6.7900 Machine Learning (Fall 2024)

Lecture 8: classification, ranking

(supporting slides)

Predicting discrete outcomes

- Lots of cases where the relevant variable to predict is not inherently "quantitative"
- E.g., is this medication helpful to me?
- E.g., who will win the election?
- E.g., is this compound toxic?
- E.g., which of these authors wrote the article?
- E.g., which grade tumor does this sample correspond to?
- E.g., which colleges are ranked top (ten) this year?

- These are examples of classification, ordinal regression, and ranking problems

Predicting discrete outcomes

- Lots of cases where the relevant variable to predict is not inherently "quantitative"
- E.g., is this medication helpful to me? [binary classification]
- E.g., who will win the election? [binary classification]
- E.g., is this compound toxic? [binary classification]
- E.g., which of these authors wrote the article? [multi-way classification]
- E.g., which grade tumor does this sample correspond to? [ordinal]
- E.g., which colleges are ranked top (ten) this year? [ranking]

- These are examples of classification, ordinal regression, and ranking problems

Self-supervised example: next word prediction

- Many at scale self-supervised tasks for AI systems are also classification tasks, e.g., next word prediction

Your

Your grade

Your grade for

Your grade for this

Your grade for this course

...

- Or learning to uncover masked words, e.g.,

course A

Your grade for this [mask] will be [mask]

Self-supervised example: cross-modal matching

- Similarly, many association tasks are often discrete classification problems
- E.g., you can draw batches of images and corresponding captions and learn to map each image to its corresponding caption (and vice versa)

images

...

captions

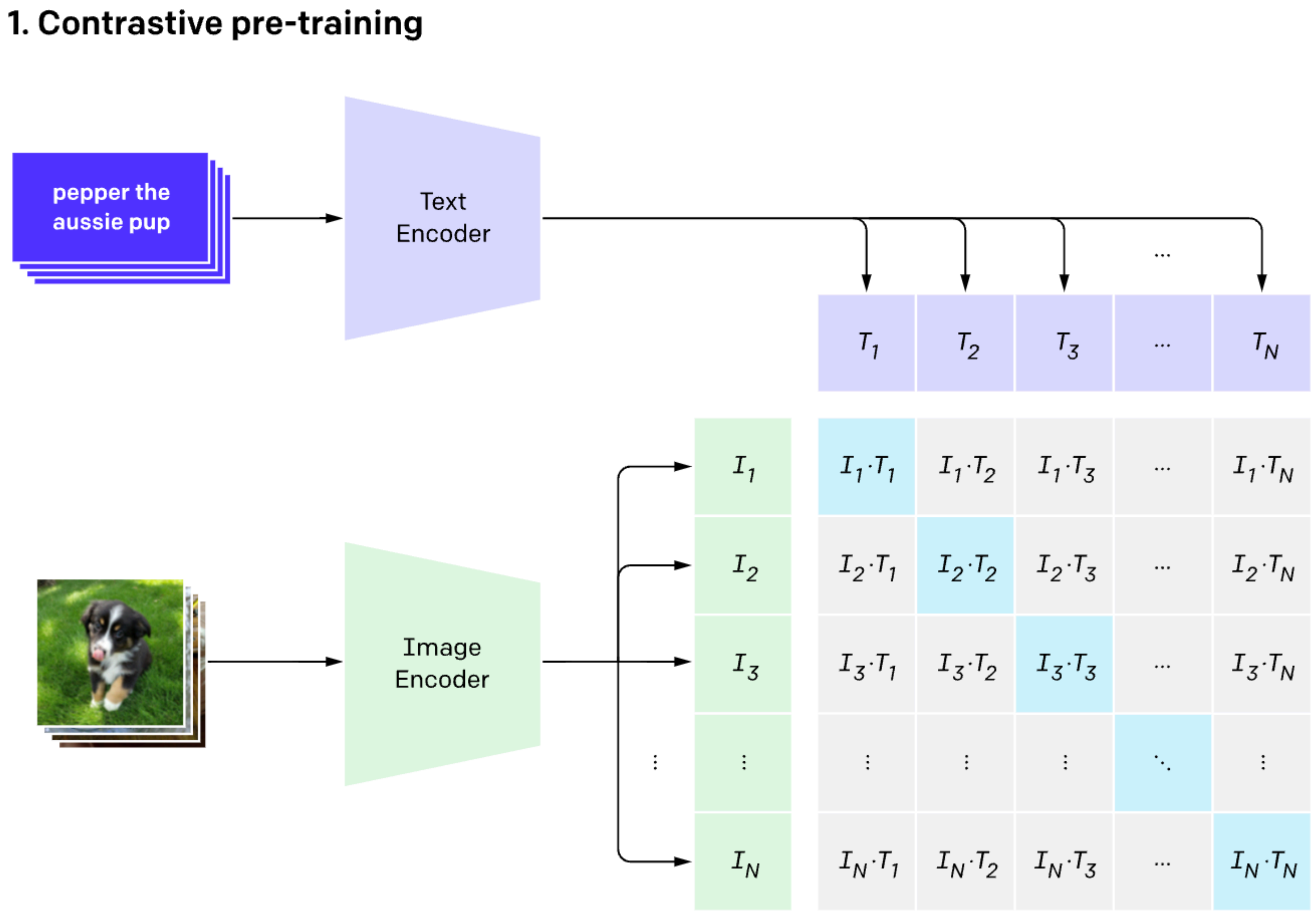
a cat looking down at an insect

...

