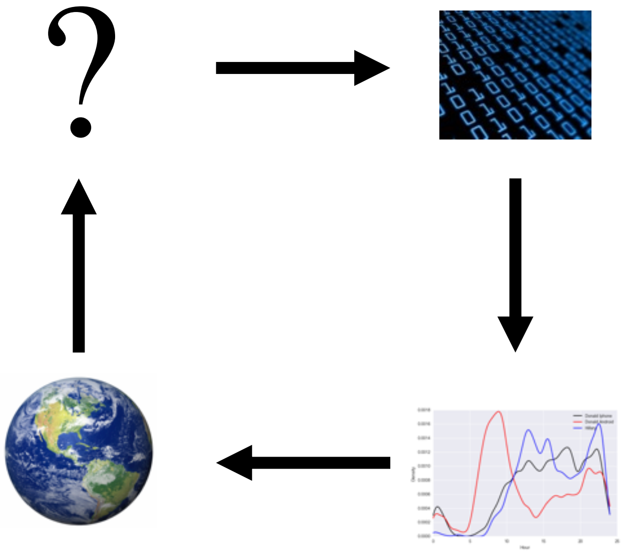
Classification & Logistic Regression

& maybe deep learning

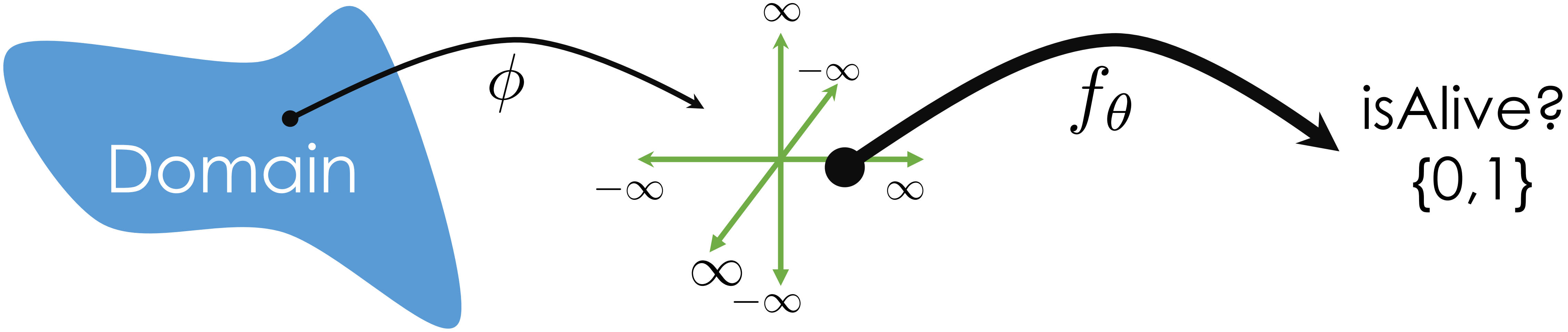
Slides by:

Joseph E. Gonzalez jegonzal@cs.berkeley.edu

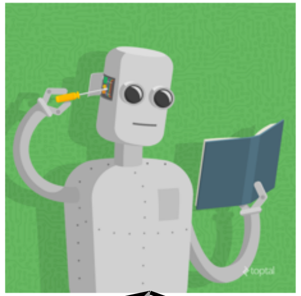


Previously…

Classification



Taxonomy of Machine Learning

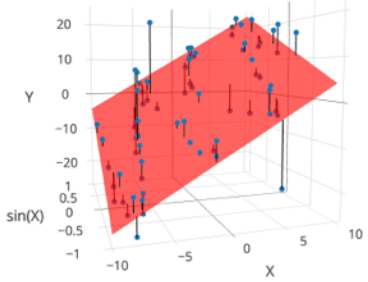


Labeled Data

Supervised Learning

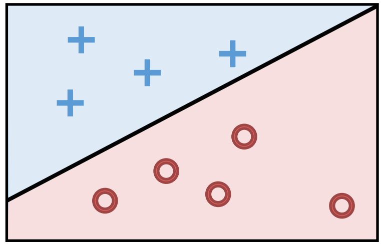
Quantitative Response

Regression



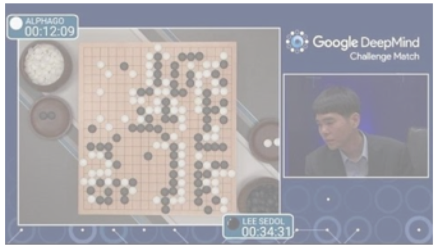
Categorical Response

Classification



Reward

Reinforcement Learning (not covered)

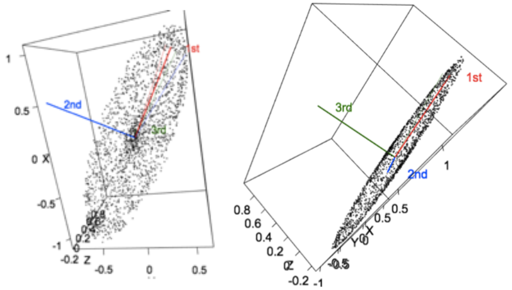


Alpha Go

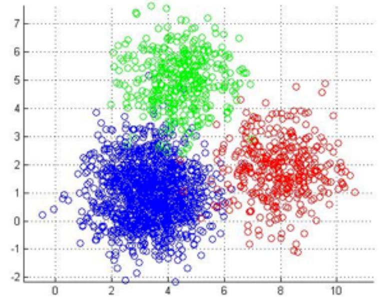
Unlabeled Data

Unsupervised Learning

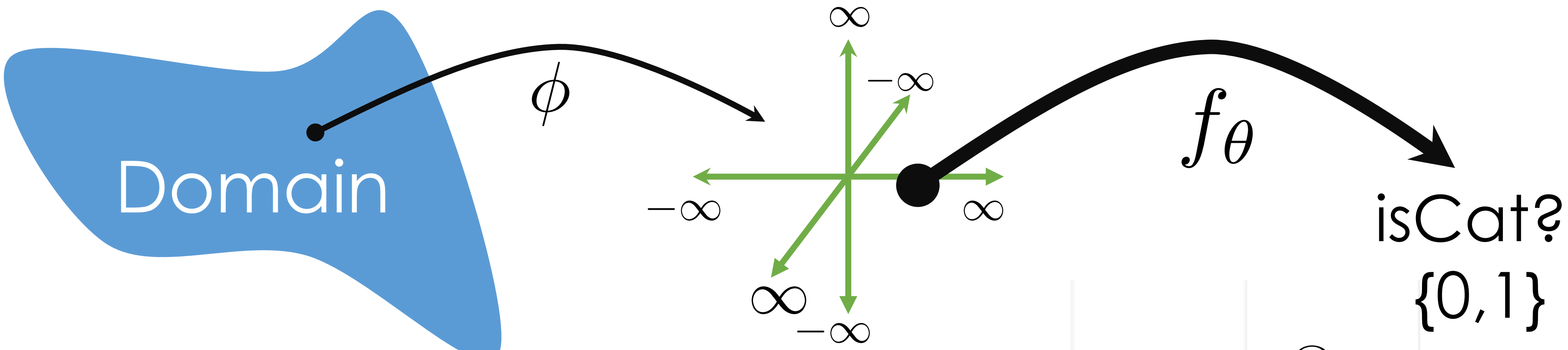
Dimensionality Reduction

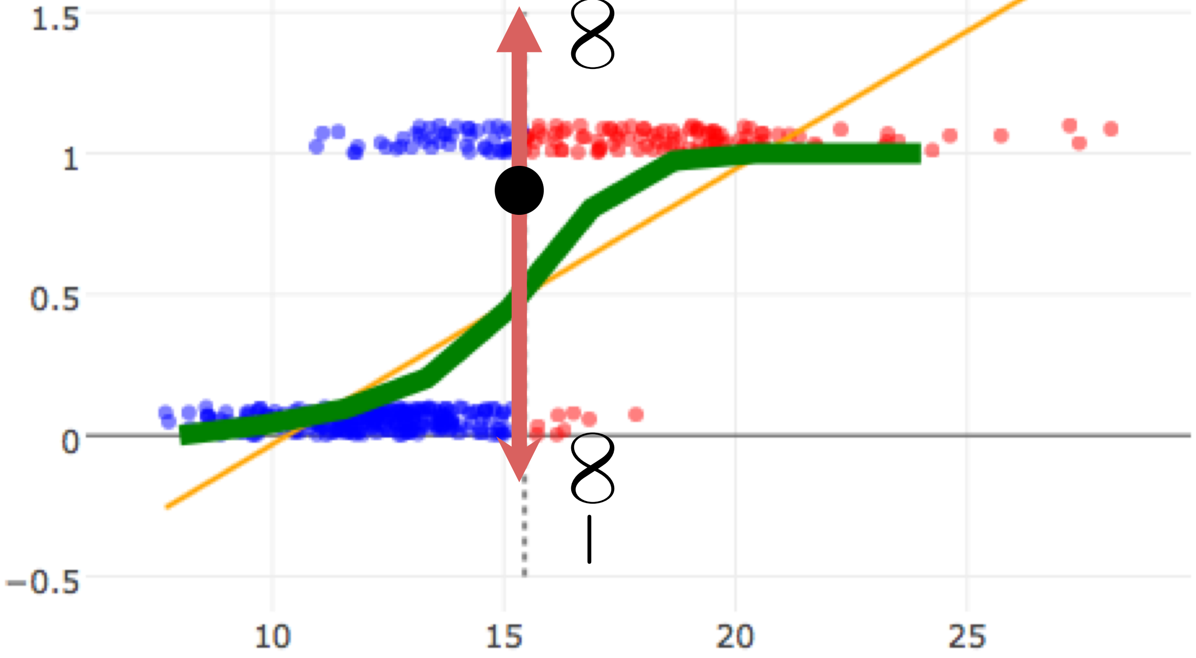


Clustering



Classification





Can we just use least squares?

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f_\theta(x_i)\right)^2 + \lambda \mathbf{R}(\theta)$$

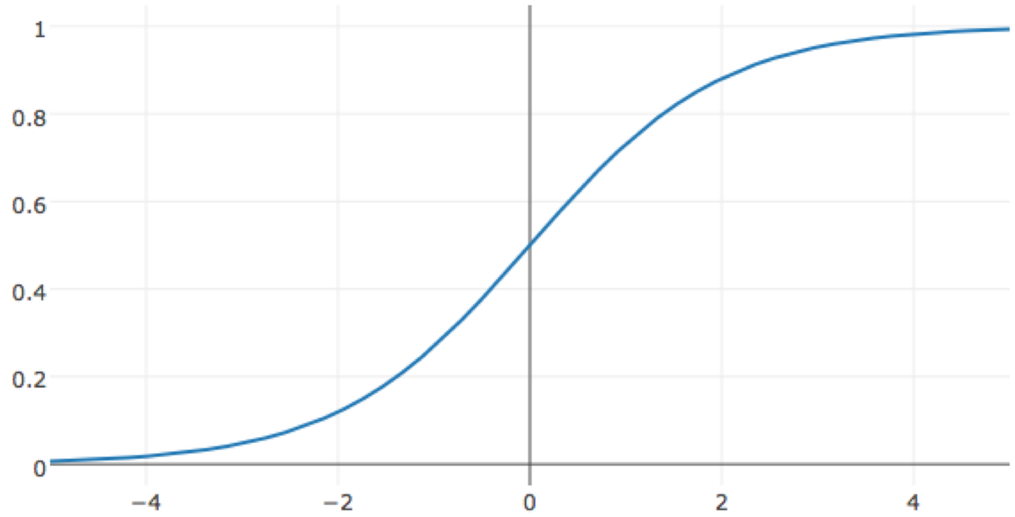
Defining a New Model for Classification

Logistic Regression

- Model the probability of a particular label:

$$\hat{\mathbf{P}}_\theta\left(y = 1 \,|\, x\right) = \sigma\Big(\underbrace{\phi(x)^T\theta}_{\textbf{Linear Model}}\Big) = \frac{1}{1 + \exp\left(-\phi(x)^T\theta\right)}$$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Motivation for the Logistic Model

- Model the "log odds" as a linear model

$$\underbrace{\phi(x_i)^T\theta}_{\text{Linear Model}} = \underbrace{\log\left(\frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{\hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)}\right)}_{\text{Log odds}}$$

$$\phi(x_i)^T\theta = 0 \quad \overset{\exp(0) = 1}{\Rightarrow} \quad \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) = \hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)$$

$$\phi(x_i)^T\theta > 0 \quad \overset{\exp(\epsilon) > 1}{\Rightarrow} \quad \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) > \hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)$$

$$\phi(x_i)^T\theta < 0 \quad \overset{\exp(-\epsilon) < 1}{\Rightarrow} \quad \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) < \hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)$$

for any positive ϵ

Motivation for the Logistic Model

$$\phi(x_i)^T\theta = \log\left(\frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{\hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)}\right)$$

$$= \log\left(\frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{1 - \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}\right)$$

Taking the exponent of both sides

$$\exp\left(\phi(x_i)^T\theta\right) = \frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{1 - \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}$$

$$\exp\left(\phi(x_i)^T\theta\right) = \frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{1 - \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}$$

Algebra

$$\exp\left(\phi(x_i)^T\theta\right)\left(1 - \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)\right) = \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)$$

Expanding terms

$$\exp\left(\phi(x_i)^T\theta\right) - \exp\left(\phi(x_i)^T\theta\right)\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) = \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)$$

Collect terms on the other side …

$$\exp\left(\phi(x_i)^T\theta\right) = \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)\left(1 + \exp\left(\phi(x_i)^T\theta\right)\right)$$

Solving for P(y=1 | x)

$$\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) = \frac{\exp\left(\phi(x_i)^T\theta\right)}{1 + \exp\left(\phi(x_i)^T\theta\right)}$$

Solving for P(y=1 | x)

$$\hat{\mathbf{P}}_{\theta}\left(y_i = 1 \,|\, x_i\right) = \frac{\exp\left(\phi(x_i)^T \theta\right)}{1 + \exp\left(\phi(x_i)^T \theta\right)}$$

Dividing numerator and denominator by exp(ϕ(x$_i$)^{T}θ)

$$\hat{\mathbf{P}}_{\theta}\left(y_i = 1 \,|\, x_i\right) = \frac{1}{1 + \exp\left(-\phi(x_i)^T \theta\right)}$$

$$= \sigma(\phi(x)^T \theta)$$

Where

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

The Logistic Regression Model

Model: $$\hat{\mathbf{P}}_\theta\left(y = 1 \,|\, x\right) \; = \sigma(\phi(x)^T\theta) \; = \frac{1}{1 + \exp\left(-\phi(x)^T\theta\right)}$$

How do we fit the model to the data?

Defining the Loss

Could we use the Squared Loss

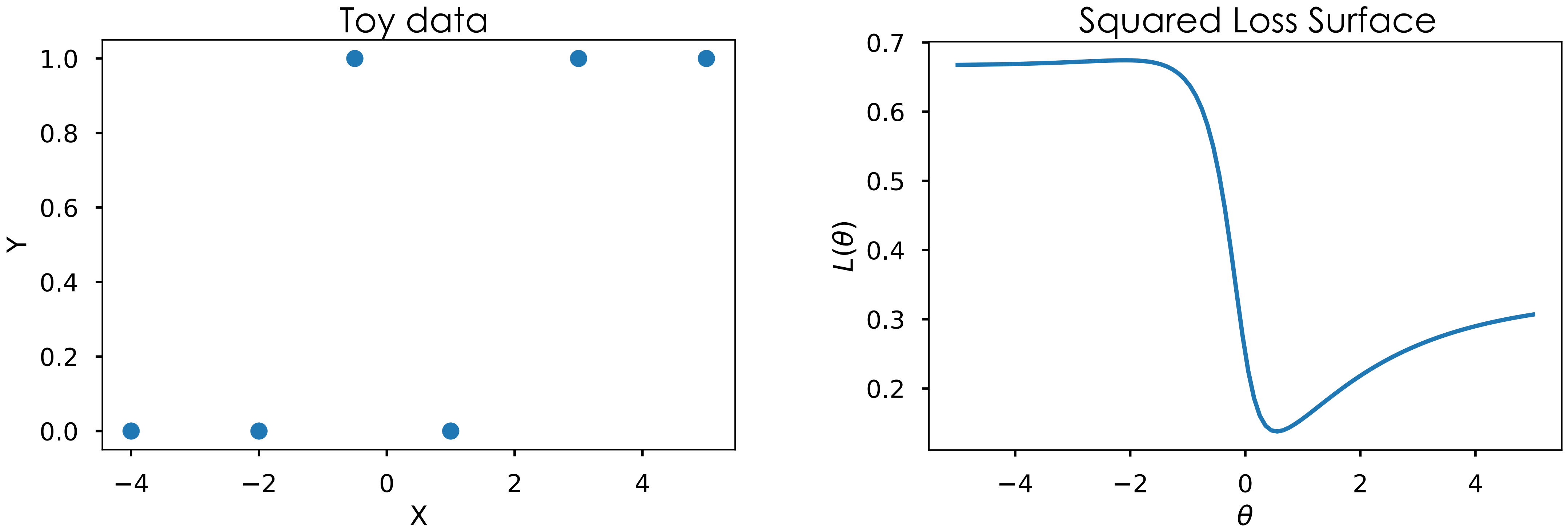
- What about squared loss and the new model:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sigma\left(\phi(x_i)^T \theta\right)\right)^2$$

  - Tries to match probability with 0/1 labels.
  - Occasionally used in some neural network applications
  - **Non-convex!**

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)^2$$

- Tries to match probability with 0/1 labels.
- Occasionally used in some neural network applications
- **Non-convex!**



Defining the Cross Entropy Loss

Loss Function

- We want our model to be close to the data:

$$\hat{\mathbf{P}}_\theta\left(y = 1 \,|\, x\right) \approx \mathbf{P}\left(y = 1 \,|\, x\right)$$

- Example: (cute or not)?

x =

y = 1 "cute"

	Cute	Not Cute
Observed Probability	$\mathbf{P}\left(y = 1 \,\|\, x\right)$ = 1.0	$\mathbf{P}\left(y = 0 \,\|\, x\right)$ = 0.0
Predicted Probability	$\hat{\mathbf{P}}_\theta\left(y = 1 \,\|\, x\right)$ = 0.8	$\hat{\mathbf{P}}_\theta\left(y = 0 \,\|\, x\right)$ = 0.2

