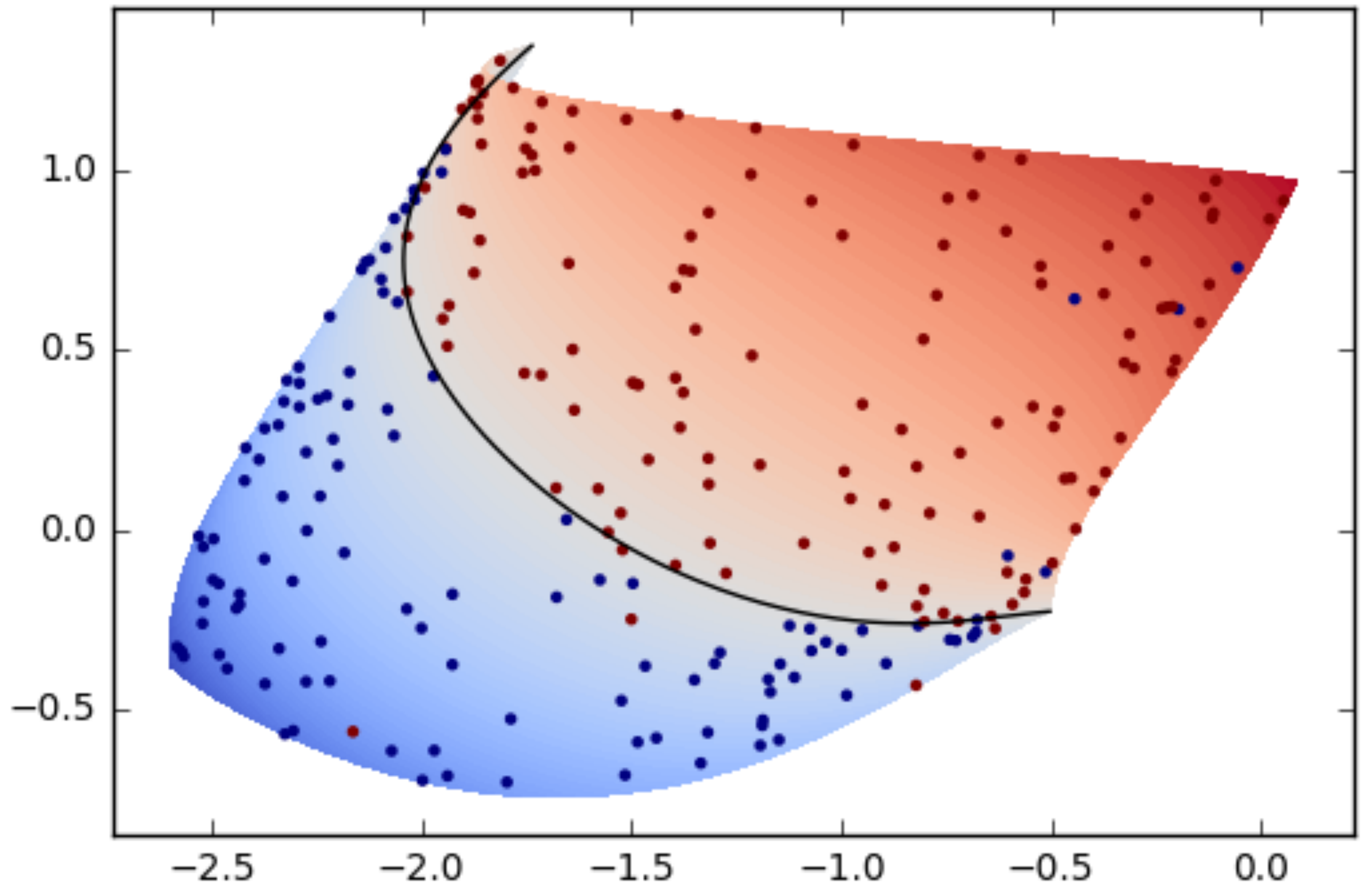
CS 273A: Machine Learning

Winter 2021

Lecture 7: Linear Classifiers

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

All slides in this course adapted from Alex Ihler & Sameer Singh



Logistics

project

- Project guidelines on Canvas:
  - https://canvas.eee.uci.edu/courses/34497/pages/projects
- Team rosters due Thursday, Jan 28 **on Canvas**
- Team-forming spreadsheet posted on piazza

midterm

- Midterm exam on Feb 9, 2–4pm **on Canvas**
- We'll accommodate other timezones — let us know

Today's lecture

Perceptrons

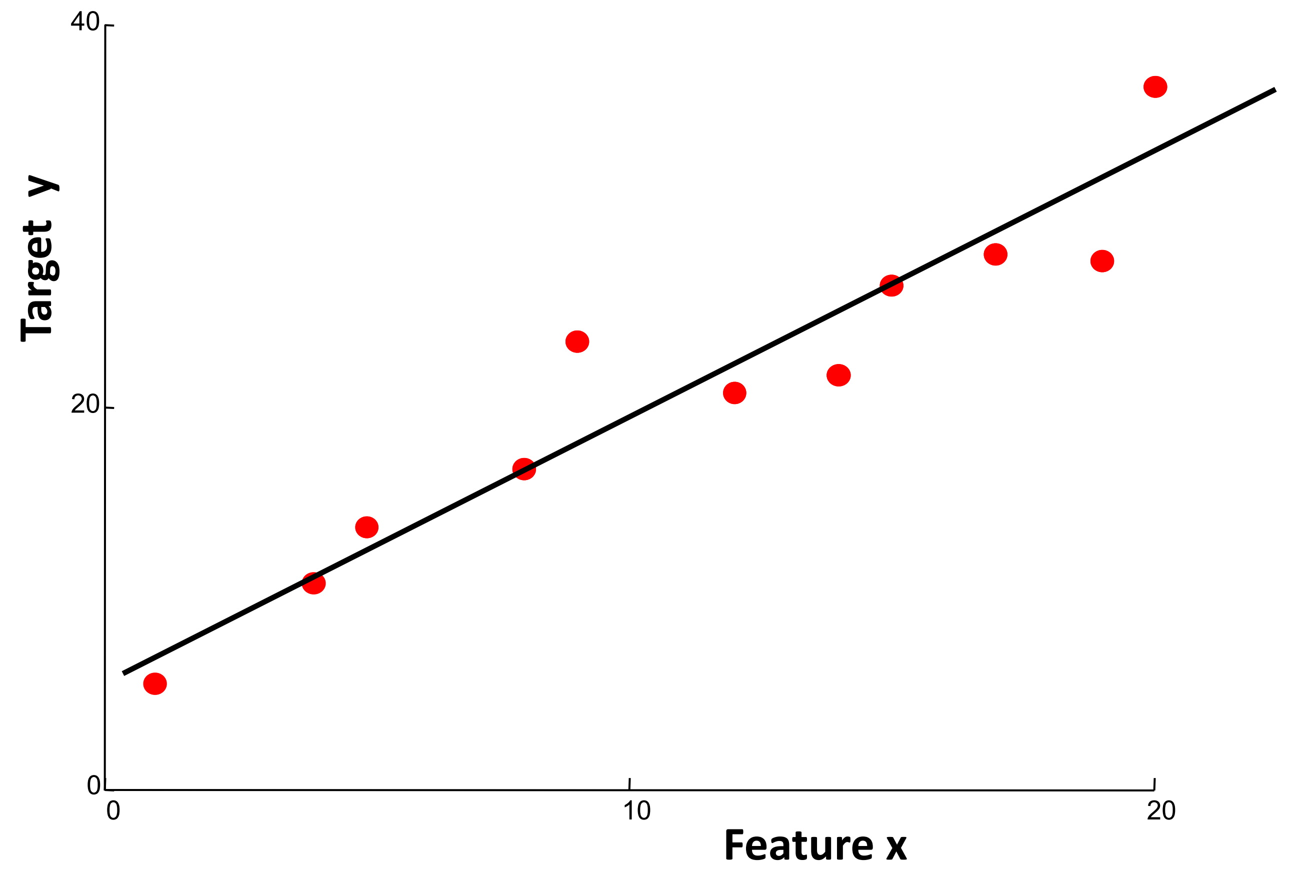
Separability

Learning perceptrons

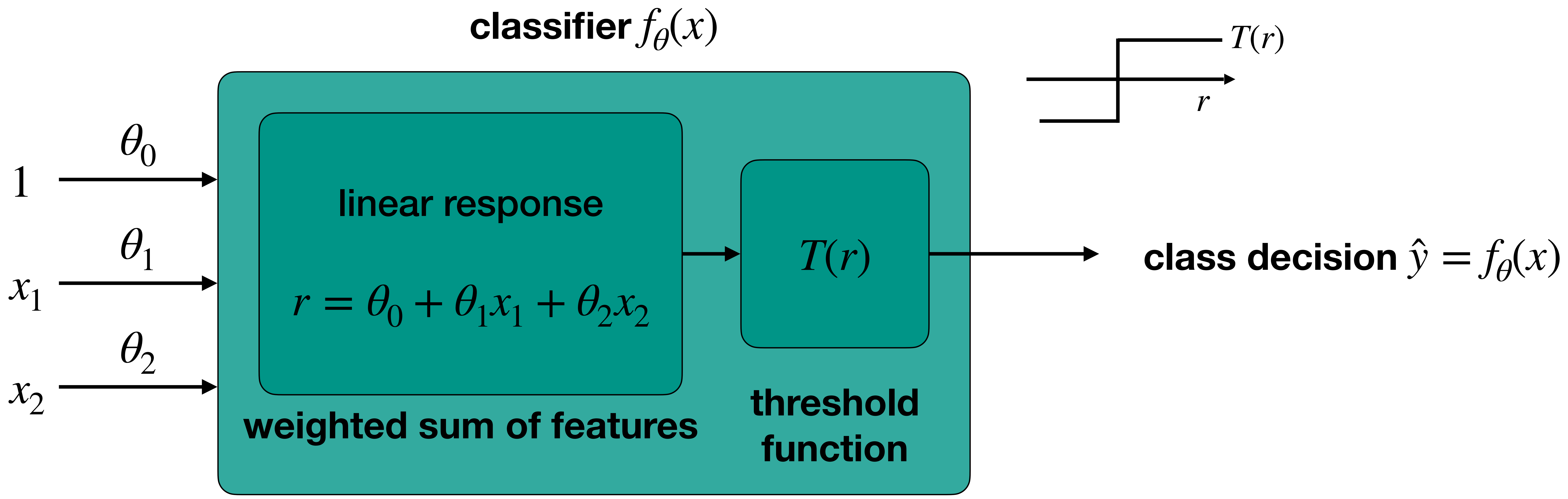
Smooth loss functions

Linear regression vs. classification

- Regression:
  - Continuous target y
  - Regressor $\hat{y} = \theta^\intercal x$
- Classification:
  - Discrete label y
  - Classifier $\hat{y} = ?$