a black lab waiting for a treat

Self-supervised example: cross-modal matching

- Similarly, many association tasks are often discrete classification problems
- E.g., you can draw batches of images and corresponding captions and learn to map each image to its corresponding caption (or vice versa)




```
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

[Radford et al. 2021]

Classification as regression?

- Why wouldn't we simply solve classification tasks as regression problems?
- E.g., we could imagine directly using {0,1} target values as if they were real numbers and solve the problem as linear (or non-linear) regression

$$\min_{\theta,\theta_0} \sum_{i=1}^{n} (y^i - \theta^T x^i - \theta_0)^2, \quad y^i \in \{0,1\}$$

- How to predict labels for new examples?

Classification as regression?

- Why wouldn't we simply solve classification tasks as regression problems?
- E.g., we could imagine directly using {0,1} target values as if they were real numbers and solve the problem as linear (or non-linear) regression

$$\min_{\theta,\theta_0} \sum_{i=1}^{n} (y^i - \theta^T x^i - \theta_0)^2, \quad y^i \in \{0,1\}$$

- How to predict labels for new examples? What might go wrong?

$$\hat{y} = \mathbb{I}(\theta^T x + \theta_0 \geq 0.5)$$

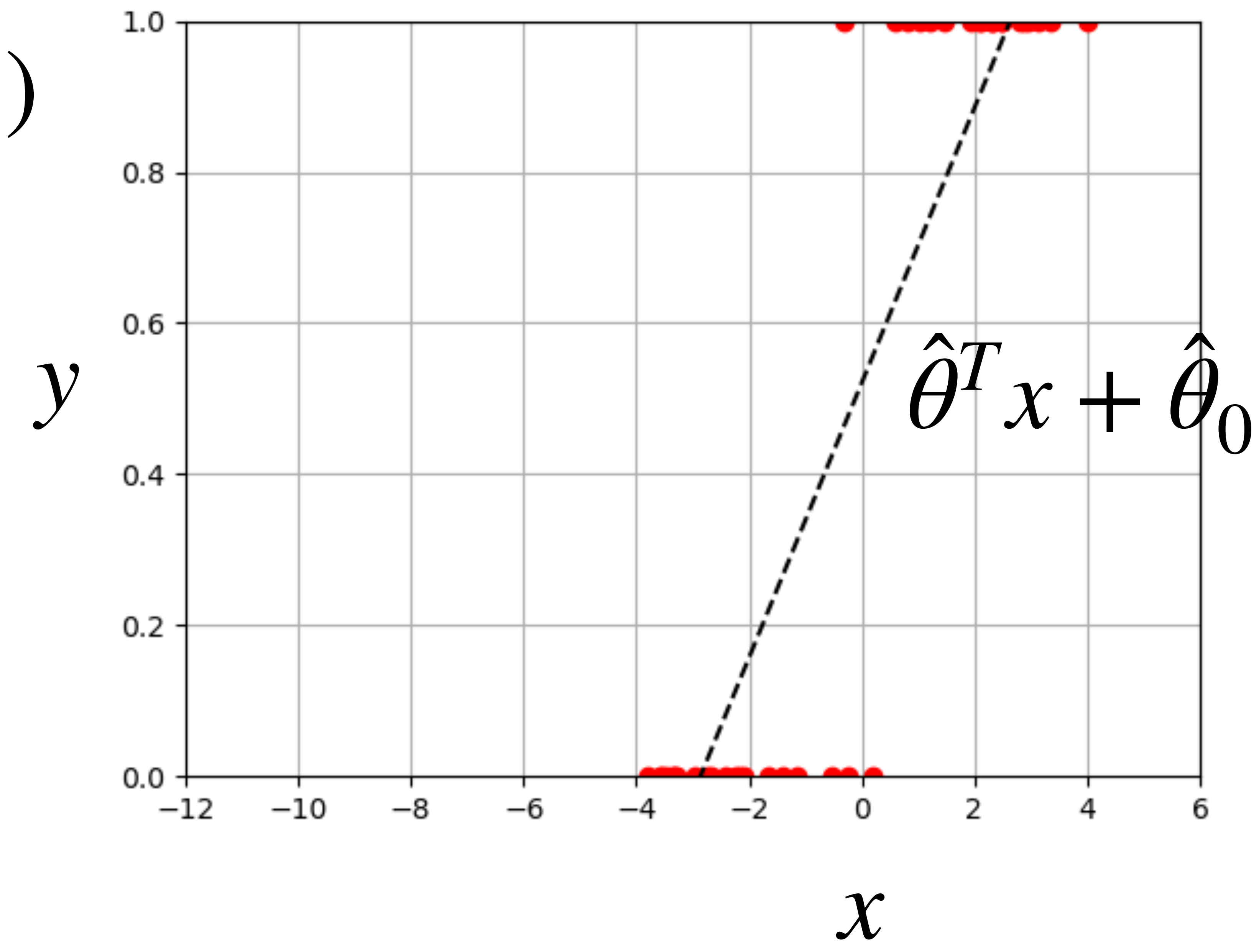
Classification as regression?

- Why wouldn't we simply solve classification tasks as regression problems?
- E.g., we could imagine directly using {0,1} target values as if they were real numbers and solve the problem as linear (or non-linear) regression

$$\min_{\theta,\theta_0} \sum_{i=1}^{n} (y^i - \theta^T x^i - \theta_0)^2, \quad y^i \in \{0,1\}$$

- How to predict labels for new examples? What might go wrong?

$$\hat{y} = \mathbb{I}(\theta^T x + \theta_0 \geq 0.5)$$



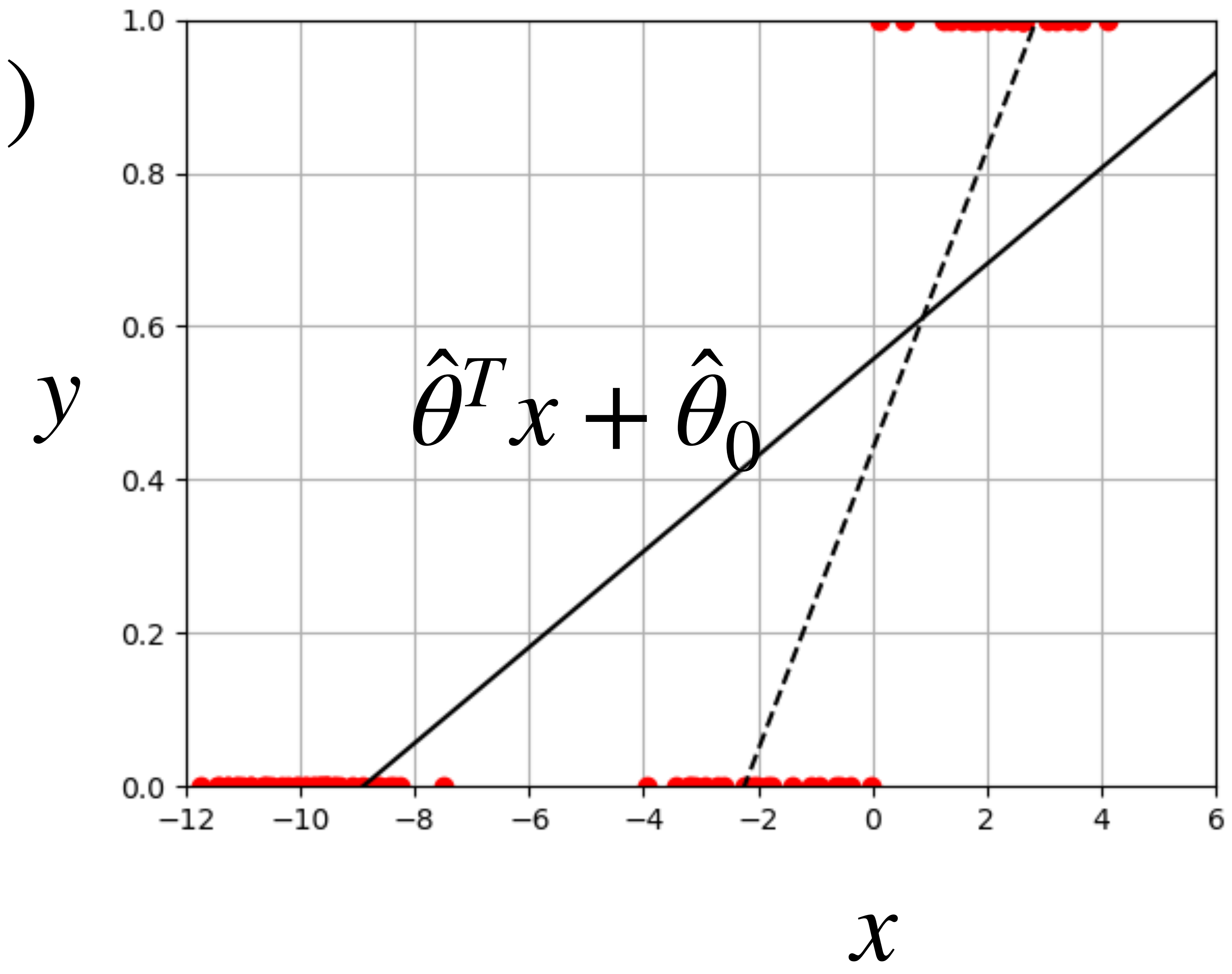
Classification as regression?

- Why wouldn't we simply solve classification tasks as regression problems?
- E.g., we could imagine directly using {0,1} target values as if they were real numbers and solve the problem as linear (or non-linear) regression

