PHYS 139/239: Machine Learning in Physics

Lecture 3: Support Vector Machine, Regularization, & Logistic Regression

Javier Duarte — January 17, 2023

Logistics

- Homework 1 draft version due Friday 1/20 5pm
  - Graded on effort (so attempt all problems!)
  - If you get stuck, explain why you're stuck
  - If you have trouble getting started, come to office hours or ask question in Slack
- Solutions will be released on Friday soon after deadline
- Homework 1 final version (where you correct things) due Wednesday 1/25 5pm
- Both versions are needed to get 100% (50% for draft, 50% for corrected version)

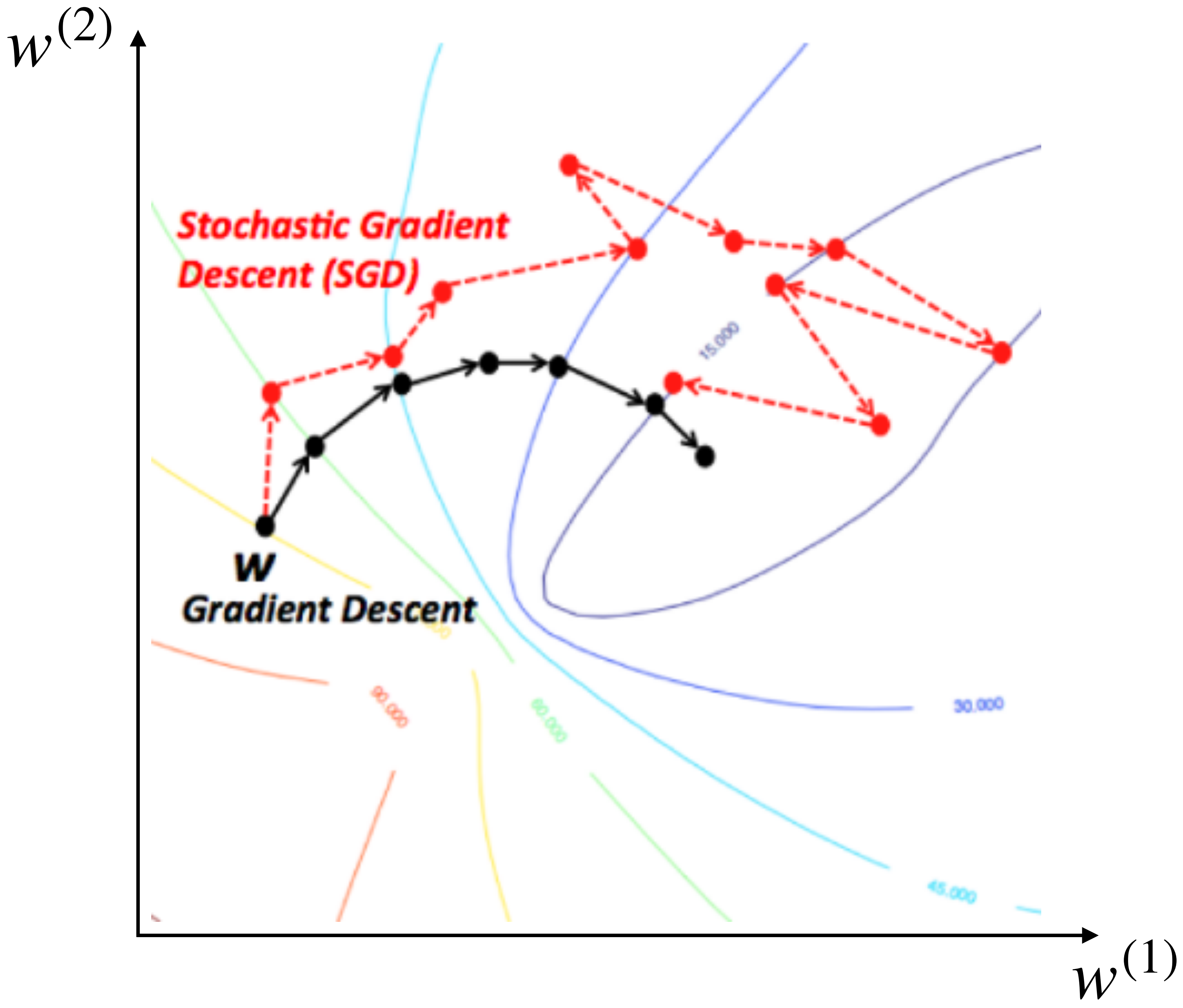
Recap: (Stochastic) gradient descent

$$l(w) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i, | w))$$

- Gradient descent update: $w(t+1) = w(t) - \eta \nabla_w l(w(t))$

- SGD update: $w(t+1) = w(t) - \eta \nabla_w L(y, f(x | w(t)))$

  for a random $(x, y) \in S$



SGD practical tips

- Divide the loss function by the number of examples (normalize):

$$w(t+1) = w(t) - \frac{\eta}{N} \nabla_w l(w)$$

(Don't want the size of our updates to depend on the N)

- Start with a large step size $\eta(t=0)$

- Whenever the validation error stops going down, lower the step size:

$$\eta(t+1) = \eta(t)/2$$

(step size must decrease over time to guarantee convergence)

- Stop when the validation error no longer decreases (early stopping)

Recap: Supervised learning pipeline

- Training dataset: $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x \in \mathbb{R}^D$ and $y \in \mathbb{R}$

For regression

- Model / hypothesis class: $f(x \mid w) = w^\intercal x$ (linear models)

or $\phi(x)$ instead of x

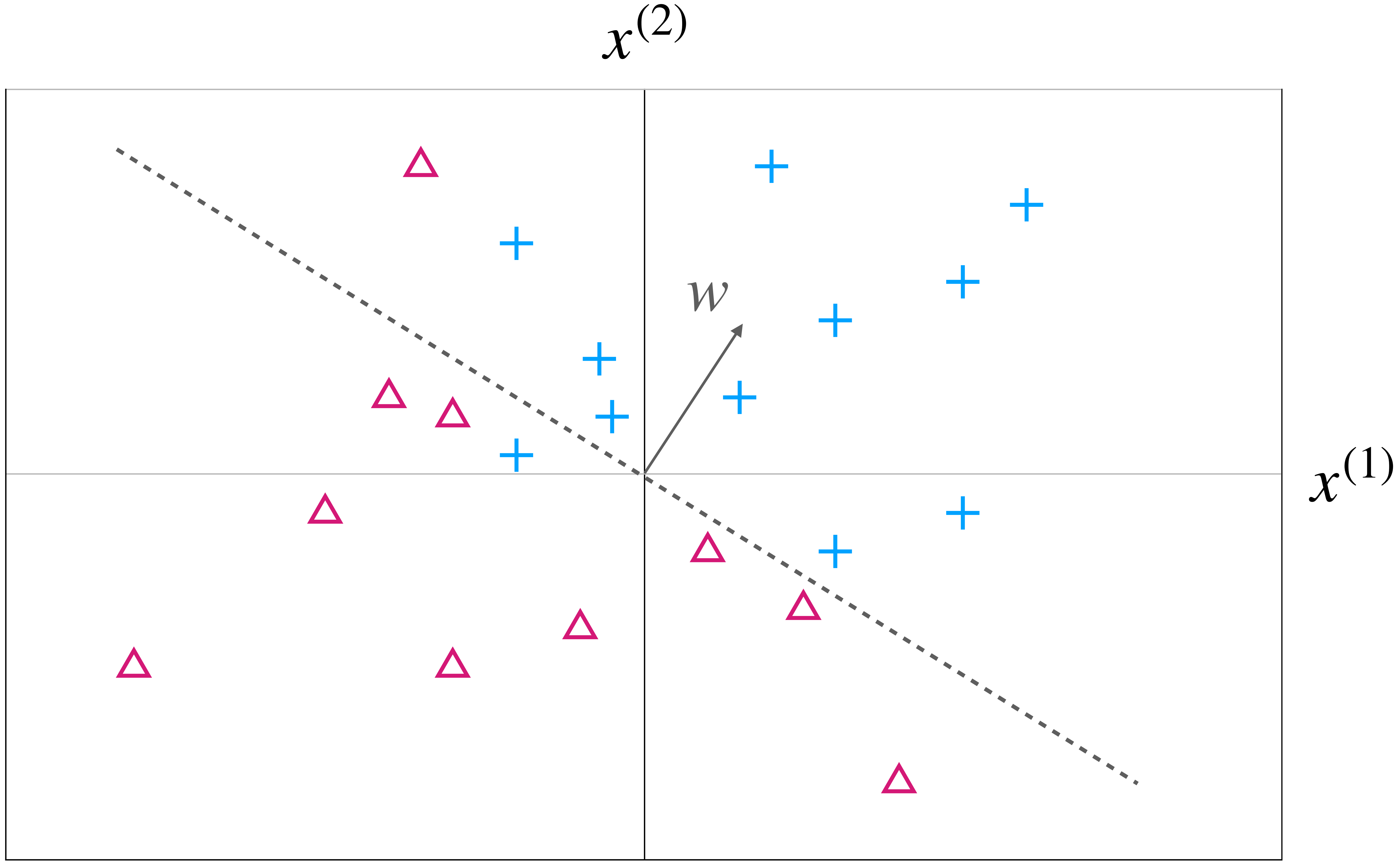
- Loss function: $L(y, y') = (y - y')^2$ (squared loss)
- Optimization algorithm: SGD
- Cross validation and model selection:

- Testing and deployment

Recap: Linear models for binary classification

- Linear model for regression: $f(x \mid w) = w^\intercal x$

- Linear model for binary classification: $f(x \mid w) = \text{sign}(\underbrace{w^\intercal x}_{\text{Raw score}}) \in \{\overset{+}{+}1, \overset{\triangle}{-}1\}$



- Usually evaluate with 0/1 loss

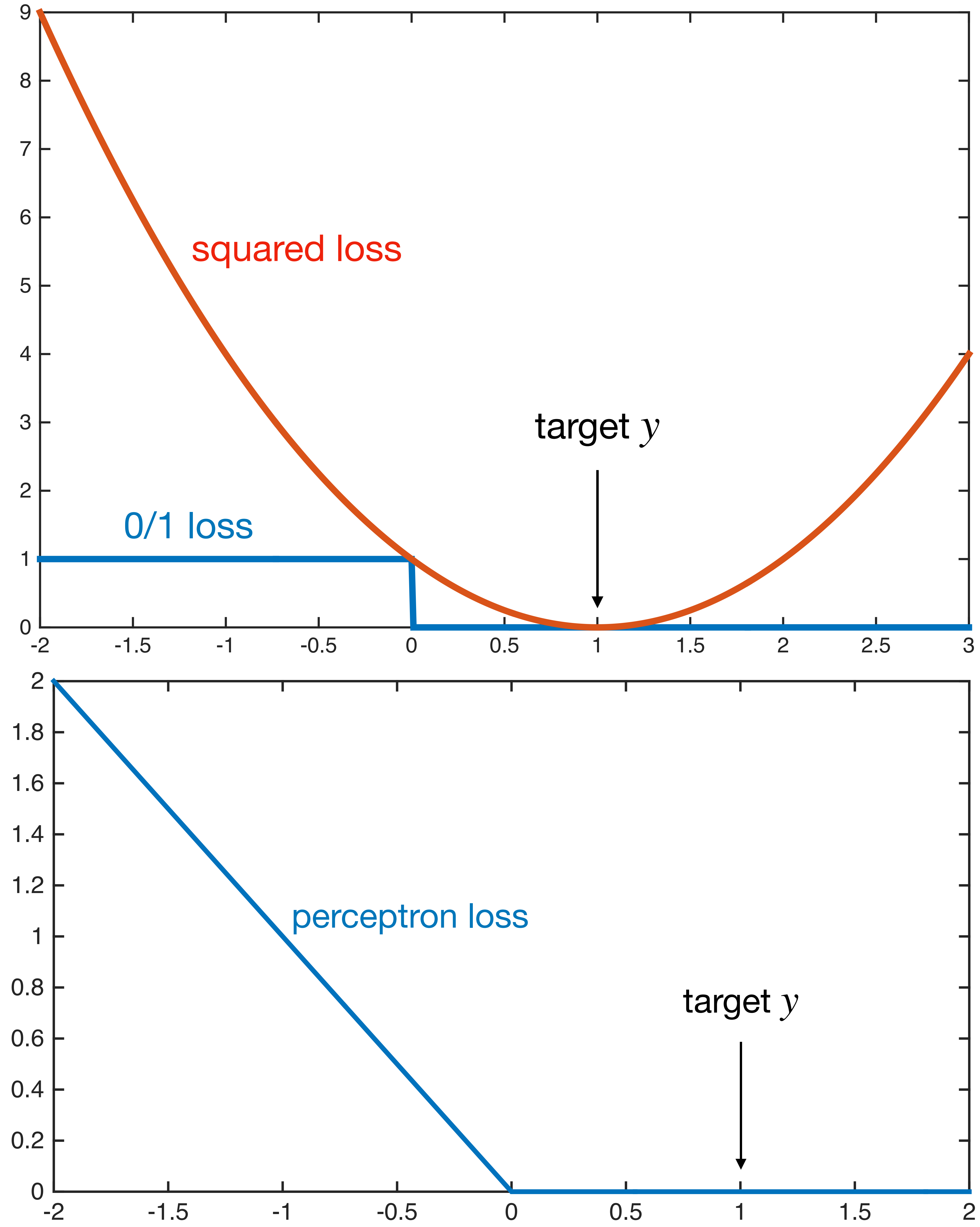
- Optimize raw score using another loss (e.g. squared loss, perceptron loss)

