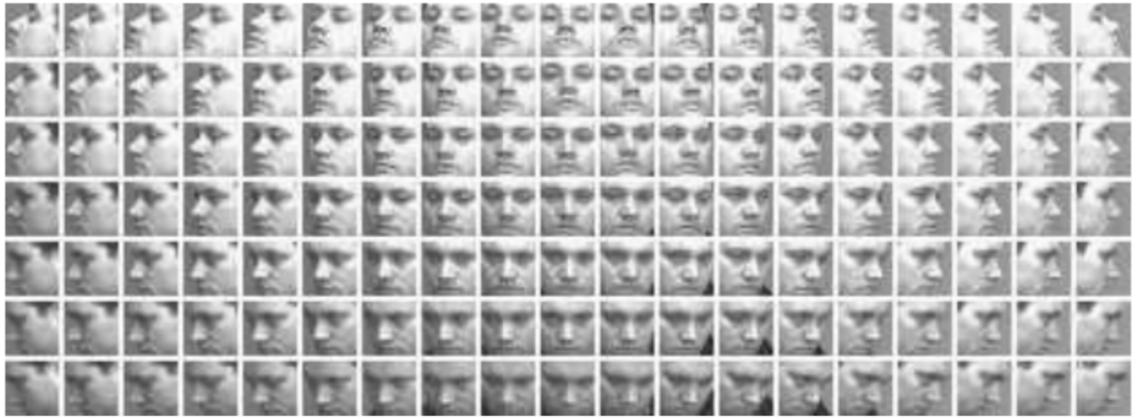
- The most sophisticated DL algorithm has save average performance (averaged over all possible tasks) as the simplest.

- Must make assumptions about probability distributions of inputs we’ll encounter in real-world

Set of 20 “images”, random Gaussian distribution



Face at different orientations = manifold n-D space

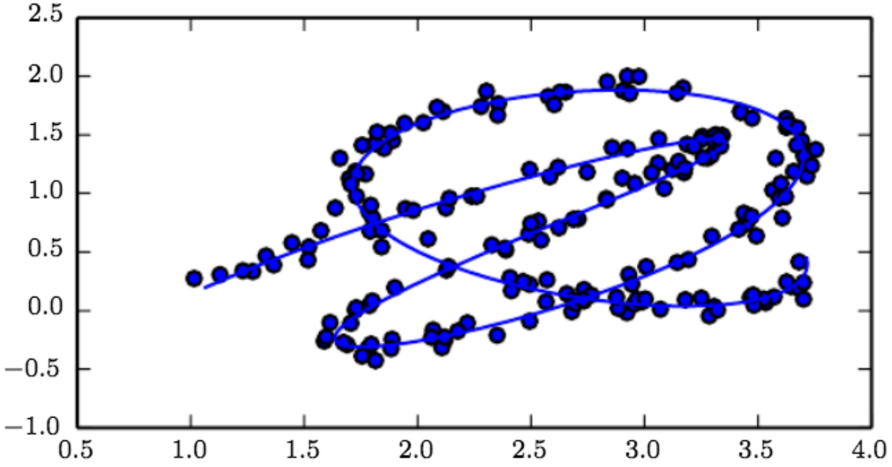


*Deep Learning*, I. Goodfellow et al., Fig. 5.12-13

# Important to remember: “No Free Lunch Theorem”

- *“Averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.”*
- The most sophisticated DL algorithm has save average performance (averaged over all possible tasks) as the simplest.
- Must make assumptions about probability distributions of inputs we’ll encounter in real-world

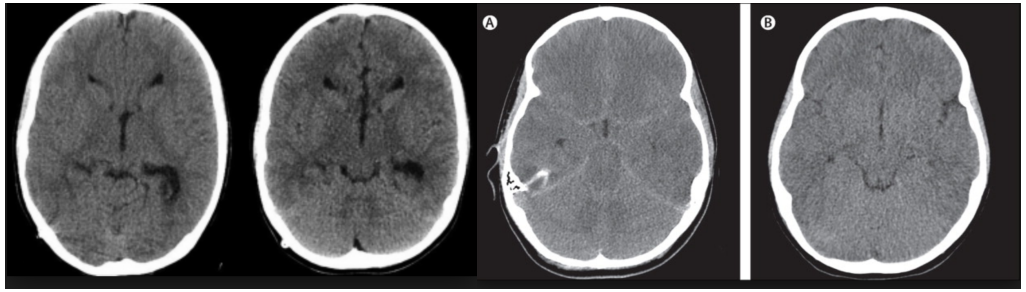
1D Manifold in 2D space



*Deep Learning*, I. Goodfellow et al., Fig. 5.11

**Manifold Hypothesis**

CT reconstructions of every brain in the world = kD manfold in nD space?



…