$$\min_{\theta,\theta_0} \sum_{i=1}^{n} (y^i - \theta^T x^i - \theta_0)^2, \quad y^i \in \{0,1\}$$

- How to predict labels for new examples? What might go wrong?

$$\hat{y} = \mathbb{I}(\theta^T x + \theta_0 \geq 0.5)$$



A typical binary classification setting

- Training set: $D = \{(x^i, y^i), i = 1,\ldots,N\}$, $(x^i, y^i) \sim P(x,y)$ iid, $x^i \in \mathbb{R}^d$, $y^i \in \{0,1\}$, P unknown
- Test cases: $(x, y) \sim P(x,y)$, iid, same unknown P

<u>Training criterion: $NNL(D;\theta)$</u>

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} -\log P(y^i \,|\, x^i, \theta)$$

<u>Test criterion: 0-1 loss (error rate)</u>

$$E_{(x,y)\sim P}\, \mathbb{I}(P(y \,|\, x, \theta) > 0.5)$$

A typical binary classification setting

- Training set: $D = \{(x^i, y^i), i = 1,\ldots,N\}$, $(x^i, y^i) \sim P(x, y)$ iid, $x^i \in \mathbb{R}^d$, $y^i \in \{0,1\}$, P unknown
- Test cases: $(x, y) \sim P(x, y)$, iid, same unknown P

<u>Training criterion: $NNL(D;\theta)$</u>

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} -\log P(y^i \,|\, x^i, \theta)$$

As $N \to \infty$ the optimal predictor* is the underlying conditional $P(y\,|\,x)$

<u>Test criterion: 0-1 loss (error rate)</u>

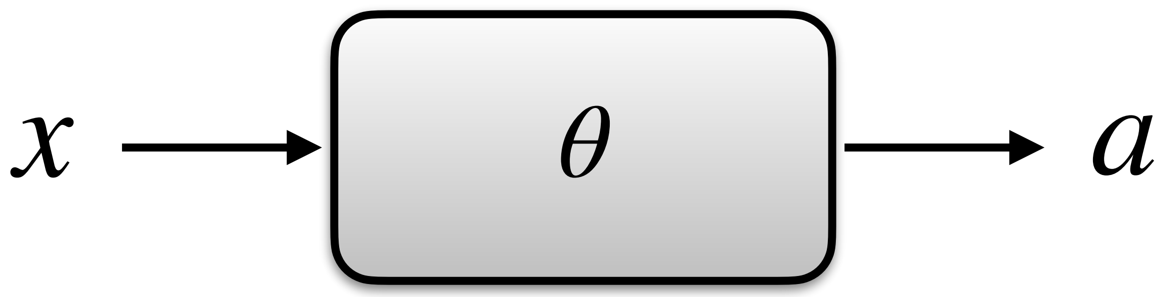
$$E_{(x,y)\sim P}\, \mathbb{I}(P(y\,|\,x, \theta) > 0.5)$$

The min probability of error classifier is the underlying conditional $P(y\,|\,x)$

*assumes consistency (e.g., finite capacity, realizability)

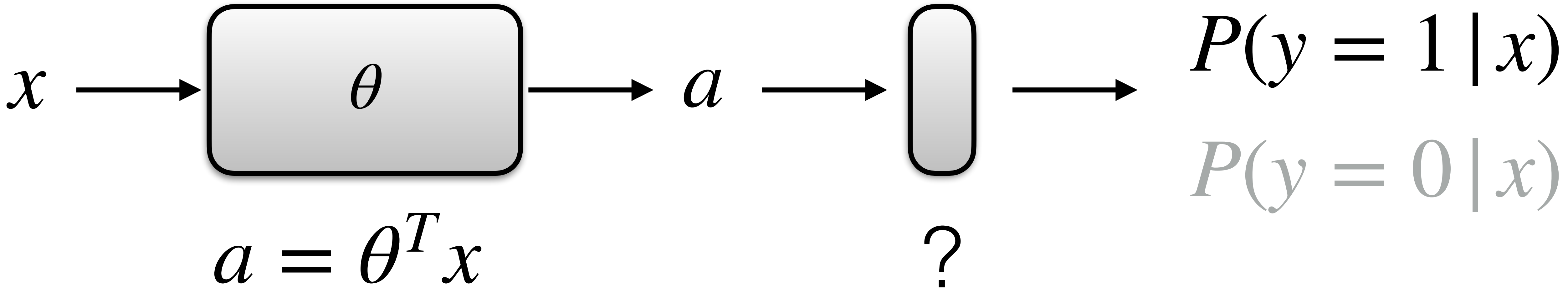
ML models as transformations: classification

- Let's begin with a simple linear model (offset parameter omitted for clarify)



$$a = \theta^T x$$

- Similar to regression case, we look for a statistical model where now the output corresponds to class probabilities



- The remaining question is how to map linear (real valued) predictions to probabilities

Logistic regression

- A natural model for binary outcomes is Bernoulli distribution (coin flip)

$$P(y\,|\,\mu) = Ber(y\,|\,\mu) = \mu^y(1-\mu)^{1-y}, \;\; y \in \{0,1\}, \mu \in [0,1]$$

  where μ specifies how biased the coin is (mean of the binary outcome)