Recap: Squared & perceptron losses

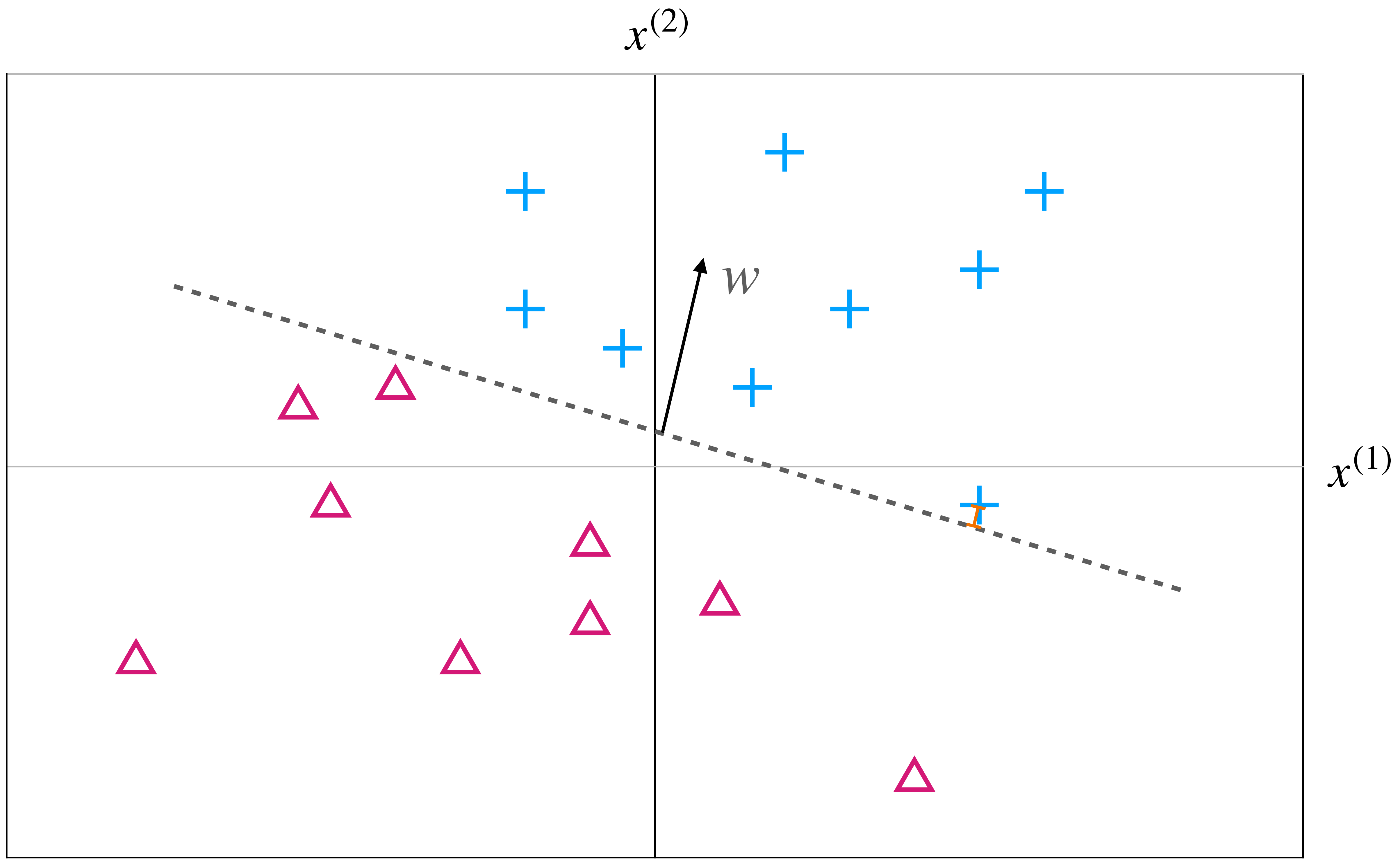
- Squared loss: Usually not good: can fail even on linearly separable data

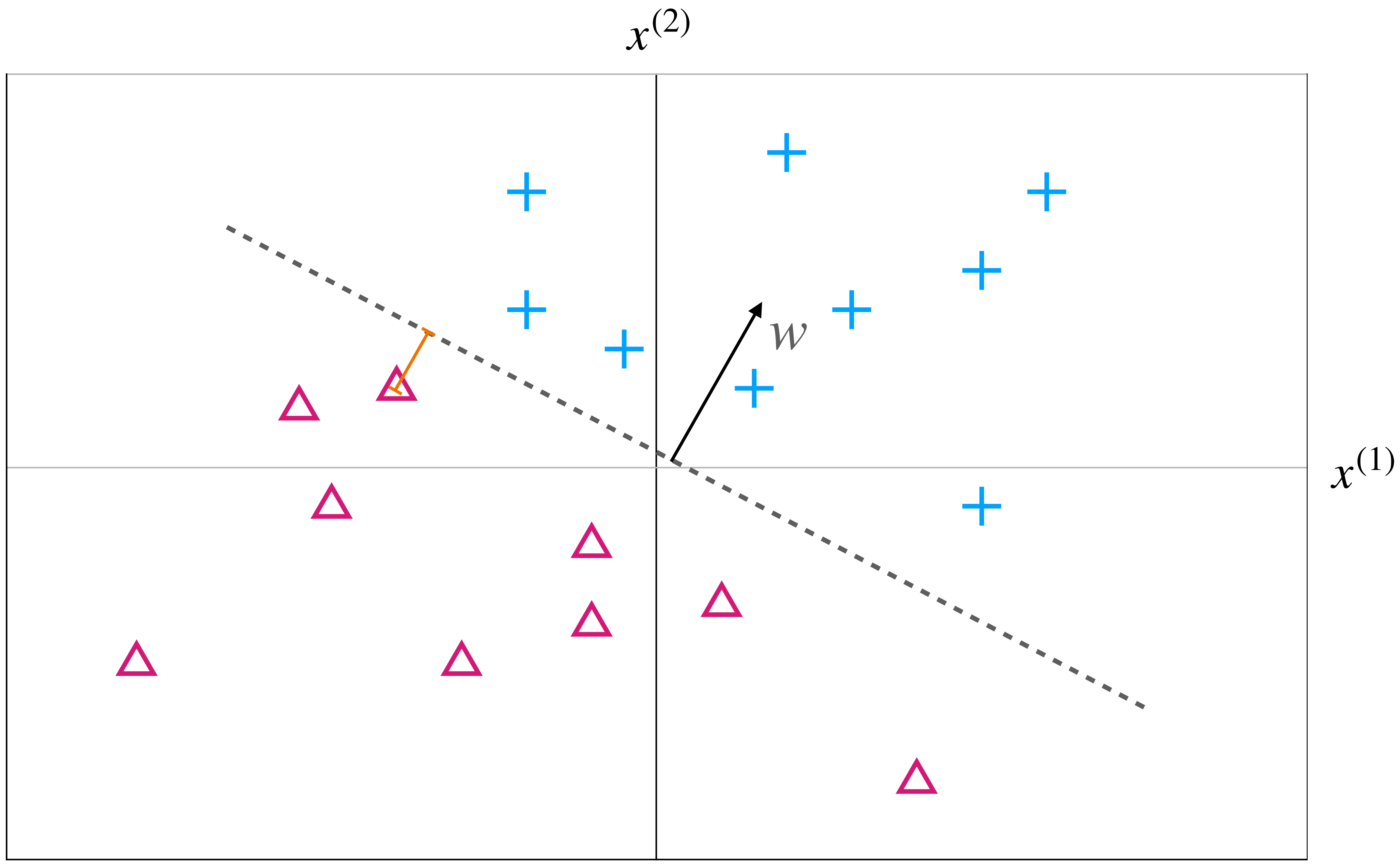
- Perceptron loss: Reproduces perceptron update

$$w(t+1) = \begin{cases} w(t) & \text{correct} \\ w(t) + yx & \text{otherwise} \end{cases}$$



Which classifier is better?

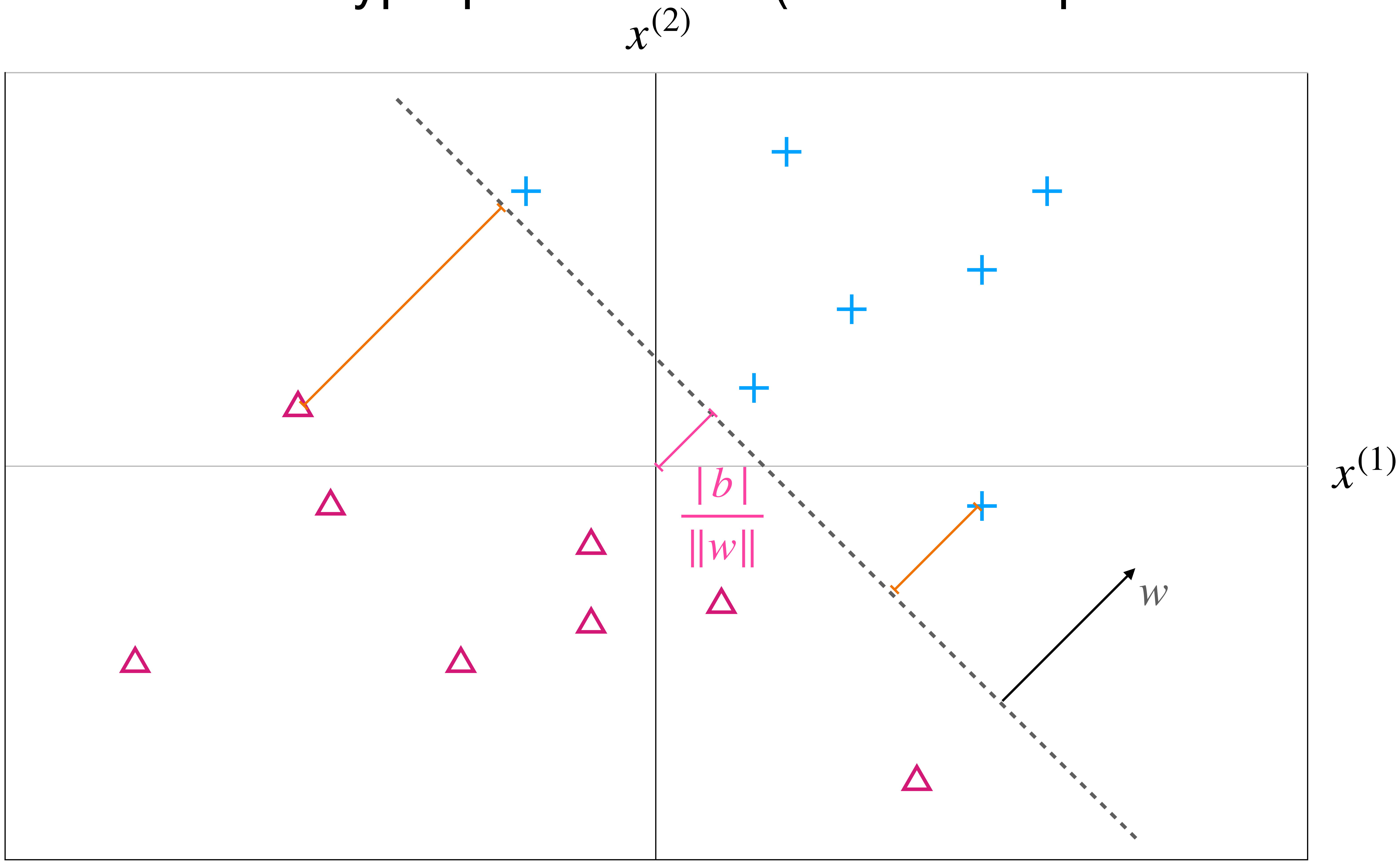




The classifier with a larger margin!
(more likely to generalize better)