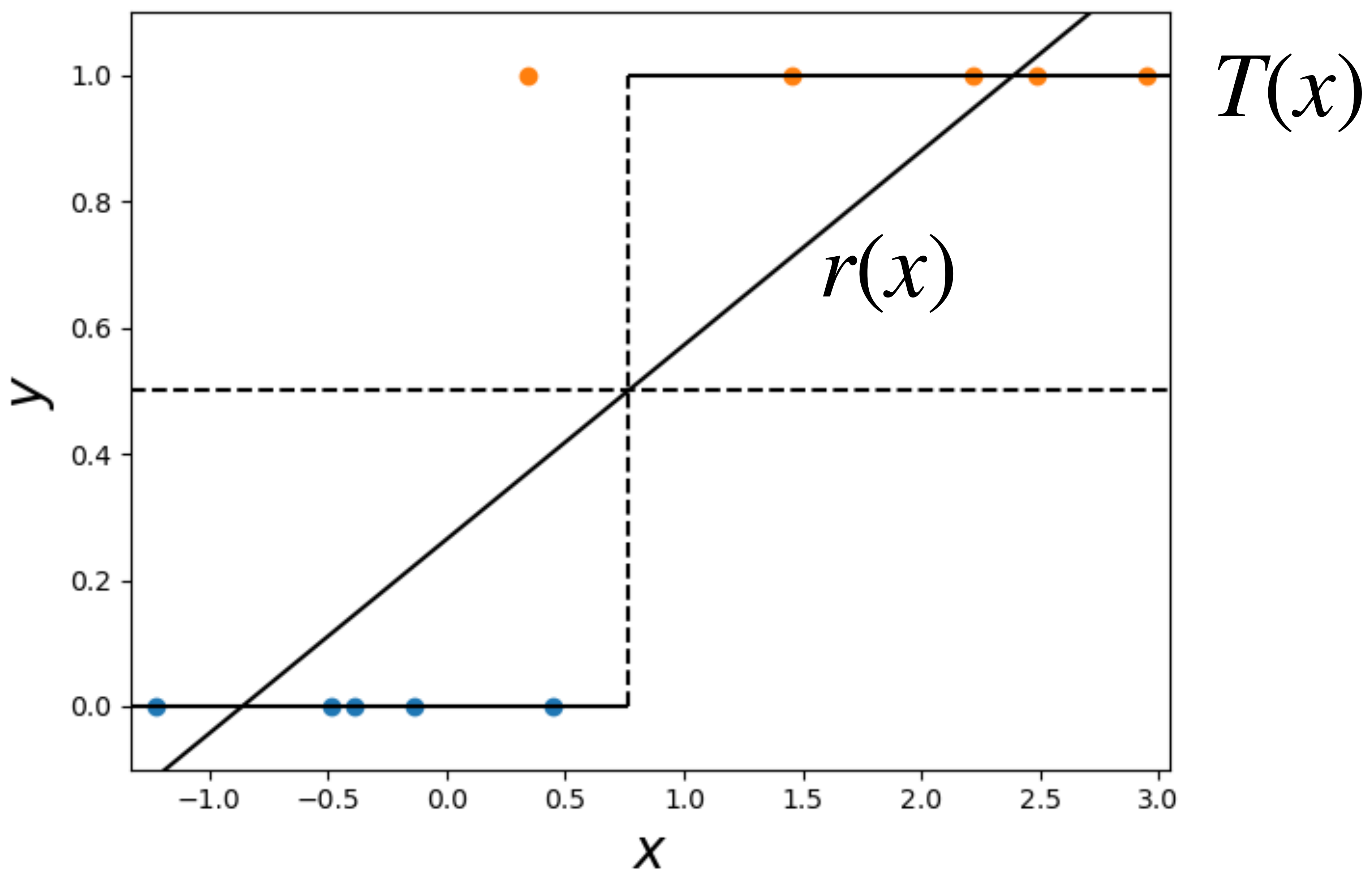


Perceptron



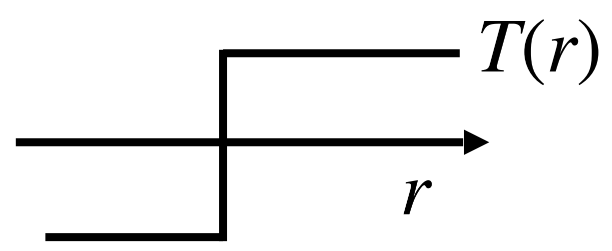

```
r = theta.T @ X           # compute linear response
y_hat = (r > 0)           # predict class 1 vs. 0
y_hat = 2*(r > 0) - 1     # predict class 1 vs. -1
```

Perceptron



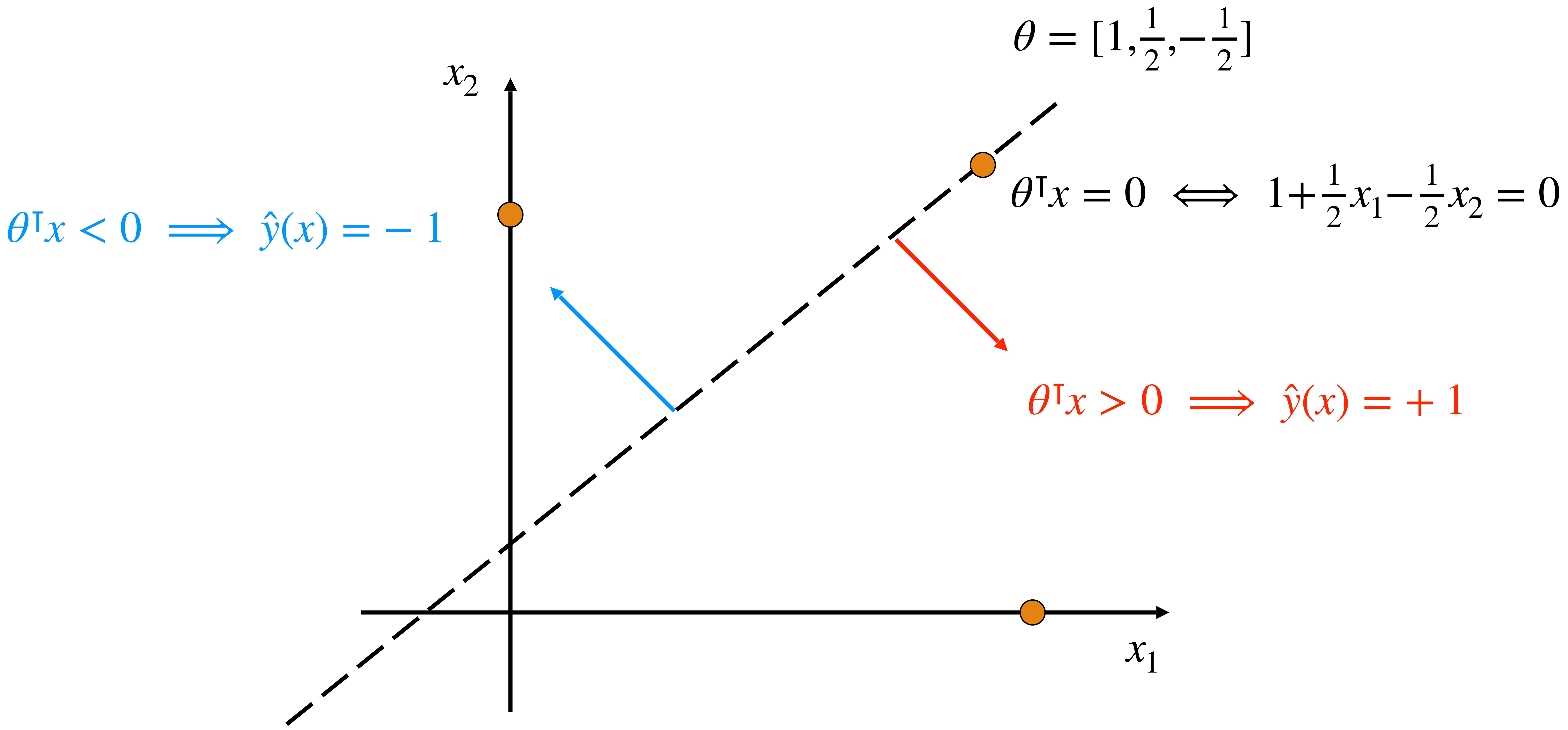
Perceptron

- Perceptron = linear classifier
  - Parameters θ = weights (also denoted w)
  - Response = weighted sum of the features $r = \theta^\intercal x$
  - Prediction = thresholded response $\hat{y}(x) = T(r) = T(\theta^\intercal x)$
  - Decision function: $\hat{y}(x) = \begin{cases} +1 & \text{if } \theta^\intercal x > 0 \\ -1 & \text{otherwise} \end{cases}$ (for $T(r) = \operatorname{sign}(r)$)



- Perceptron: a simple (vastly inaccurate) model of human neurons
  - Weights = "synapses"
  - Prediction = "neural firing"

Example



Adapted from Padhraic Smyth

Today's lecture

Perceptrons

Separability

Learning perceptrons

Smooth loss functions

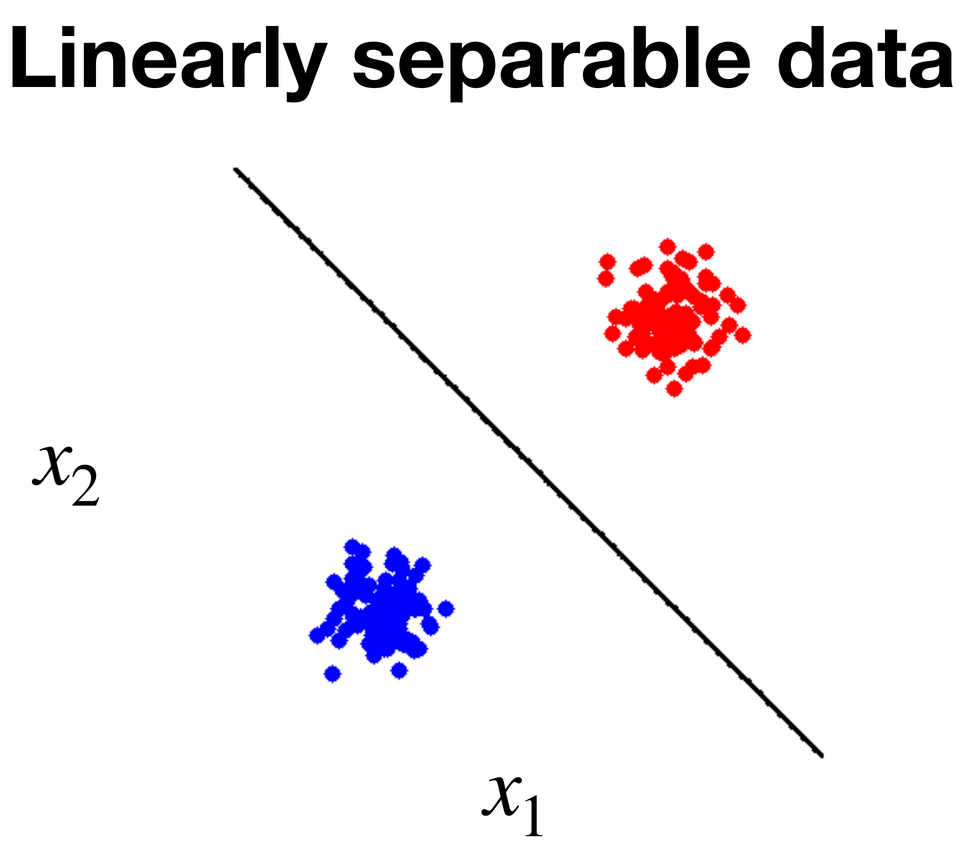
Separability

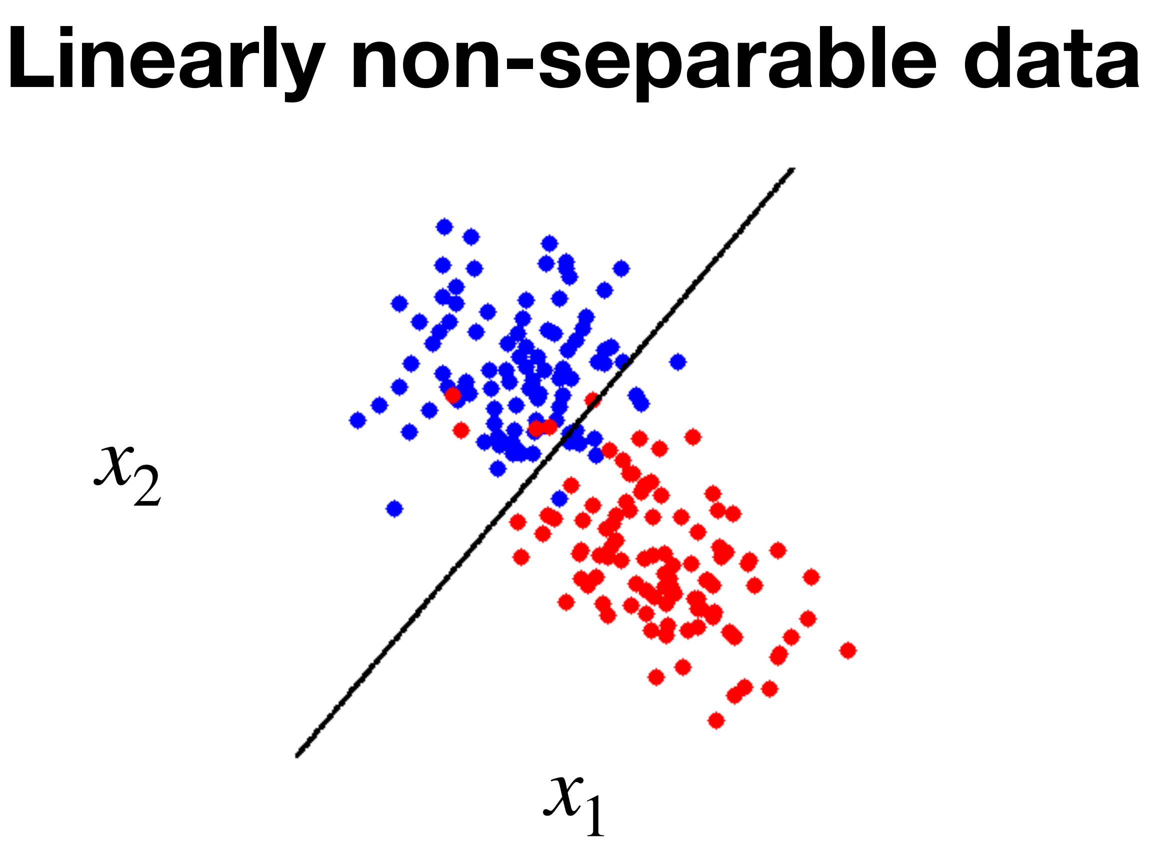
- Separable dataset = there's a model (in our class) with perfect prediction

- Separable problem = there's a model with 0 test loss $\mathbb{E}_{x,y\sim p}[\ell(y, \hat{y}(x))] = 0$

  - Also called realizable

- Linearly separable = separable by a linear classifiers (hyperplanes)