- To turn this into a conditional model, conditioned on covariates x, we need to specify how μ depends on x, i.e., we need to define $\mu(x)$ based on $\theta^T x$

$$a = \theta^T x \in \mathbb{R} \qquad\qquad ? \qquad P(y=1\,|\,x) = \mu(x) \in [0,1]$$

Logistic regression

- A natural model for binary outcomes is Bernoulli distribution (coin flip)

$$P(y\,|\,\mu) = Ber(y\,|\,\mu) = \mu^y(1-\mu)^{1-y}, \;\; y \in \{0,1\}, \mu \in [0,1]$$

  where μ specifies how biased the coin is (mean of the binary outcome)
- To turn this into a conditional model, conditioned on covariates x, we need to specify how μ depends on x, i.e., we need to define $\mu(x)$ based on $\theta^T x$

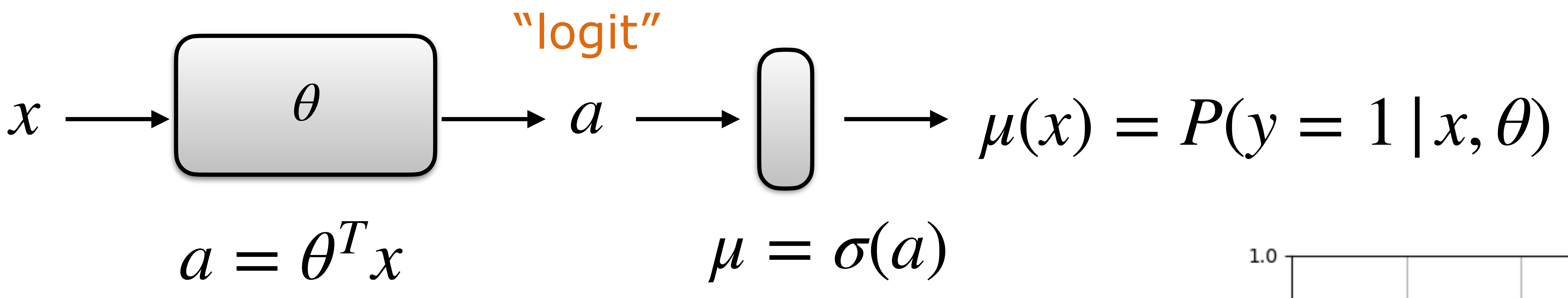
$$a = \theta^T x \in \mathbb{R} \qquad ? \qquad P(y = 1\,|\,x) = \mu(x) \in [0,1]$$

- A common way to link the two is via modeling log-odds ratio as a (linear) function of covariates (logistic regression)

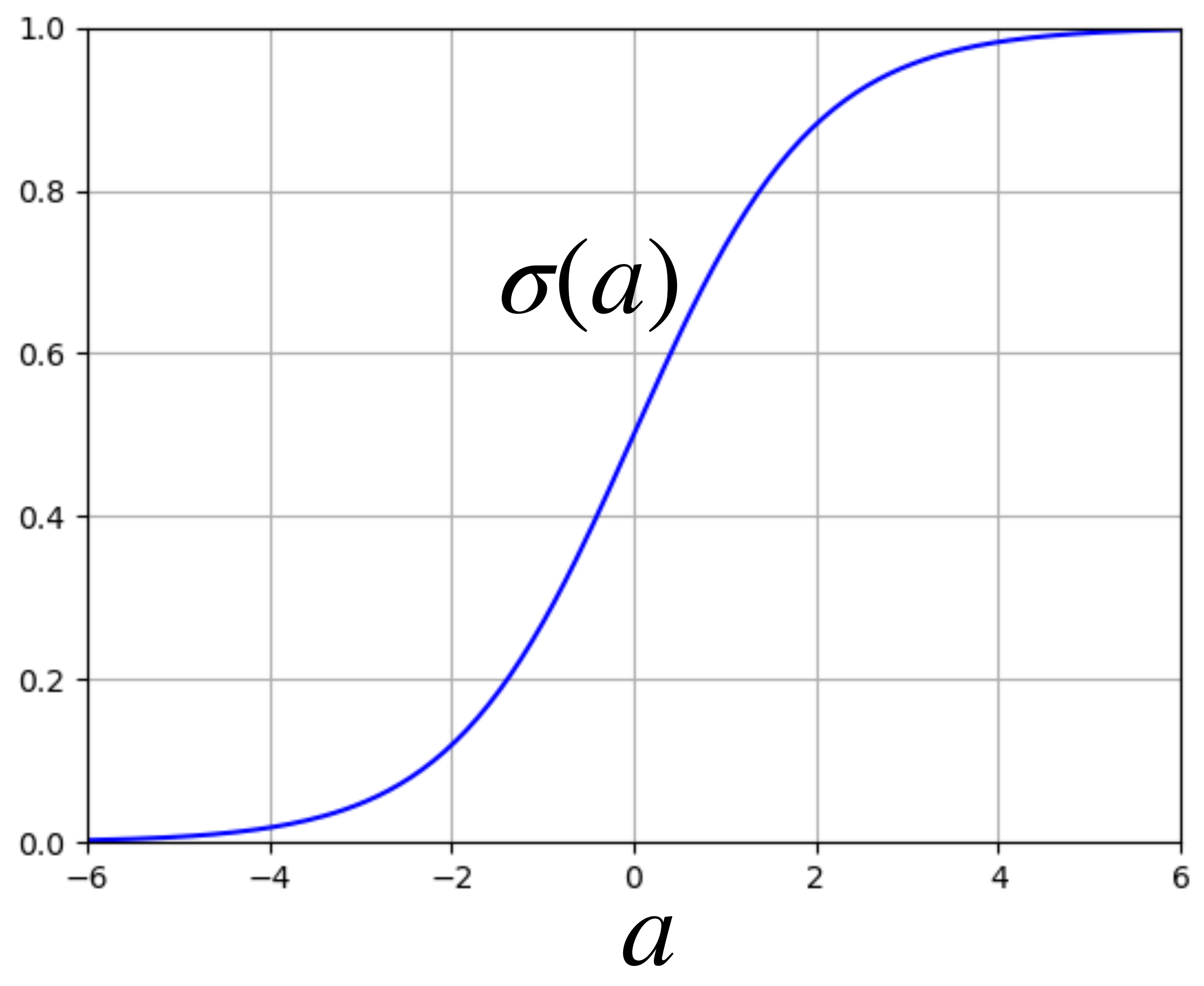
$$\log \frac{\mu(x)}{1-\mu(x)} = \theta^T x \quad \Leftrightarrow \mu(x) = \sigma(\theta^T x), \quad \sigma(a) = \frac{1}{1+\exp(-a)}$$

sigmoid/logistic function

ML models as transformations: logistic regression



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

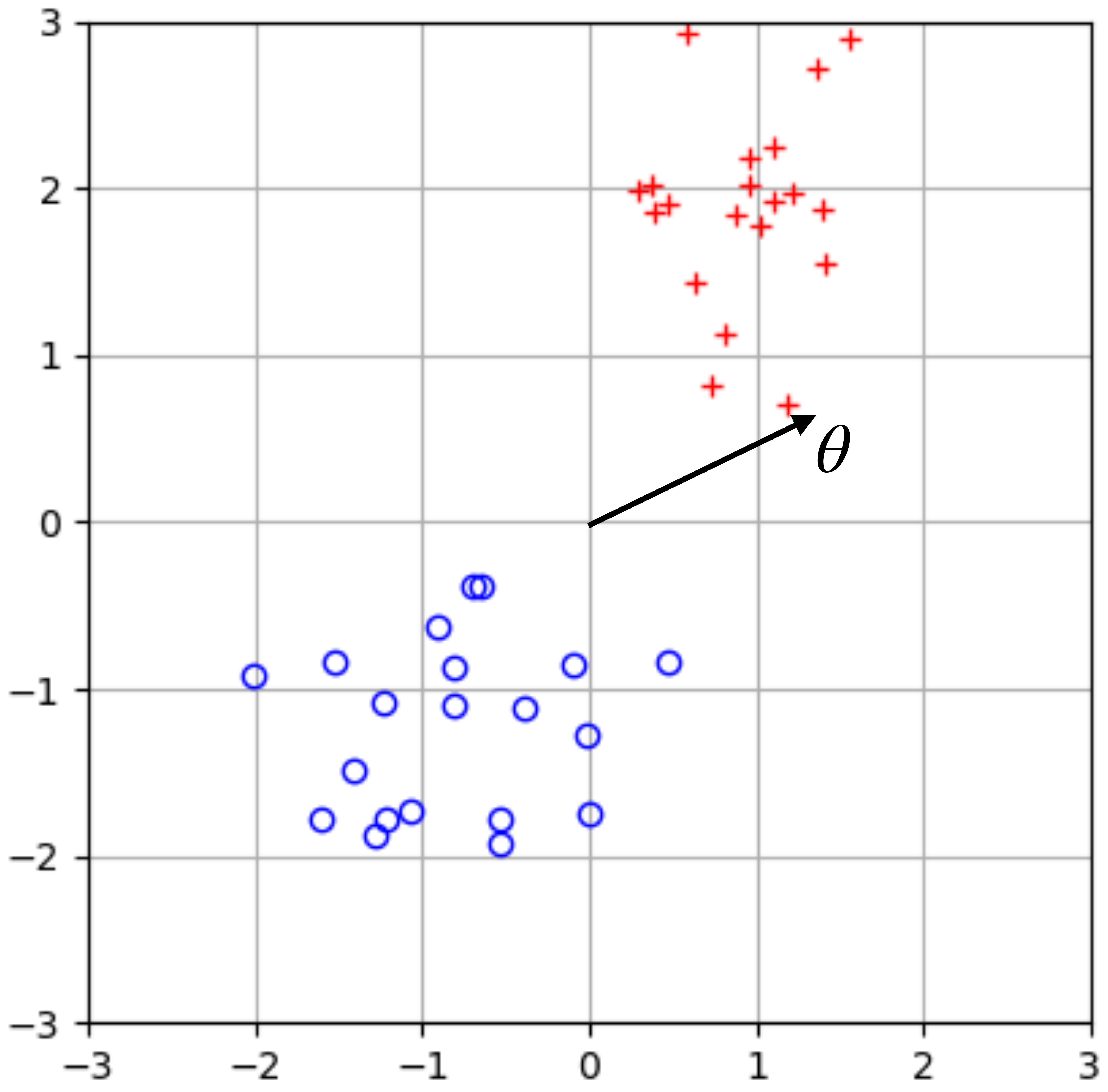


- If we are forced to make a binary {0,1} prediction rather than a probability value, we would predict

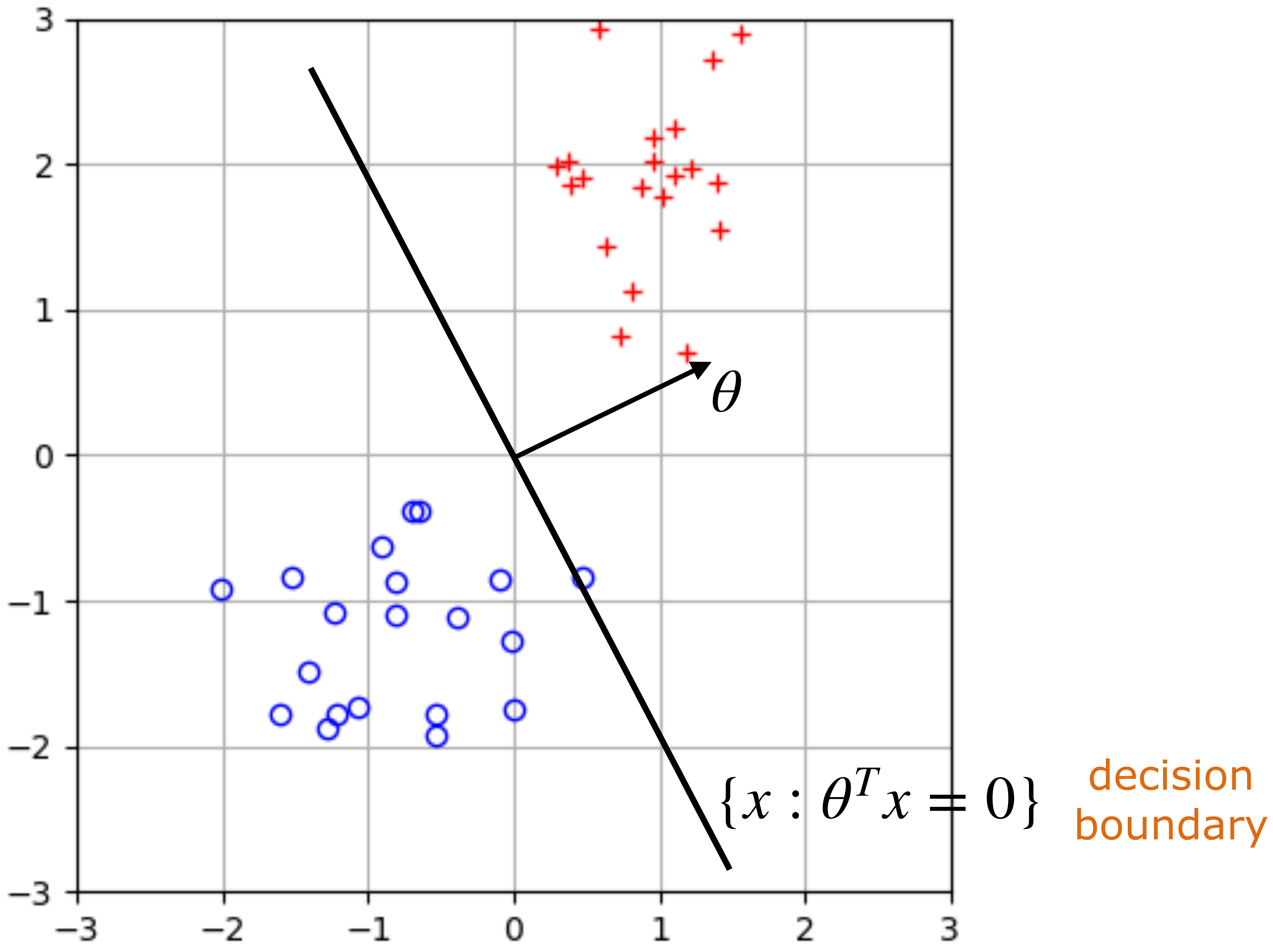
$$\hat{y} = \mathbb{I}(P(y = 1 \mid x, \theta) > 0.5) = \mathbb{I}(\sigma(\theta^T x) > 0.5) = \mathbb{I}(\theta^T x > 0)$$

- why different from the non-statistical proposal?

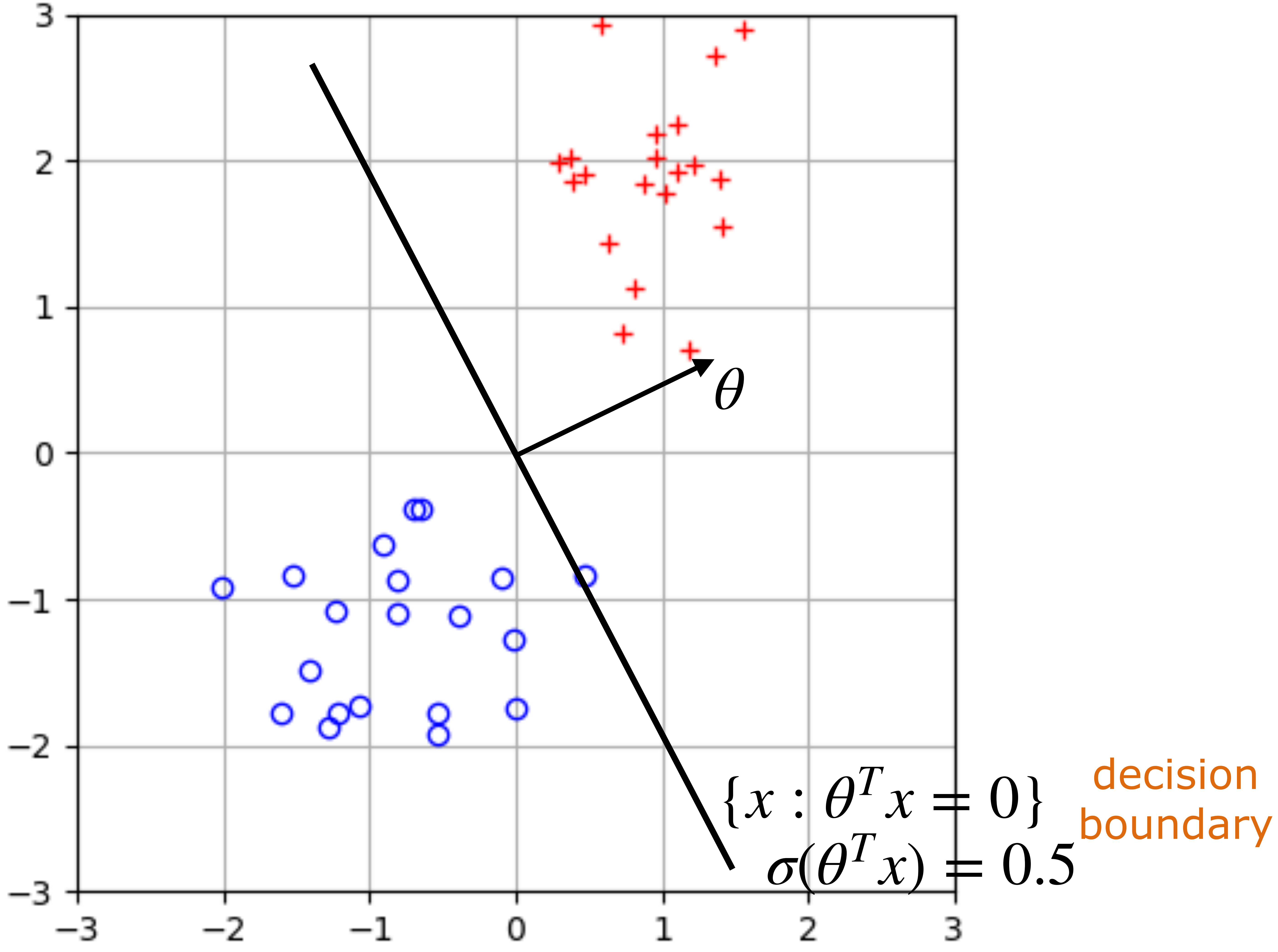
Logistic regression: assumptions



Logistic regression: assumptions

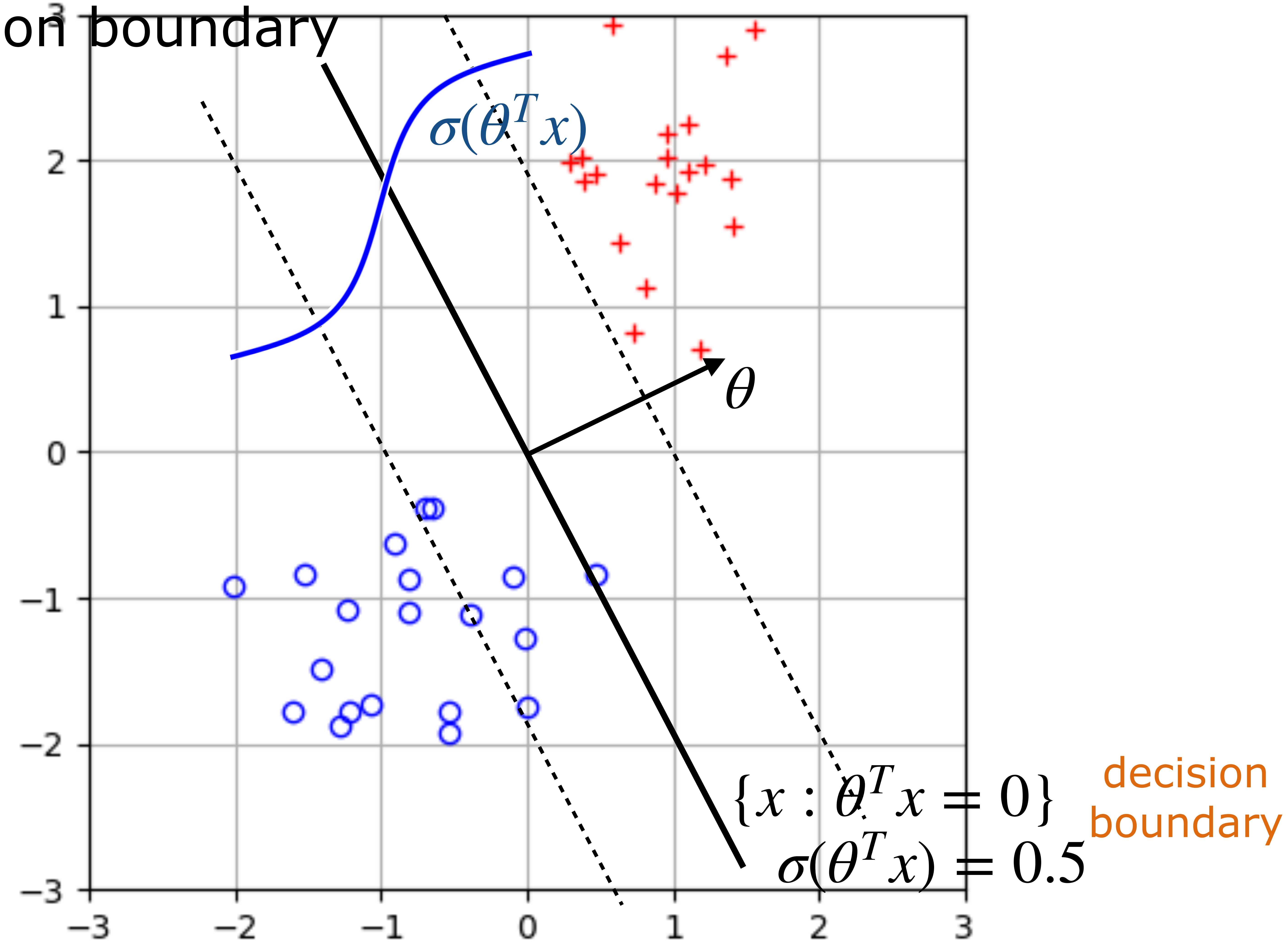


Logistic regression: assumptions



Logistic regression: assumptions

- Assumes a specific functional form for how predicted class probability varies away from the decision boundary



Exercise: class-conditional Gaussians

- Suppose the true underlying class conditional distributions are multi-variate Gaussian $P(x\,|\,y) = N(x\,|\,\mu_y, \Sigma_y)$ with class dependent means and covariances