Distance from a hyperplane

- $w^\intercal x + b = 0$ defines a hyperplane in $\mathbb{R}^D$ (affine subspace of dimension $D - 1$)



Raw score of linear model

Distance: $\dfrac{|w^\intercal x + b|}{\|w\|}$ ← L^2 norm: $\|w\| = \sqrt{w^\intercal w}$ → Signed distance: $\dfrac{w^\intercal x + b}{\|w\|}$

How to maximize margin?

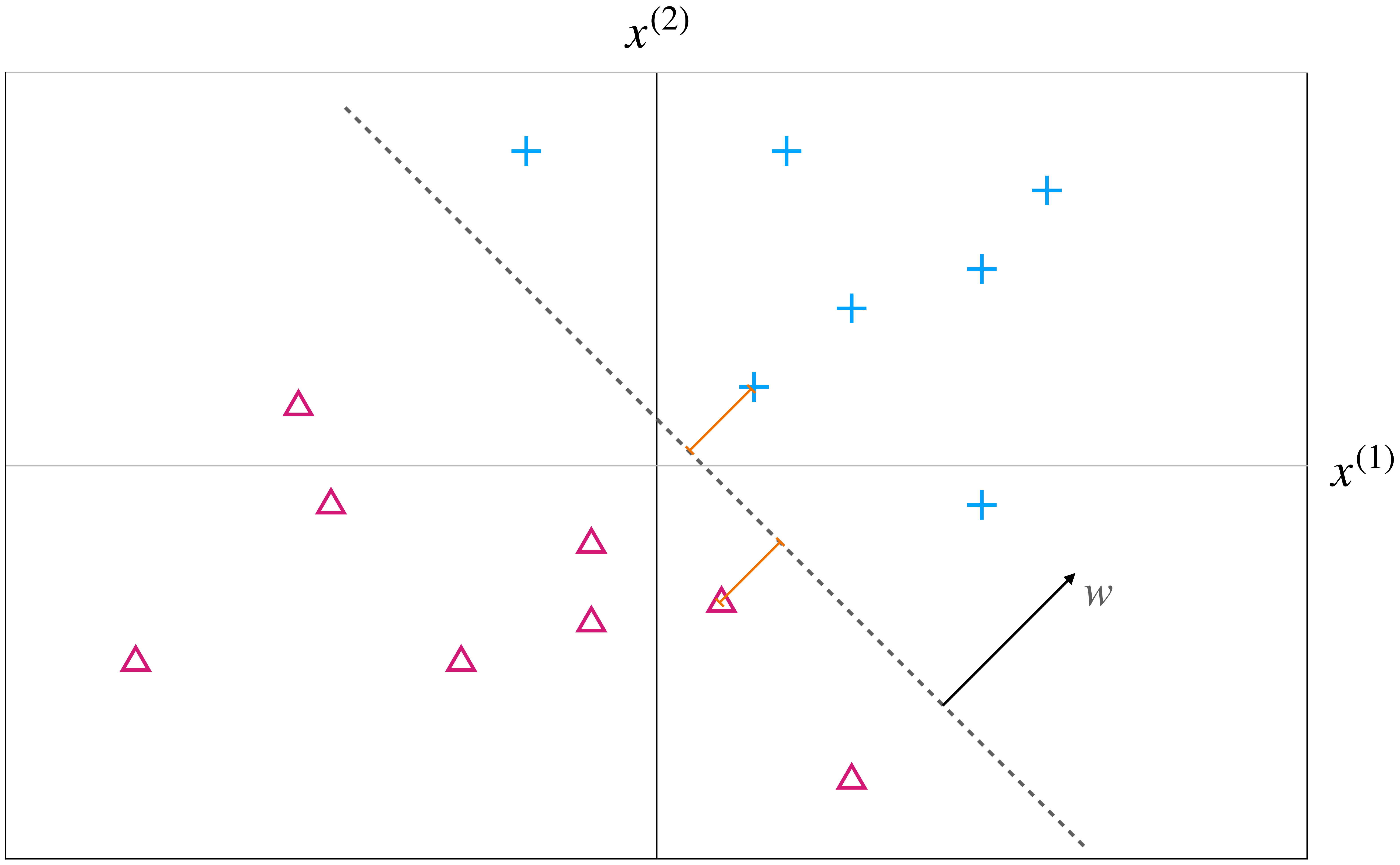
- Assuming linearly separable

- Choose w, b that maximize

$$\min_{(x,y)\in S} \frac{y(w^\intercal x + b)}{||w||}$$

- Equivalently, minimize $||w||^2$ with the constraint

$$\min_{(x,y)\in S} y(w^\intercal x + b) = 1$$



See Bishop Ch. 7 for more details

Support vector machine (SVM)

- Assuming linearly separable: Max margin classifier

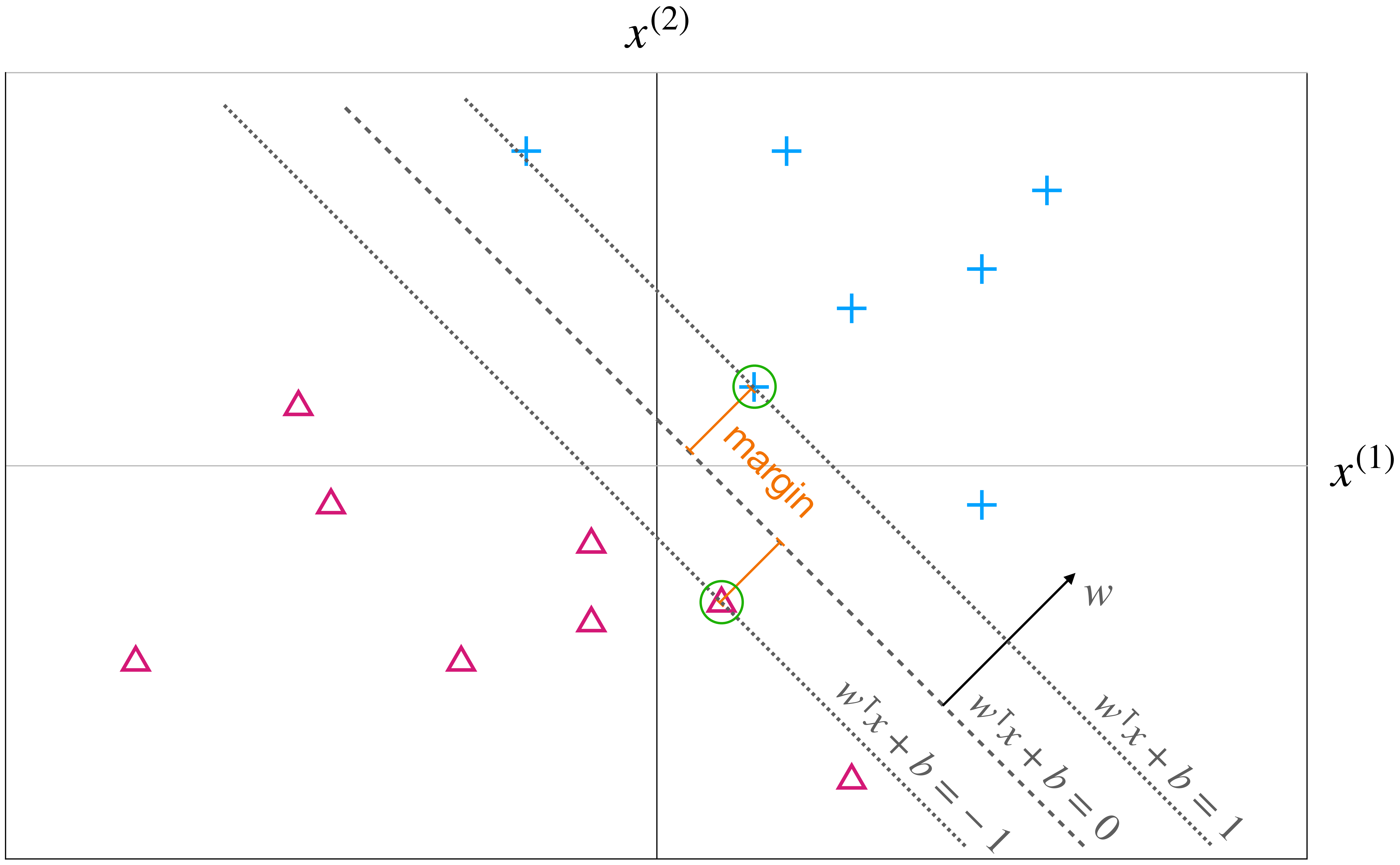
- SVM optimization problem

$$\arg\min_{w,b} \|w\|^2$$

subject to

$$y_i(w^\intercal x_i + b) \geq 1 \ \forall i$$

some constraints are tight
at the optimum (support vectors)