Loss Function

- We want our model to be close to the data:

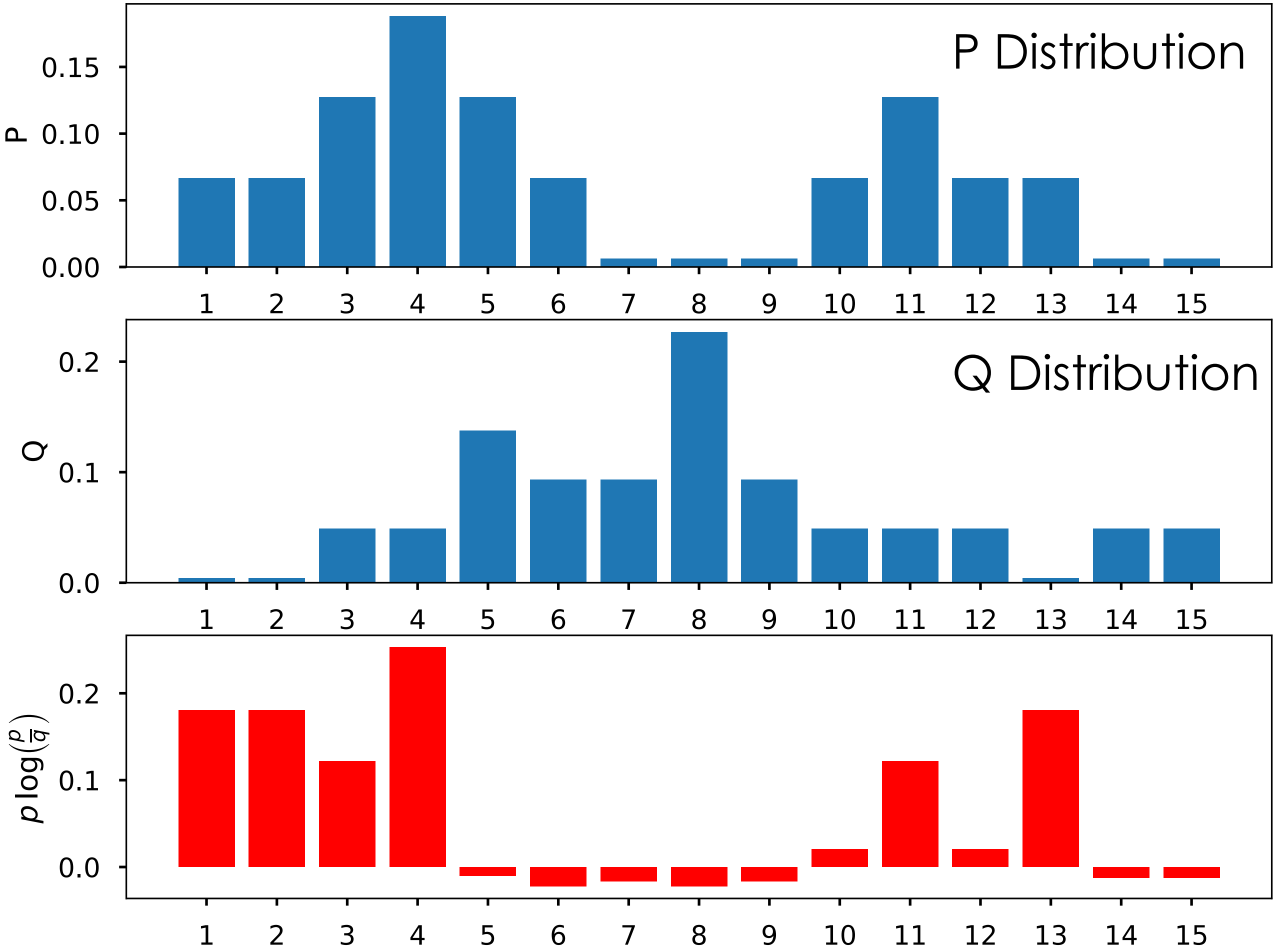
$$\hat{\mathbf{P}}_{\theta}\left(y = 1 \,|\, x\right) \;\approx\; \mathbf{P}\left(y = 1 \,|\, x\right)$$

- Kullback–Leibler (KL) Divergence provides a measure of difference between two distributions:
  - Difference between two discrete distributions P and Q

$$\mathbf{D}(P||Q) = \sum_{k=0}^{K-1} P(k) \log\left(\frac{P(k)}{Q(k)}\right)$$

Kullback–Leibler (KL) Divergence

$$\mathbf{D}(P||Q) = \sum_{k=0}^{K-1} P(k) \log \left(\frac{P(k)}{Q(k)} \right)$$



- ➢ The average log difference between P and Q weighted by P
- ➢ Does not penalize mismatch for rare events with respect to P
- ➢ Note that it is not symmetric

$$\mathbf{D}(P||Q) \neq \mathbf{D}(Q||P)$$

Loss Function

- We want our model to be close to the data:

$$\hat{\mathbf{P}}_\theta\left(y = 1 \,|\, x\right) \;\approx\; \mathbf{P}\left(y = 1 \,|\, x\right)$$

- Kullback–Leibler (KL) divergence for classification
  - For a **single** (x,y) data point

Binary Classification

$$\mathbf{D}_{KL}\left(\mathbf{P} || \hat{\mathbf{P}}_\theta\right) = \sum_{k=0}^{\overset{=2}{K}-1} \mathbf{P}\left(y = k \,|\, x\right) \log\left(\frac{\mathbf{P}\left(y = k \,|\, x\right)}{\hat{\mathbf{P}}_\theta\left(y = k \,|\, x\right)}\right)$$

  - Average KL Divergence for all the data:

- Kullback–Leibler (KL) divergence for classification
  - For a **single** (x,y) data point

$=2$ Binary Classification

$$\mathbf{D}_{KL}\left(\mathbf{P}||\hat{\mathbf{P}}_\theta\right) = \sum_{k=0}^{K-1} \mathbf{P}\left(y = k \,|\, x\right) \log\left(\frac{\mathbf{P}\left(y = k \,|\, x\right)}{\hat{\mathbf{P}}_\theta\left(y = k \,|\, x\right)}\right)$$

  - Average KL Divergence for all the data:

$$\arg\min_\theta \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{K-1} \underbrace{\mathbf{P}\left(y_i = k \,|\, x_i\right) \log\left(\frac{\mathbf{P}\left(y_i = k \,|\, x_i\right)}{\hat{\mathbf{P}}_\theta\left(y_i = k \,|\, x_i\right)}\right)}_{\text{Log(a/b) = Log(a) – Log(b)}}$$

Doesn't depend on θ

$$\cancel{\mathbf{P}\left(y_i = k \,|\, x_i\right) \log\left(\mathbf{P}\left(y_i = k \,|\, x_i\right)\right)}$$

$$- \mathbf{P}\left(y_i = k \,|\, x_i\right) \log\left(\hat{\mathbf{P}}_\theta\left(y_i = k \,|\, x_i\right)\right)$$

- Average **cross entropy loss**

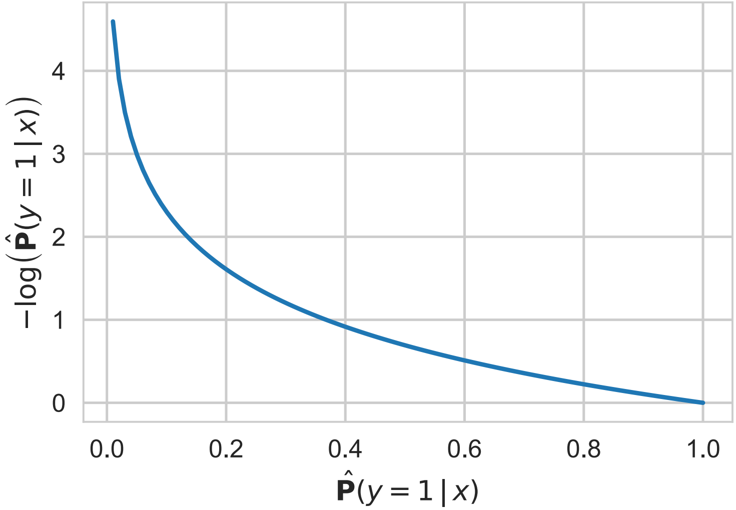
$$\arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \overbrace{\sum_{k=0}^{K-1} - \mathbf{P}\left(y_i = k \mid x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = k \mid x_i\right)\right)}$$

➢ Average **cross entropy loss**

$$\arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \overbrace{\sum_{k=0}^{K-1} - \mathbf{P}\left(y_i = k \mid x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = k \mid x_i\right)\right)}$$

Cute Cat Example

x =  , y = 1 "cute"



	Cute	Not Cute
Observed Probability	$\mathbf{P}(y = 1 \mid x)$ = 1.0	$\mathbf{P}(y = 0 \mid x)$ = 0.0
Predicted Probability	$\hat{\mathbf{P}}_{\theta}(y = 1 \mid x)$ = 0.8	$\hat{\mathbf{P}}_{\theta}(y = 0 \mid x)$ = 0.2
Cross Ent. $-\mathbf{P}\log\hat{\mathbf{P}}_{\theta}$	-1.0 Log(0.8) ≈ 0.22	-0.0 Log(0.2) = 0.0

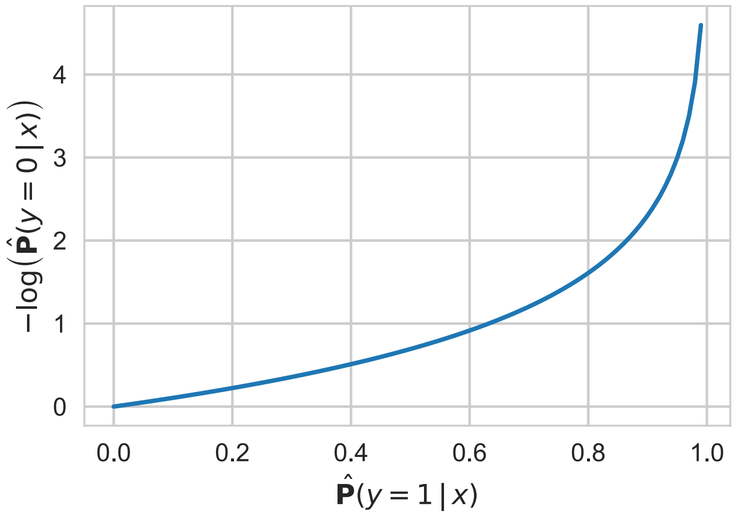
Also called the log loss because it is the log of the predicted probability for the true class

➢ Average **cross entropy loss**

$$\arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \overbrace{\sum_{k=0}^{K-1} - \mathbf{P}\left(y_i = k \mid x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = k \mid x_i\right)\right)}$$

Cute Cat Example

x =  , y = 0 “not cute"



	Cute	Not Cute
Observed Probability	$\mathbf{P}(y=1 \mid x)$ = 0.0	$\mathbf{P}(y=0 \mid x)$ = 1.0
Predicted Probability	$\hat{\mathbf{P}}_{\theta}(y=1 \mid x)$ = 0.7	$\hat{\mathbf{P}}_{\theta}(y=0 \mid x)$ = 0.3
Cross Ent. $-\mathbf{P}\log\hat{\mathbf{P}}_{\theta}$	-0.0 Log(0.7) = 0.0	-1.0 Log(0.3) ≈ 1.20

Also called the log loss because it is the log of the predicted probability for the true class

- Average **cross entropy loss**

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \overbrace{\sum_{k=0}^{K-1} - \mathbf{P}\left(y_i = k \,|\, x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = k \,|\, x_i\right)\right)}$$

- Computing the more general version for (x_i, y_i)

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \left(\begin{aligned} \mathbf{P}\left(y_i = 0 \,|\, x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = 0 \,|\, x_i\right)\right) + \\ \mathbf{P}\left(y_i = 1 \,|\, x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = 1 \,|\, x_i\right)\right) \end{aligned} \right)$$

$$\mathbf{P}\left(y_i = 1 \,|\, x_i\right) = y_i$$

$$\mathbf{P}\left(y_i = 0 \,|\, x_i\right) = \left(1 - y_i\right)$$

$$\hat{\mathbf{P}}_{\theta}\left(y_i = 1 \,|\, x_i\right) = \sigma(\phi(x_i)^T \theta)$$

$$\hat{\mathbf{P}}_{\theta}\left(y_i = 0 \,|\, x_i\right) = 1 - \sigma(\phi(x_i)^T \theta)$$

- Average **cross entropy loss**

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{K-1} - \mathbf{P}\left(y_i = k \mid x_i\right) \log \left(\hat{\mathbf{P}}_{\theta}\left(y_i = k \mid x_i\right) \right)$$

- Computing the more general version for (x_i, y_i)

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \left(\begin{array}{ccc} (1 - y_i) & \log & \left(1 - \sigma(\phi(x_i)^T \theta)\right) + \\ y_i & \log & \left(\sigma(\phi(x_i)^T \theta) \right) \end{array} \right)$$

Rewriting on one line:

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log \left(\sigma \left(\phi(x_i)^T \theta \right) \right) + (1 - y_i) \log \left(1 - \sigma \left(\phi(x_i)^T \theta \right) \right) \right)$$

- Average **cross entropy loss**

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{K-1} - \mathbf{P}\left(y_i = k \mid x_i\right) \log\left(\hat{\mathbf{P}}_{\theta}\left(y_i = k \mid x_i\right)\right)$$

Rewriting on one line:

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log\left(\sigma\left(\phi(x_i)^T \theta\right)\right) + (1 - y_i) \log\left(1 - \sigma\left(\phi(x_i)^T \theta\right)\right)\right)$$

After much algebra (see last lecture) we obtain:

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \phi(x_i)^T \theta + \log\left(\sigma\left(-\phi(x_i)^T \theta\right)\right)\right)$$

The Loss for Logistic Regression

- Average **cross entropy** (simplified):

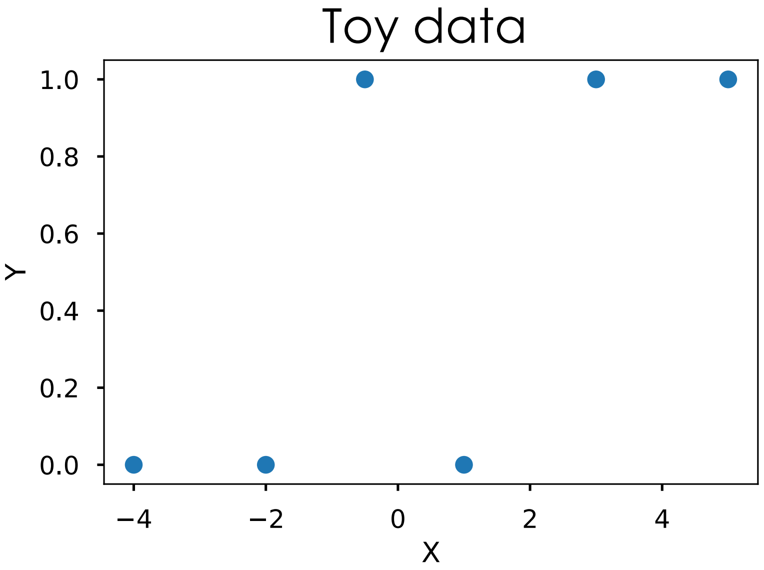
$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

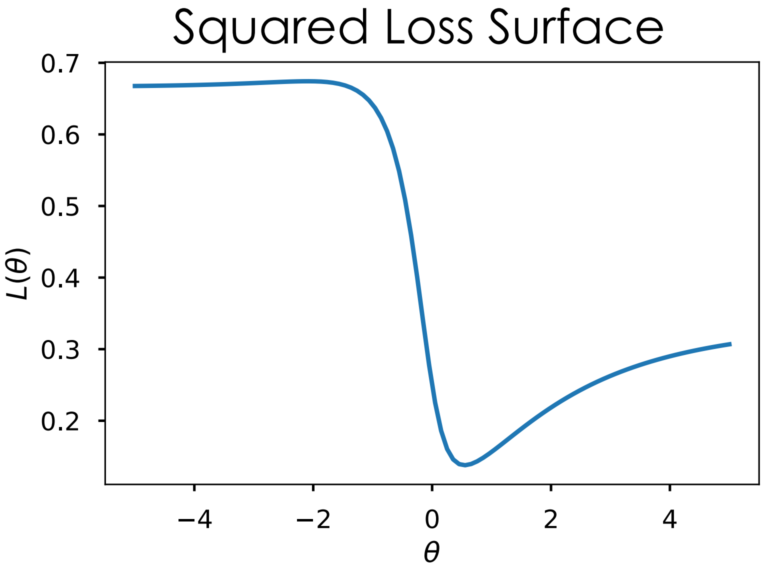
  - Equivalent to (derived from) **minimizing the KL divergence**
  - Also equivalent to **maximizing the log-likelihood of the data** … (not covered in Data100 this semester)

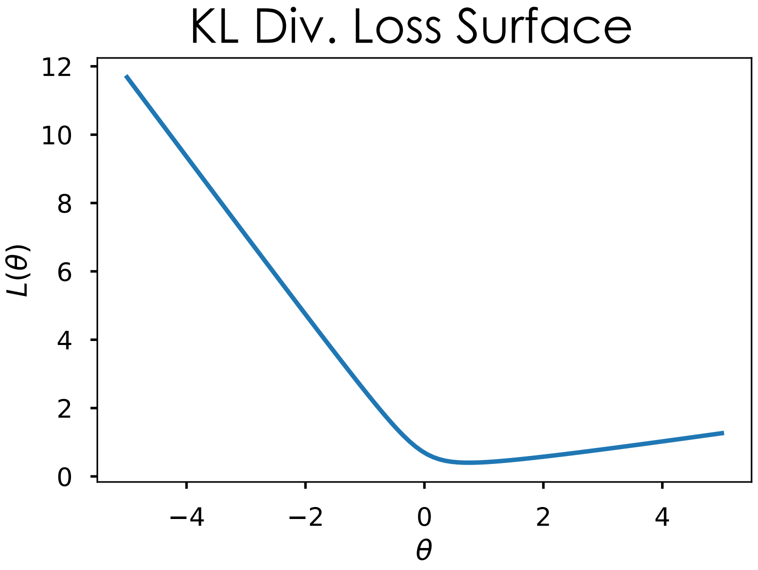
Is this loss function reasonable?