- Assume also that the classes have prior frequencies $p(y), y \in \{0,1\}$ (in the absence of any knowledge about the covariates)
- Then the conditional $P(y\,|\,x)$ can be represented by a logistic regression model

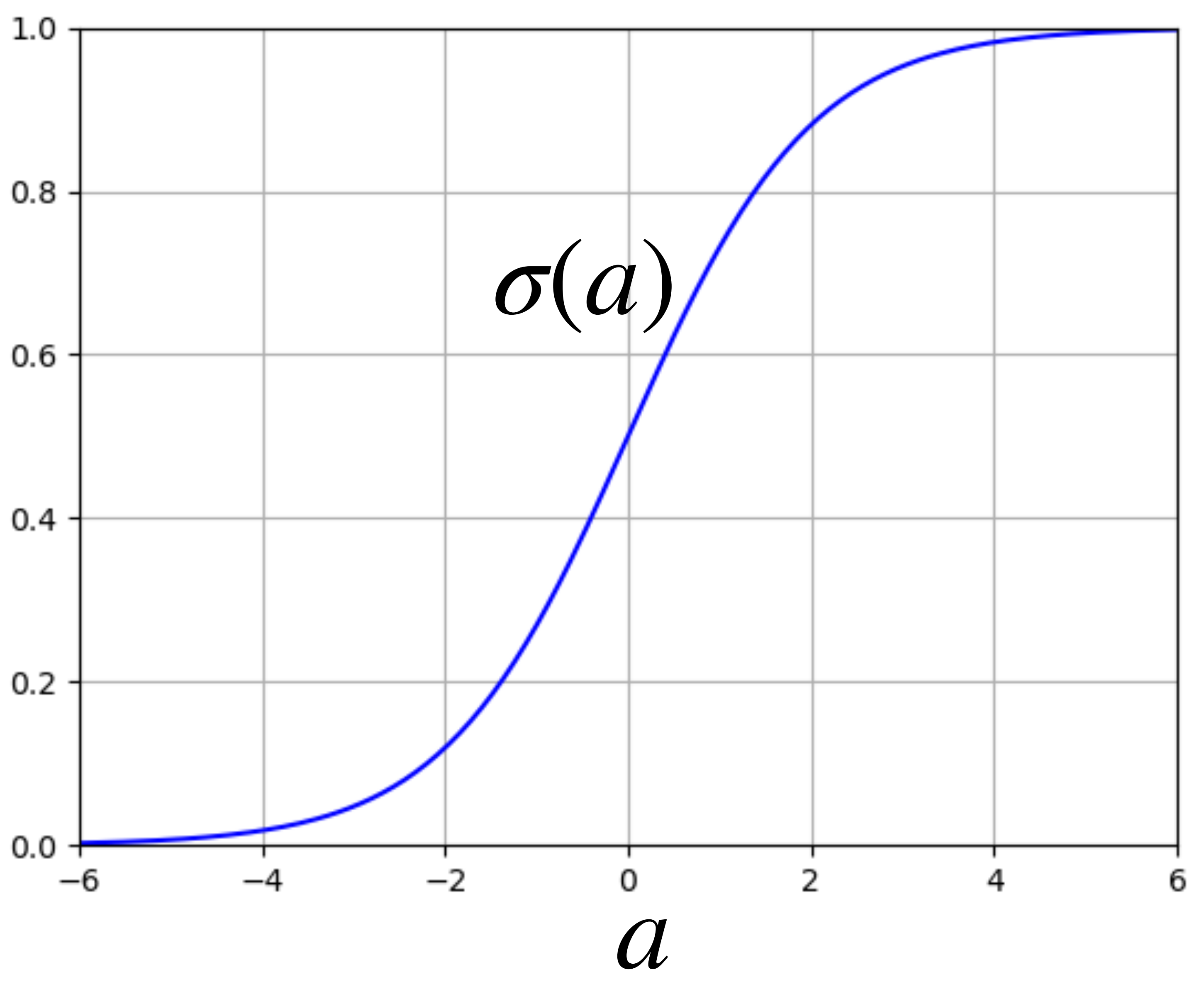
$$P(y\,|\,x) = \frac{p(y)P(x\,|\,y)}{\sum_{y'=0,1} p(y')P(x\,|\,y')} = \frac{p(y)N(x\,|\,\mu_y, \Sigma_y)}{\sum_{y'=0,1} p(y')N(x\,|\,\mu_{y'}, \Sigma_{y'})} = \sigma\Big(\overset{?}{\ldots} \Big)$$

where the argument is either a) a linear expression in x when $\Sigma_1 = \Sigma_2$, b) requires 2nd order polynomial features when $\Sigma_1 \neq \Sigma_2$

Properties

$$\sigma(a) = \frac{1}{1+\exp(-a)} \quad \Leftrightarrow \quad \log \frac{\sigma(a)}{1-\sigma(a)} = a$$

$1 - \sigma(a) = \sigma(-a)$ symmetric tails $\quad \sigma'(a) = \sigma(a)(1-\sigma(a))$

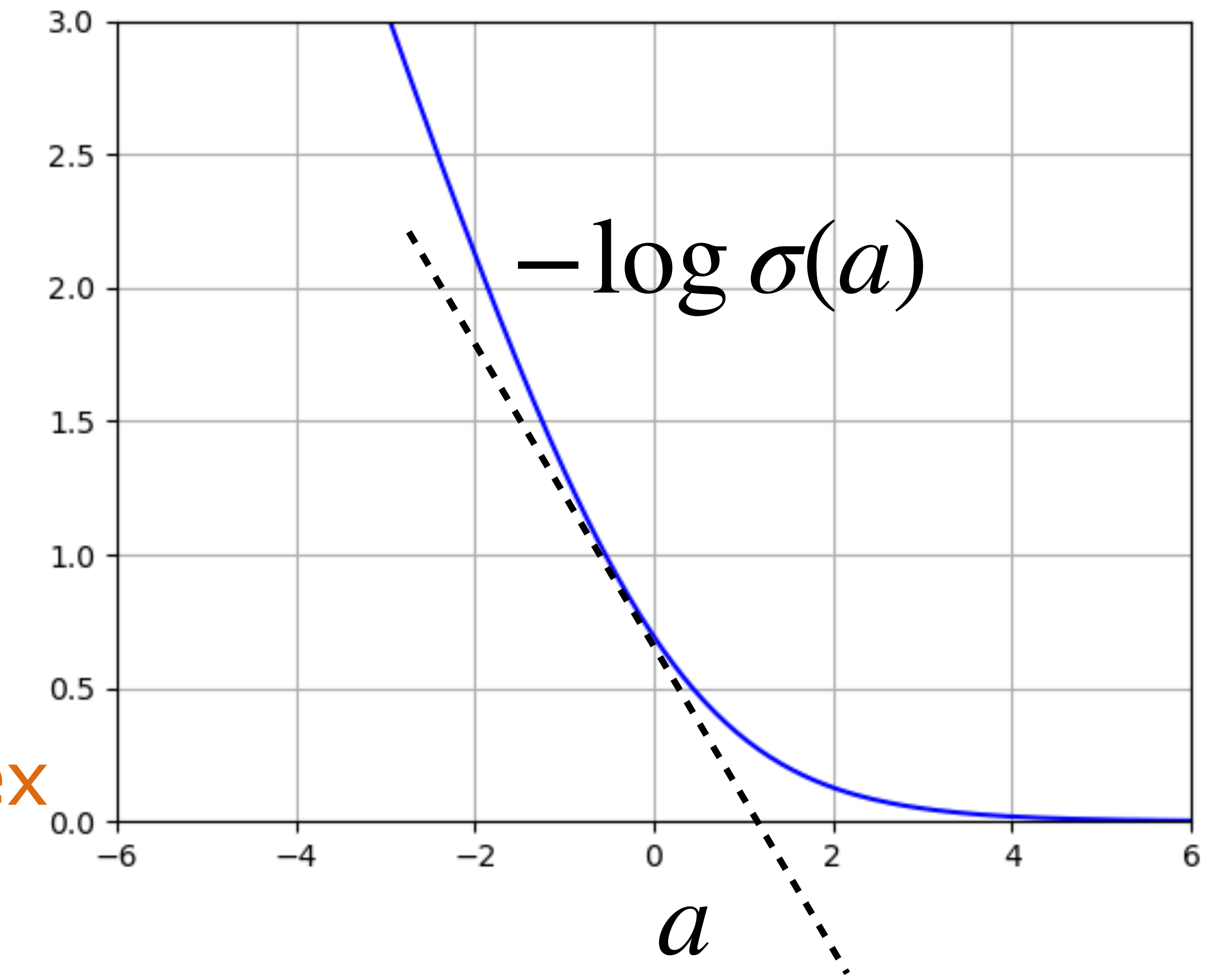


$-\log \sigma(a) = \log(1+\exp(-a))$ strictly convex in a $a \in (-\infty, \infty)$

$-\log \sigma(\theta^T x)$ convex in θ (not strictly convex)

$c(z) \geq (z-z_0)^T c'(z_0) + c(z_0), \;\; \forall z, z_0$ convex

$c(z) > (z-z_0)^T c'(z_0) + c(z_0), \;\; \forall z \neq z_0$ strictly convex



Estimation: neg-log-likelihood

- Training data $D = \{(x^i, y^i), i = 1,\ldots,N\}$
- Average negative log-likelihood of data

$$NLL(D;\theta) = \frac{1}{N}\left(-\log\left[\prod_{i=1}^{N} P(y^i \,|\, x^i, \theta)\right]\right) = \frac{1}{N}\left(-\log\left[\prod_{i=1}^{N} \sigma(\theta^T x^i)^{y^i}(1-\sigma(\theta^T x^i))^{1-y^i}\right]\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\underbrace{\left[-y^i \log\sigma(\theta^T x^i) - (1-y^i)\log(1-\sigma(\theta^T x^i))\right]}_{\text{cross-entropy loss (convex in } \theta)} = \frac{1}{N}\sum_{i=1}^{N} H_{ce}(y^i, \sigma(\theta^T x^i))$$

$$H_{ce}(p,q) = -p\log q - (1-p)\log(1-q)$$

Estimation: neg-log-likelihood

- Training data $D = \{(x^i, y^i), i = 1,\dots,N\}$
- Average negative log-likelihood of data