margin = $\dfrac{1}{\|w\|}$

Soft-margin SVM

- Don't assume linearly separable

- Soft-margin SVM optimization problem

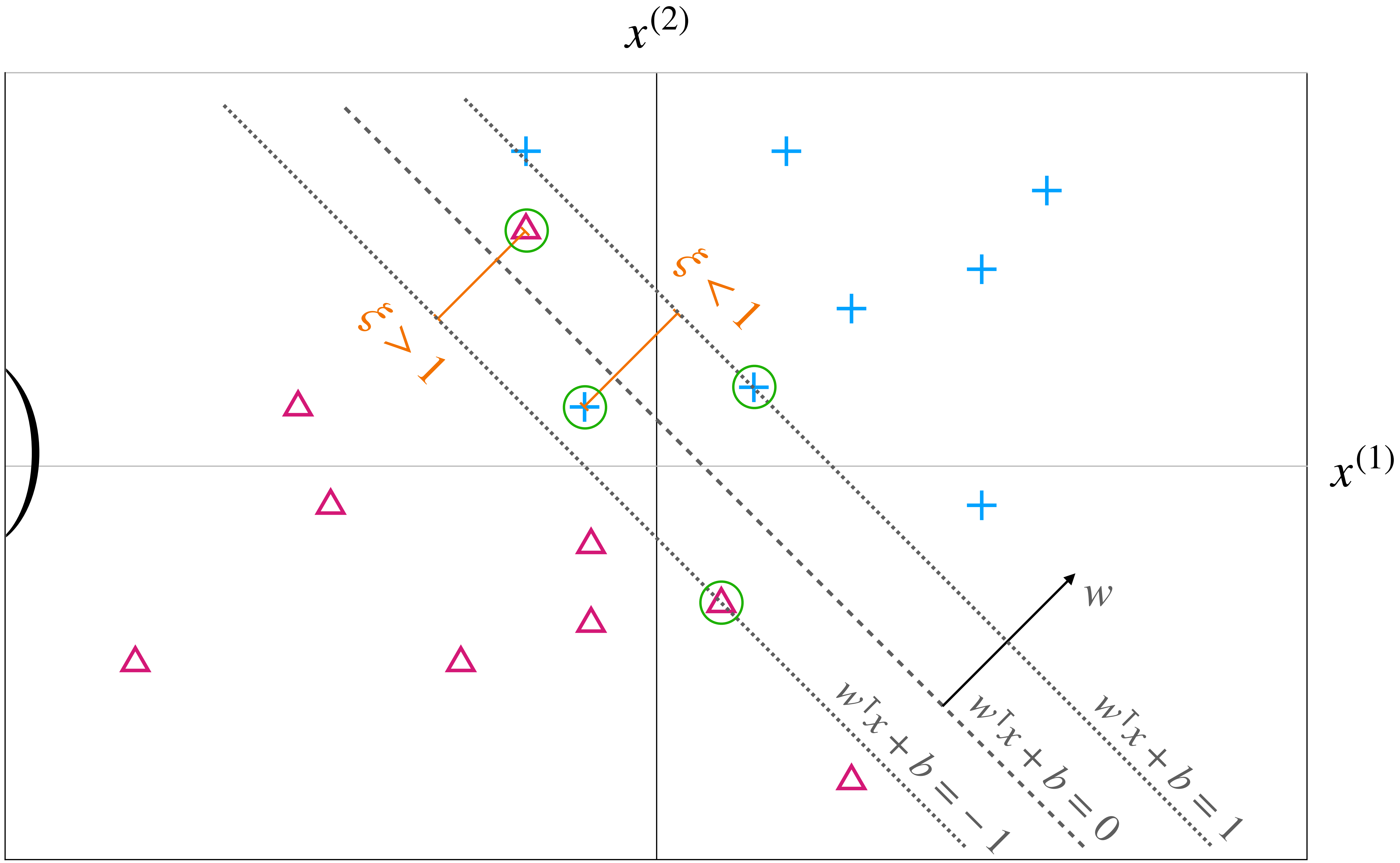
$$\arg\min_{w,b} \left(\|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \right)$$

subject to

$$y_i(w^\intercal x_i + b) \geq 1 - \xi_i \ \forall i$$

$$\xi_i > 0 \ \forall i$$

Slack



C controls the trade-off between size of margins and margin violations

Hinge loss

- Soft-margin SVM optimization problem

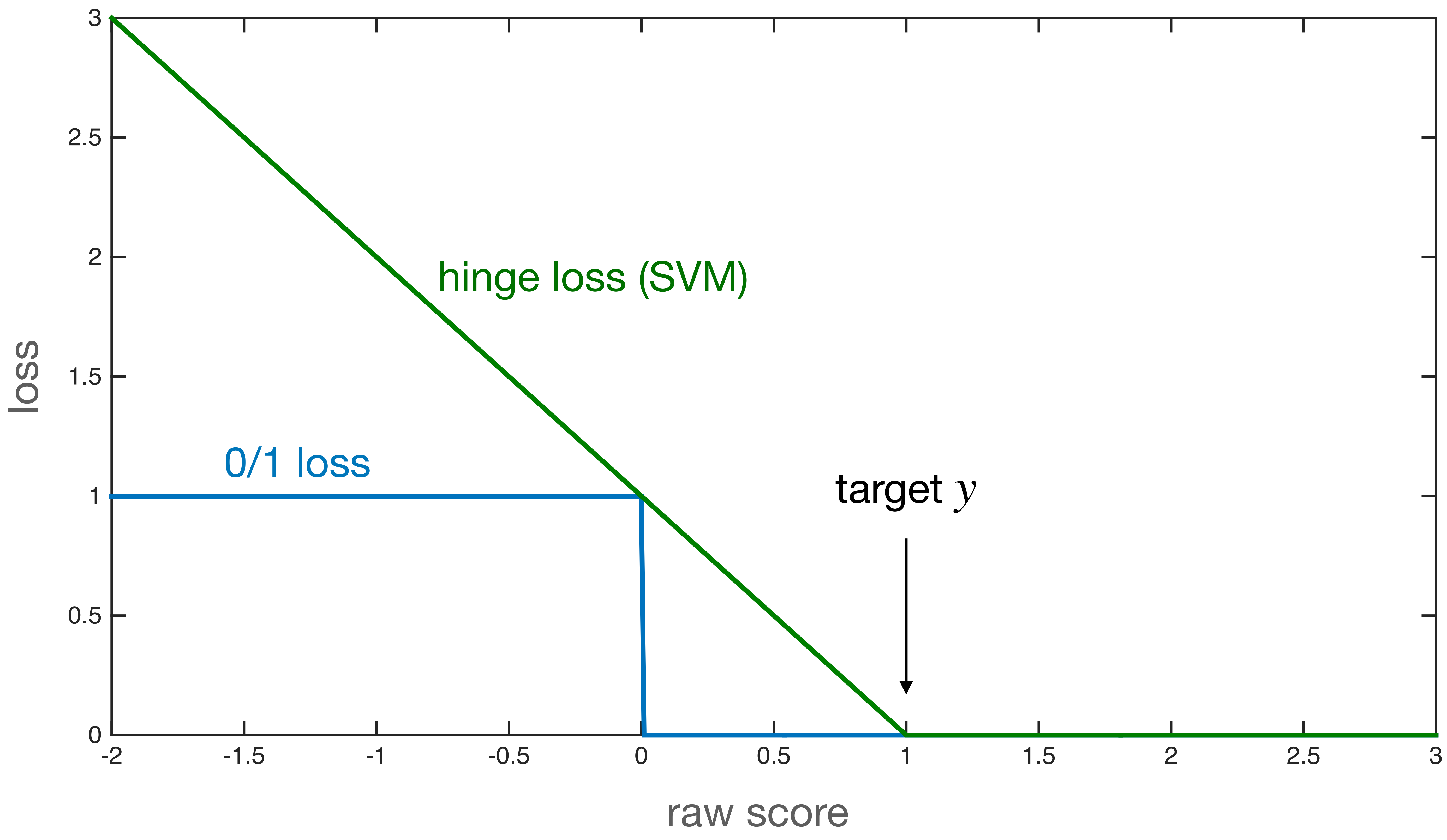
$$\arg\min_{w,b} \|w\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$

subject to

$$y_i(w^\intercal x_i + b) \geq 1 - \xi_i \;\forall i$$
$$\xi_i > 0 \;\forall i$$

$$\Updownarrow$$

- $\xi_i \geq \max(0, 1 - y_i(w^\intercal x_i + b))$



SVM equivalent to minimizing: $\frac{1}{N}\sum_{i=1}^{N}\underbrace{\max(0, 1 - y_i(w^\intercal x_i + b))}_{\text{Hinge loss}} + \underbrace{\frac{1}{C}\|w\|^2}_{\text{Regularization}}$

Kernel trick preface

- Theorem:

  Optimal w has the form $w = \sum_{i=1}^{N} \alpha_i x_i$ ($\alpha_i \neq 0$ only if x_i is a support vector)

- Alternative (dual) formulation of SVM:

$$\arg\min_{\alpha_i, b} \sum_{i,j} \alpha_i \alpha_j x_i^\intercal x_j + \frac{C}{N} \sum_{i=1}^{N} \xi_i \quad \text{subject to} \quad \sum_{j=1}^{N} y_i(\alpha_j x_j^\intercal x_i + b) \geq 1 - \xi_i \ \forall i$$

$$\xi_i \geq 0 \ \forall i$$

- Predictions only depend on support vectors

- Optimization and predictions only depend on dot products $x^\intercal x'$ of input vector x, x'