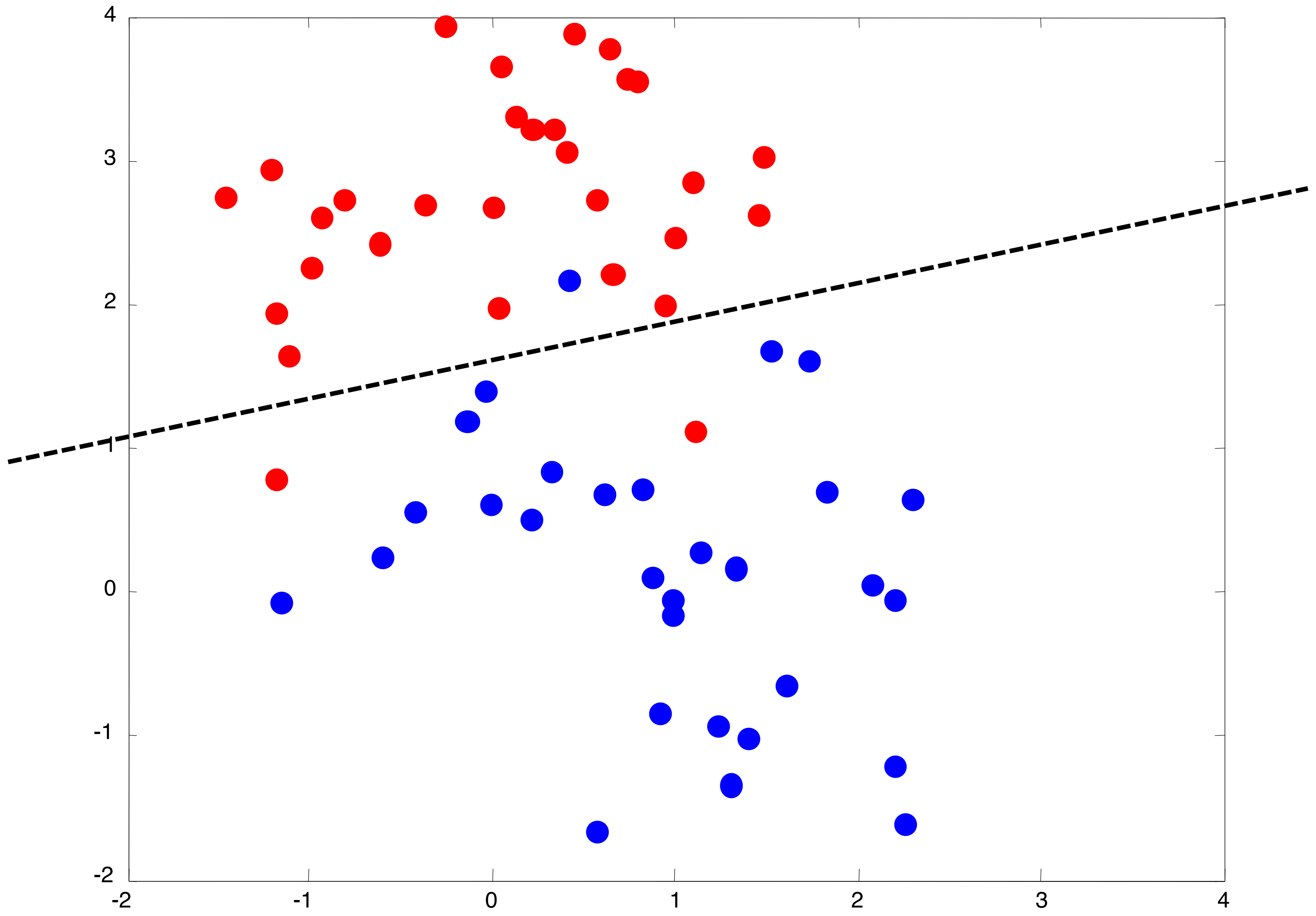


Why do classes overlap?

- Non-separable data means no model can perfectly predicted it
  - Feature ranges for different classes overlap
  - Given an instance in the overlap range — we have uncertainty
- How to improve separation / reduce loss?
  - More complex model class may include a separating model
  - May need more features for that
- Realistically, we must live with some uncertainty / loss
  - But sometimes we can get less of it...

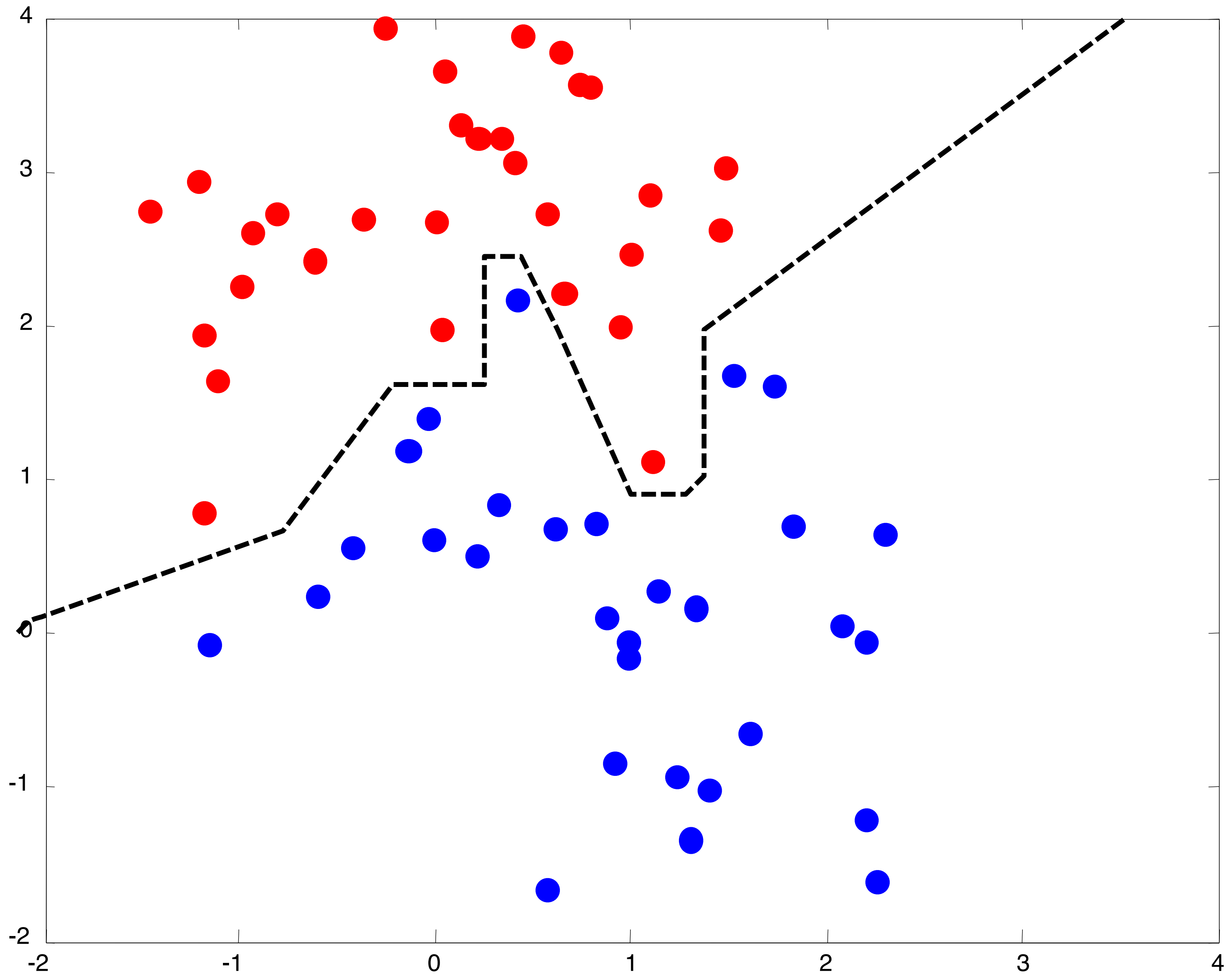
Example: linearly non-separable data

- Data is non-separable with linear classifier



Example: linearly non-separable data

- Data is non-separable with linear classifier
  - ...but separable with non-linear classifier



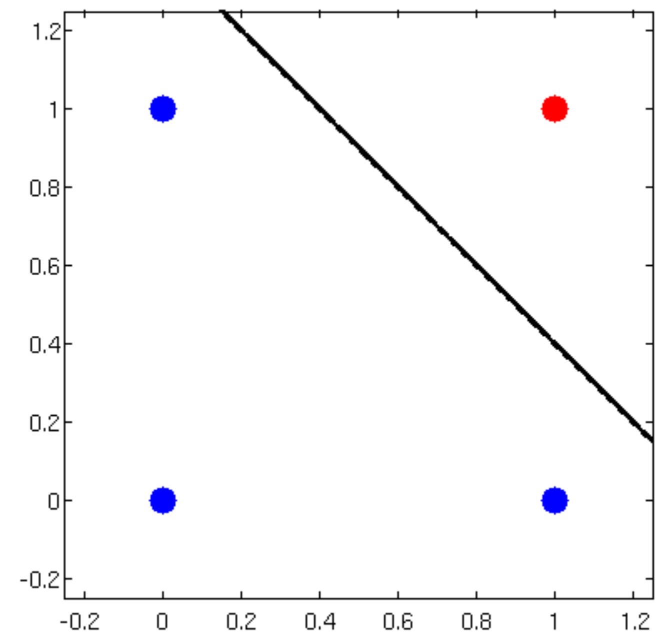
- Is this good? Probably high test loss (overfitting)
  - Problem may be separable by complex model, but no hope of finding a good one

Perceptron: representational power

- A perceptron can represent linearly separable data
- Which functions can a perceptron represent?
  - Those that are linearly separable over all x
- A family of functions that are easy to analyze: boolean functions
  - A perceptron can represent AND but not XOR