Convexity Using Pictures

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$


Toy data


Squared Loss Surface

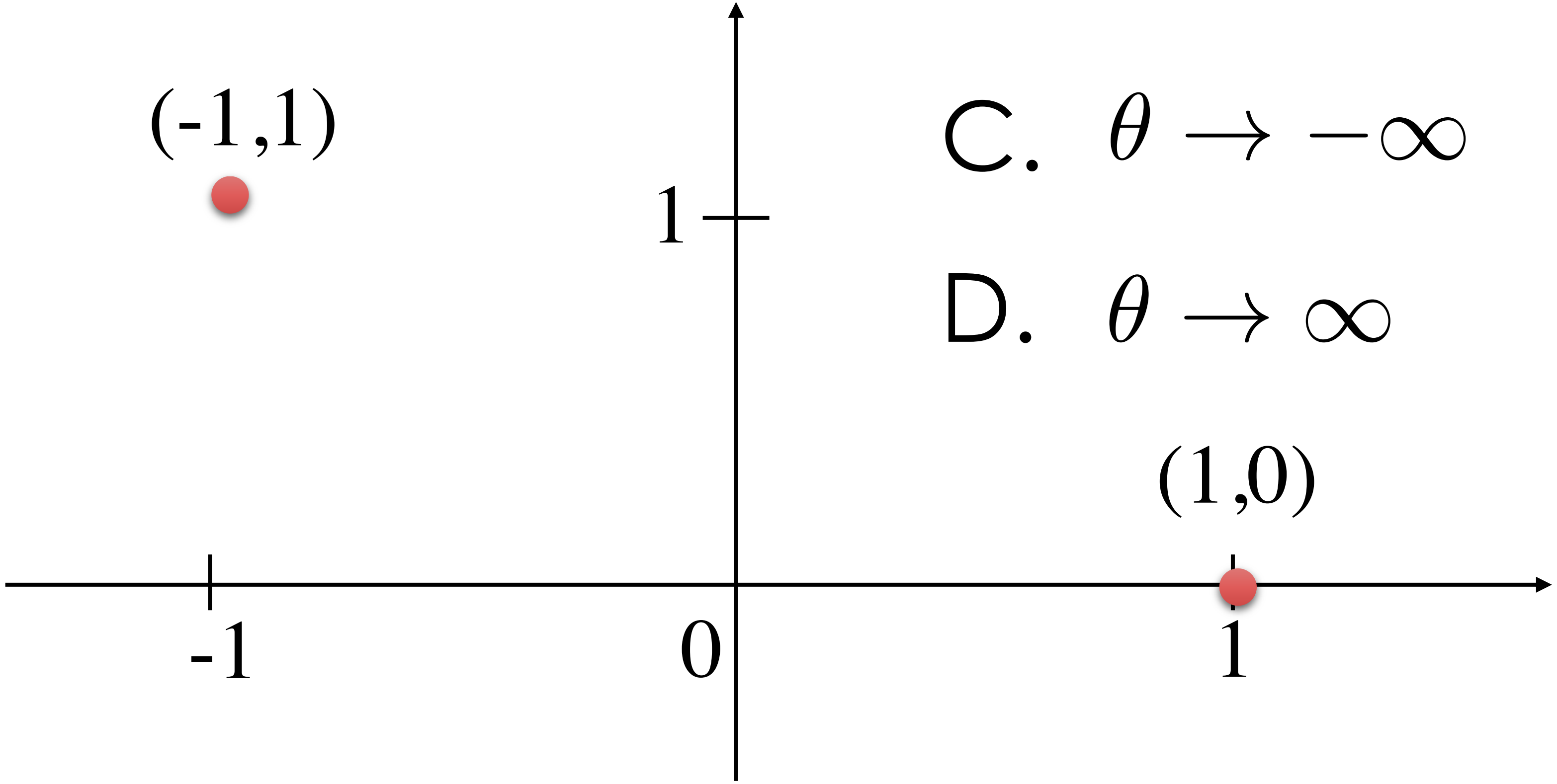

KL Div. Loss Surface

What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

Assume: $\phi(x) = x$

The Data

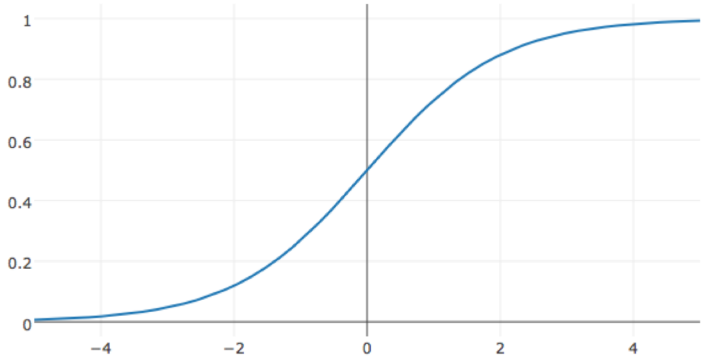


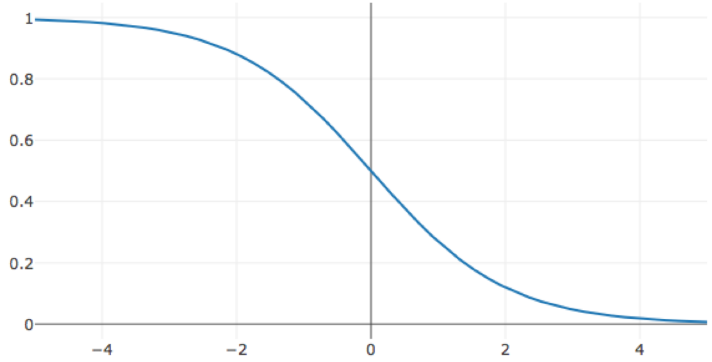
A. $\theta = -1$

B. $\theta = 1$

C. $\theta \to -\infty$

D. $\theta \to \infty$





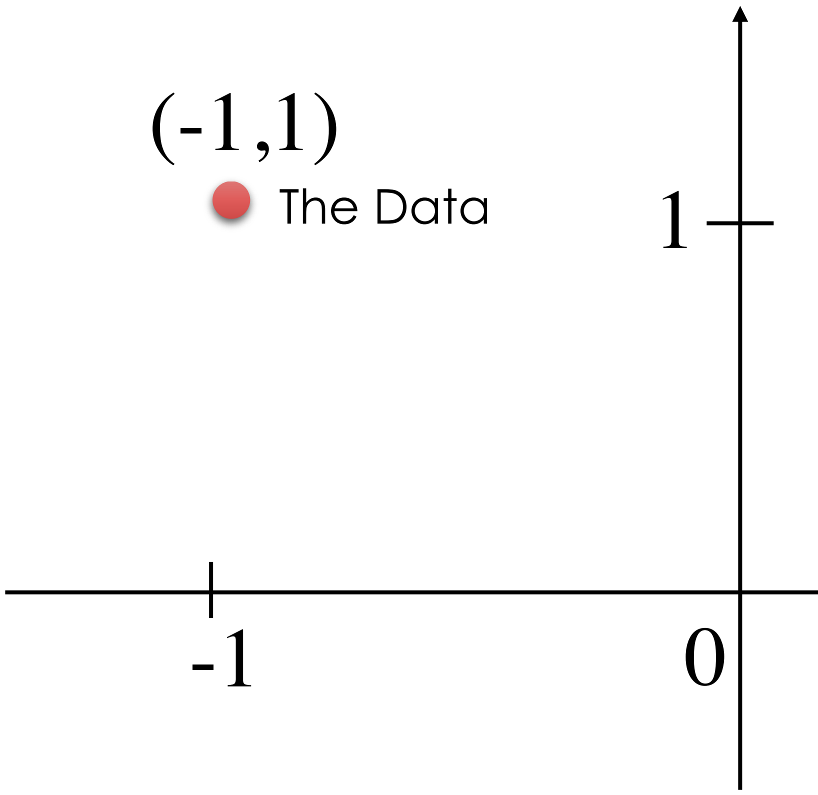
What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

Assume: $\phi(x) = x$

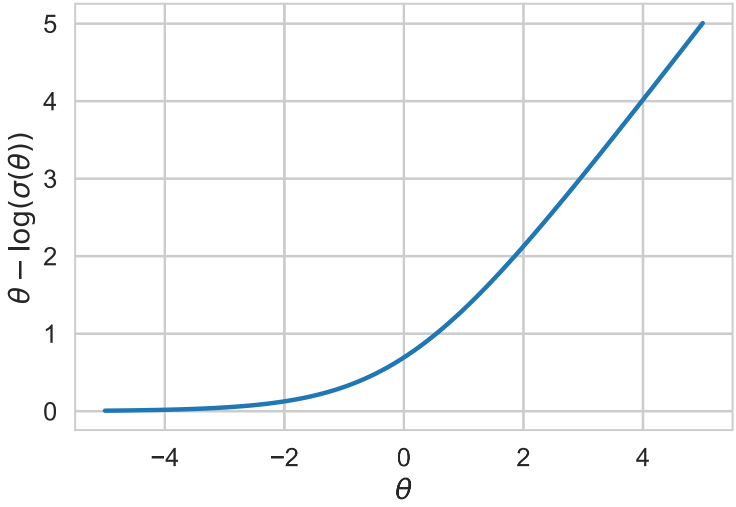
For the point (-1,1): $y_i\phi(x_i)^T = -1$

$$-\phi(x_i)^T = 1$$



Objective:

$$\theta - \log\left(\sigma\left(\theta\right)\right)$$



$$\theta \rightarrow -\infty$$

What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

For the point (-1,1): $\theta - \log\left(\sigma\left(\theta\right)\right)$

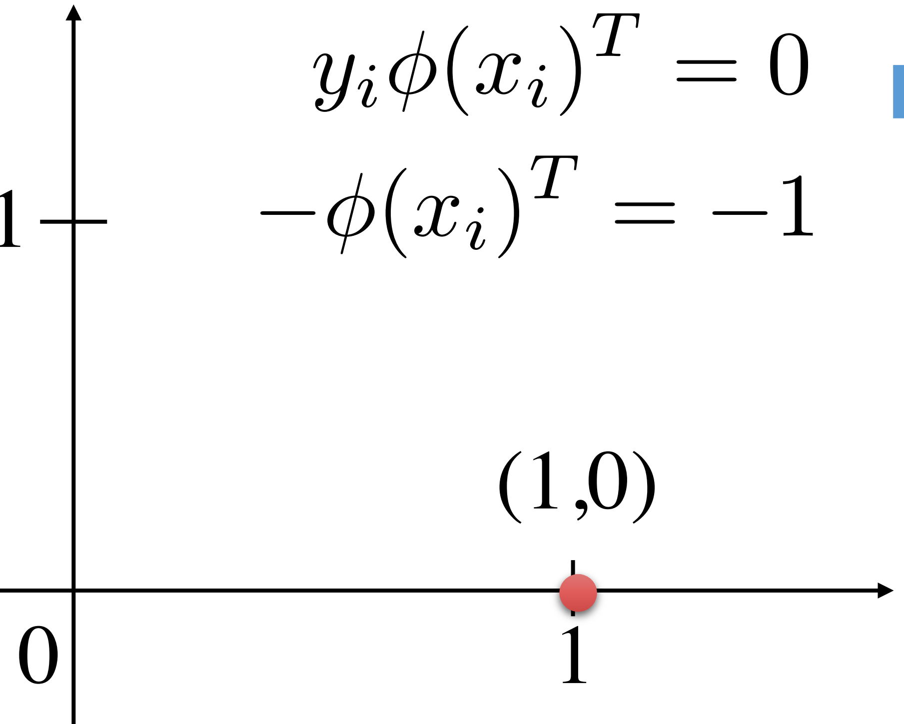
$$\theta \rightarrow -\infty$$

Assume: $\phi(x) = x$

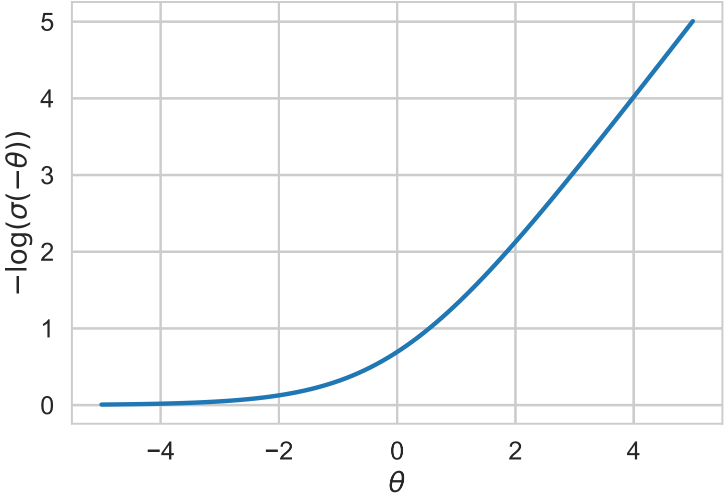
For the point (1, 0):

$$y_i\phi(x_i)^T = 0$$

$$-\phi(x_i)^T = -1$$



➡ $0 - \log(\sigma(-\theta))$



$$\theta \rightarrow -\infty$$

What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

Assume: $\phi(x) = x$

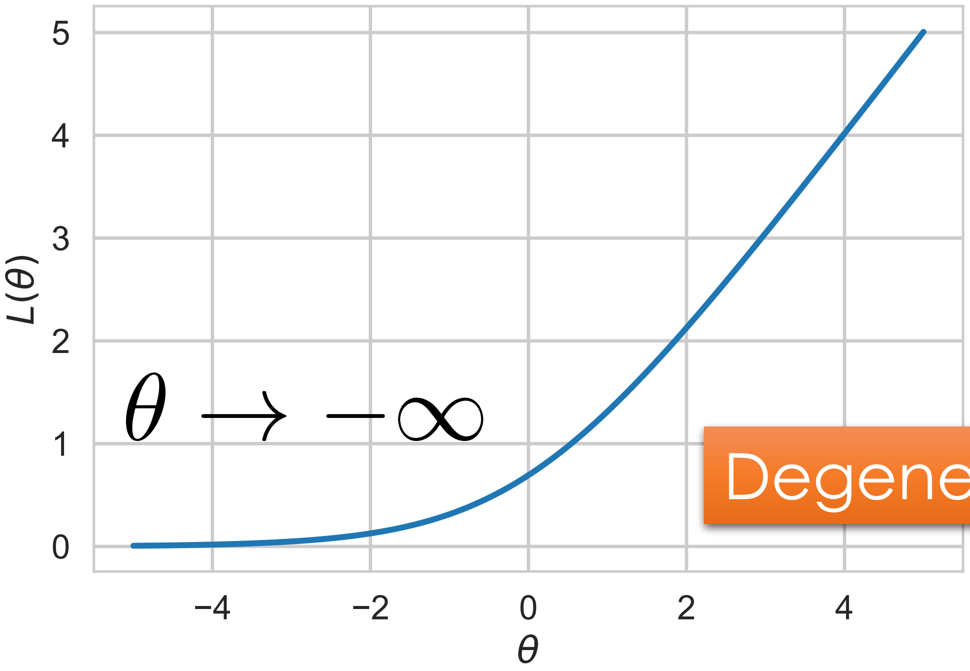
For the point (-1,1): $\theta - \log\left(\sigma\left(\theta\right)\right)$

$$\theta \to -\infty$$

For the point (1, 0): $0 - \log(\sigma(-\theta))$

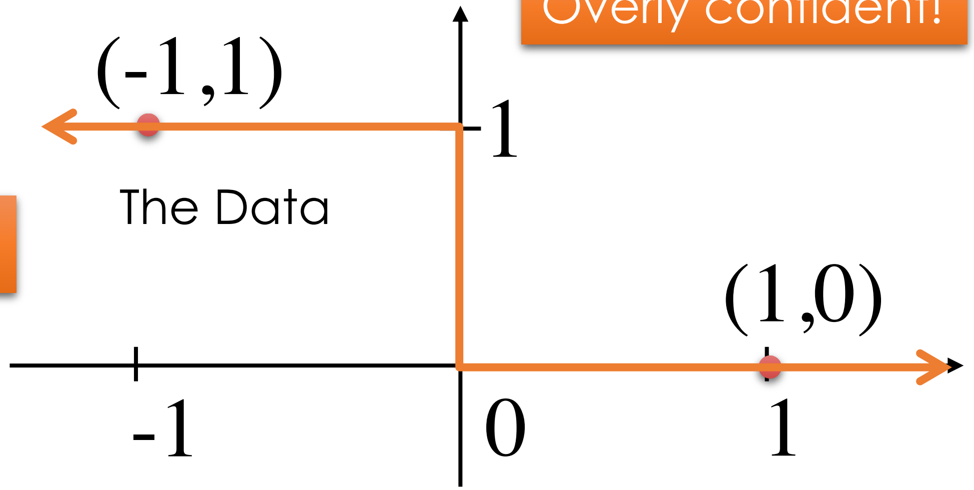
$$\theta \to -\infty$$

Total Loss



$\theta \to -\infty$

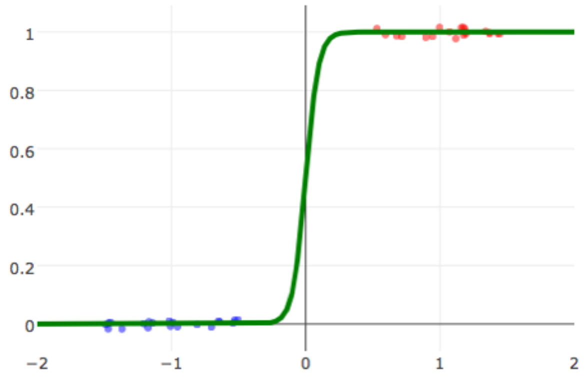
Degenerate Solution!



Overly confident!