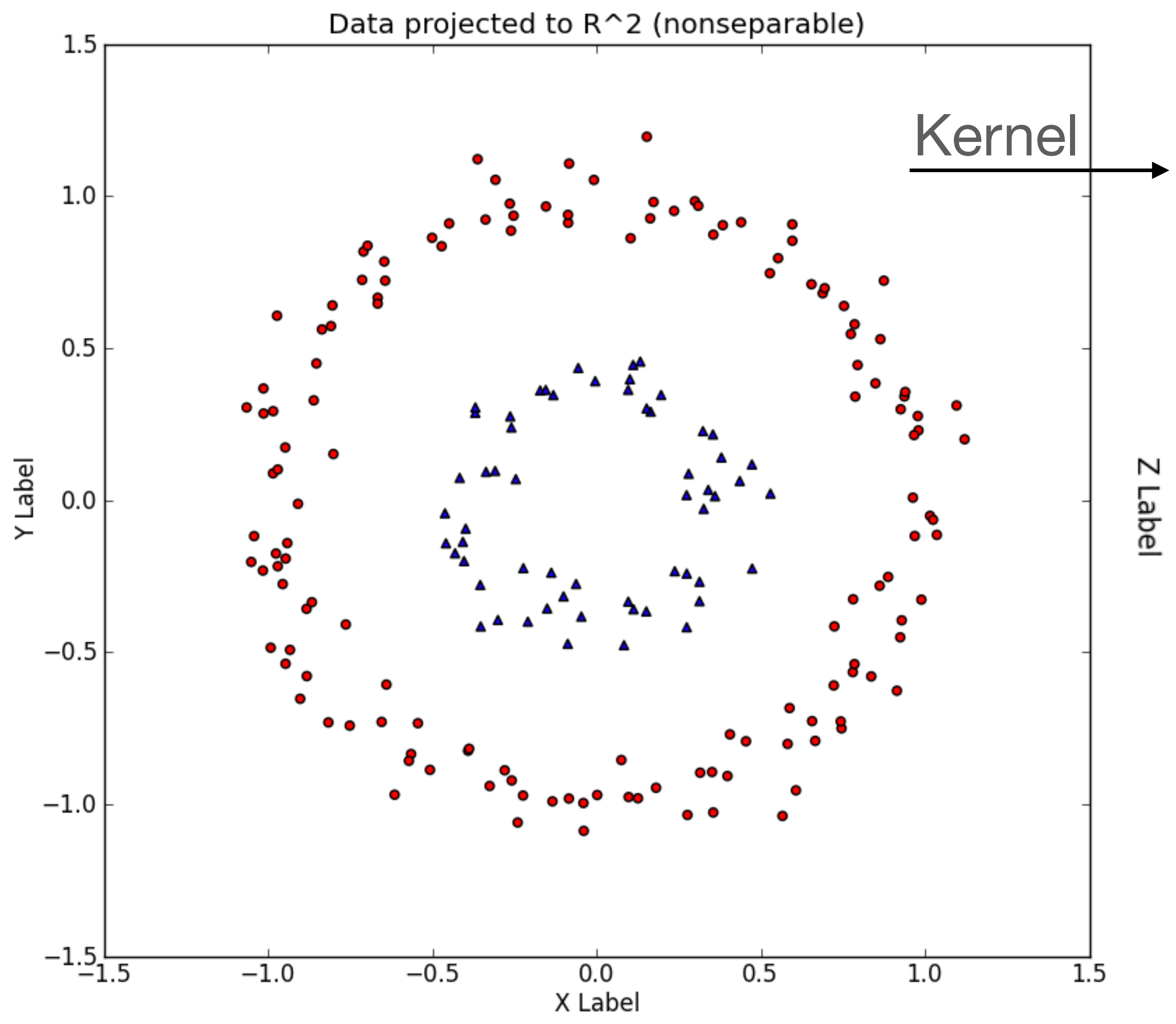
Kernel trick

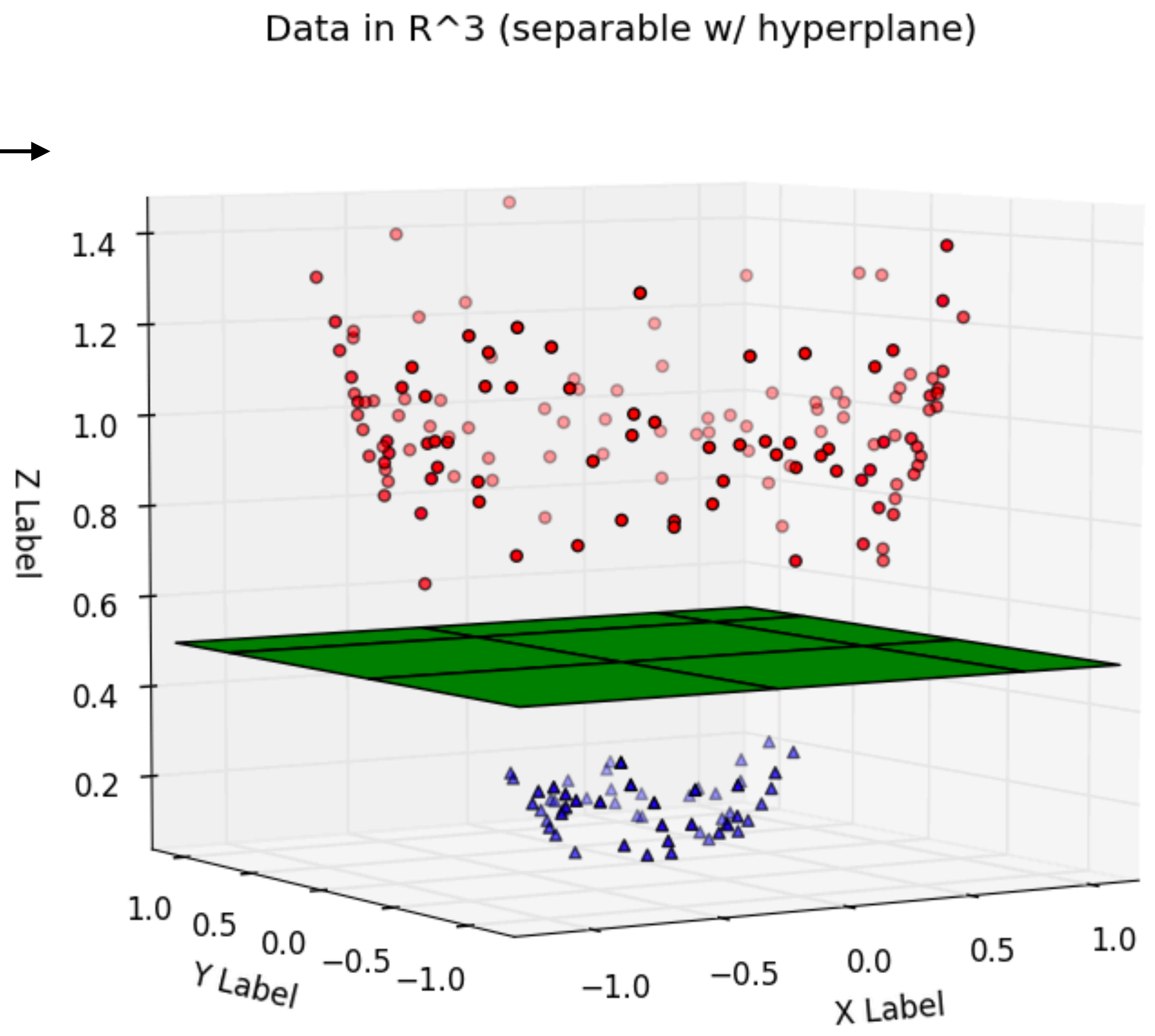
- Can replace dot products $x^\intercal x'$ with arbitrary kernels $k(x, x')$!
  - Polynomial kernel: $k(x, x') = (x^\intercal x' + c)^d$
  - Gaussian kernel: $k(x, x') = \exp(-\|x - x'\|^2 / 2\sigma^2)$

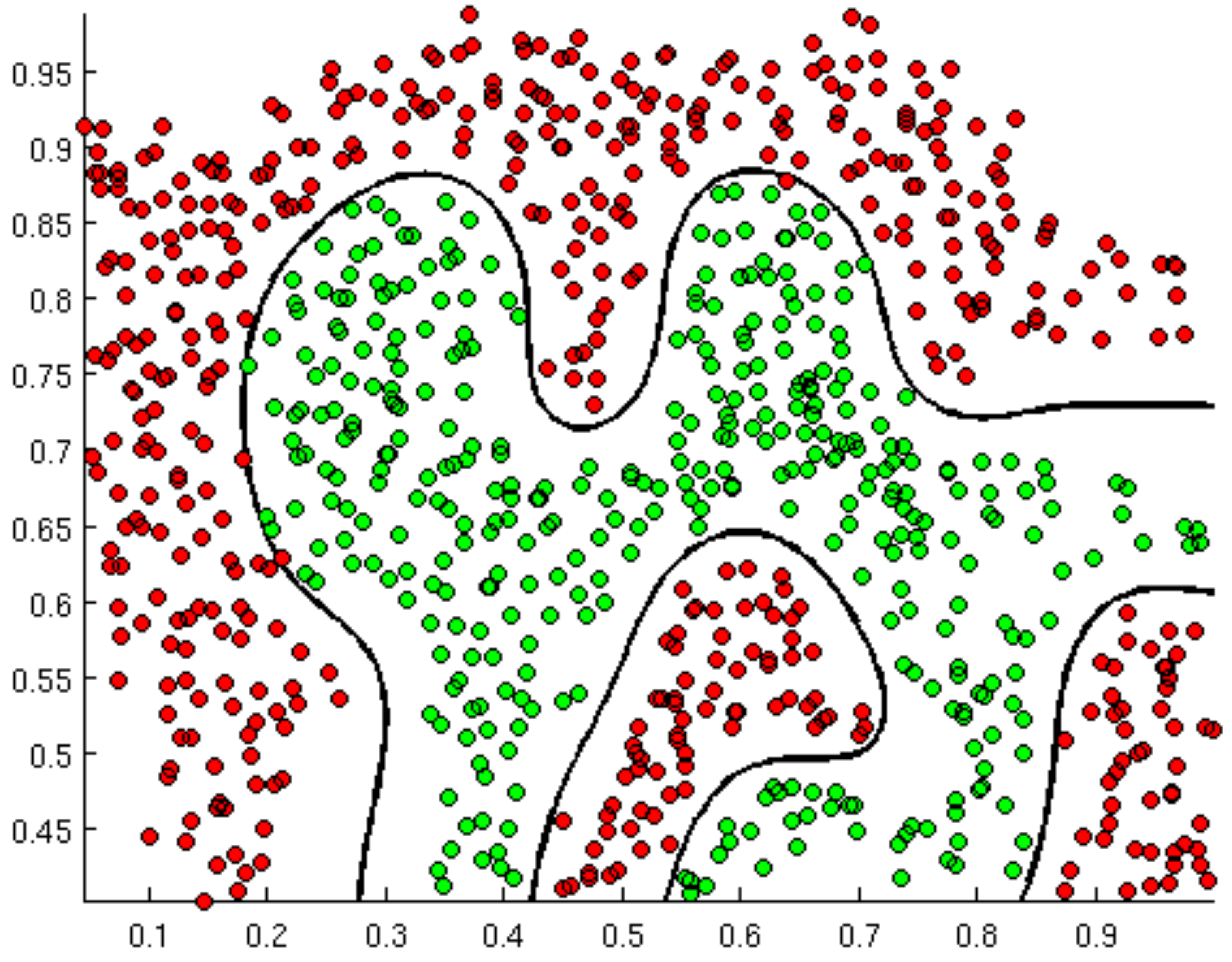
A kernel formally behaves as

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

(without explicitly computing $\phi(x), \phi(x')$)