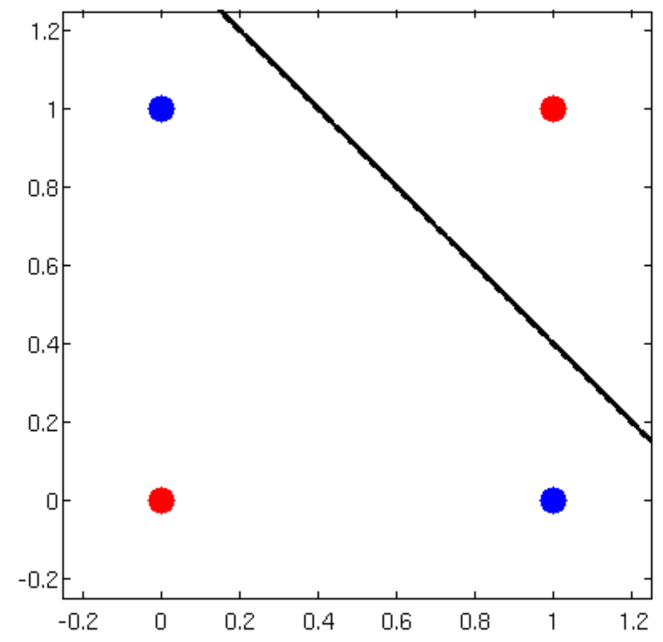
AND

x_1	x_2	y
0	0	-1
0	1	-1
1	0	-1
1	1	1



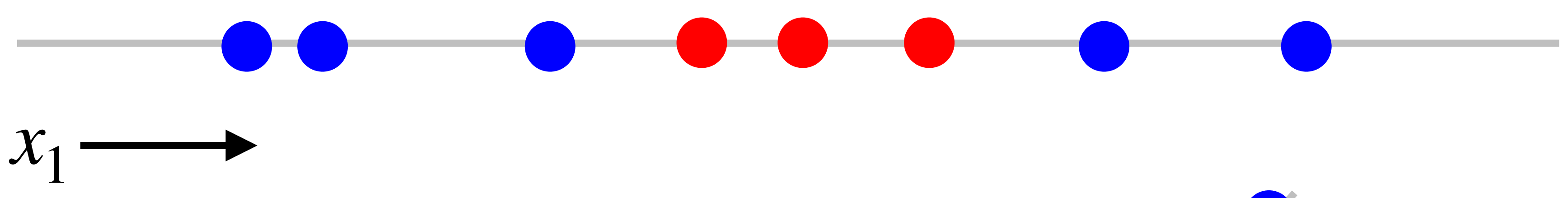
XOR

x_1	x_2	y
0	0	-1
0	1	1
1	0	1
1	1	-1

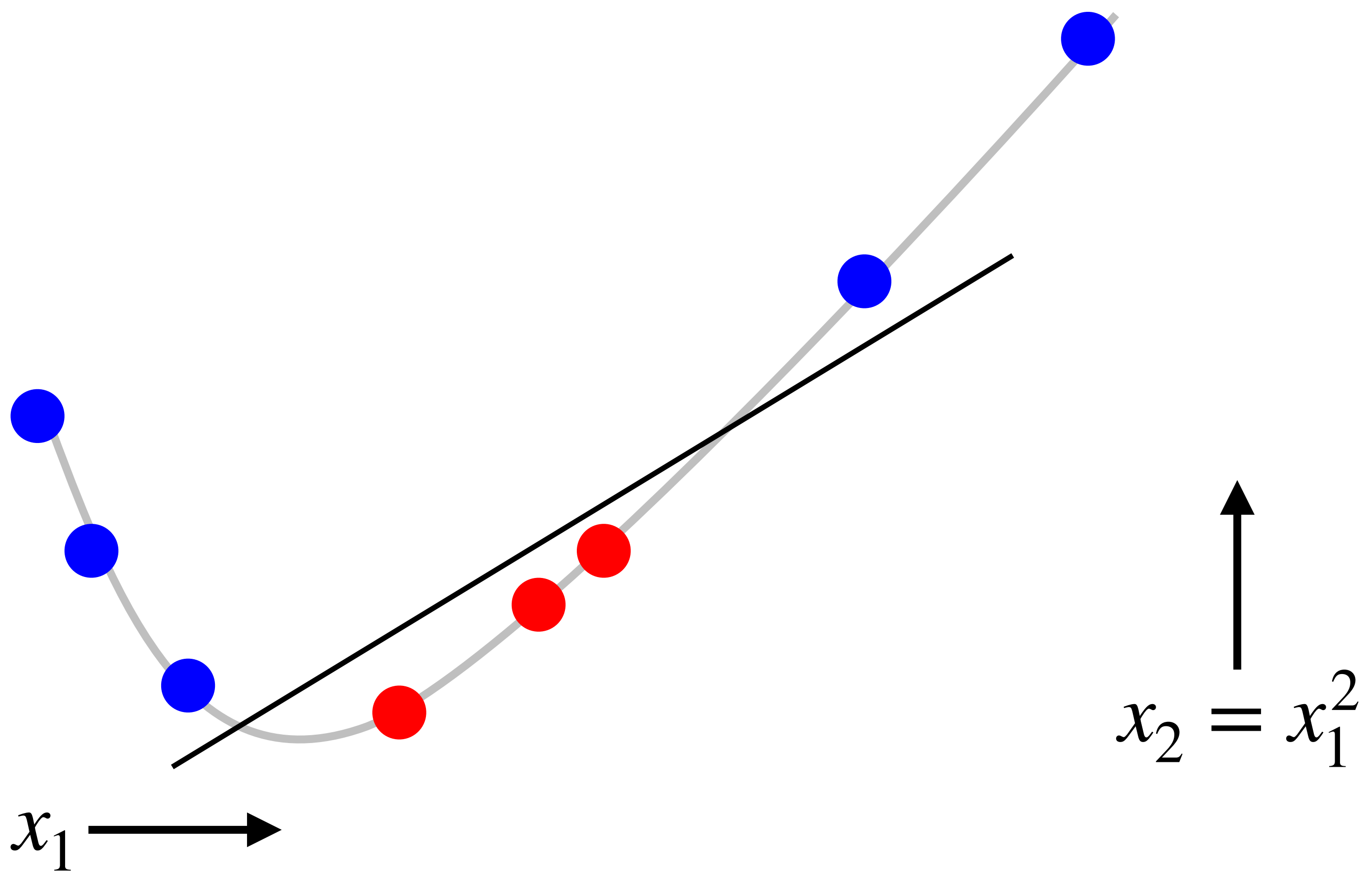


Adding features

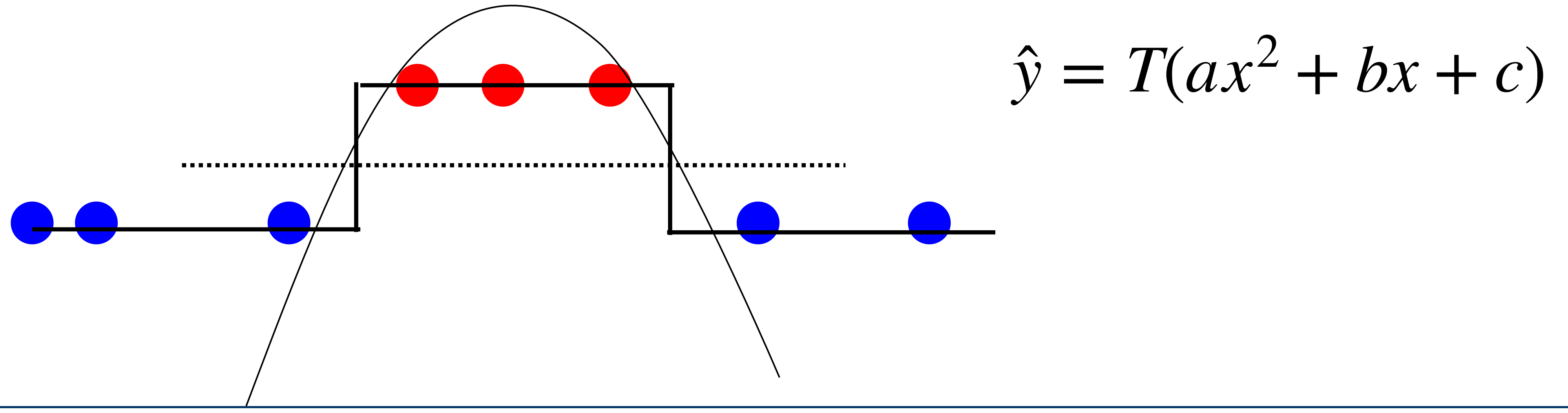
- How to make the perceptron more expressive?
  - Add features — recall linear → polynomial regression
- Linearly non-separable:



- Linearly separable in quadratic features:



- Visualized in original feature space:



  - Decision boundary: $ax^2 + bx + c = 0$

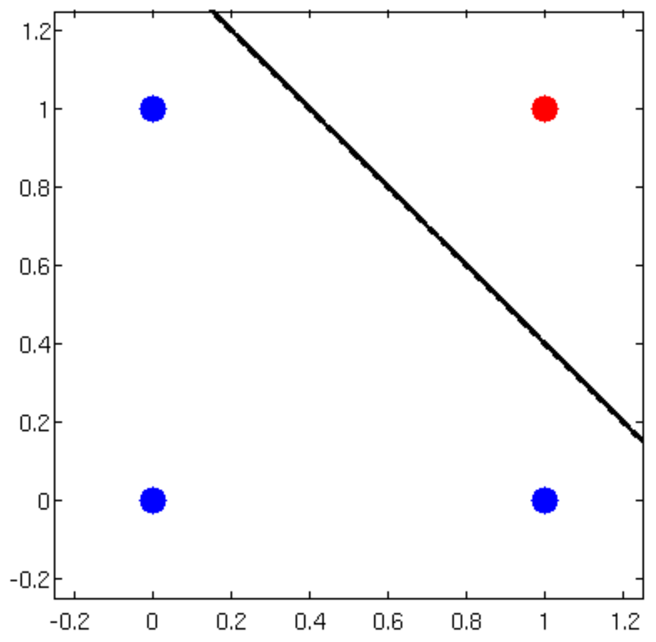
Adding features

- Which functions do we need to represent the decision boundary?
  - When linear functions aren't sufficiently expressive
  - Perhaps quadratic functions are

$$ax_1^2 + bx_1 + cx_2^2 + dx_2 + ex_1x_2 + f = 0$$

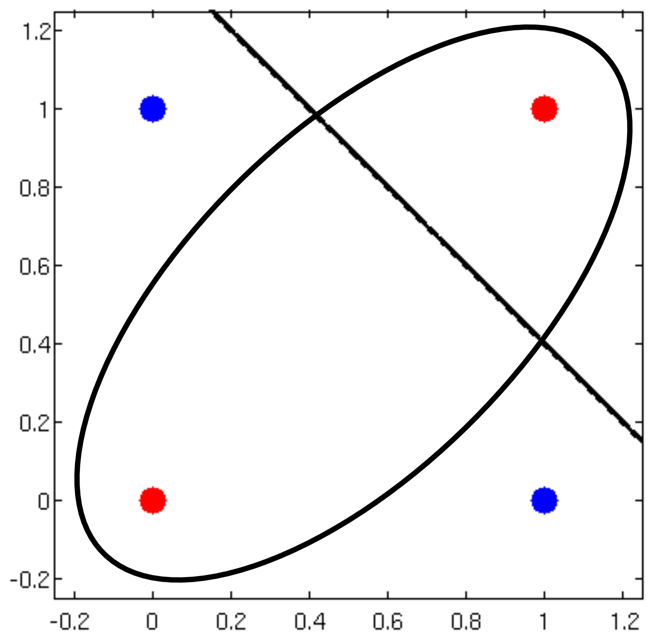
AND

x_1	x_2	y
0	0	-1
0	1	-1
1	0	-1
1	1	1



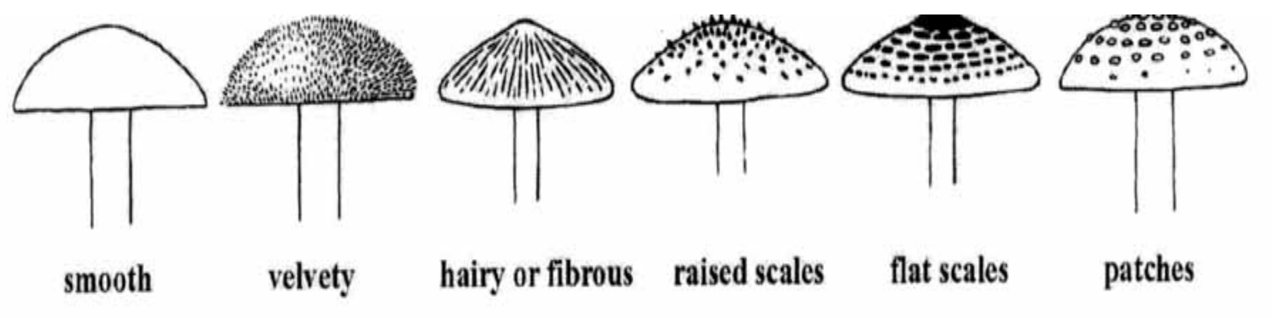
XOR

x_1	x_2	y
0	0	-1
0	1	1
1	0	1
1	1	-1



Representing discrete features

- To define "linear" functions of discrete features: represent as real numbers
- Example: classify poisonous mushroom



  - Surface $\in \{\text{fibrous}, \text{grooves}, \text{scaly}, \text{smooth}\}$
  - Represent as $\{1,2,3,4\}$? Is $\text{smooth} - \text{fibrous} = 3(\text{scaly} - \text{grooves})$?
  - Better: one-hot representation: $\{[1000], [0100], [0010], [0001]\}$
    - Requires 4 binary features instead of 1 integer
    - Preserves the original <u>lack</u> of "topology"

Separability in high dimension

- As we add more features → dimensionality of instance x increases:
  - Separability becomes easier: more parameters, more models that could separate
  - Add enough (good) features, and even a linear classifier can separate
    - Given a decision boundary $f(x) = 0$: add $f(x)$ as a feature → linearly separable!
- However:
  - Do these features explain test data or just training data?
  - Increasing model complexity can lead to overfitting

Recap

- Perceptron = linear classifier
  - Linear response → step decision function → discrete class prediction
  - Linear decision boundary
- Separability = existence of a perfect model (in the class)
  - Separable data: 0 loss on this data
  - Separable problem: 0 loss on the data <u>distribution</u>
  - Perceptron: linear separability
- Adding features:
  - Complex features: complex decision boundary, easier separability
  - Can lead to overfitting

Today's lecture

Perceptrons

Separability

Learning perceptrons

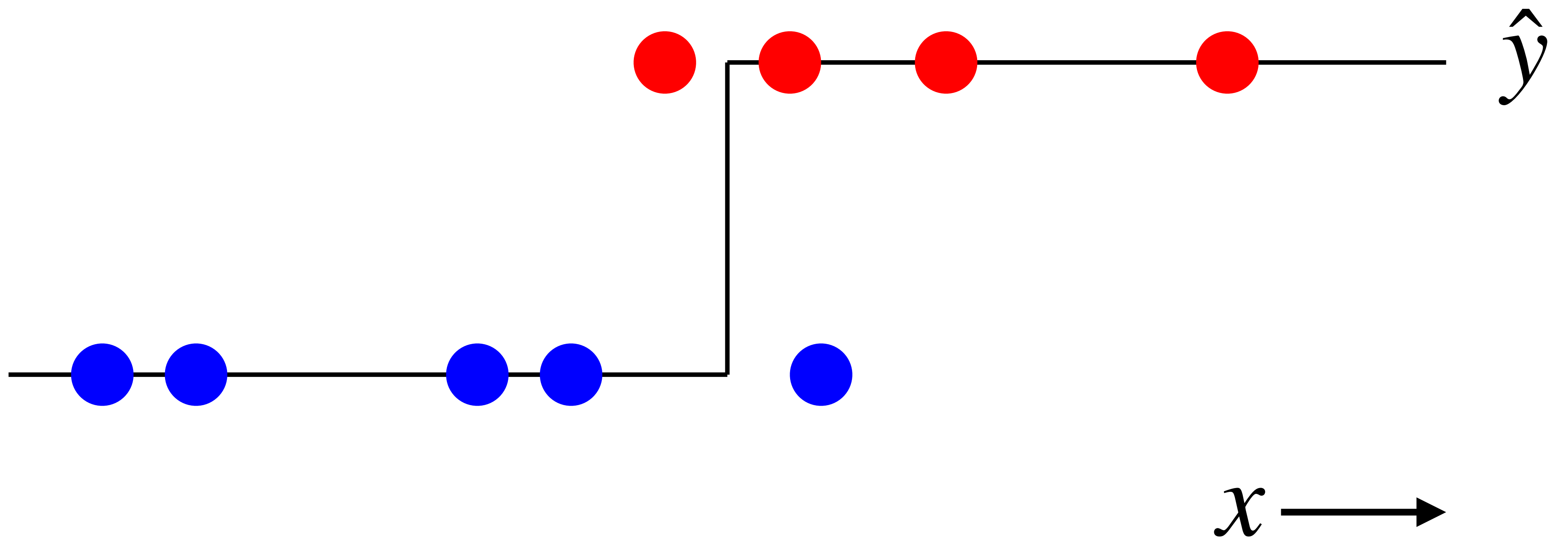
Smooth loss functions

Learning a perceptron

- What do we need to learn the parameters θ of a perceptron?
  - Training data $\mathscr{D}$ = labeled instances
  - Loss function $\mathscr{L}_\theta$ = error rate on labeled data
  - Optimization algorithm = method for minimizing training loss