Linearly Separable Data

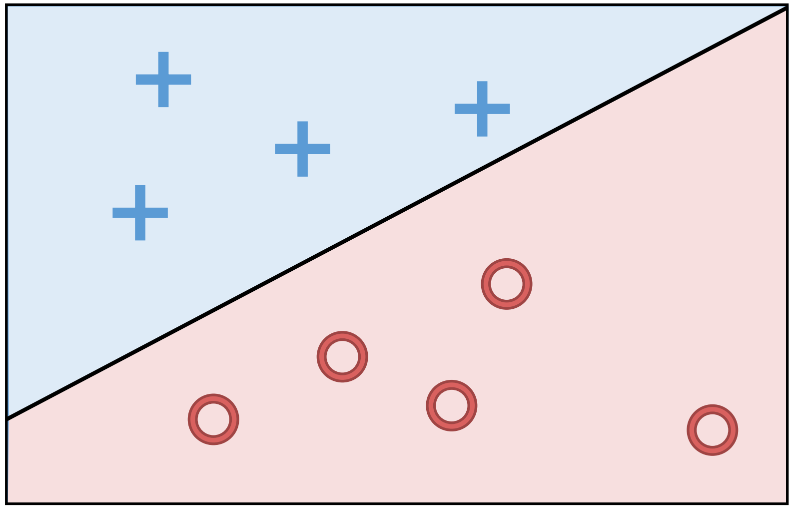
- A classification dataset is said to be linearly separable if there exists a hyperplane that separates the two classes.
- If data is linearly separable, logistic regression requires regularization



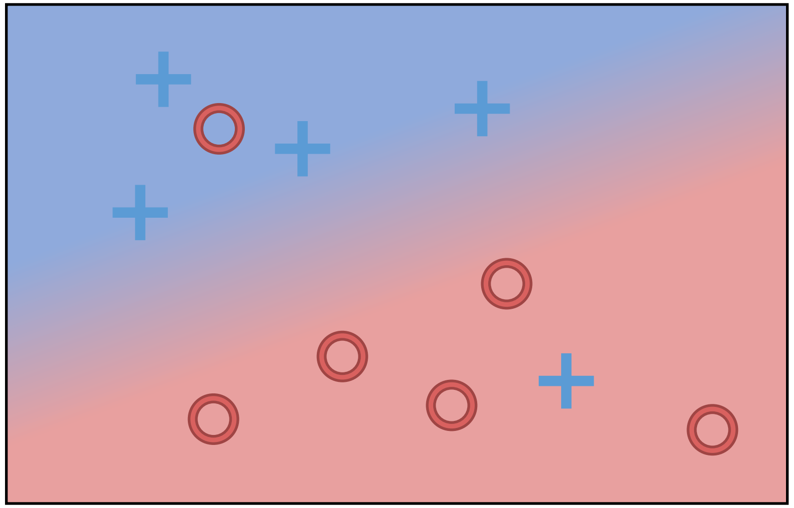
Weights go to infinity!

Solution?

Linearly Separable Data



Not Linearly Separable Data

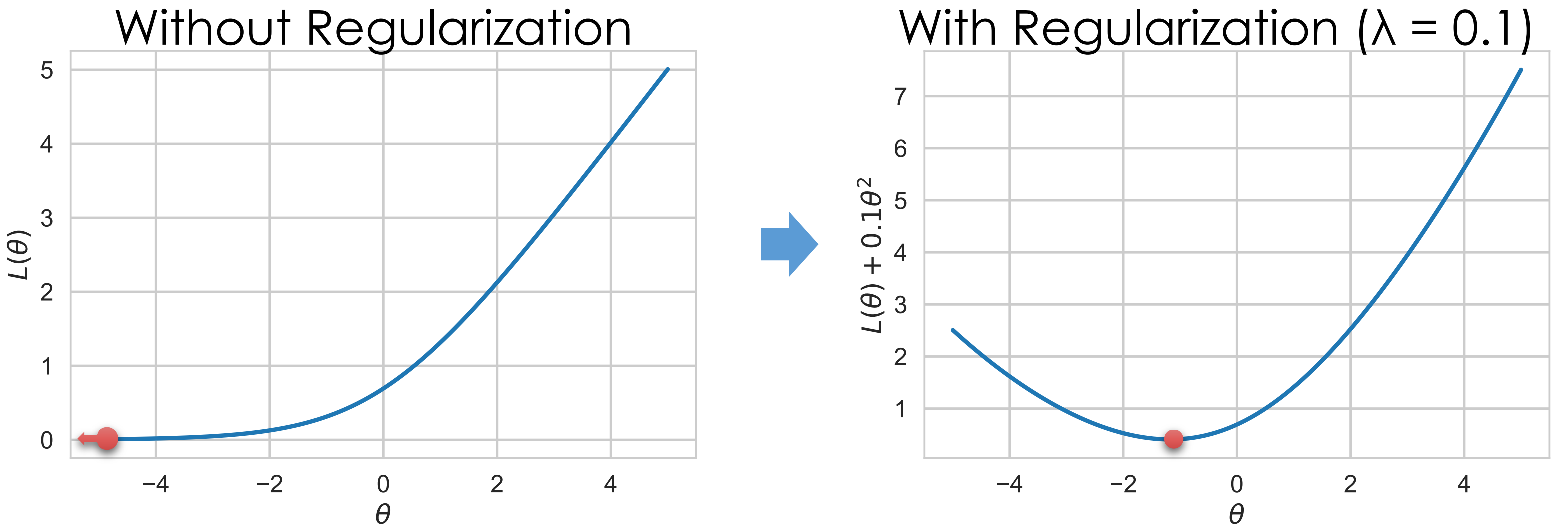


Adding Regularization to Logistic Regression

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right) + \lambda\sum_{j=1}^{d}\theta_j^2$$

- ➢ Prevents weights from diverging on linearly separable data

Earlier Example



Without Regularization

With Regularization (λ = 0.1)

Minimize the Loss

Logistic Loss Function

- Average KL divergence (simplified)

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Take Derivative:

$$\begin{aligned}\nabla_{\theta}\mathbf{L}(\theta) &= -\frac{1}{n}\sum_{i=1}^{n}\nabla_{\theta}y_i\phi(x_i)^T\theta + \nabla_{\theta}\log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\\ &= -\frac{1}{n}\sum_{i=1}^{n}y_i\phi(x_i) + \nabla_{\theta}\log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\end{aligned}$$

- Average KL divergence (simplified)

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\nabla_\theta y_i\phi(x_i)^T\theta + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i\phi(x_i) + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i\phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)}\nabla_\theta \sigma\left(-\phi(x_i)^T\theta\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)} \nabla_\theta \sigma\left(-\phi(x_i)^T\theta\right)$$

Useful Identity

$$\frac{\partial}{\partial t}\sigma(t) = \frac{\partial}{\partial t}\frac{1}{1+e^{-t}} \overset{\text{Chain Rule}}{=} \frac{-1}{\left(1+e^{-t}\right)^2}\frac{\partial}{\partial t}\left(1+e^{-t}\right)$$

$$\overset{\text{Chain Rule}}{=} \frac{e^{-t}}{\left(1+e^{-t}\right)^2} \overset{\text{Alg.}}{=} \left(\frac{1}{1+e^{-t}}\right)\left(\frac{e^{-t}}{1+e^{-t}}\right)$$

$$\overset{\text{Alg.}}{=} \left(\frac{1}{1+e^{-t}}\right)\left(\frac{1}{e^{t}+1}\right) \overset{\text{Defn. of }\sigma}{=} \sigma(t)\sigma(-t)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)} \nabla_\theta \sigma\left(-\phi(x_i)^T\theta\right)$$

Useful Identity $\frac{\partial}{\partial t}\sigma(t) = \sigma(t)\sigma(-t)$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{\sigma\left(-\phi(x_i)^T\theta\right)}{\sigma\left(-\phi(x_i)^T\theta\right)} \sigma\left(\phi(x_i)^T\theta\right) \nabla_\theta\left(-\phi(x_i)^T\theta\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} \left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)\phi(x_i)$$

Logistic Loss Function

- Average KL divergence (simplified)

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Take Derivative:

$$\nabla_{\theta}\mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)\phi(x_i)$$

- Set derivative = 0 and solve for θ
  - No general analytic solution
  - Solved using numeric methods

The Gradient Descent Algorithm

```
$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)
For $\tau$ from 0 to convergence:
    $\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left( \left. \nabla_\theta \mathbf{L}(\theta) \right|_{\substack{\text{Evaluated} \\ \text{at} \\ \theta=\theta^{(\tau)}}} \right)$
```

- **$\rho(\tau)$** is the step size (learning rate)
  - typically $1/\tau$
- Converges when gradient is ≈ 0 (or we run out of patience)

Gradient Descent for Logistic Regression

Logistic Regression

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left(\frac{1}{n} \sum_{i=1}^{n} \left(\sigma \left(\phi(x_i)^T \theta^{(\tau)} \right) - y_i \right) \phi(x_i) \right)$$

- ***ρ(τ)*** is the step size (learning rate)
  - typically $1/\tau$
- Converges when gradient is ≈ 0 (or we run out of patience)

Stochastic Gradient Descent

- For many learning problems the gradient is a sum:

$$\nabla_\theta \mathbf{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\sigma\left(\phi(x_i)^T\theta\right) - y_i\right)\phi(x_i)$$

  - For large n this can be costly

- What if we approximated the gradient by looking at a few random points:

$$\nabla_\theta \mathbf{L}(\theta) \approx \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\left(\sigma\left(\phi(x_i)^T\theta\right) - y_i\right)\phi(x_i)$$

- What if we approximated the gradient by looking at a few random points:

$$\nabla_\theta \mathbf{L}(\theta) \approx \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \left(\sigma\left(\phi(x_i)^T\theta\right) - y_i\right)\phi(x_i)$$

Batch Size

Random sample of records

- This is a reasonable estimator for the gradient
  - Unbiased …
- Often batch size is one! (why is this helpful)
  - Fast to compute!
- A key ingredient in the recent success of deep learning

Stochastic Gradient Descent

```
$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)
For $\tau$ from 0 to convergence:
    $\mathcal{B} \sim \text{ Random subset of indices}$
    $\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left( \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \mathbf{L}_i(\theta) \Big|_{\theta = \theta^{(\tau)}} \right)$
```

Decomposable Loss

$$\mathbf{L}(\theta) = \sum_{i=1}^{n} \mathbf{L}_i(\theta) = \sum_{i=1}^{n} \mathbf{L}(\theta, x_i, y_i)$$

Loss can be written as a sum of the loss on each record.

Gradient Descent

```
$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)
For $\tau$ from 0 to convergence:
    $\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left( \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathbf{L}_i(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$
```

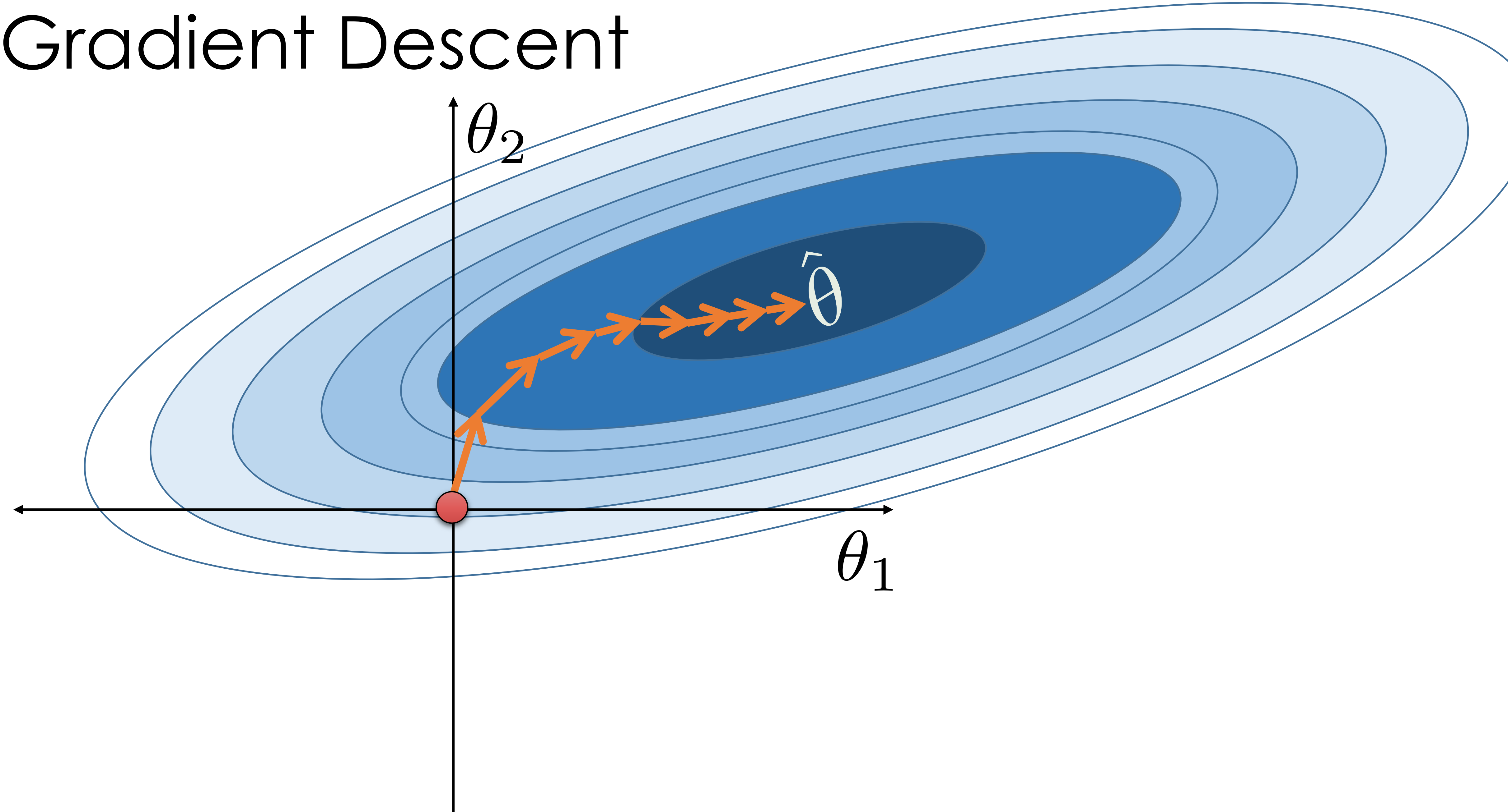
Stochastic Gradient Descent

```
$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)
For $\tau$ from 0 to convergence:
    $\mathcal{B} \sim \text{Random subset of indices}$
    $\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left( \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \mathbf{L}_i(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$
```

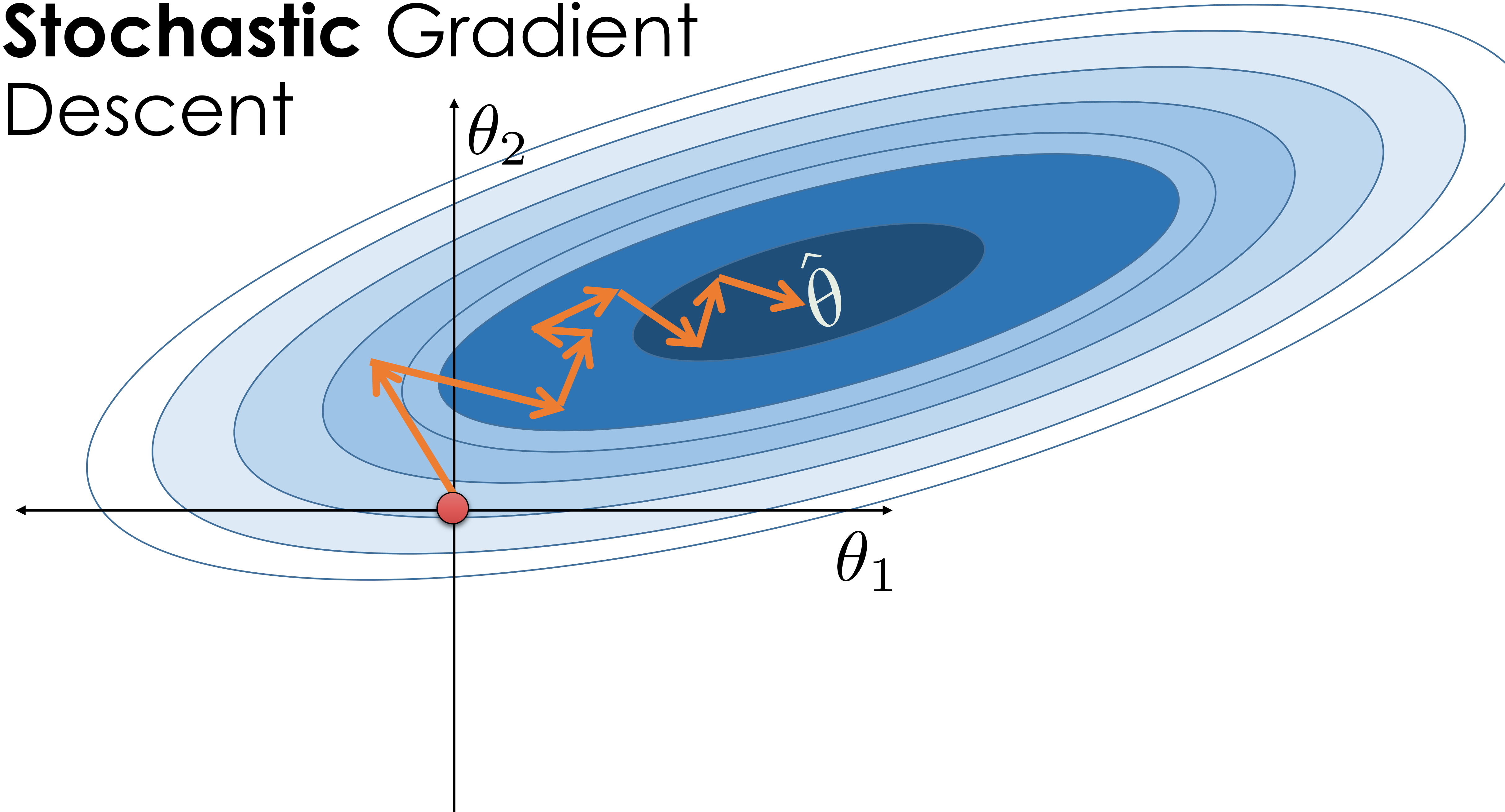
Very Similar Algorithms

Assuming Decomposable Loss Functions

Gradient Descent



Stochastic Gradient Descent



Logistic Regression in Scikit Learn

```
from sklearn.linear_model import LogisticRegression

# By default SK learn adds regularization
# C = 1/lambda the inverse regularization parameter.

model = LogisticRegression(C=100.00)

# Train the model

model.fit(df[[‘feat1’, ‘feat2’]], df[‘label’])

# Make Predictions

test_df[‘label’] = model.predict(test_df[[‘feat1’, ‘feat2’]])

test_df[‘P(Y|X)’] = model.predict_proba(test_df[[‘feat1’, ‘feat2’]])
```

Python Demo!