$$NLL(D;\theta) = \frac{1}{N}\left(-\log\left[\prod_{i=1}^{N} P(y^i|x^i,\theta)\right]\right) = \frac{1}{N}\left(-\log\left[\prod_{i=1}^{N}\sigma(\theta^T x^i)^{y^i}(1-\sigma(\theta^T x^i))^{1-y^i}\right]\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\underbrace{-y^i\log\sigma(\theta^T x^i) - (1-y^i)\log(1-\sigma(\theta^T x^i))}_{\text{cross-entropy loss (convex in }\theta)}\right] = \frac{1}{N}\sum_{i=1}^{N} H_{ce}(y^i, \sigma(\theta^T x^i))$$

$$H_{ce}(p,q) = -p\log q - (1-p)\log(1-q)$$

$$\nabla_\theta NLL(D;\theta) = \dots = -\frac{1}{N}\sum_{i=1}^{N}\underbrace{(y^i - \sigma(\theta^T x^i))}_{\text{prediction error}}x^i$$

Estimation: stochastic gradient descent

- **Stochastic gradient descent (SGD)**
- initialize θ (e.g., zero vector or randomize)
- Repeat (step t = 1,2,…),

  sample example (x^i, y^i) at random

  calculate loss gradient $g_t = \nabla_\theta H_{ce}(y^i, \sigma(\theta^T x^i)) = -(y^i - \sigma(\theta^T x^i))x^i \quad \in \mathbb{R}^d$

  update $\theta \leftarrow \theta - \eta_t g_t$ $\quad \eta_t$ is a scalar learning rate
- Until a stopping criterion (many options) is reached

Estimation: stochastic gradient descent

- **Stochastic gradient descent (SGD)**
- initialize θ (e.g., zero vector or randomize)
- Repeat (step t = 1,2,…),

  sample example (x^i, y^i) at random

  calculate loss gradient $g_t = \nabla_\theta H_{ce}(y^i, \sigma(\theta^T x^i)) = -(y^i - \sigma(\theta^T x^i))x^i \quad \in \mathbb{R}^d$

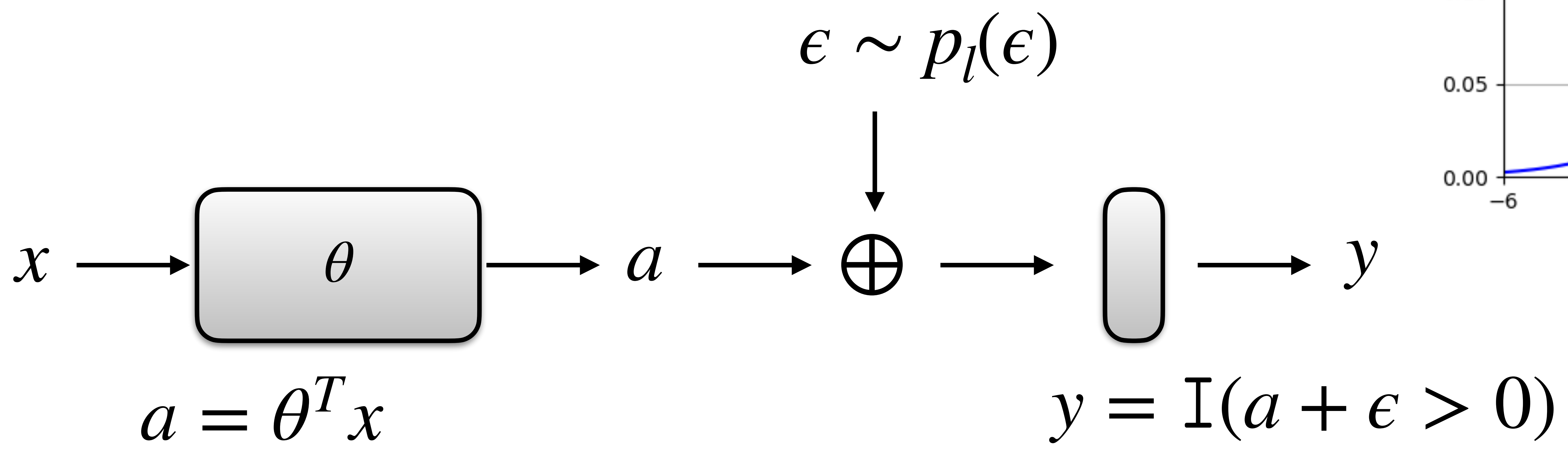
  update $\theta \leftarrow \theta - \eta_t g_t$ η_t is a scalar learning rate
- Until a stopping criterion (many options) is reached

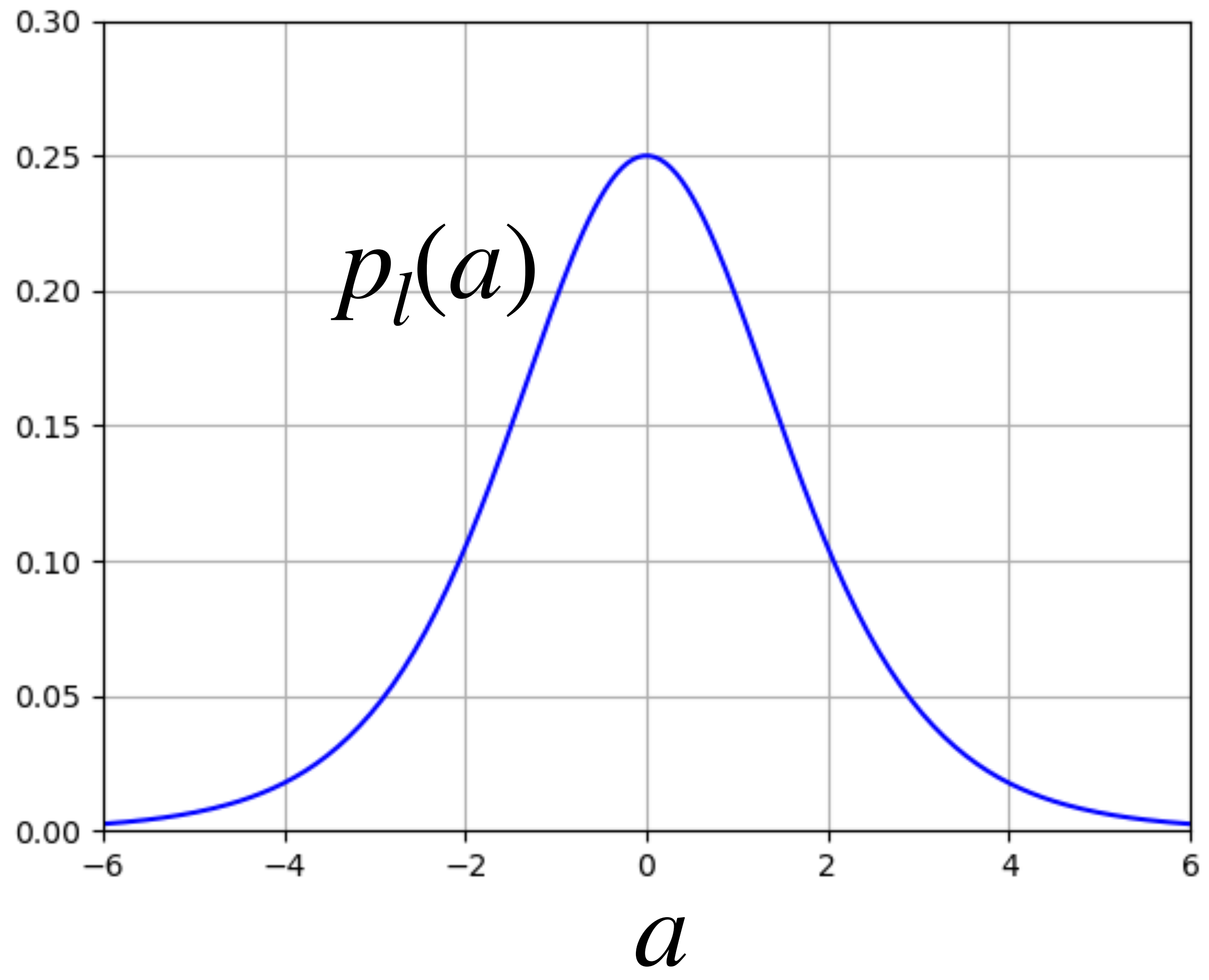
- **Convergence "theorem"** (generic, one of many): if per example losses are convex with Lipschitz continuous gradients, finite variance, then SGD with $\eta_t = c/\sqrt{t}$ converges at rate $O(1/\sqrt{t})$ where t is the number of updates

Equivalent implicit construction

$$\frac{d}{da}\sigma(a) = \sigma'(a) = \sigma(a)\sigma(-a) \quad \equiv p_l(a)$$ (logistic density)

$$\sigma(a) = \int_{-\infty}^{a} p_l(s)ds$$ (logistic function as a cdf)

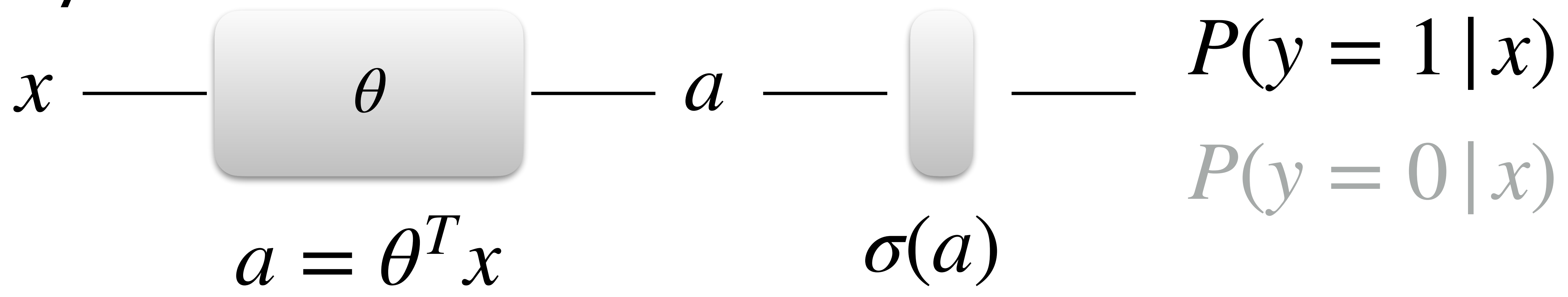