Error rate

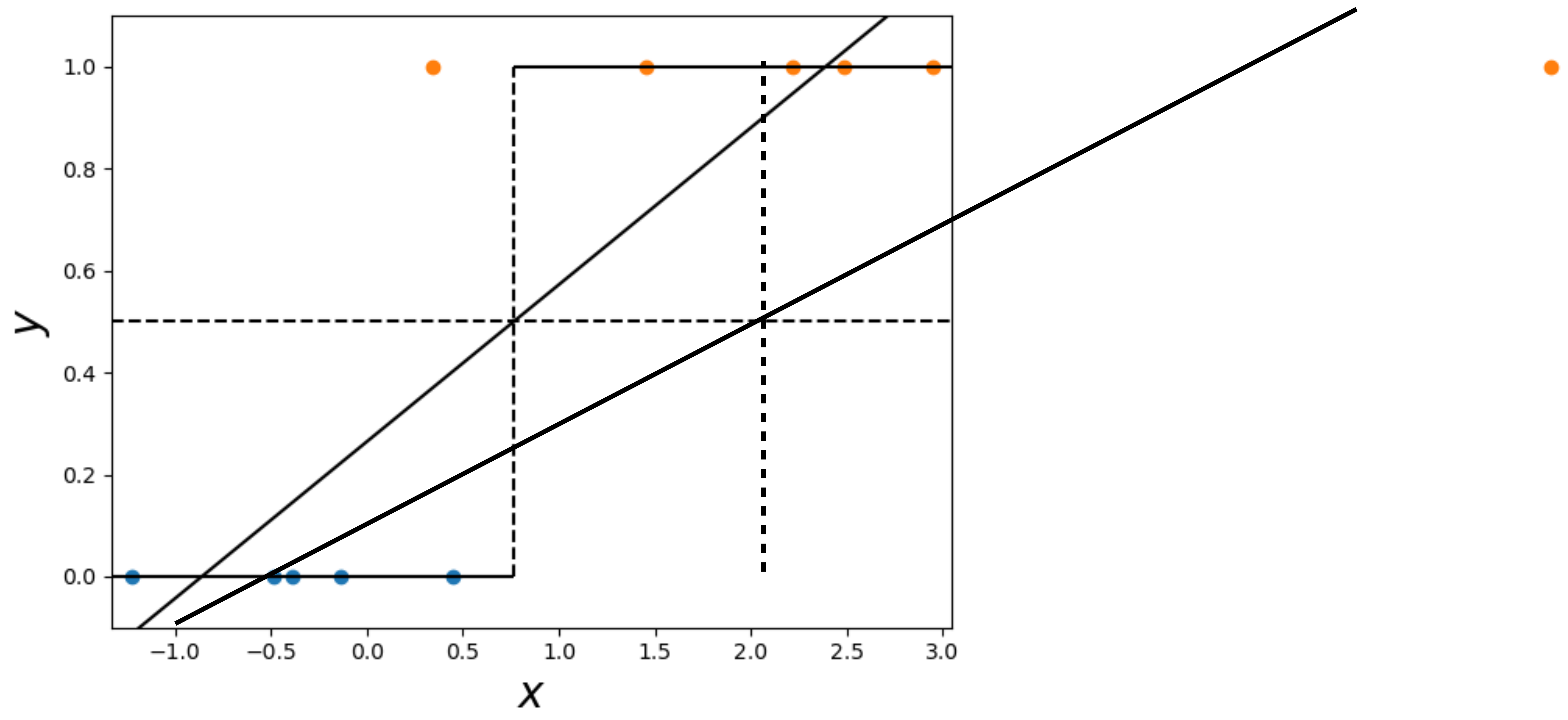
- Error rate: $\mathscr{L}_\theta = \frac{1}{m}\sum_i \delta(y^{(i)} \neq f_\theta(x^{(i)}))$
  - With the indicator $\delta(y \neq \hat{y}) = \begin{cases} 1 & y \neq \hat{y} \\ 0 & else \end{cases}$



$\mathscr{L}_\theta = \frac{2}{9}$

Use linear regression?

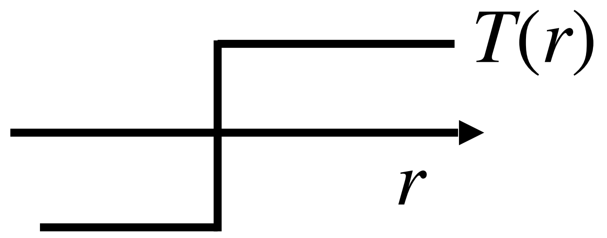
- Idea: find θ using linear regression



- Affected by large regression losses
  - We only care about the <u>classification</u> loss

Perceptron: gradient-based learning

- Problem: loss function not differentiable $\mathscr{L}_\theta(x, y) = \delta(y \neq \operatorname{sign}(\theta^\intercal x))$
  - Write differently: $\mathscr{L}_\theta(x, y) = \frac{1}{4}(y - \operatorname{sign}(\theta^\intercal x))^2$
    - $\nabla_\theta \operatorname{sign}(\theta^\intercal x) = 0$ almost everywhere



  - But we also don't want MSE = $\mathscr{L}_\theta(x, y) = \frac{1}{2}(y - \theta^\intercal x)^2$
  - Compromise: $\mathscr{L}_\theta(x, y) = (y - \operatorname{sign}(\theta^\intercal x))(y - \theta^\intercal x)$
    - $\nabla_\theta \mathscr{L}_\theta = -(y - \operatorname{sign}(\theta^\intercal x))x = -(y - \hat{y})x$

Perceptron training algorithm

$$\text{while } \neg \text{ done:}$$
$$\text{for each data point } j\text{:}$$
$$\hat{y}^{(j)} = \text{sign}(\theta \cdot x^{(j)})$$ **predict output for point j**
$$\theta \leftarrow \theta + \alpha(y^{(j)} - \hat{y}^{(j)})x^{(j)}$$ **gradient step on weird loss**

- Similar to linear regression with MSE loss
  - Except that $\hat{y}$ is the class prediction, not the linear response
  - No update for correct predictions $y^{(j)} = \hat{y}^{(j)}$
  - For incorrect predictions: $y^{(j)} - \hat{y}^{(j)} = \pm 2$
    - $\implies$ update towards x (for false negative) or $-x$ (for false positive)

Perceptron training algorithm

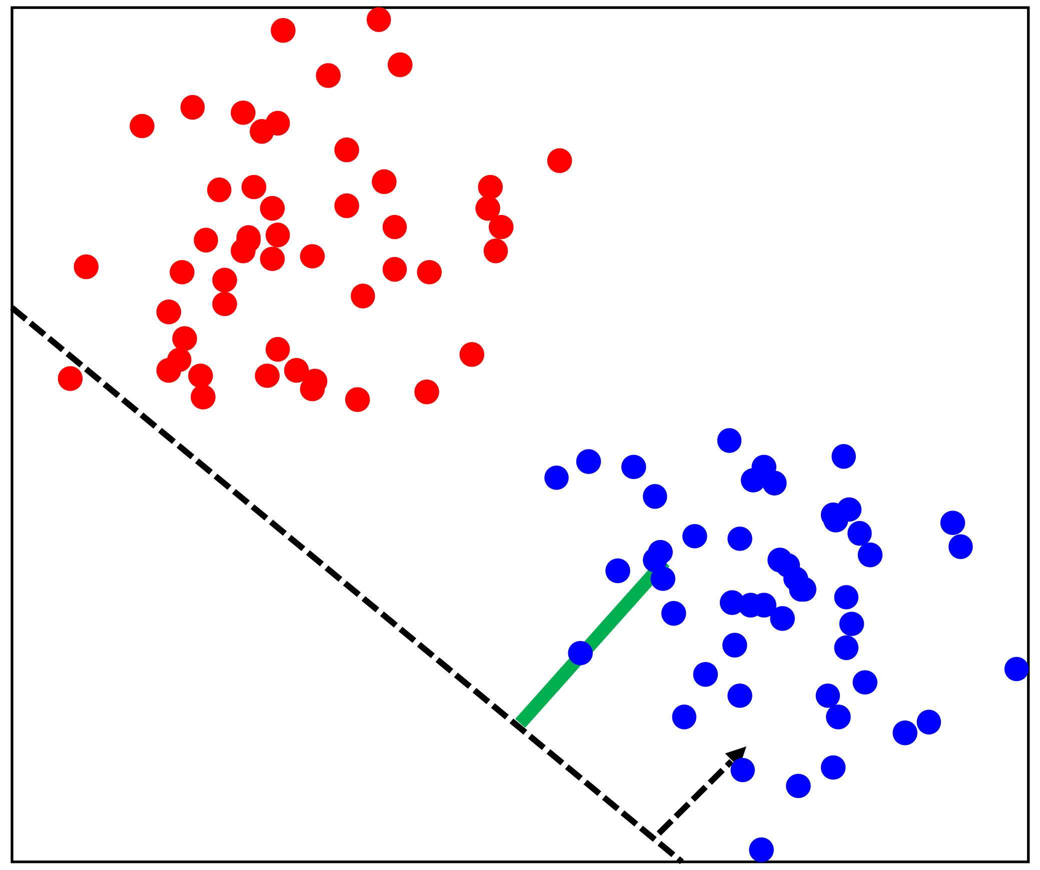
$$\text{while} \neg \text{done:}$$

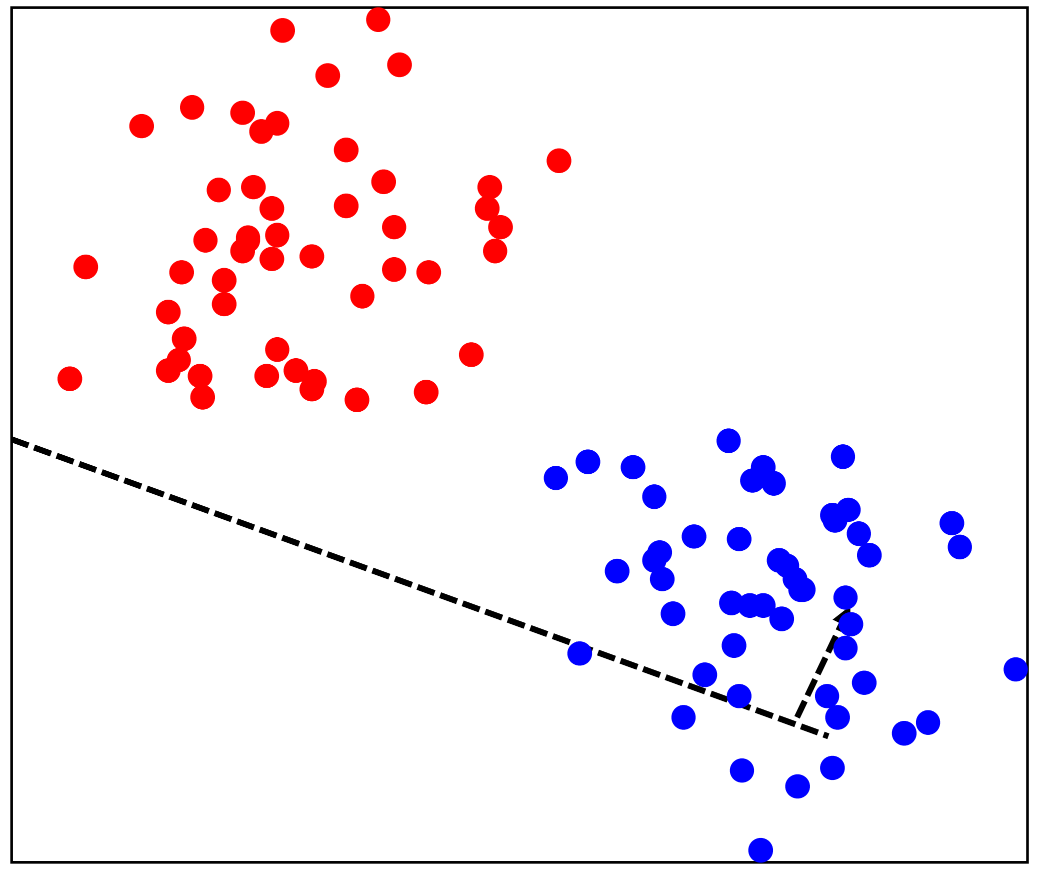
$$\quad \text{for each data point } j\text{:}$$

$$\qquad \hat{y}^{(j)} = \text{sign}(\theta \cdot x^{(j)})$$ **predict output for point j**