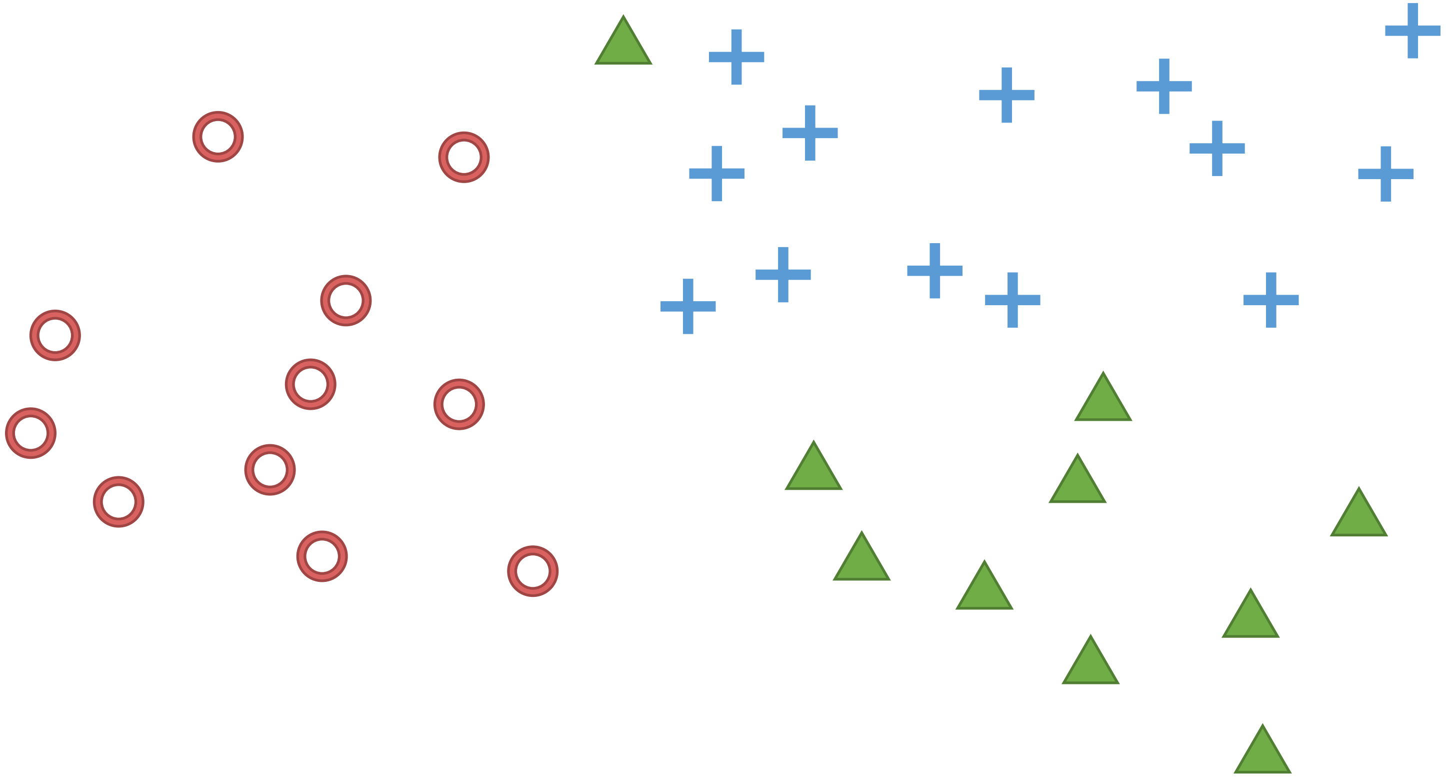
Attendance Quiz

http://bit.ly/ds100-sp18-sgd

1. Is the gradient of a **simple random sample** of the data an **unbiased estimate** of the gradient of the entire dataset? (T/F)
2. By decreasing the batch size we:
   1. **Increase** the **variance** (T/F)
   2. **Decrease** the **bias** (T/F)
   3. **Reduce** the **computational cost** of each iteration (T/F)

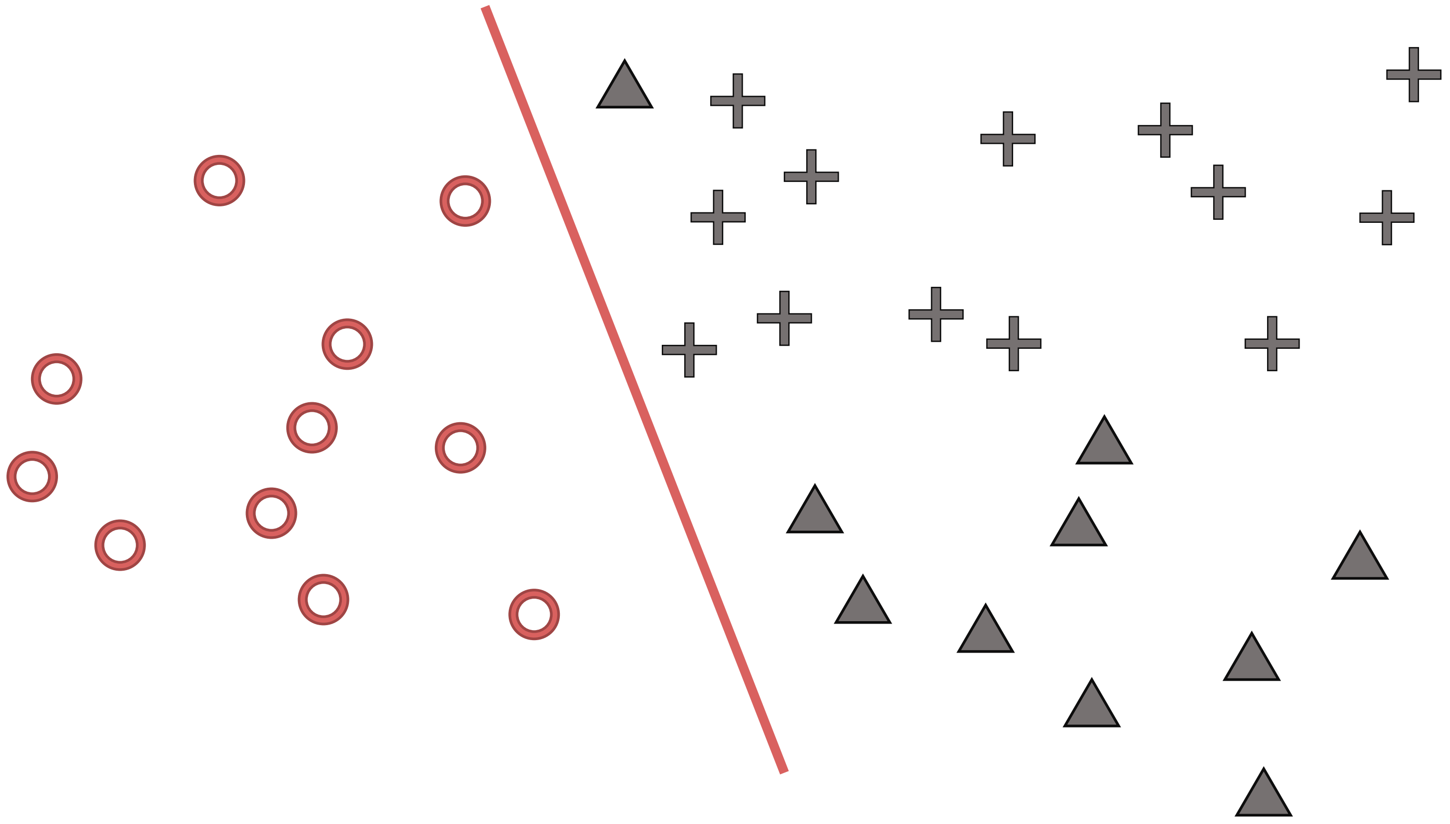
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



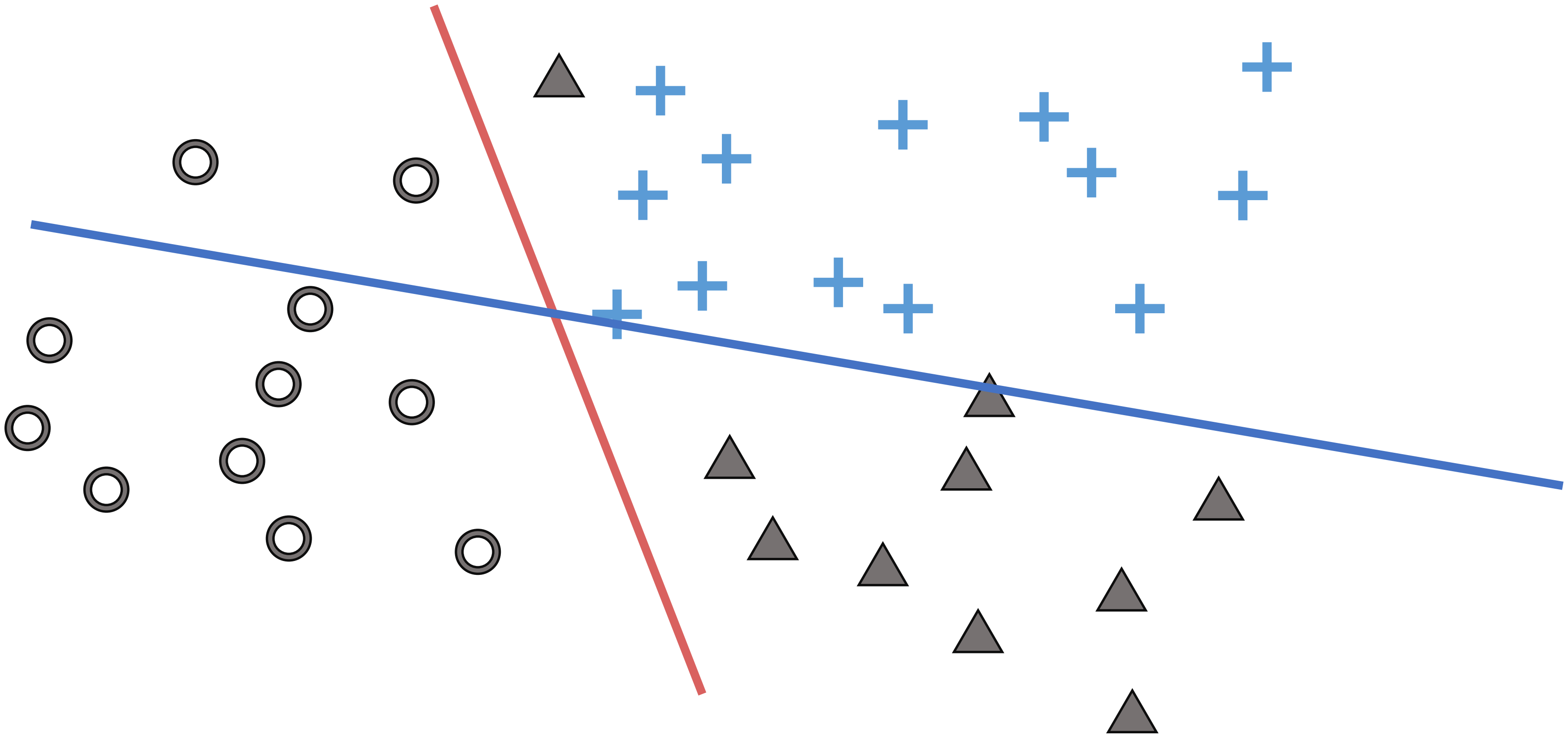
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



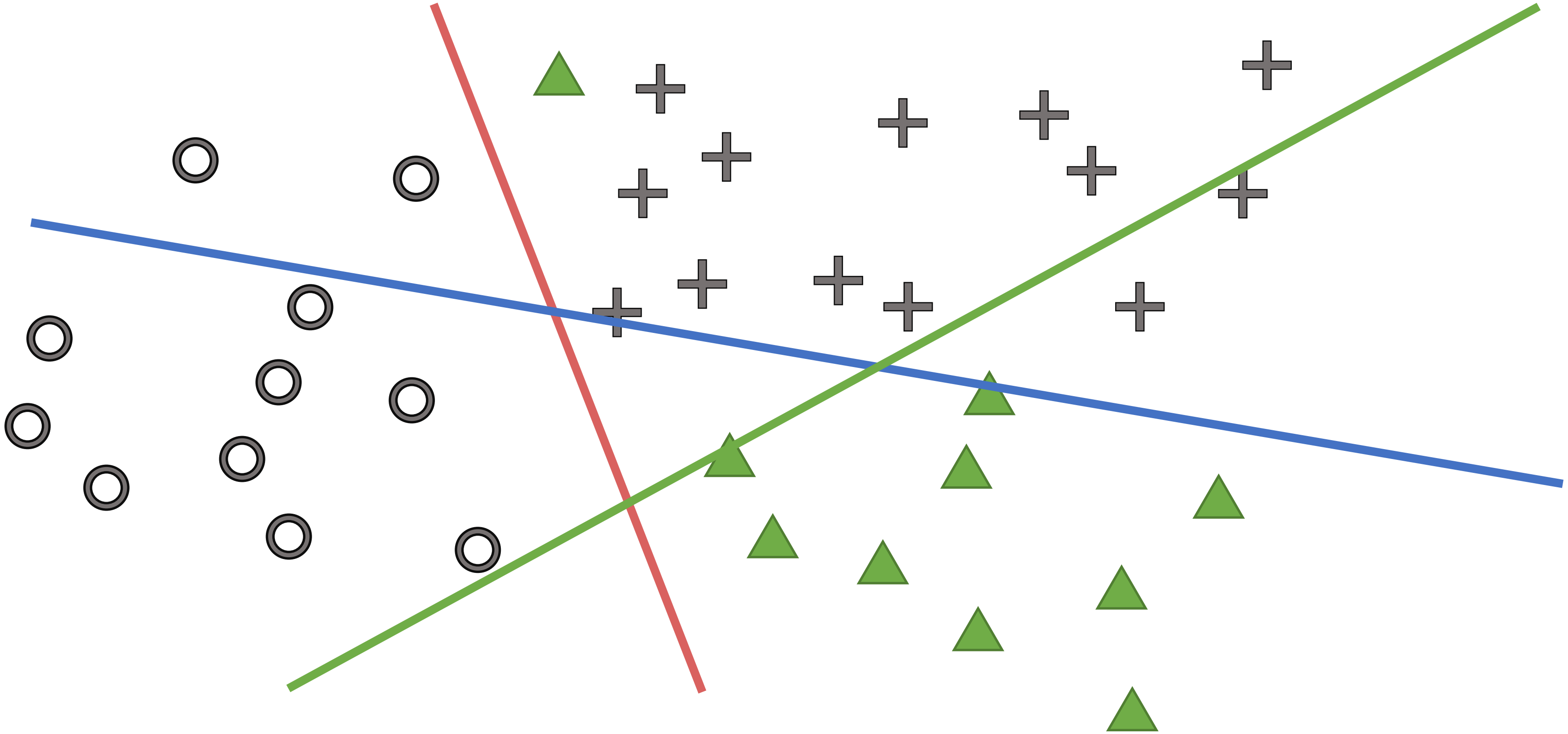
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



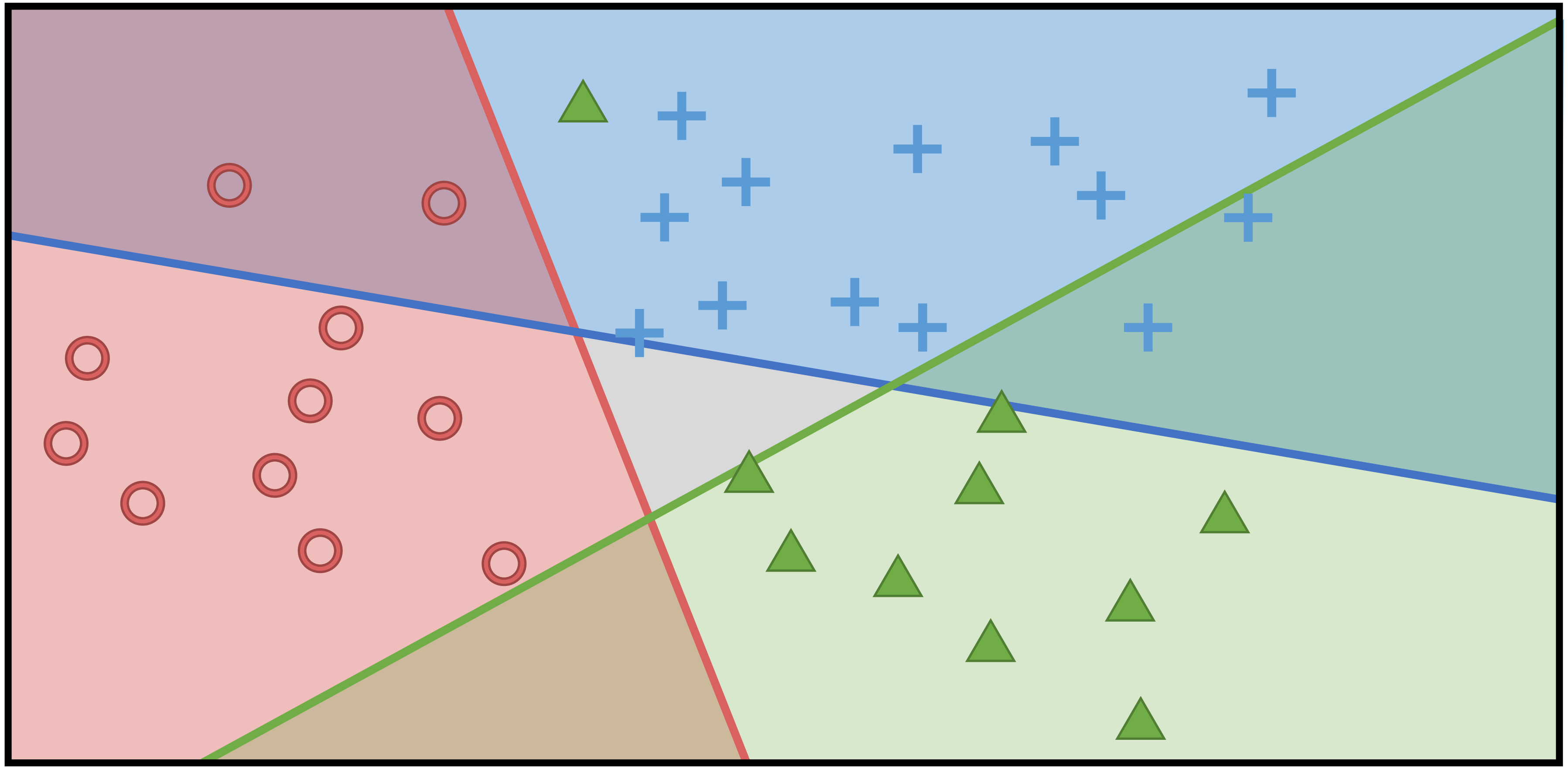
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