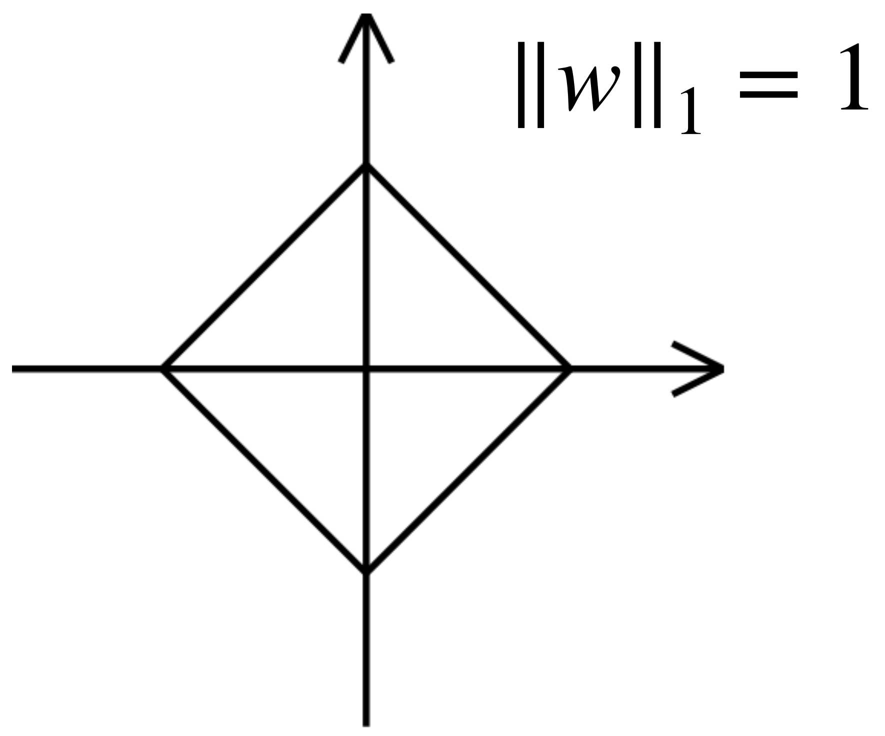


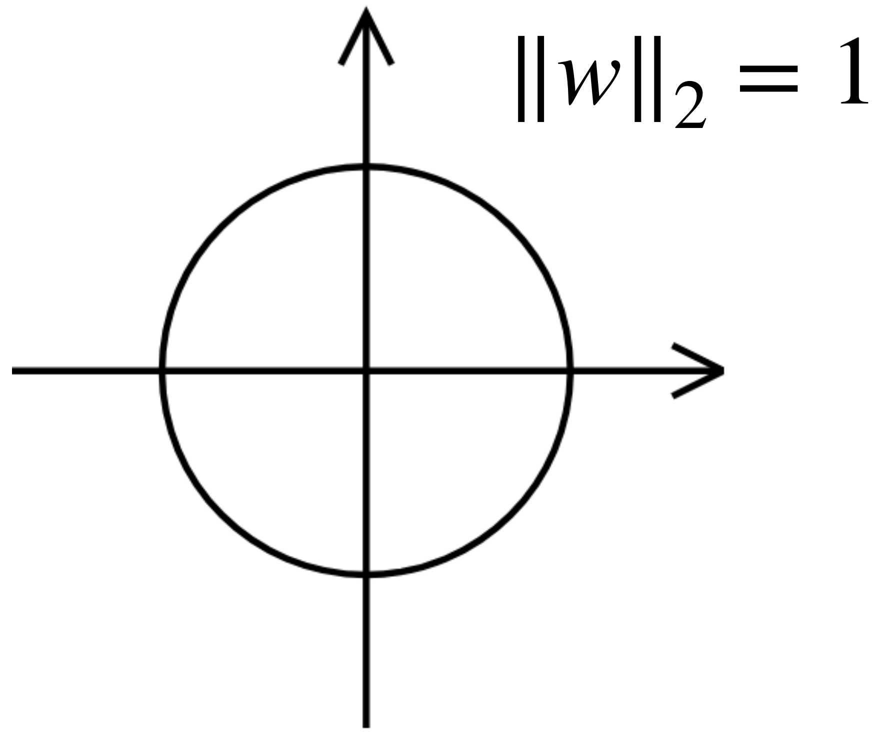


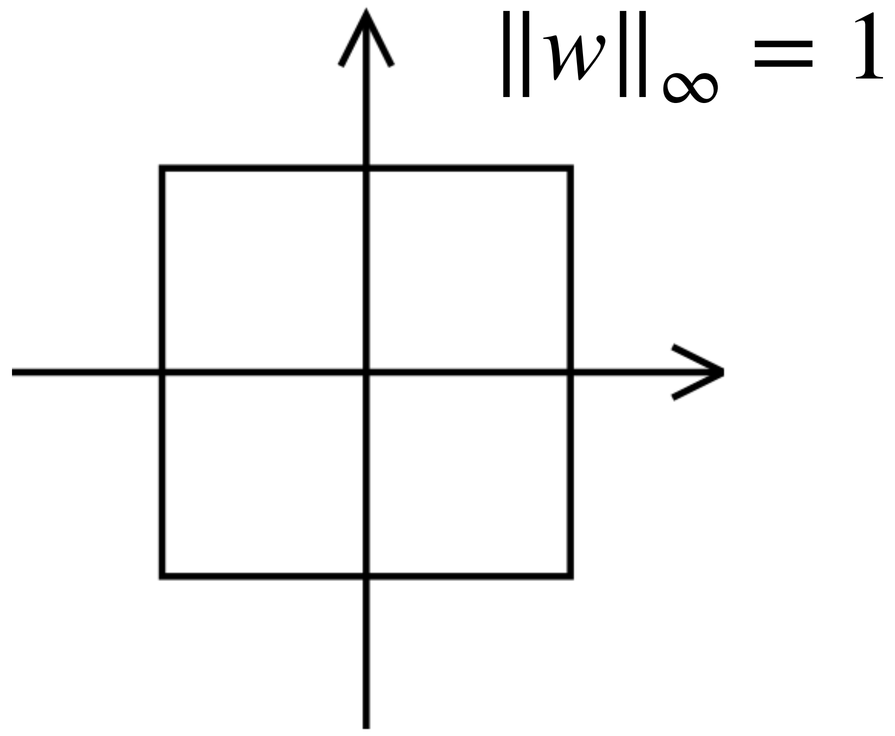


Norms

- L^0 norm: $\|w\|_0$ = number of nonzero entries of w (not a norm!)
- L^1 norm: $\|w\|_1 = \sum_j |w^{(j)}|$ (sum of abs. values of entries)
- L^2 norm: $\|w\| = \|w\|_2 = \sqrt{w^\intercal w} = \sqrt{\sum_j (w^{(j)})^2}$
- L^∞ norm: $\|w\|_\infty = \max_j |w^{(j)}|$ (max abs. value of entries)



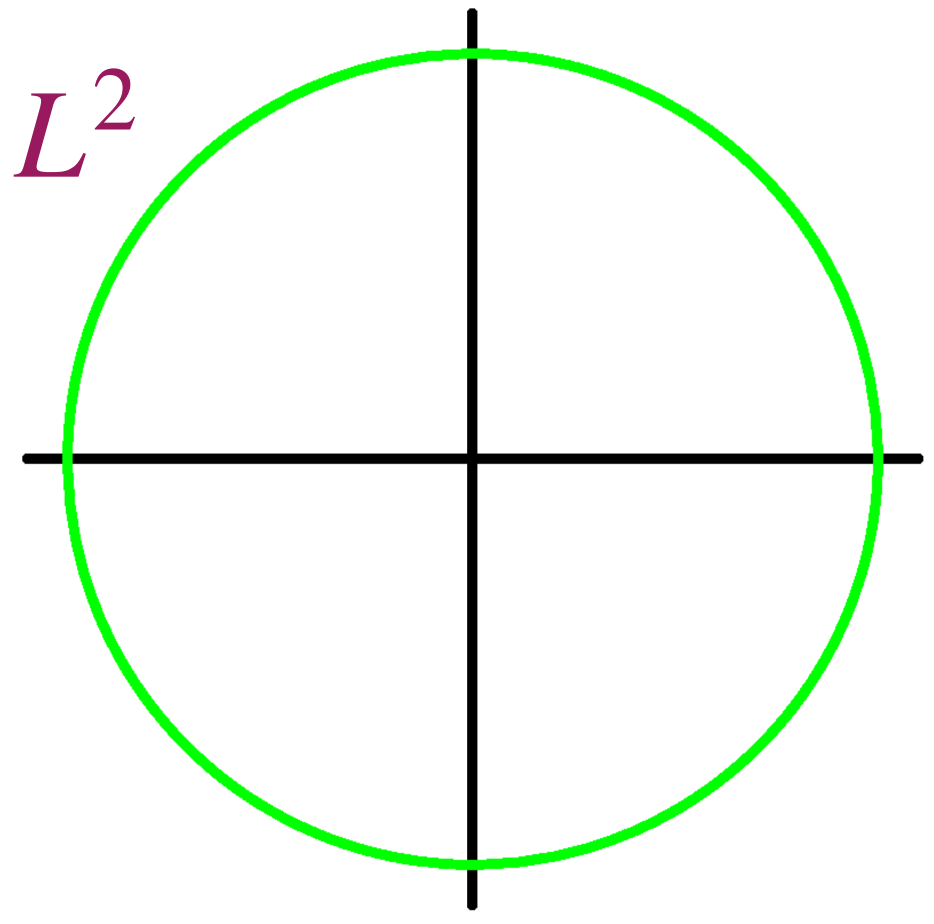




Regularization

- L^2-regularization for regression (ridge regression)

$$\arg\min_{w} \sum_{i=1}^{N} \underbrace{(y_i - w^\intercal x_i)^2}_{\text{Training loss}} + \underbrace{\lambda \|w\|^2}_{\text{Regularization}}$$



$$\|w\|_2 = c$$

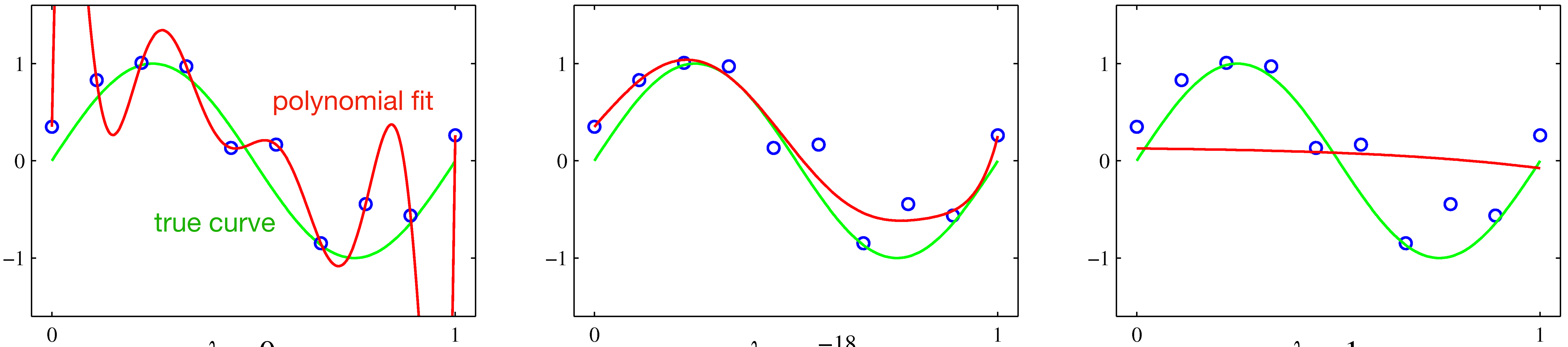
- Trades off model complexity vs. training loss

- Each choice of λ gives a model class (larger λ constrains w to be smaller)

- Regularization can be combined with any loss

Regularization example

- Example: polynomial curve fitting (10 points, degree 9)



$\lambda = 0$ | $\lambda = e^{-18}$ | $\lambda = 1$

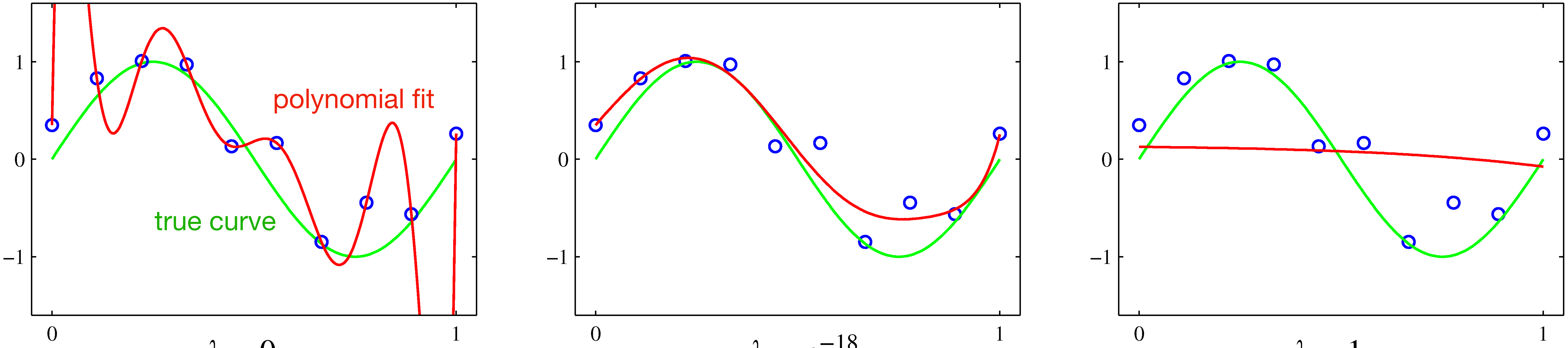
low λ: overfitting
(high variance)

high λ: underfitting
(high bias)

$$w = \begin{bmatrix} 0.35 \\ 232.37 \\ -5321.83 \\ 48568.31 \\ -231639.30 \\ 640042.26 \\ -1061800.52 \\ 1042400.18 \\ -557682.99 \\ 125201.43 \end{bmatrix} \qquad w = \begin{bmatrix} 0.35 \\ 4.74 \\ -0.77 \\ -31.97 \\ -3.89 \\ 55.28 \\ 41.32 \\ -45.95 \\ -91.53 \\ 72.68 \end{bmatrix} \qquad w = \begin{bmatrix} 0.13 \\ -0.05 \\ -0.06 \\ -0.05 \\ -0.03 \\ -0.02 \\ -0.01 \\ 0.00 \\ 0.00 \\ 0.01 \end{bmatrix}$$