$$\qquad \theta \leftarrow \theta + \alpha(y^{(j)} - \hat{y}^{(j)})x^{(j)}$$ **gradient step on weird loss**

incorrect prediction: update weights





Perceptron training algorithm

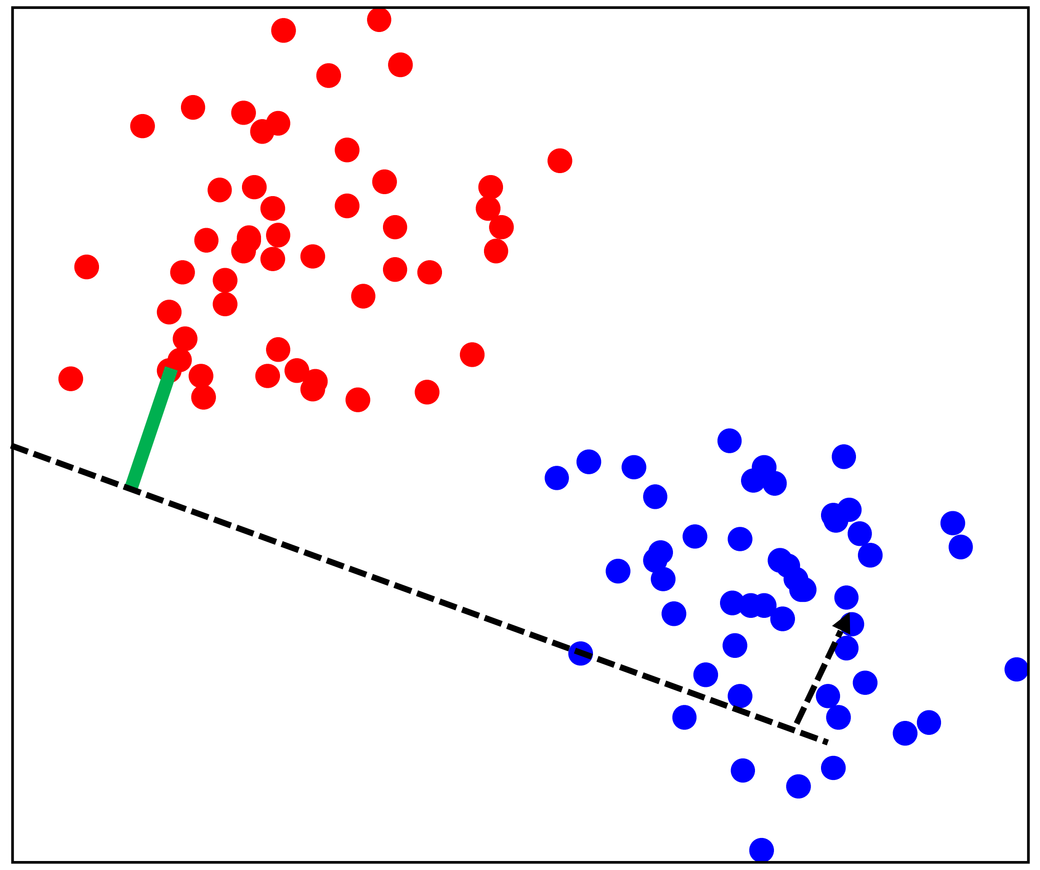
while $\neg$ done:

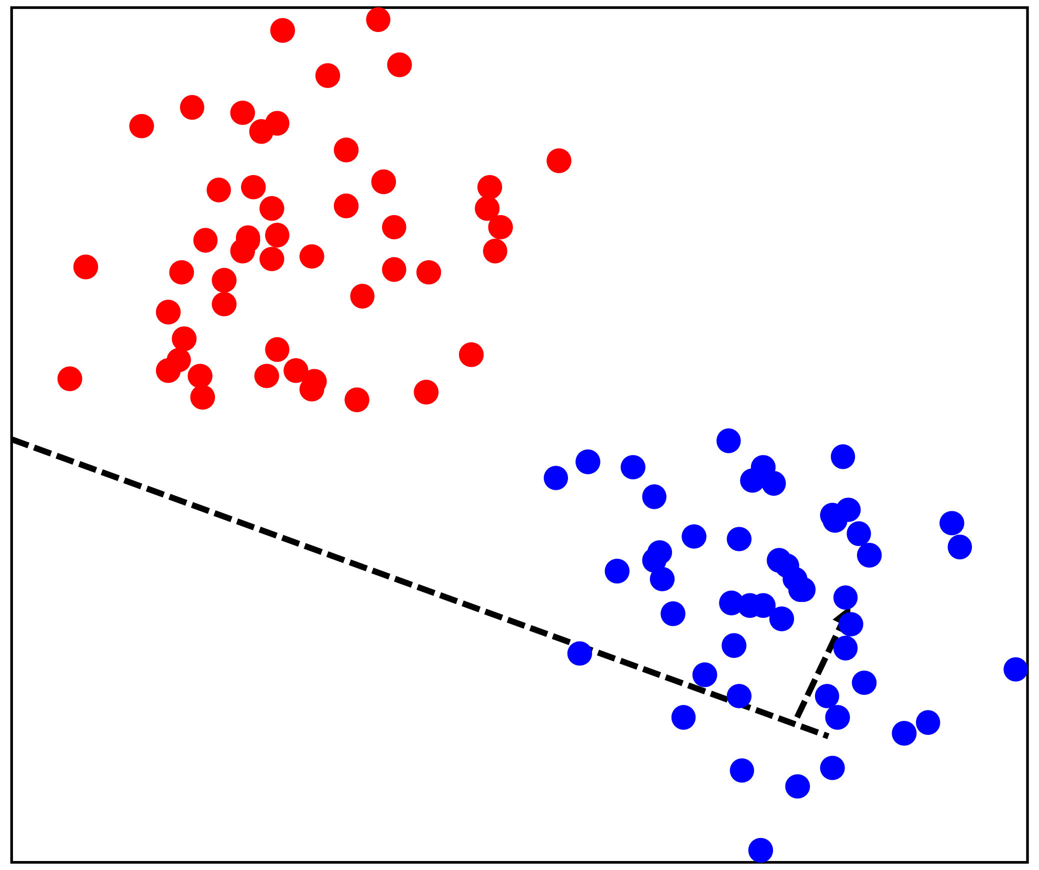
for each data point j:

$$\hat{y}^{(j)} = \operatorname{sign}(\theta \cdot x^{(j)})$$ **predict output for point j**

$$\theta \leftarrow \theta + \alpha(y^{(j)} - \hat{y}^{(j)})x^{(j)}$$ **gradient step on weird loss**

correct prediction: no update





Perceptron training algorithm

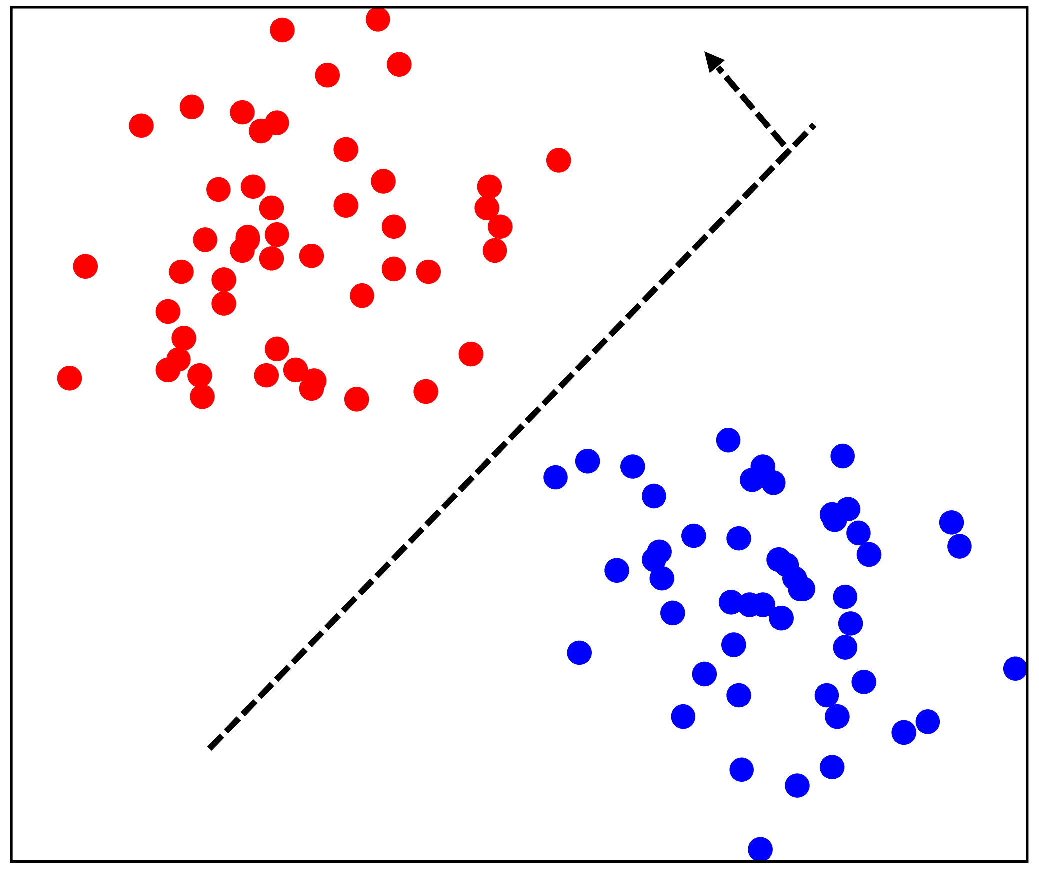
while $\neg$ done:

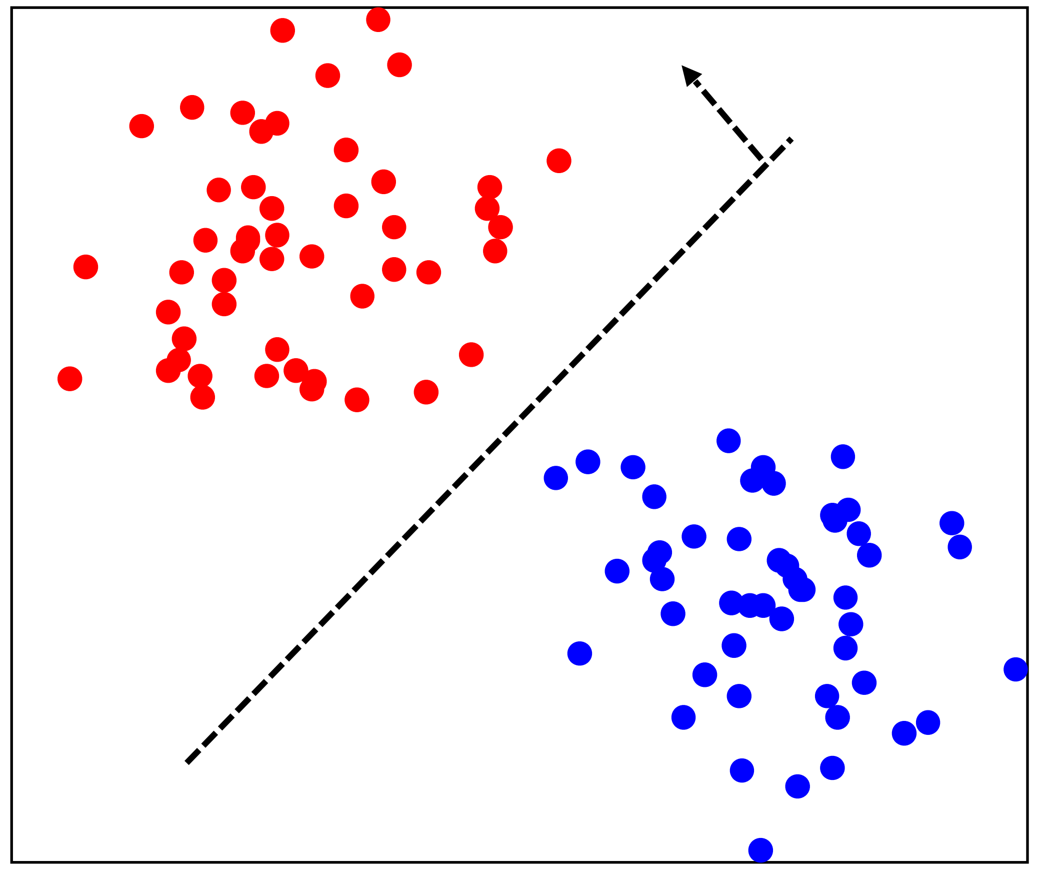
for each data point j:

$$\hat{y}^{(j)} = \mathrm{sign}(\theta \cdot x^{(j)})$$ **predict output for point j**

$$\theta \leftarrow \theta + \alpha(y^{(j)} - \hat{y}^{(j)})x^{(j)}$$ **gradient step on weird loss**

convergence: no more updates





Today's lecture

Perceptrons

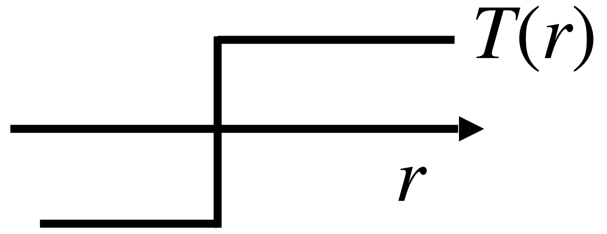
Separability

Learning perceptrons

Smooth loss functions

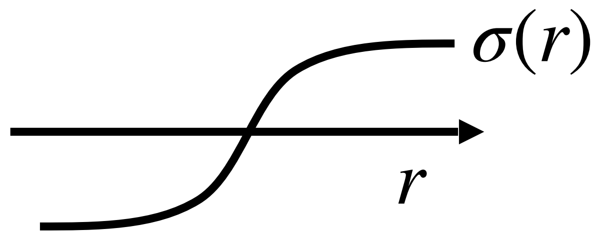
Surrogate loss functions

- Alternative: use differentiable loss function
  - E.g., approximate the step function with a smooth function