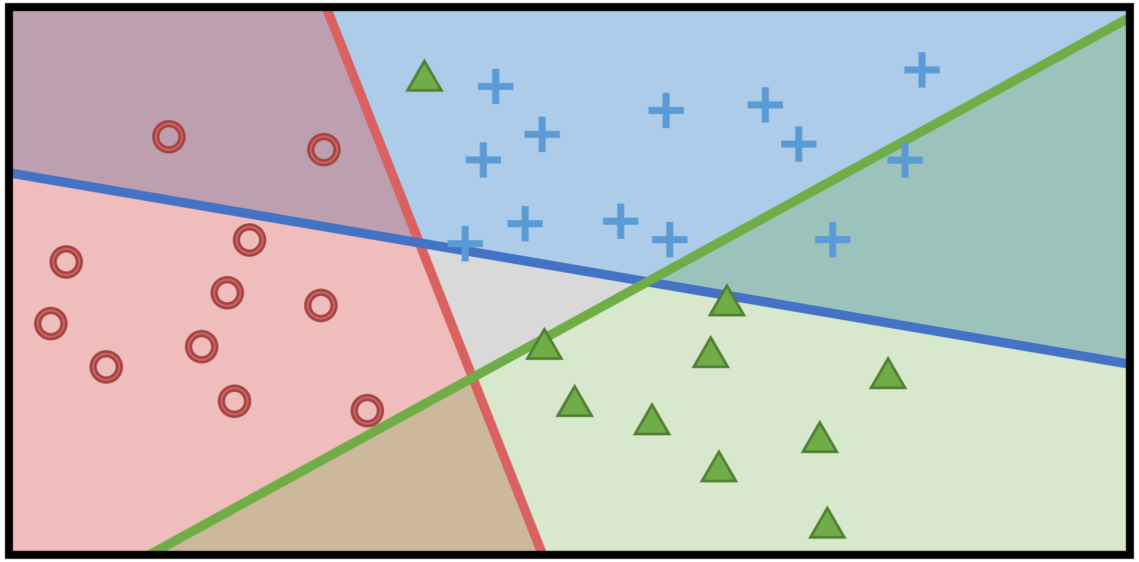
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class
  - Class with highest confidence wins
  - Need to address class imbalance issue



- **Soft-Max** multiclass classification

- **Soft-Max** multiclass classification

$$\mathbf{P}\left(Y = j \,|\, x\right) = \frac{\exp\left(x^T \theta^{(j)}\right)}{\sum_{m=1}^{k} \exp\left(x^T \theta^{(m)}\right)}$$

- Separate $\theta^{(j)} \in \mathbb{R}^p$ for each class
- Trained using gradient descent methods
- Over parameterized. Why?
  - k sets of parameters one for each class
  - Only need K-1 parameters

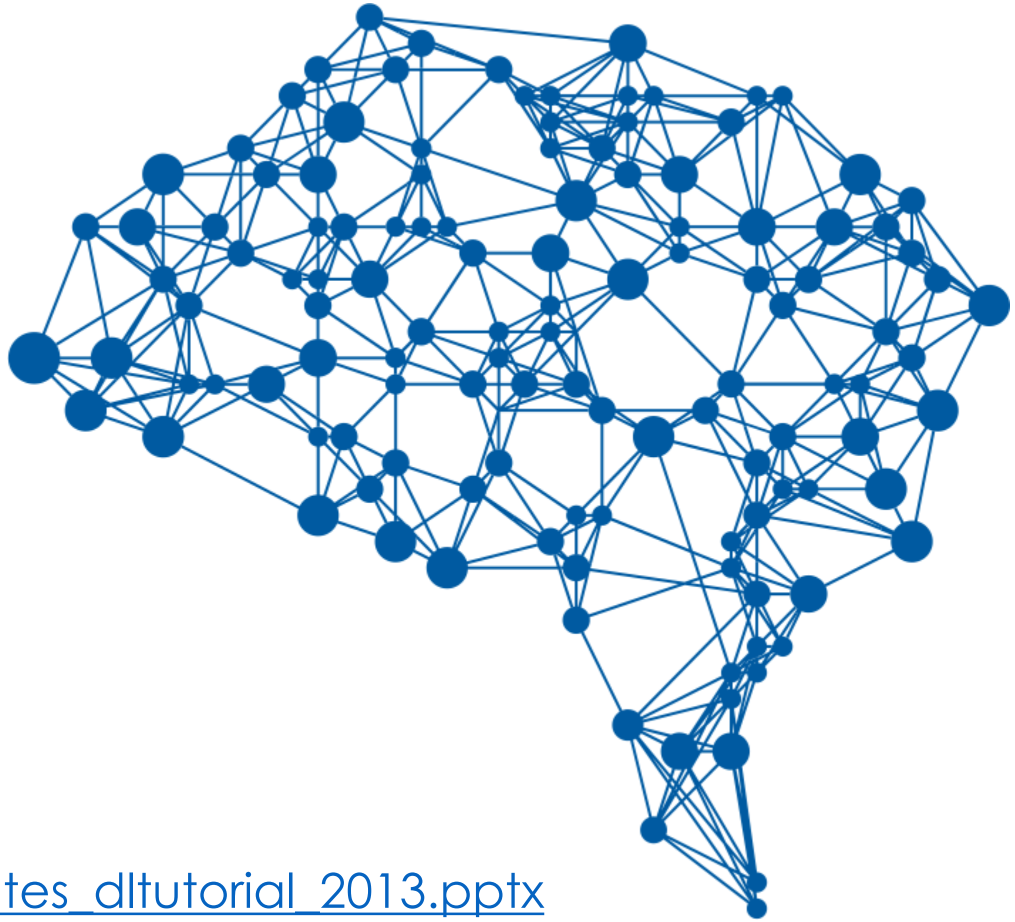
$$\mathbf{P}\left(y = k \,|\, x\right) = 1 - \sum_{j=1}^{K} \mathbf{P}\left(y = j \,|\, x\right)$$

  - Often use k parameters + regularization to address “redundancy”.

Python Demo!

Deep Learning Overview

Bonus Material



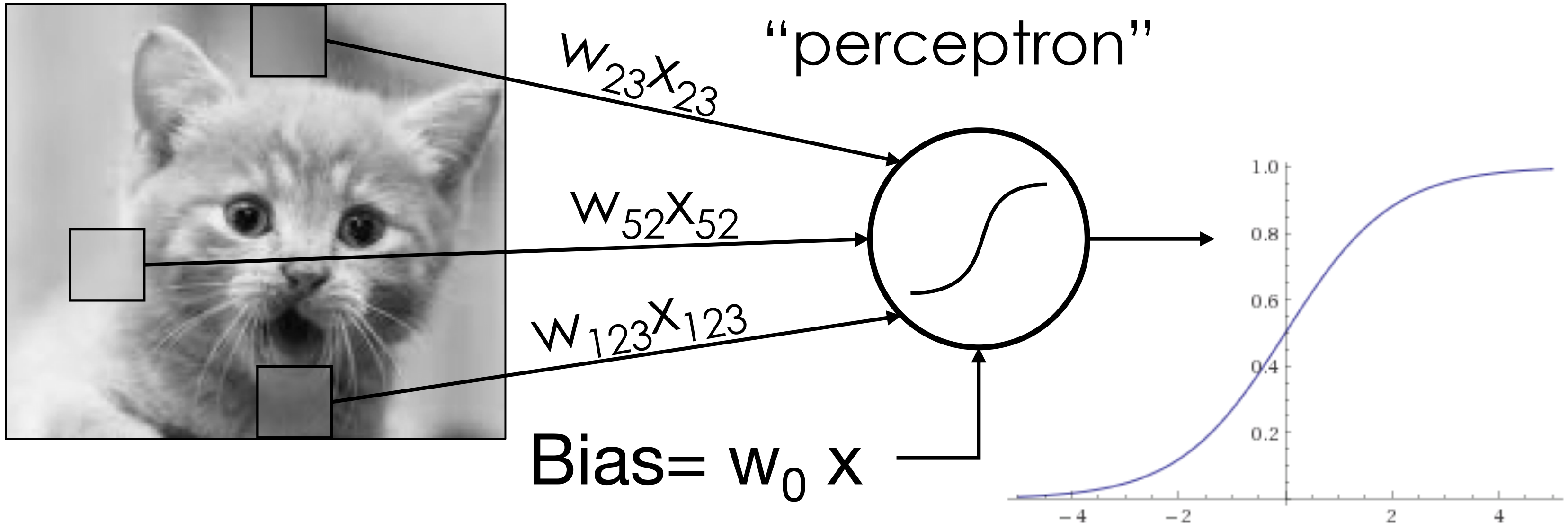
Borrowed from excellent talks by:

- **Adam Coates:** http://ai.stanford.edu/~acoates/coates_dltutorial_2013.pptx
- **Fei-Fei Li and Andrej Karpathy:** http://cs231n.stanford.edu/syllabus.html

Logistic Regression as a "Neuron"

- Consider the simple function family: $$\sigma(u) = \frac{1}{1+\exp(-u)}$$

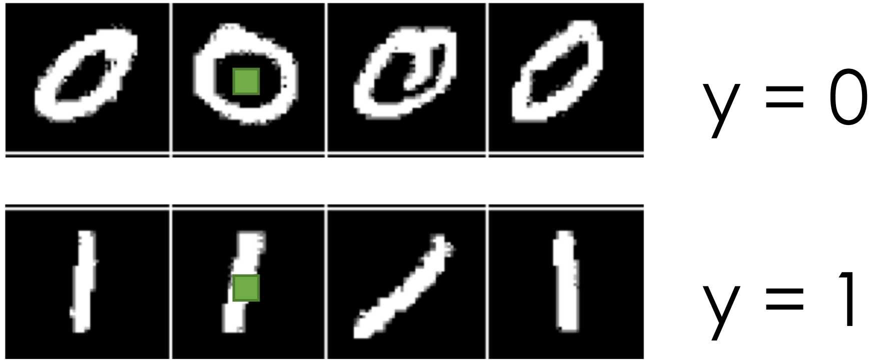
$$f_w(x) = \sigma\left(w^T x\right) = \sigma\left(\sum_{j=1}^{d} w_j x_j\right) = P(y = 1 \,|\, x)$$



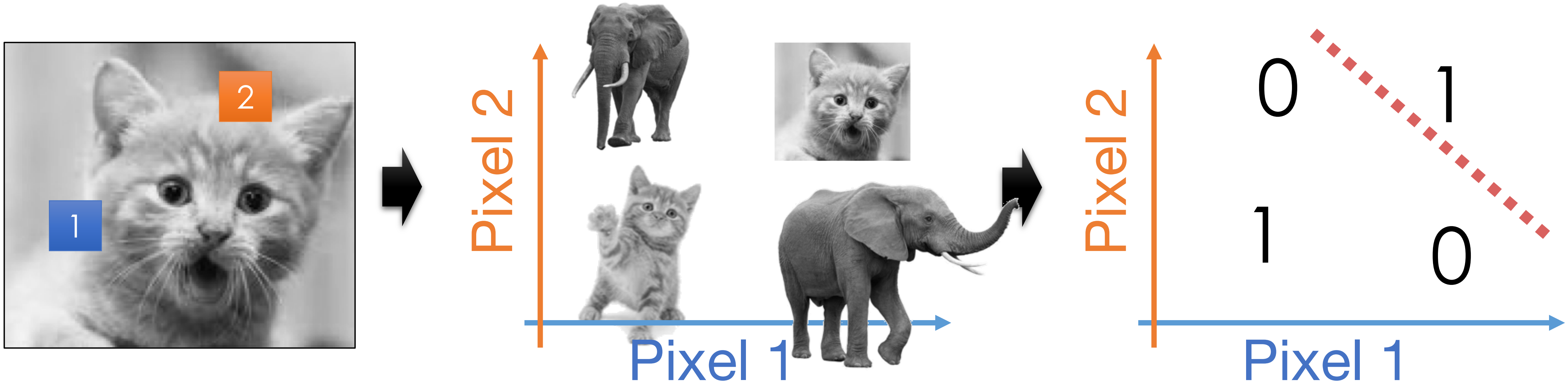
Neuron "fires" if weighted sum of input is greater than zero.

Logistic Regression: Strengths and Limitations

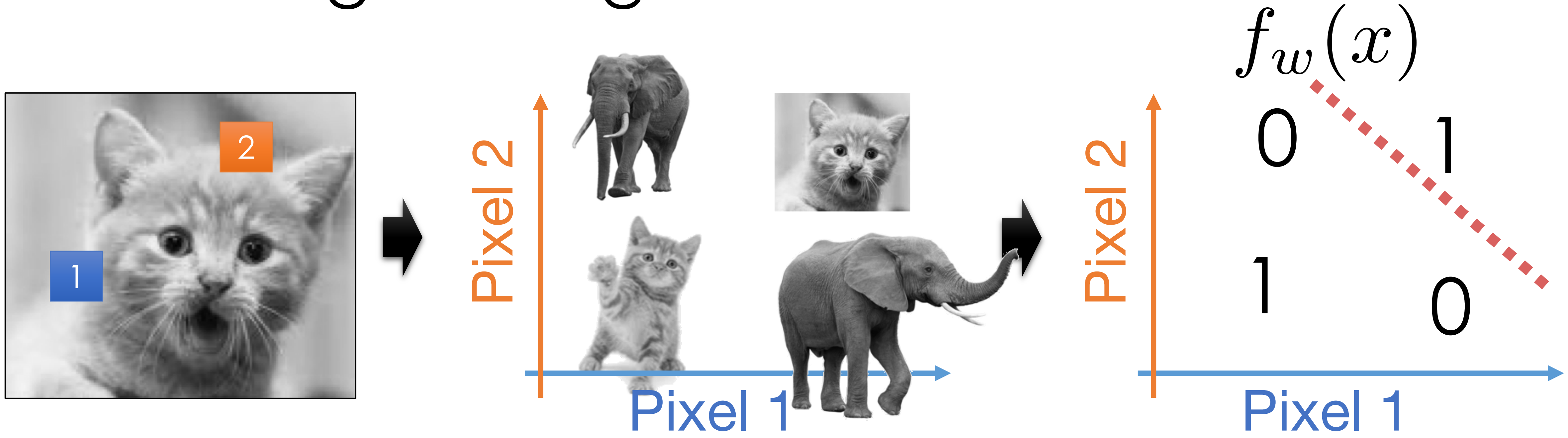
- Widely used machine learning technique
  - convex → efficient to learn
  - easy to interpret model weights
  - works well given good features



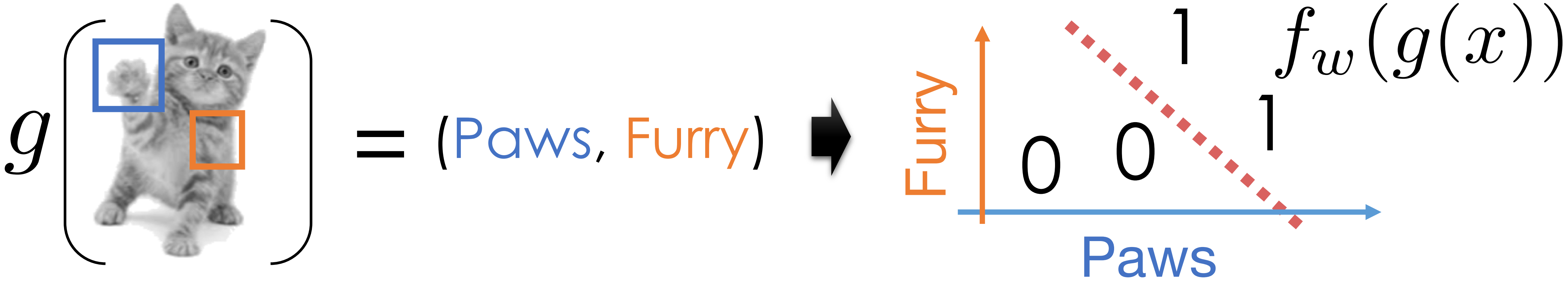
- Limitations:
  - Restricted to linear relationships → sensitive to choice of features