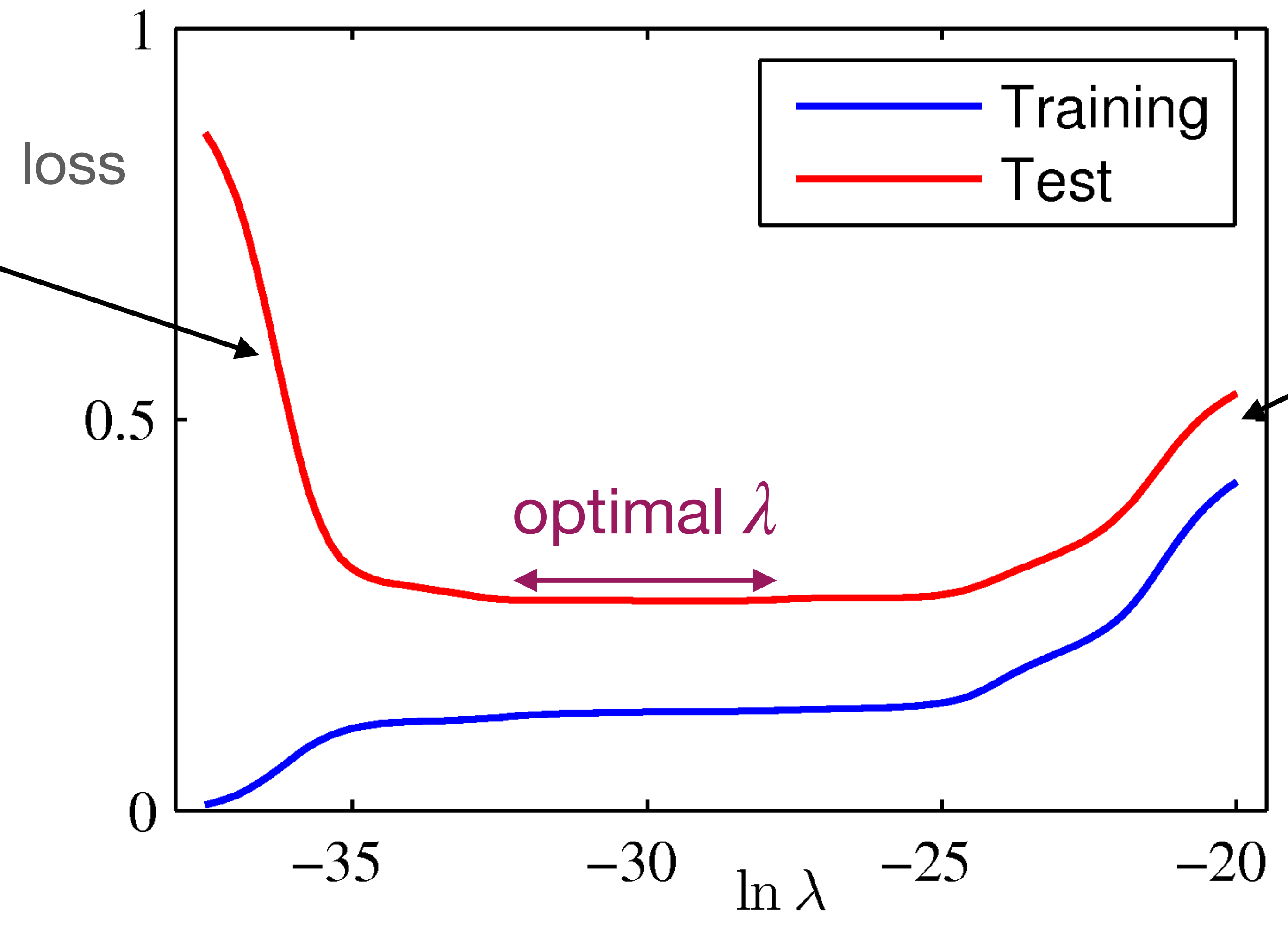
Regularization example

- Example: polynomial curve fitting (10 points, degree 9)



$\lambda = 0$ $\lambda = e^{-18}$ $\lambda = 1$

low λ: overfitting
(high variance)



high λ: underfitting
(high bias)

The optimal choice of λ depends on the training size N

Model class interpretation

- Minimize

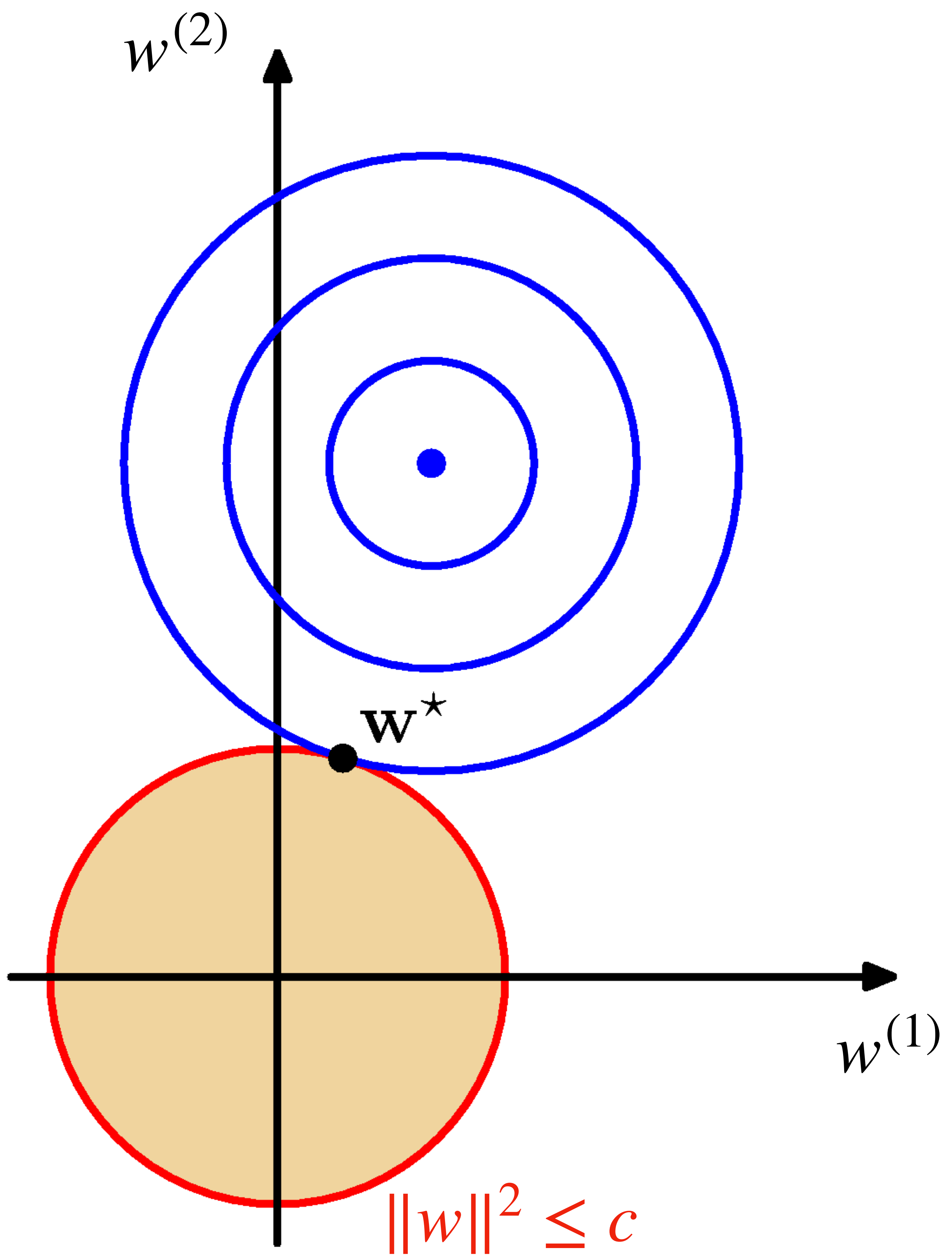
$$\sum_{i=1}^{N} (y_i - w^\intercal x_i)^2 + \lambda \|w\|^2$$

$\Updownarrow$ Equivalent

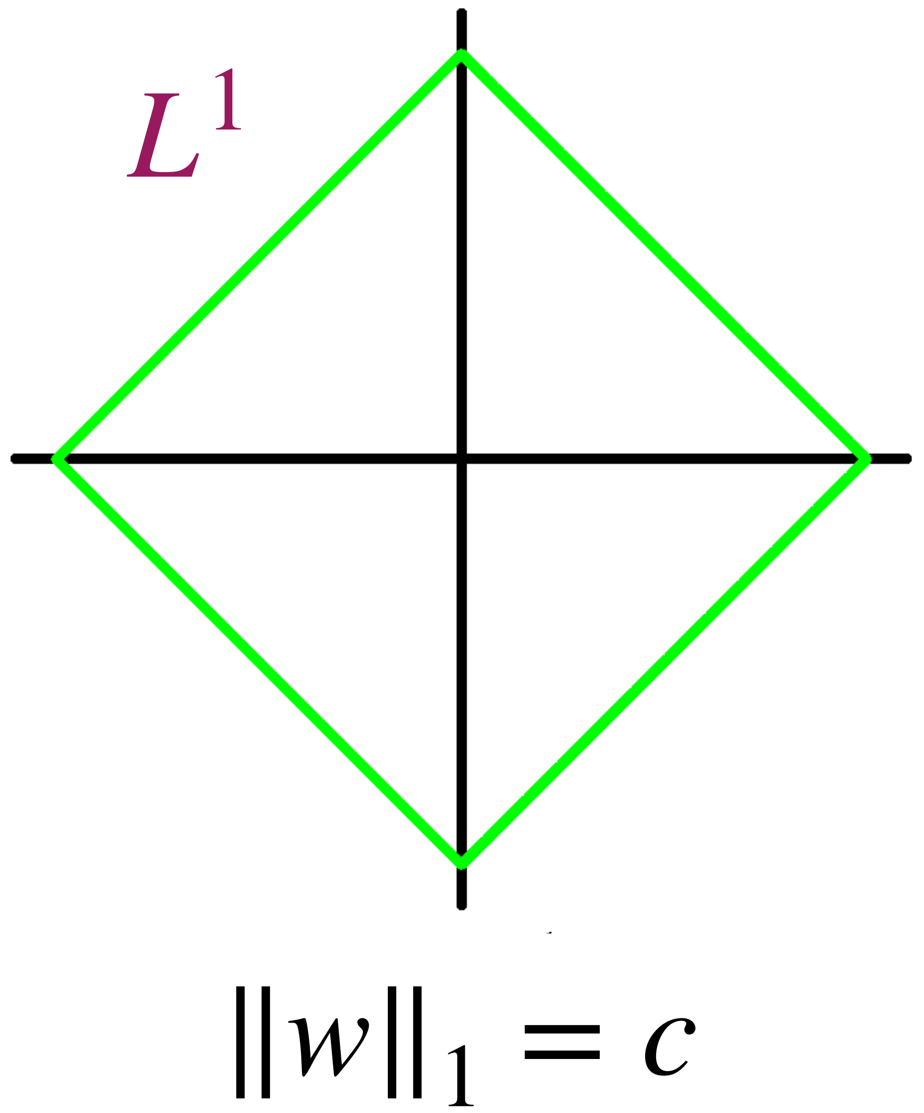
- Minimize

$$\sum_{i=1}^{N} (y_i - w^\intercal x_i)^2 \ \text{ with constraint } \ \|w\|^2 < c$$

(using Lagrange multipliers)



Lasso & sparsity



- L^1-regularization for (lasso) regression

$$\underset{w}{\arg\min} \sum_{i=1}^{N} \underbrace{(y_i - w^\intercal x_i)^2}_{\text{Training loss}} + \underbrace{\lambda \|w\|_1}_{\text{Regularization}}$$

- We say that w is sparse if several of its entries are zero ($\|w\|_0$ is small)

- Finding a sparse w is useful, e.g. time/memory efficiency (only some entries are needed to compute $w^\intercal x$)