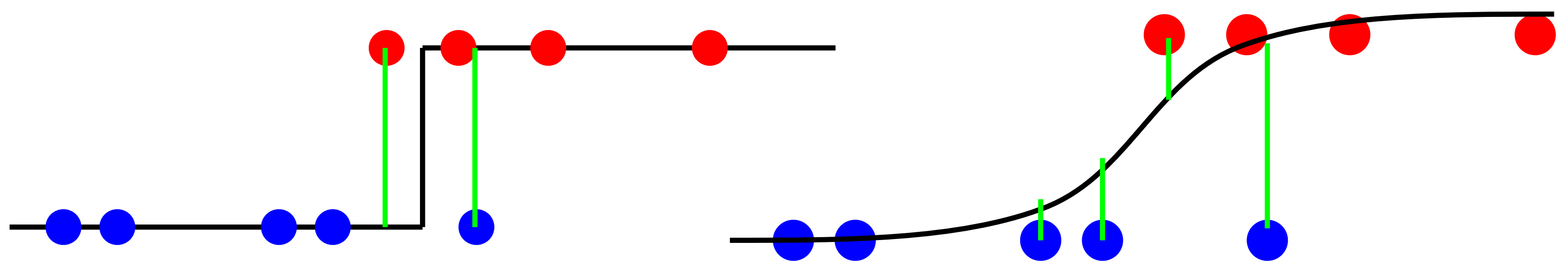


  - Popular choice: logistic / sigmoid function (sigmoid = ”looks like s”)

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



- MSE loss: $\mathscr{L}_\theta(x, y) = (y - \sigma(r(x)))^2$



  - Far from the boundary: $\sigma \approx 0$ or 1, loss approximates 0–1 loss
  - Near the boundary: $\sigma \approx \frac{1}{2}$, loss near $\frac{1}{4}$, but clear improvement direction

Widening the classification margin

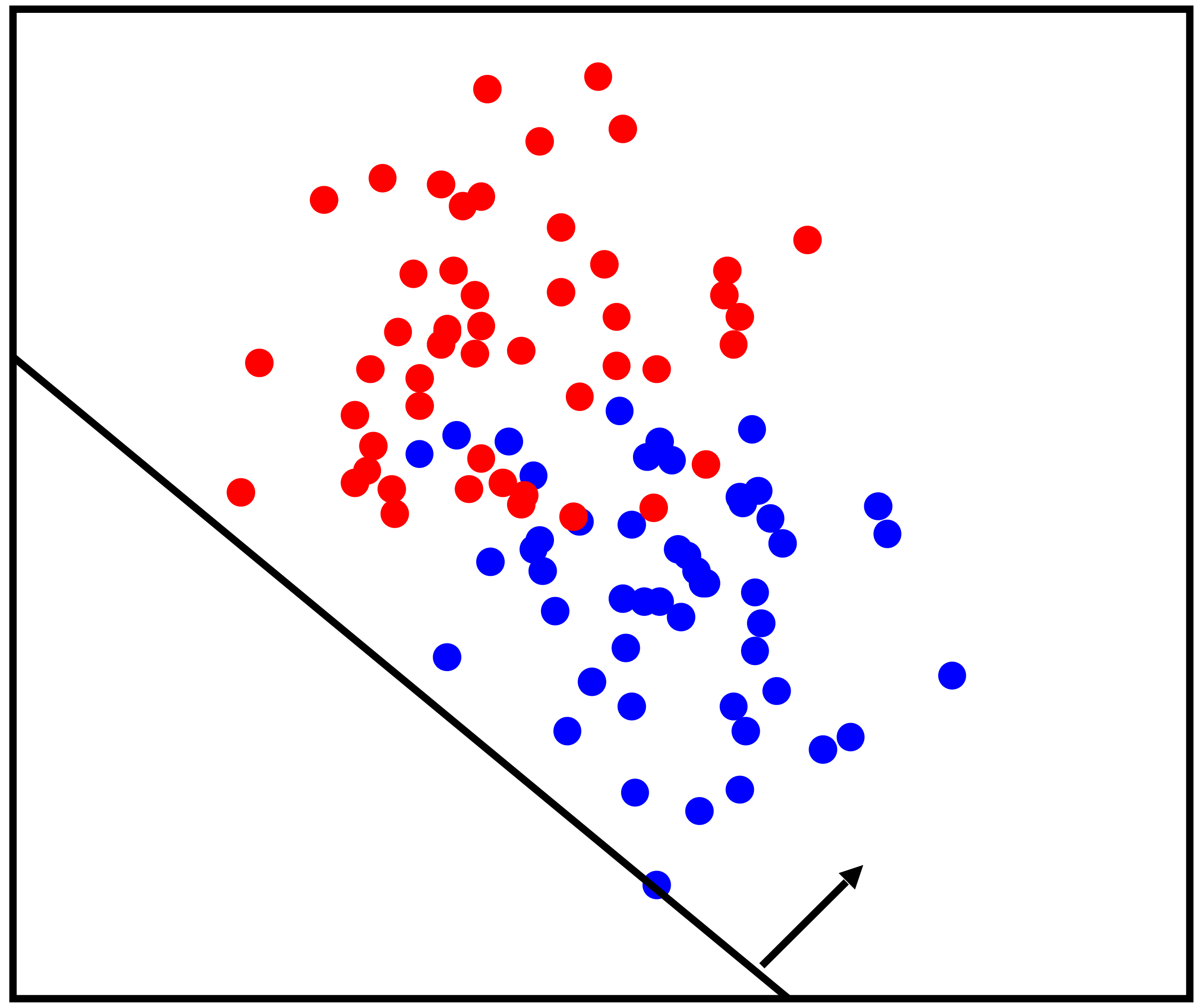
- Which decision boundary is “better”?
  - Both have 0 training loss
  - But one seems more robust, expected to generalize better

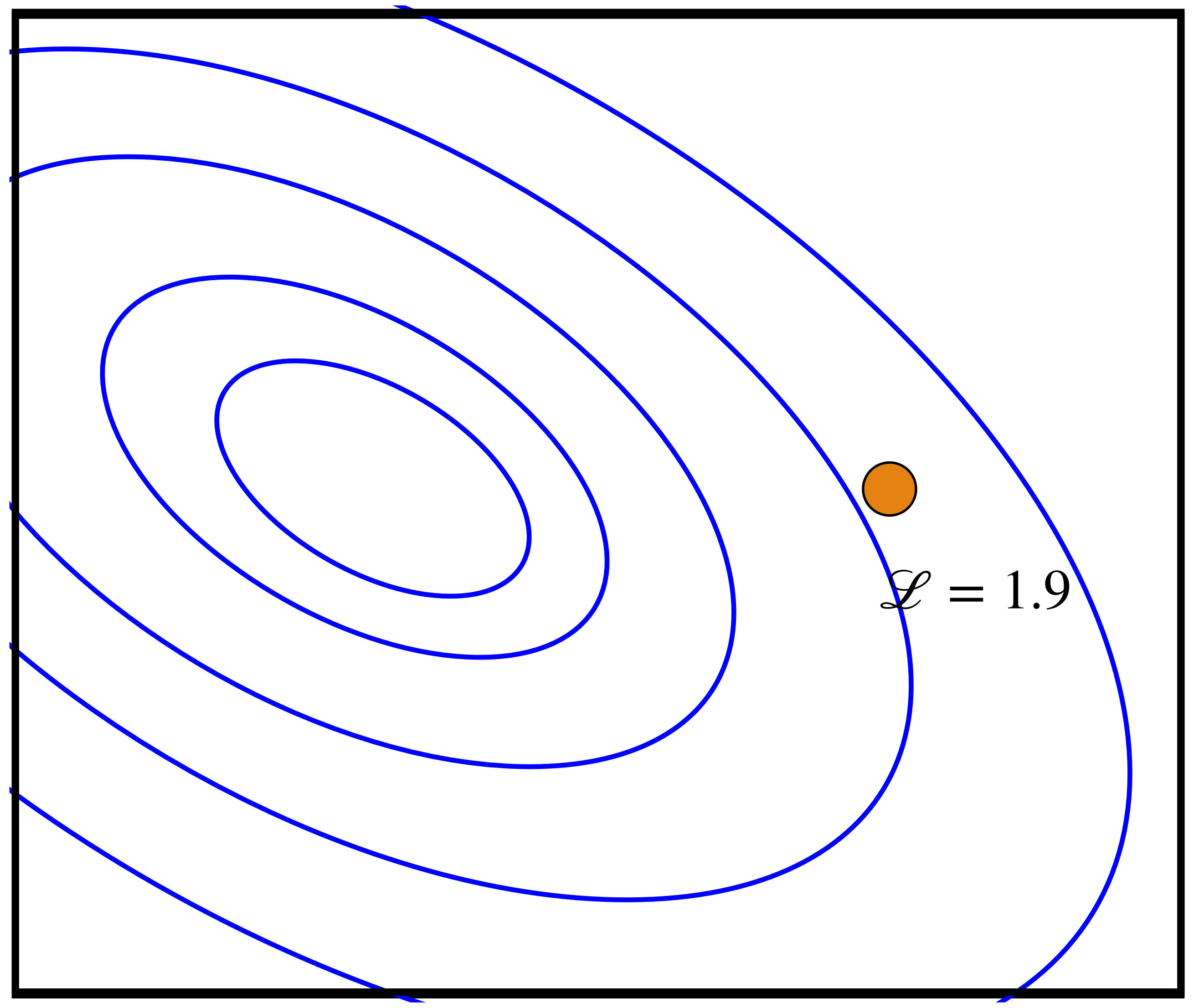
- Benefit of smooth loss function: care about margin
  - Encourage distancing the boundary from data points