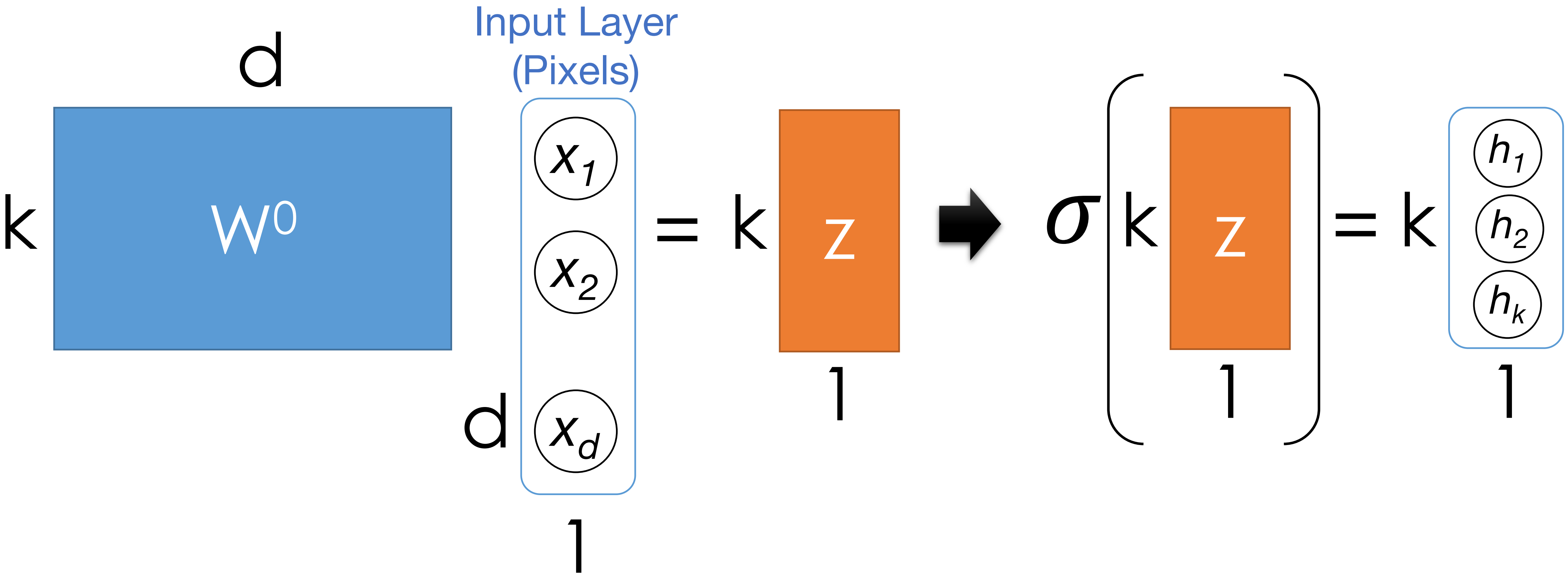
Feature Engineering



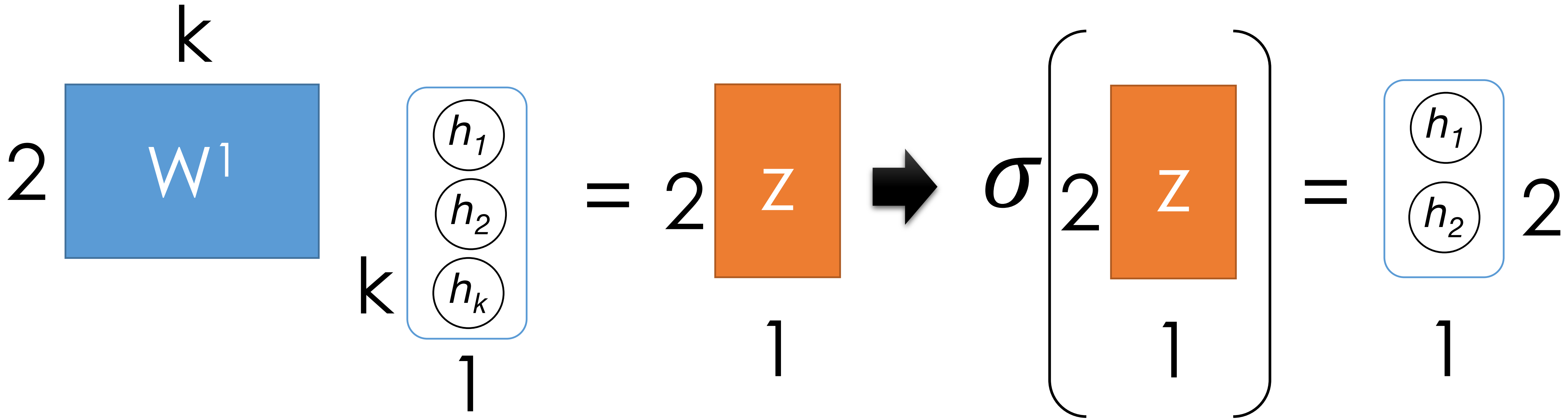
- Rather than use raw **pixels build/train feature functions:**



Composition Linear Models and Nonlinearities



Composition Linear Models and Nonlinearities

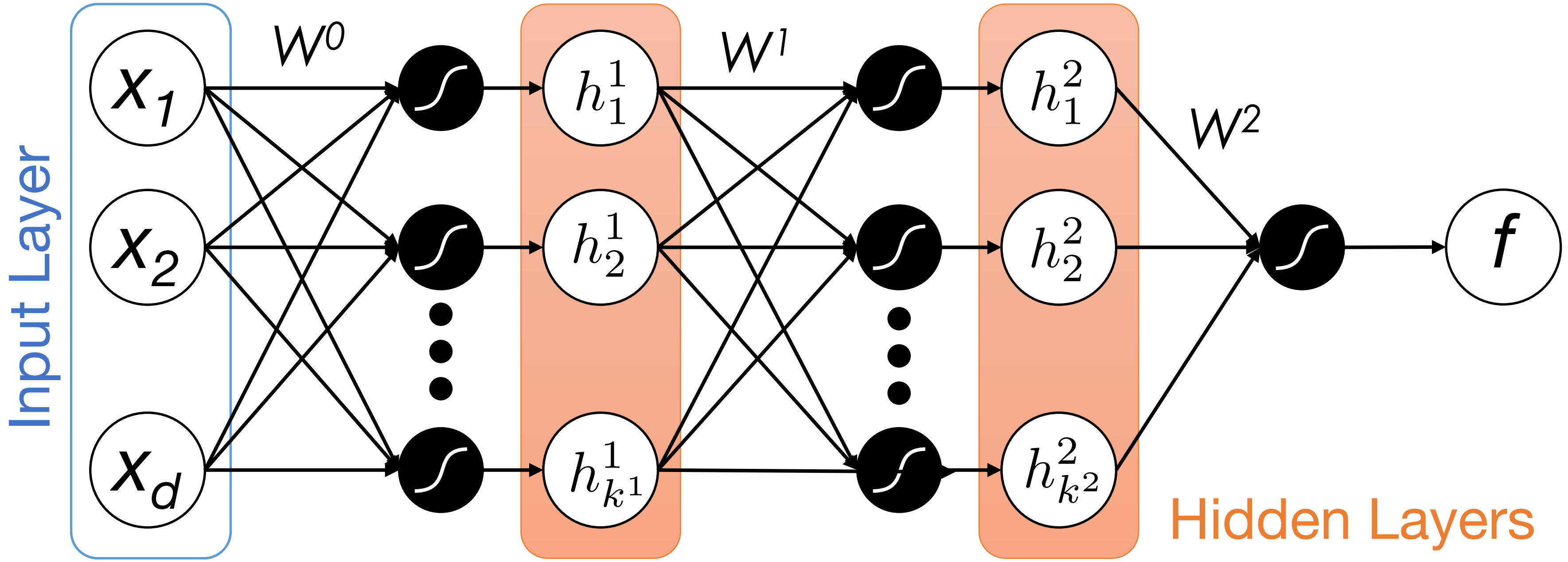


Neural Networks

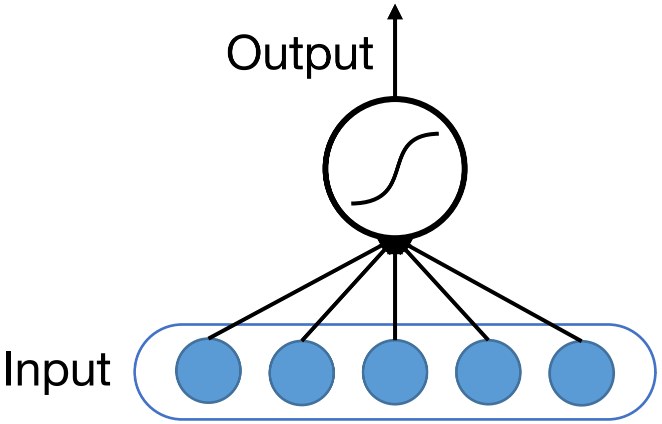
- Composing “perceptrons”

$$x \rightarrow \sigma(W^0 x) \rightarrow h^1 \rightarrow \sigma(W^1 h^1) \rightarrow h^2 \rightarrow \sigma(W^2 h^2) \rightarrow f$$

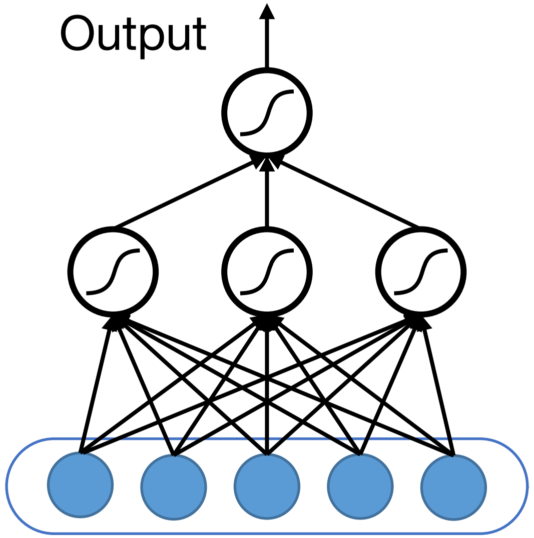
$$y = f_{W^0,W^1,W^2}(x) = \sigma\left(W^2\ \sigma\left(W^1\ \sigma\left(W^0 x\right)\right)\right)$$