- (if the noise is replaced by N(0,1) we get a probit regression model where the logistic function is replaced by Gaussian cdf)

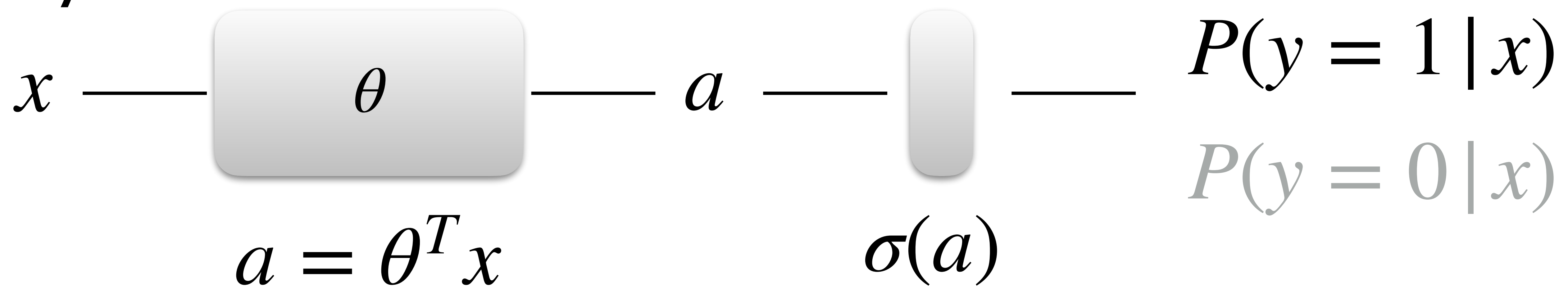
Multi-way classification: softmax regression

- Recall the binary classification formulation: single linear prediction is mapped to a single probability value

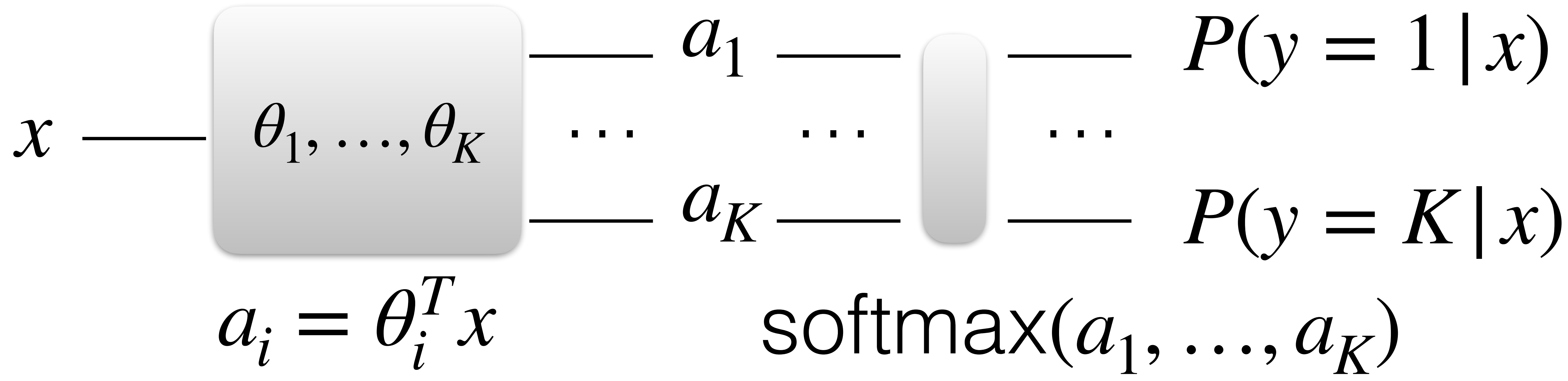


Multi-way classification: softmax regression

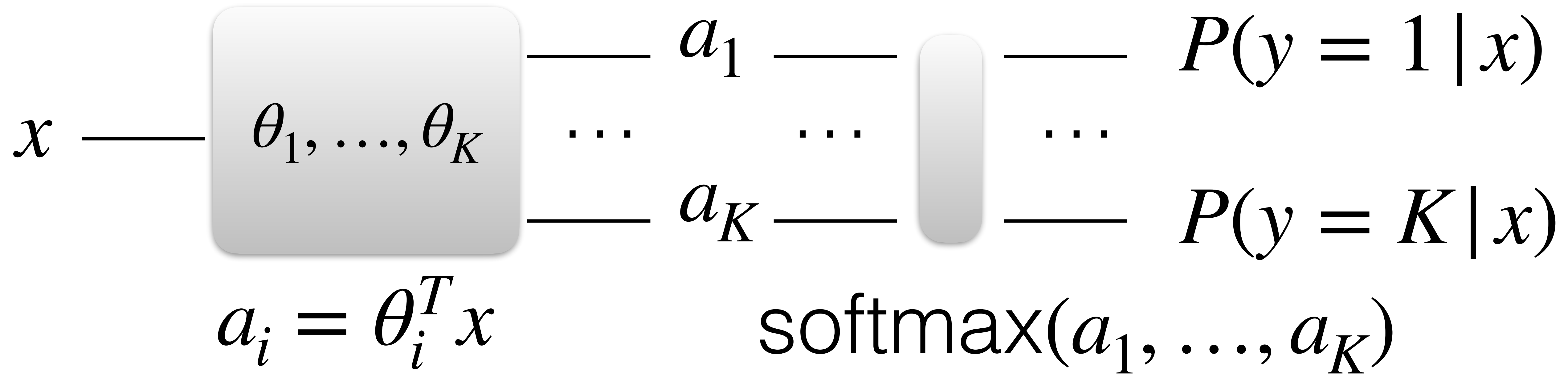
- Recall the binary classification formulation: single linear prediction is mapped to a single probability value



- We just generalize this to multiple outputs



Multi-way classification: softmax regression



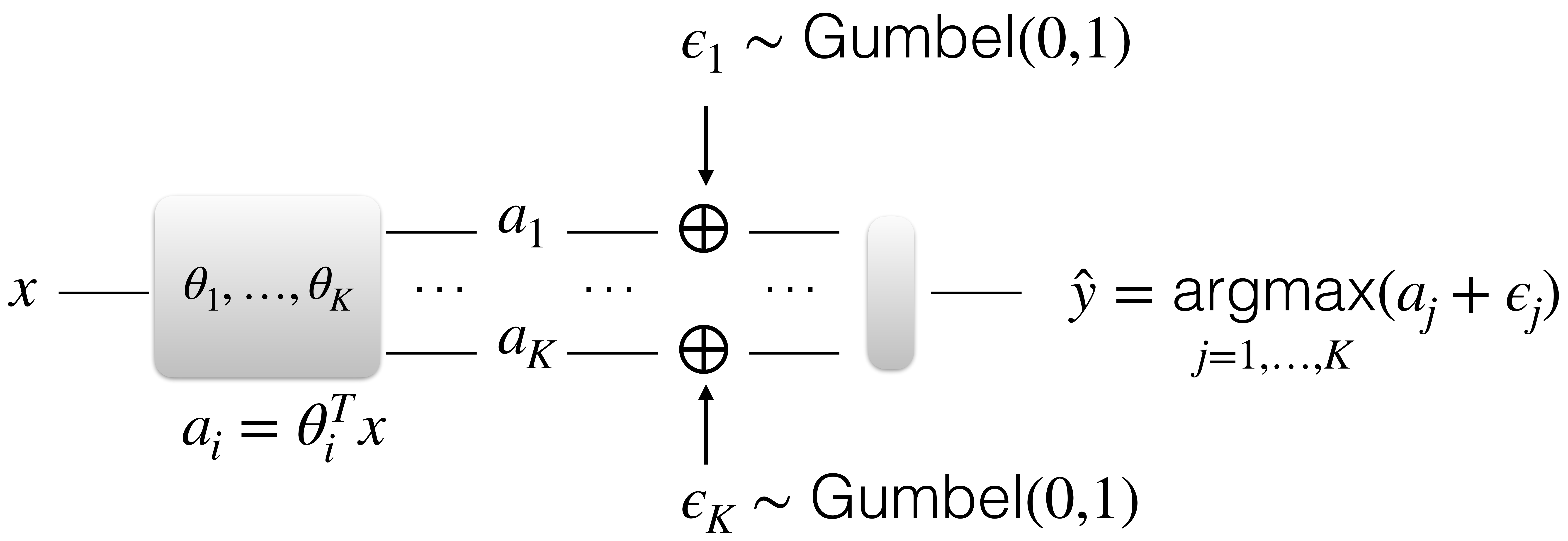
- The link between probabilities and linear predictions is again based on log-ratios of probabilities

$$\log \frac{P(y = i \mid x)}{P(y = j \mid x)} = \theta_i^T x - \theta_j^T x \quad \Leftrightarrow \quad P(y = i \mid x) = \frac{\exp(\theta_i^T x)}{\sum_{l=1}^{K} \exp(\theta_l^T x)}$$

$$= \text{softmax}(i \mid \theta_1^T x, \ldots, \theta_K^T x)$$

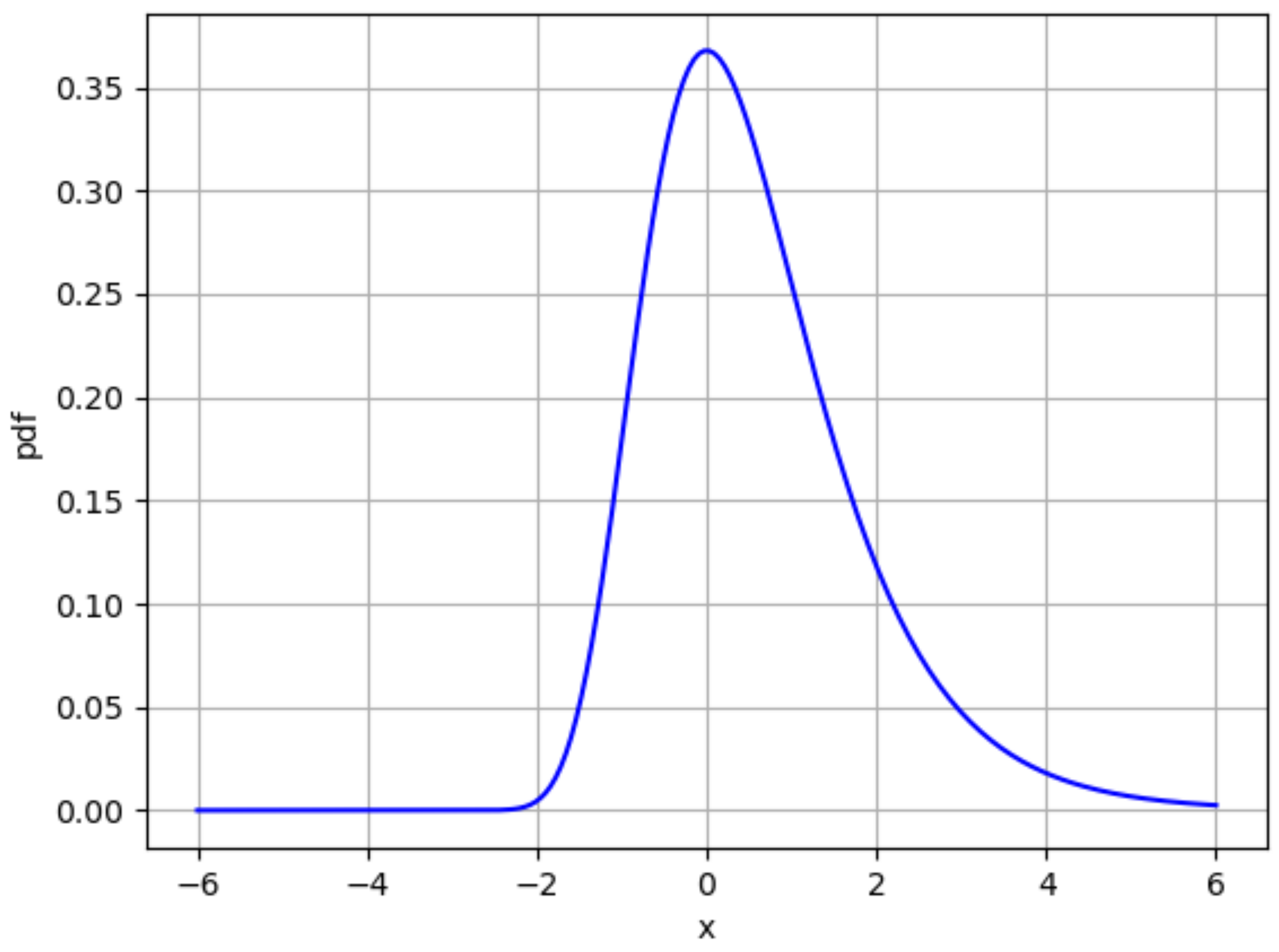
$$\nabla_{\theta_l}[-\log P(y \mid x, \theta)] = -\underbrace{(\delta_{y,l} - P(y = l \mid x, \theta))}_{\text{prediction error}}\, x$$

Equivalent implicit construction



$$P(\hat{y} = i \,|\, x, \theta) = \text{softmax}(i \,|\, \theta_1^T x, \ldots, \theta_K^T x)$$

$$\text{Gumbel}(x \,|\, 0,1) = \exp(-x - \exp(-x))$$

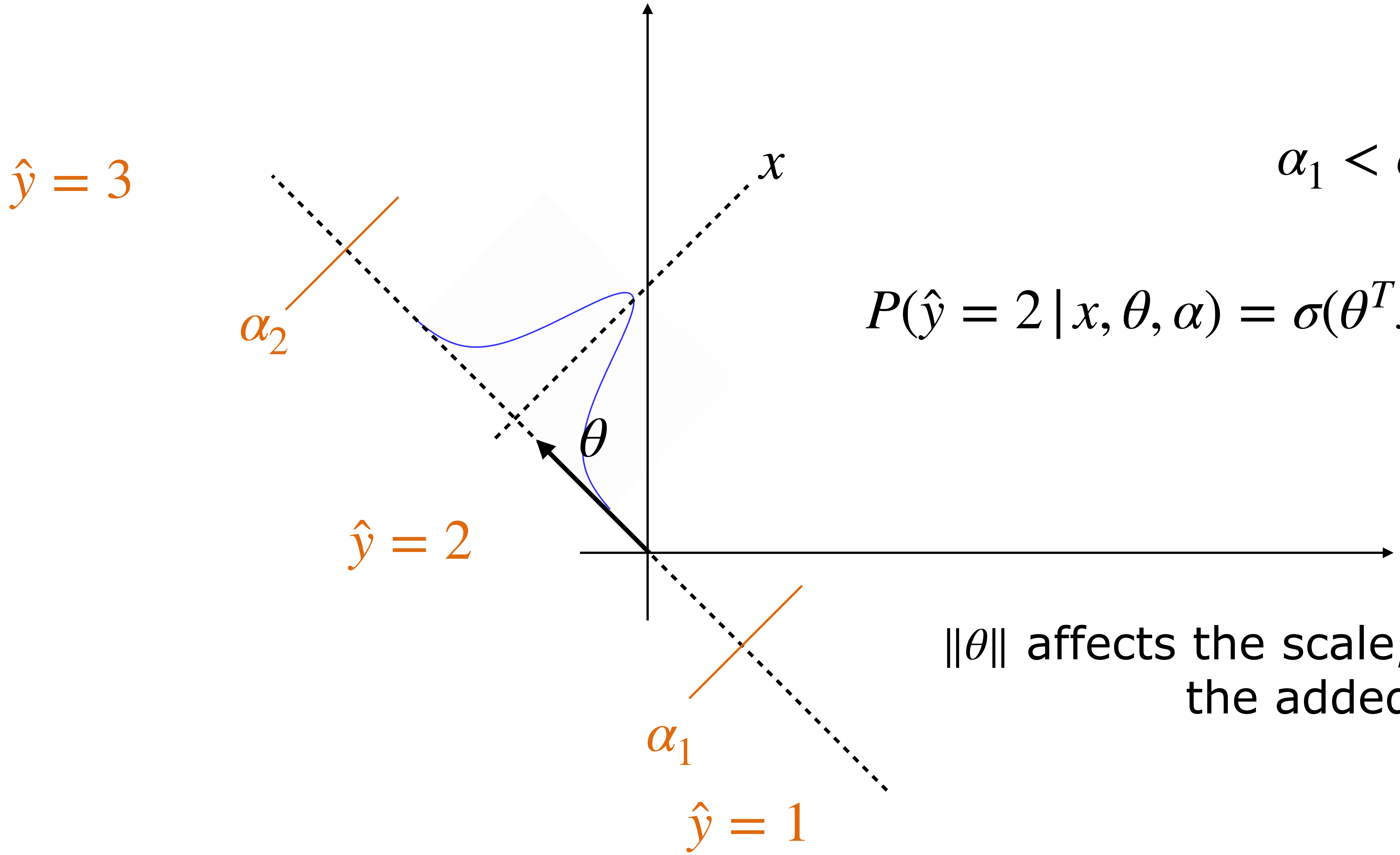


Ordinal regression

- The labels we predict may not be all just symbols but actually have a natural ordering
- E.g., predicting tumor category (from benign to adverse)
- E.g., predicting rating value for content (from 1 to 5 stars)
- etc.
- How should we setup the statistical model in order to capture this?

Ordinal regression model example

- For simplicity, let's assume in the figure that $\|\theta\| = 1$ so we can interpret $\theta^T x$ as the (length of the) orthogonal projection of x in the direction θ



$$\alpha_1 < \alpha_2 < \dots$$

$$P(\hat{y} = 2 \,|\, x, \theta, \alpha) = \sigma(\theta^T x - \alpha_1) - \sigma(\theta^T x - \alpha_2)$$