Learning smooth linear classifiers

- With a smooth loss function with can use Stochastic Gradient Descent

$$\mathscr{L}_\theta(x, y) = (y - \sigma(r(x)))^2$$

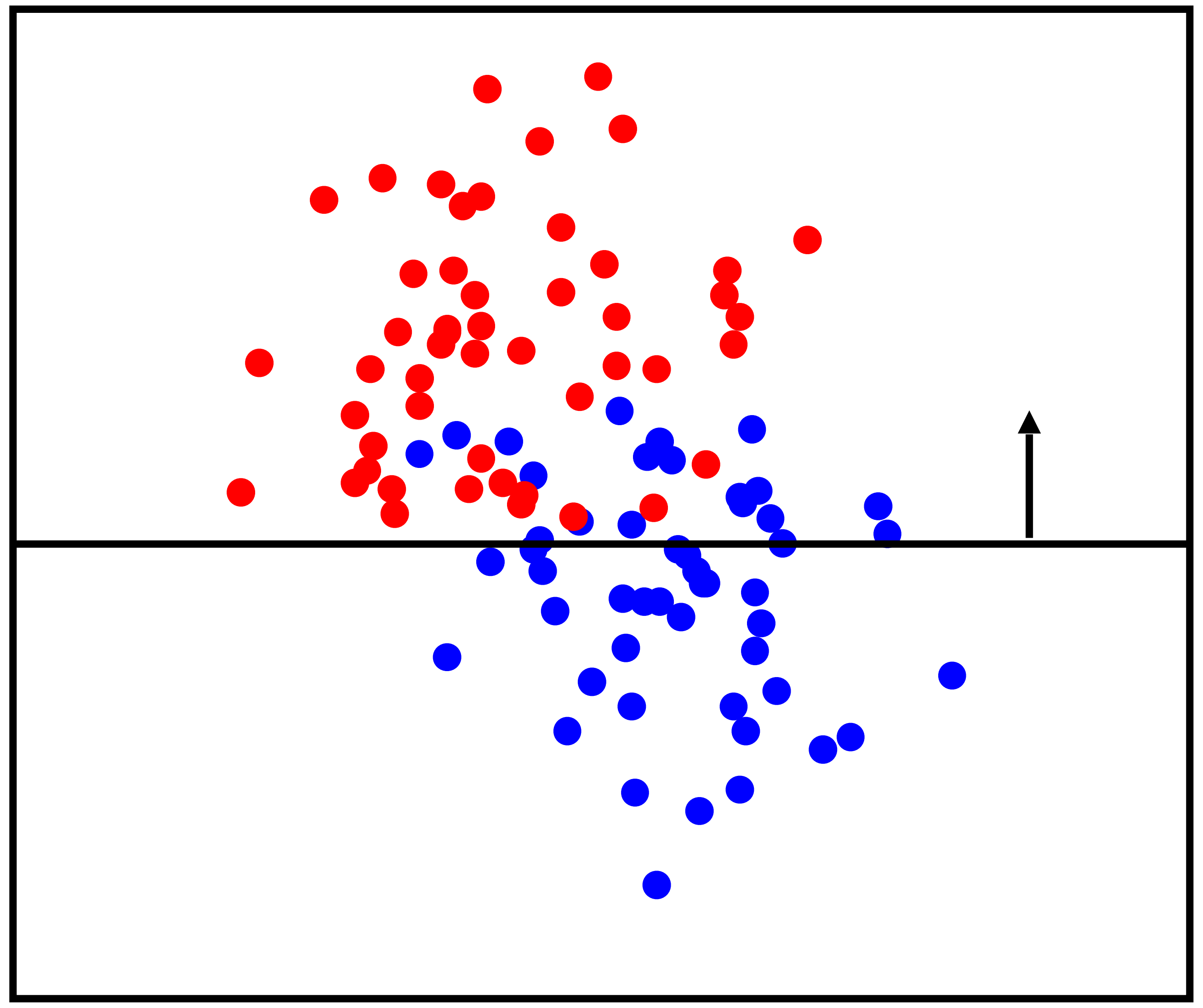


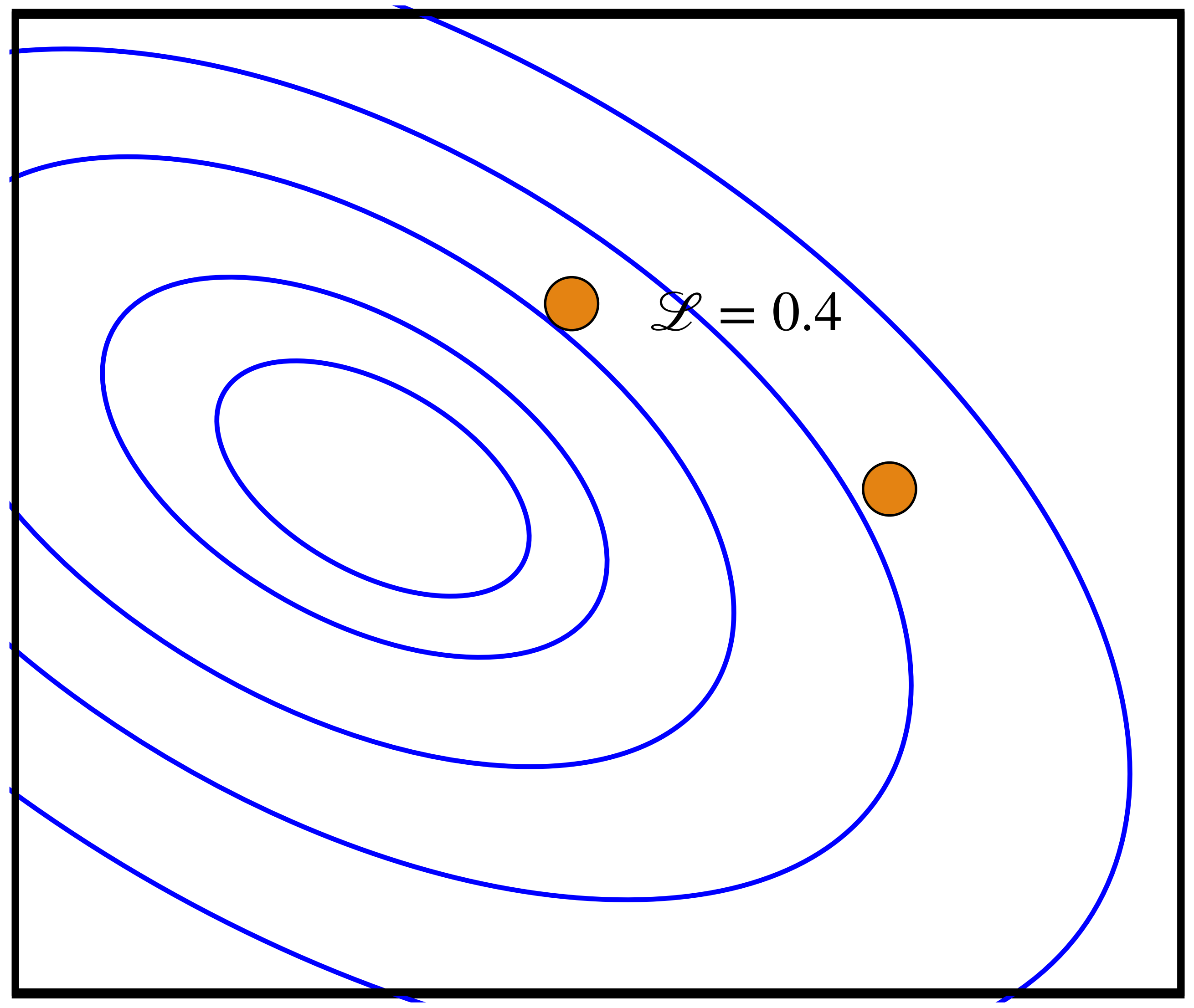


Learning smooth linear classifiers

- With a smooth loss function with can use Stochastic Gradient Descent

$$\mathscr{L}_\theta(x, y) = (y - \sigma(r(x)))^2$$

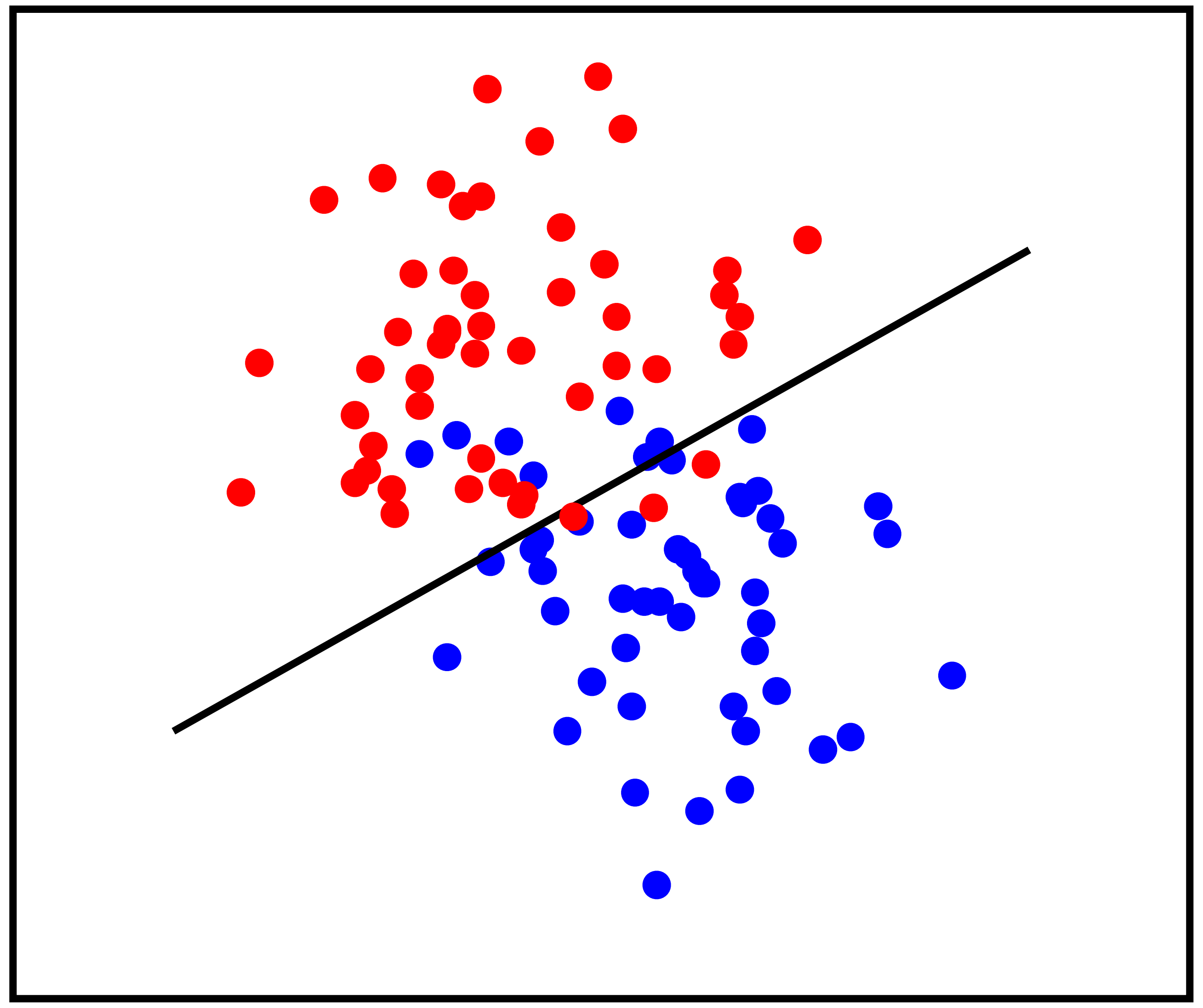


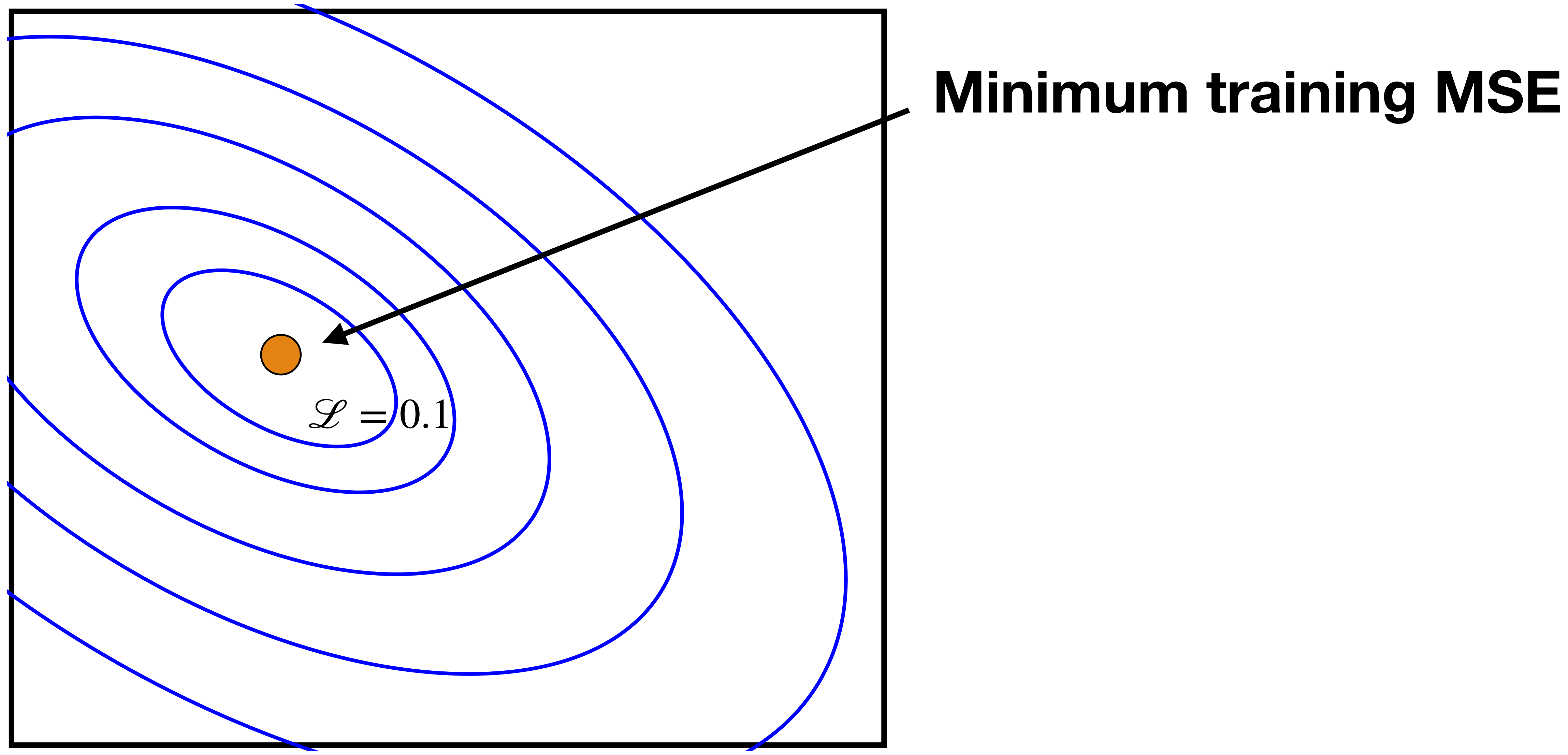


Learning smooth linear classifiers

- With a smooth loss function with can use Stochastic Gradient Descent

$$\mathscr{L}_\theta(x, y) = (y - \sigma(r(x)))^2$$





Logistics

project

- Project guidelines on Canvas:
  - https://canvas.eee.uci.edu/courses/34497/pages/projects
- Team rosters due Thursday, Jan 28 **on Canvas**
- Team-forming spreadsheet posted on piazza

midterm

- Midterm exam on Feb 9, 2–4pm **on Canvas**
- We'll accommodate other timezones — let us know