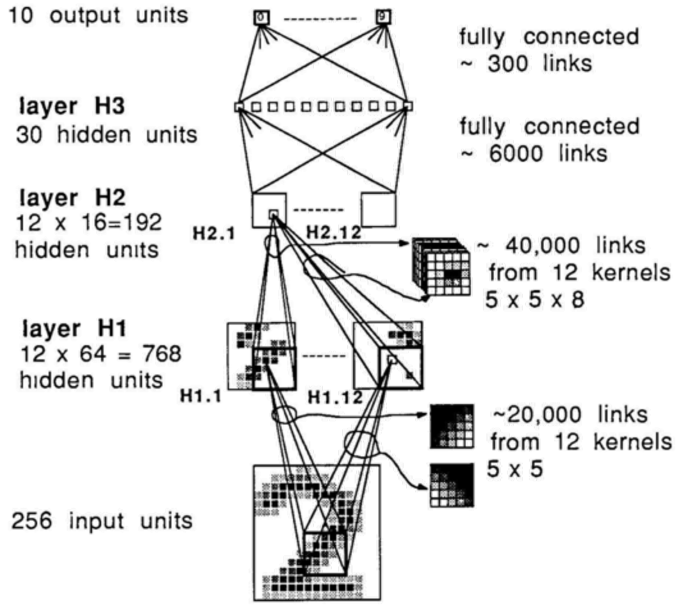


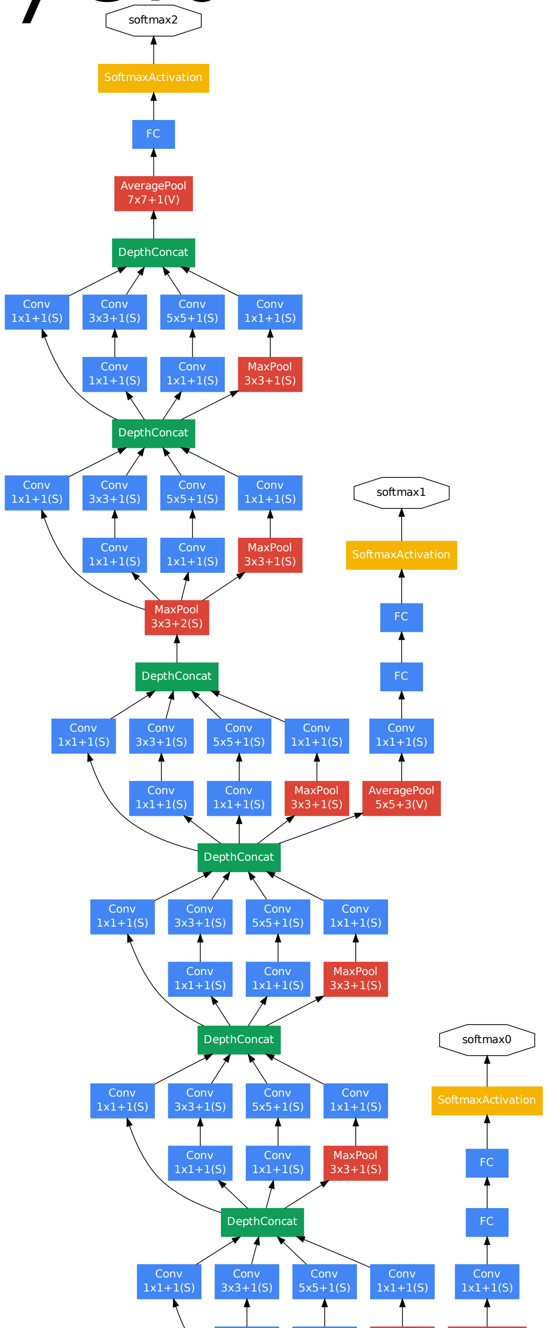
Deep Learning →Many hidden layers …



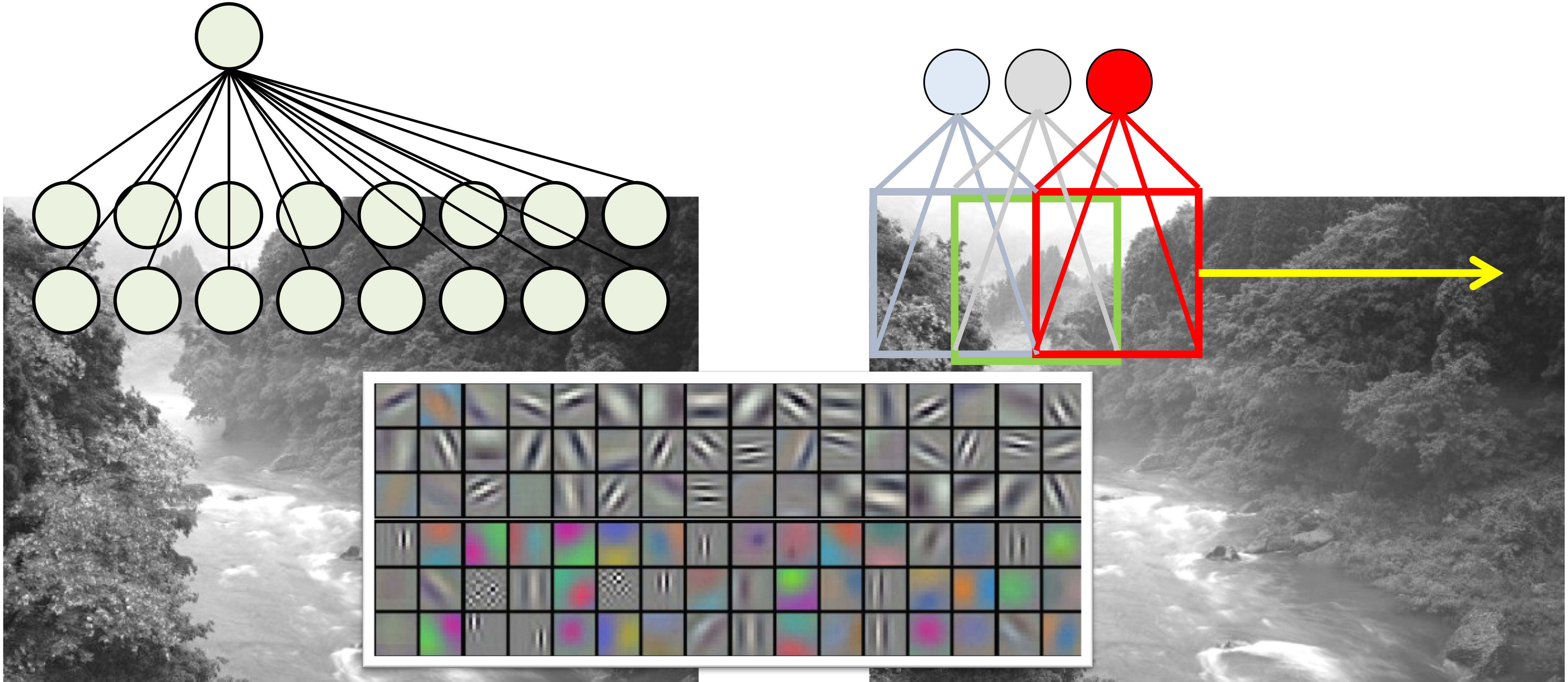
Logistic Regression



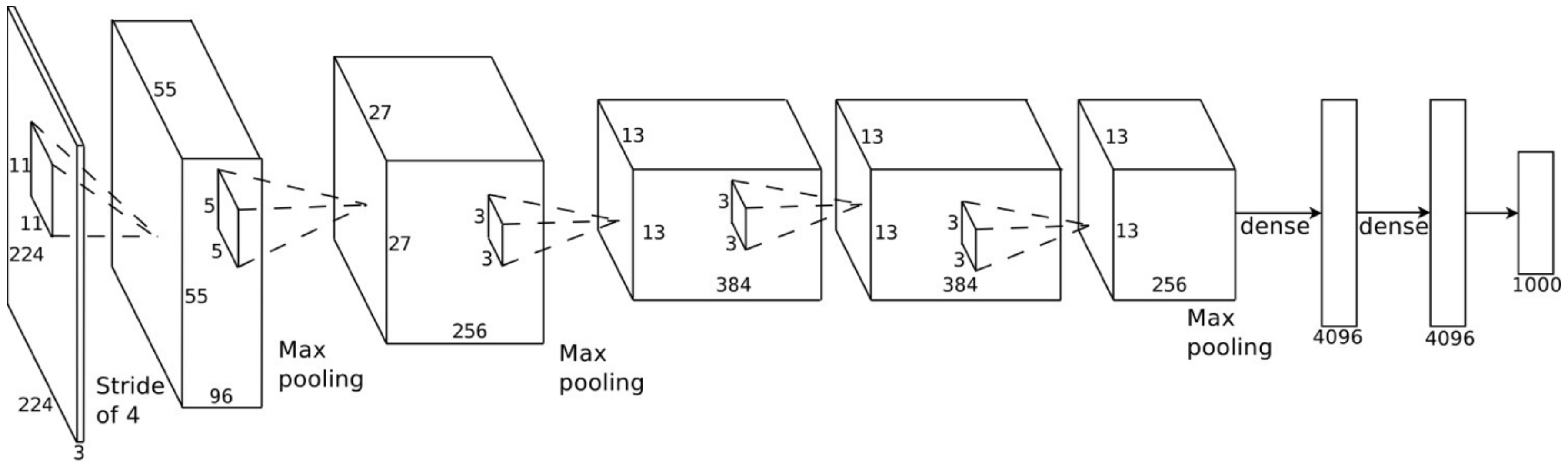




Convolutional Neural Networks: Exploiting Spatial Sparsity

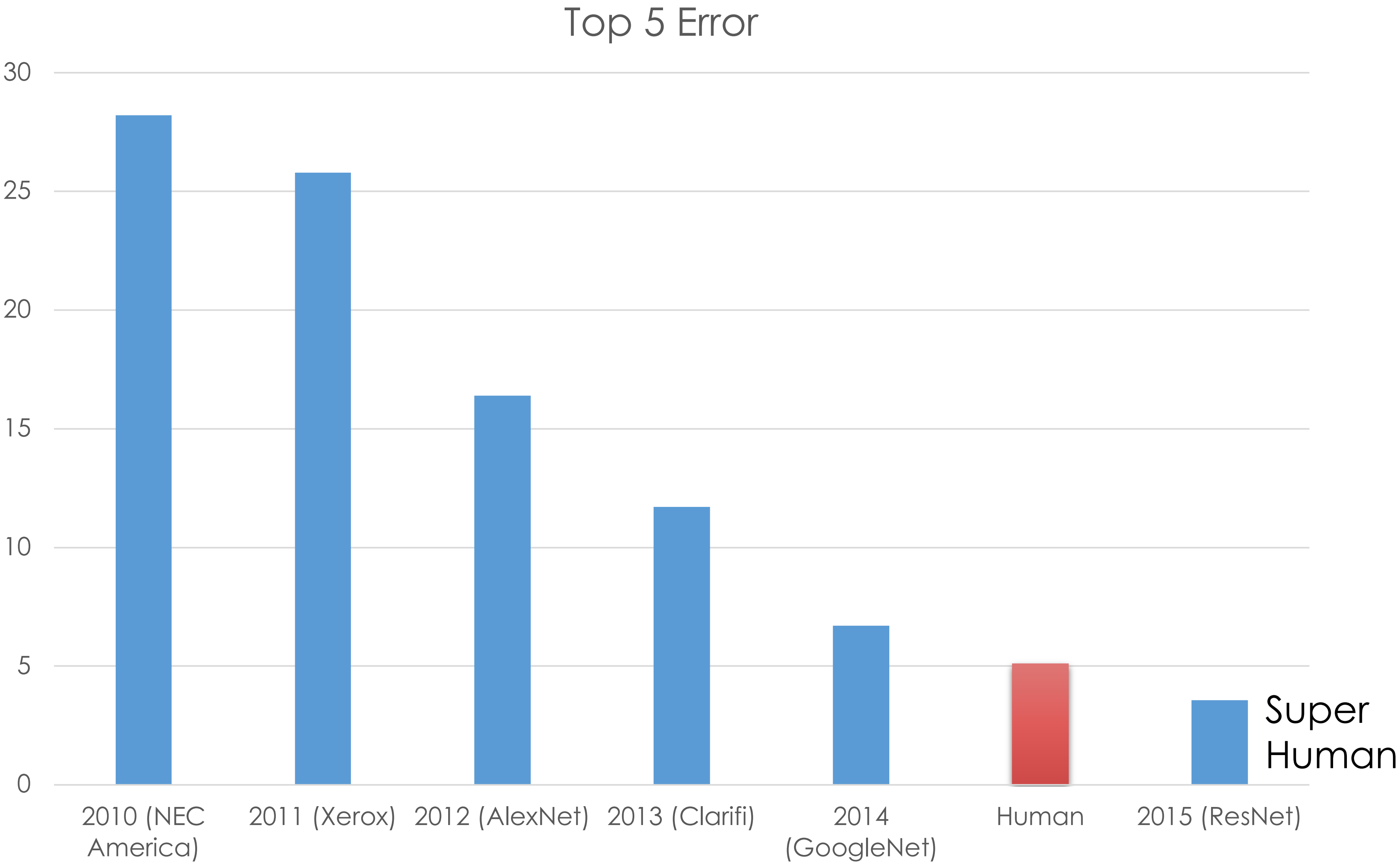


Example: AlexNet (Krizhevsky et al., NIPS 2012)

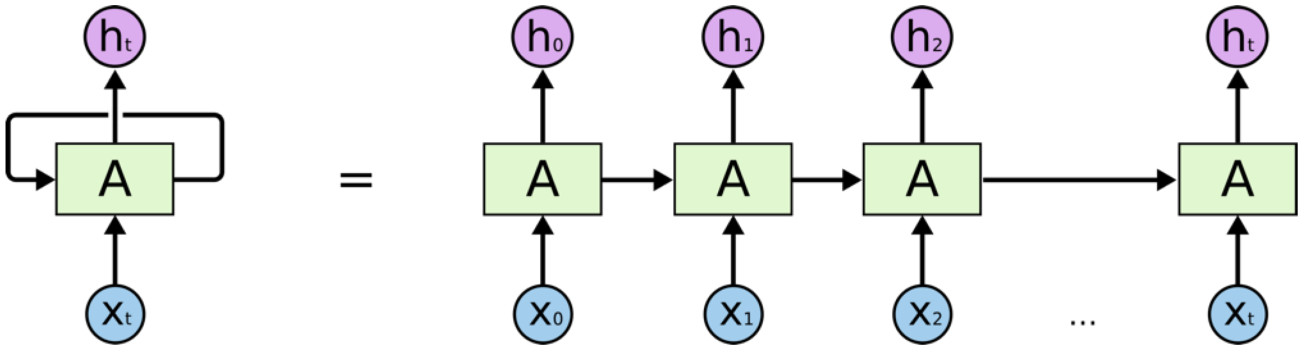


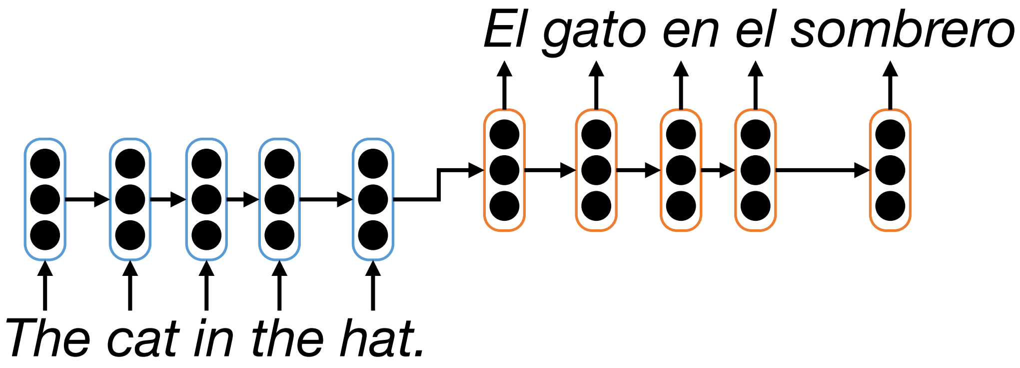
- Introduced in 2012, significantly outperformed state-of-the-art (top 5 error of 16% compared to runner-up with 26% error)

Improvement on ImageNet Benchmark



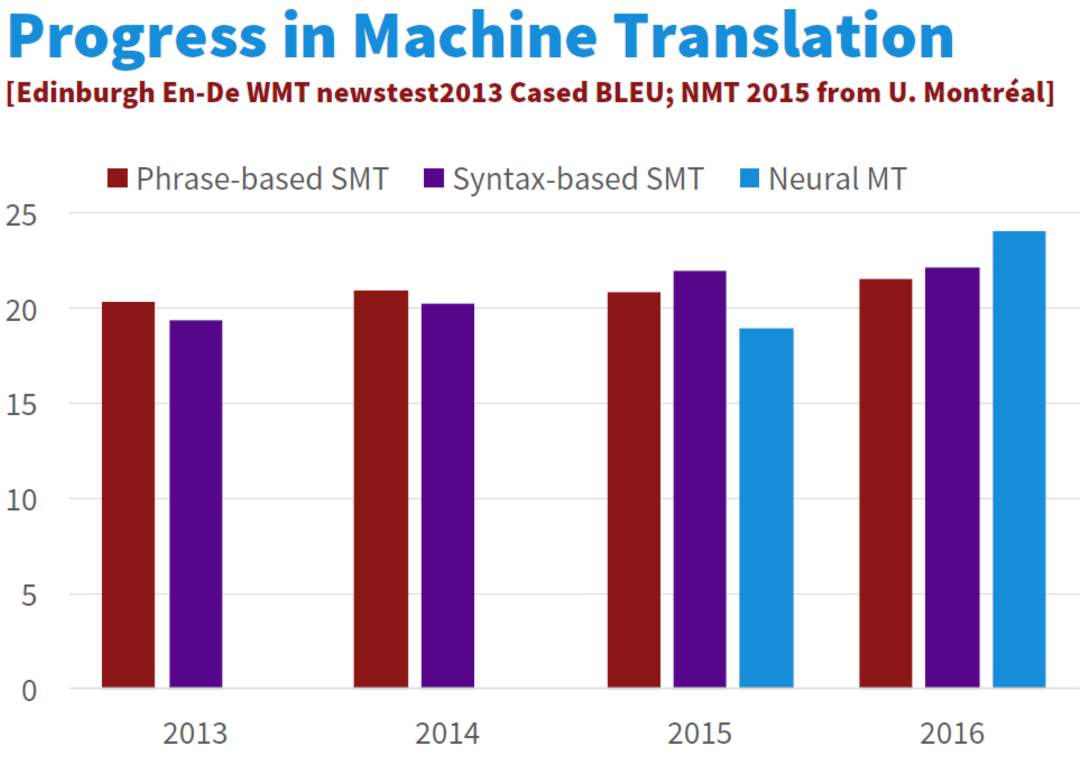
Recurrent Neural Networks: Modeling Sequence Structure



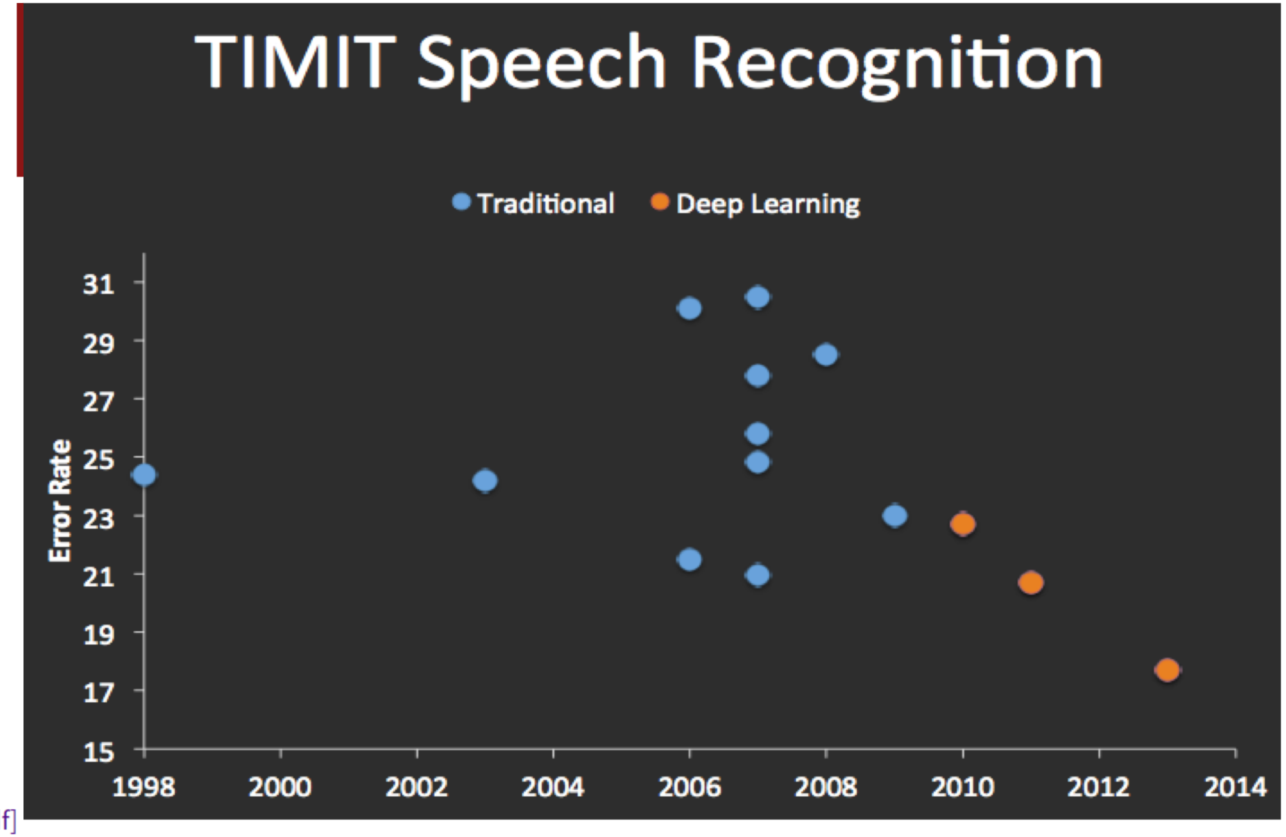


- ➢ input + previous output → new output
- ➢ State of the art in modeling sequential data
  - ➢ speech recognition and machine translation

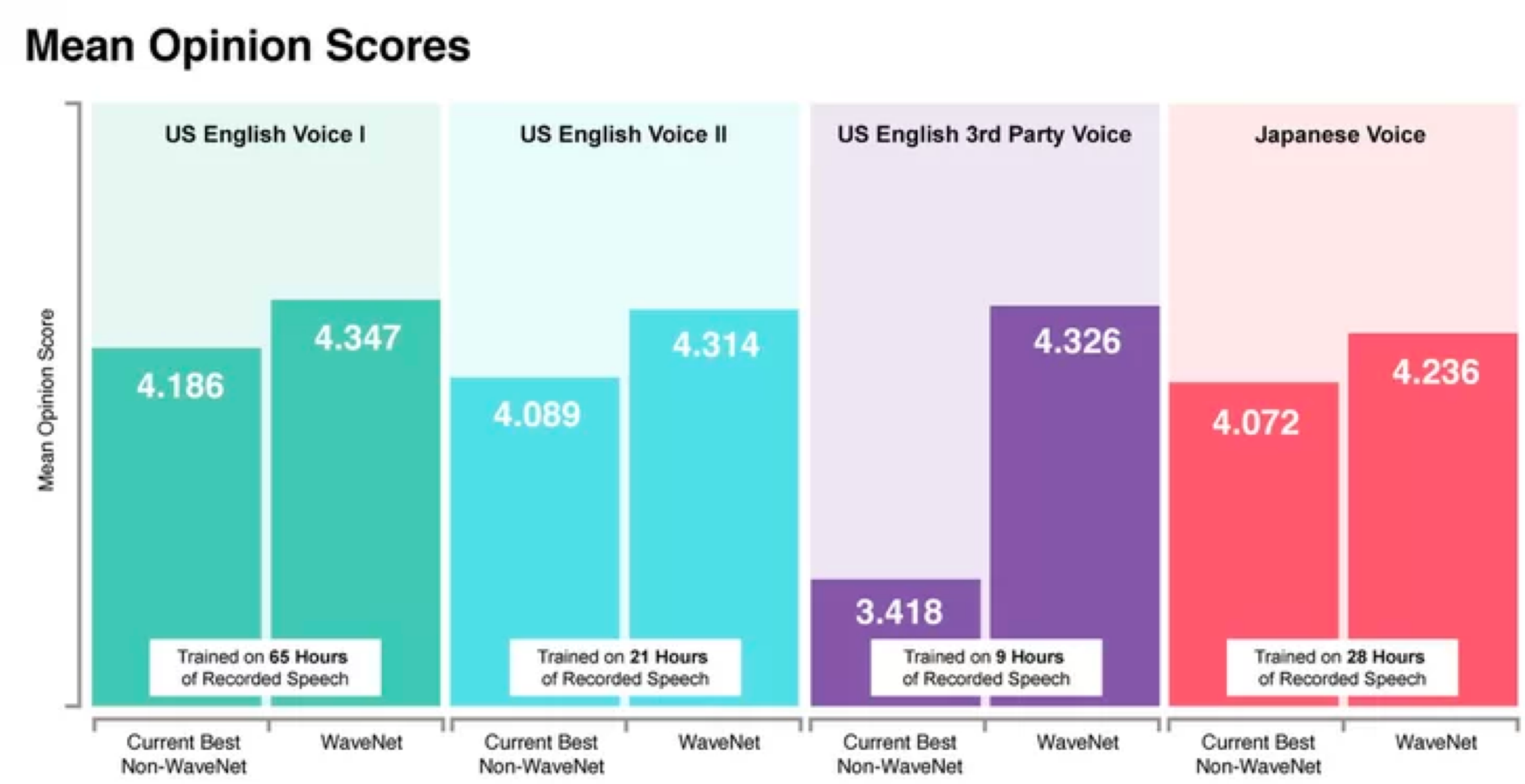
Improvements in Machine Translation & Automatic Speech Recognition



From [Sennrich 2016, http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf]



State of the art in Text to Speech (TTS)



Interested in Deep Learning?

- RISE Lab Deep Learning Overview:
  - https://ucbrise.github.io/cs294-rise-fa16/deep_learning.html
- TensorFlow Python Tutorial
- Stanford CS231 Labs
  - http://cs231n.github.io/linear-classify/
  - http://cs231n.github.io/optimization-1/
  - http://cs231n.github.io/optimization-2/