$\|\theta\|$ affects the scale, effectively controlling the added noise level

Predicting ranked lists

- Instead of classifying into K categories, we could predict the ordering of K alternatives (e.g., ranking of teams in a tournament)
- Let $(\pi(1), \ldots, \pi(K))$ be a particular ordering of $(1, \ldots, K)$ and x define the context for our prediction.

Predicting ranked lists

- Instead of classifying into K categories, we could predict the ordering of K alternatives (e.g., ranking of teams in a tournament)
- Let $(\pi(1), \ldots, \pi(K))$ be a particular ordering of $(1, \ldots, K)$ and x define the context for our prediction.
- We can adapt the softmax model to define how likely this ordering would be in context x by just taking a product of probabilities of choosing the first, then the second among the remaining ones, and so on.

$$P(\pi(1), \ldots, \pi(K) \,|\, x, \theta) = \prod_{i=1}^{K} \frac{\exp(\theta_{\pi(1)}^T x)}{\sum_{j=i}^{k} \exp(\theta_{\pi(j)}^T x)}$$

Predicting ranked lists

- Instead of classifying into K categories, we could predict the ordering of K alternatives (e.g., ranking of teams in a tournament)
- Let $(\pi(1), \ldots, \pi(K))$ be a particular ordering of $(1, \ldots, K)$ and x define the context for our prediction.
- We can adapt the softmax model to define how likely this ordering would be in context x by just taking a product of probabilities of choosing the first, then the second among the remaining ones, and so on.

$$P(\pi(1), \ldots, \pi(K) \,|\, x, \theta) = \prod_{i=1}^{K} \frac{\exp(\theta_{\pi(1)}^T x)}{\sum_{j=i}^{k} \exp(\theta_{\pi(j)}^T x)}$$

- The parameters are learned from (context, ranking) pairs (x^i, π^i)
- If we wish to rank varying sets of elements with features, we can replace $\theta_{\pi(i)}^T x$ in the model with $z_{\pi(i)}^T x$ or $z_{\pi(i)}^T \Theta x$ where $z_1, \ldots, z_K$ are feature descriptions of objects

References

- Garrigos et al. (2024), "Handbook of Convergence Theorems for (Stochastic) Gradient Methods", https://arxiv.org/abs/2301.11235
- K. Murphy, "Probabilistic Machine Learning", chapter 10 (logistic regression)