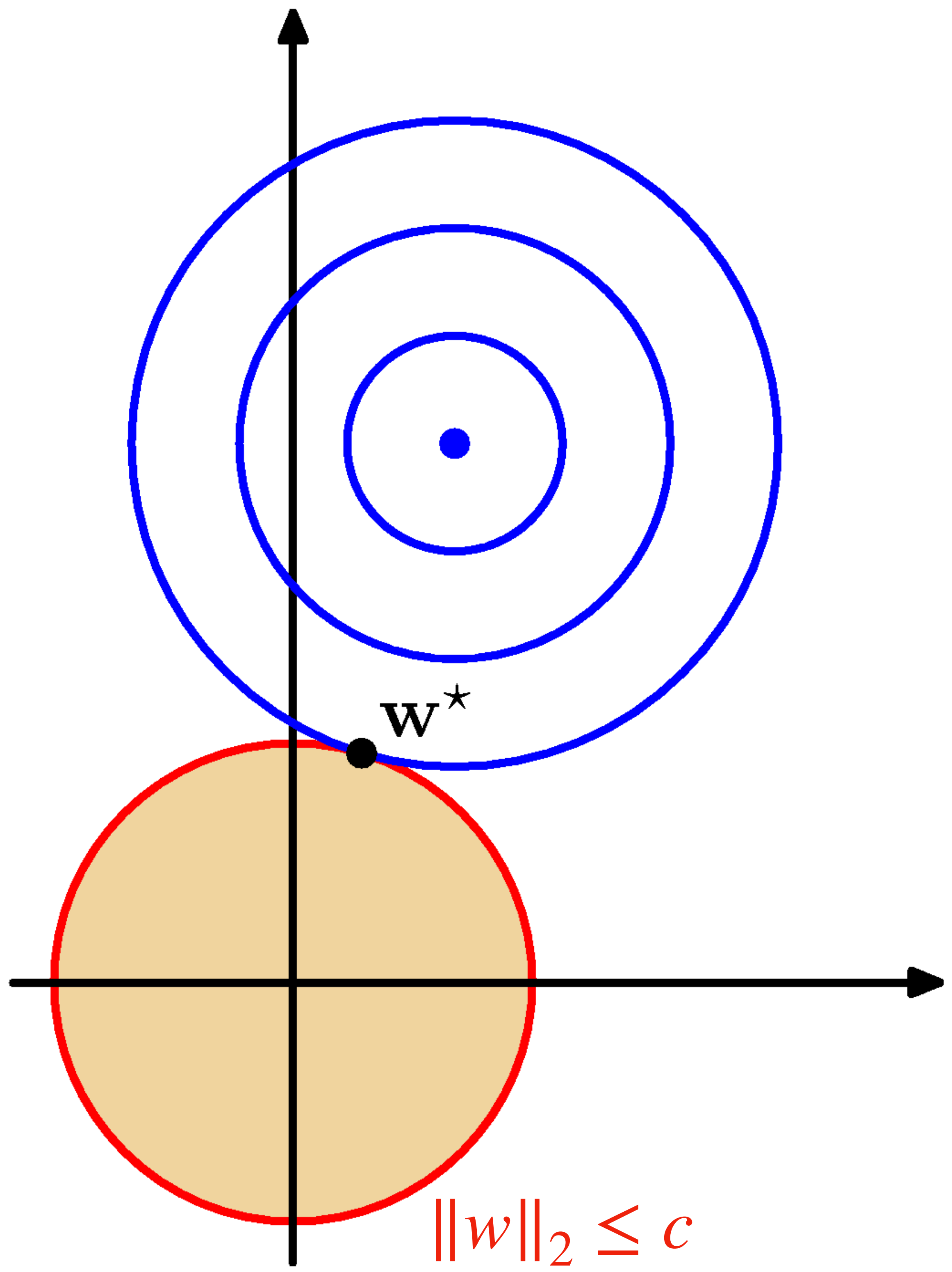
- We cannot use L^0 regularization directly (not continuous)

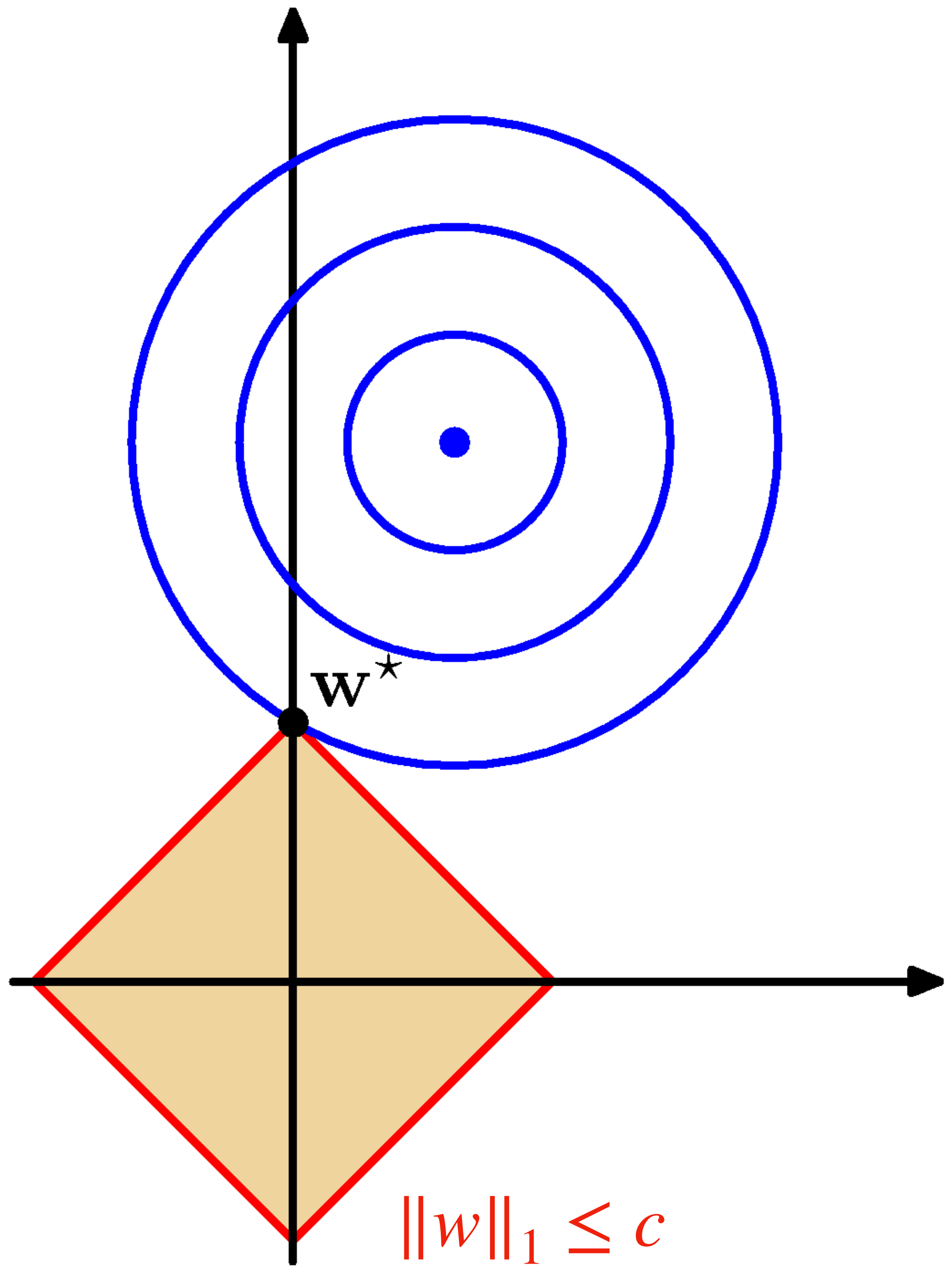
- However, L^1 regularization (lasso) induces **sparsity**!

Model class interpretation: L^1 vs. L^2

$$\arg\min_w \sum_{i=1}^{N} \underbrace{(y_i - w^\intercal x_i)^2}_{\text{Training loss}} + \underbrace{\lambda \|w\|^2}_{L^2 \text{ regularization}}$$

$$\arg\min_w \sum_{i=1}^{N} \underbrace{(y_i - w^\intercal x_i)^2}_{\text{Training loss}} + \underbrace{\lambda \|w\|_1}_{L^1 \text{ regularization}}$$





Updated supervised learning pipeline

- Training dataset: $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x \in \mathbb{R}^D$ and $y \in \mathbb{R}$

For regression

- Model / hypothesis class: $f(x \mid w) = w^\intercal x$ (linear models)

or $\phi(x)$ instead of x

- Loss function: $L(y, y') = (y - y')^2$ (squared loss)
- Optimization algorithm: SGD with regularization (L^1 or L^2)
- Cross validation and model selection:

Select λ

- Testing and deployment

Probabilistic approach

Parametrized by w

- Idea: Model a probability distribution $p(y \mid x; w)$ of labels y given inputs x
- Choose a form for $p(y \mid x; w)$ (different for regression and classification)
- Write the likelihood of w, i.e. the probability of observing the labels y_i of the training dataset S given the inputs x_i:

$$p(S \mid w) = \prod_{i=1}^{N} p(y_i \mid x_i)$$

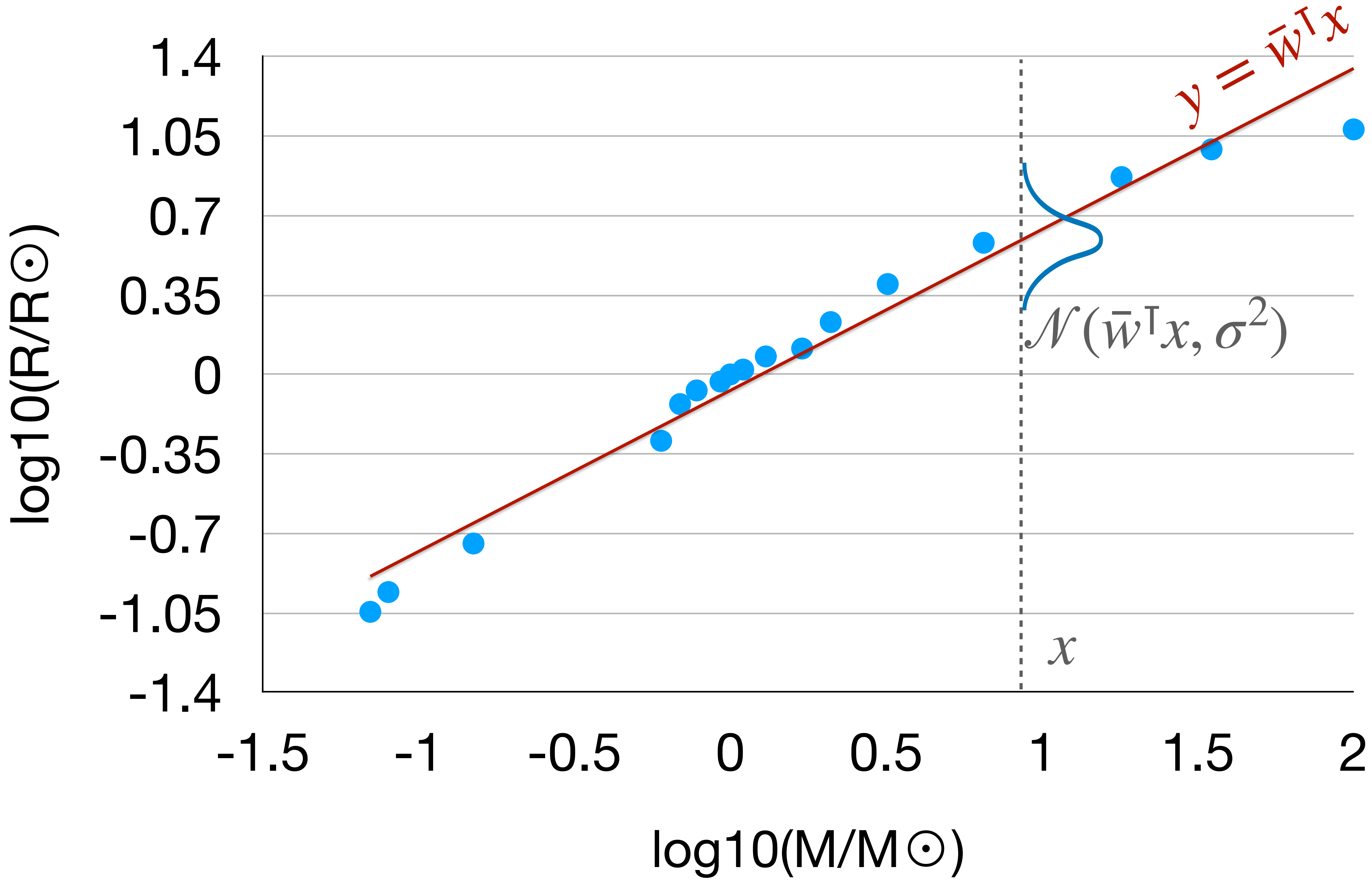
Assuming training examples are independent

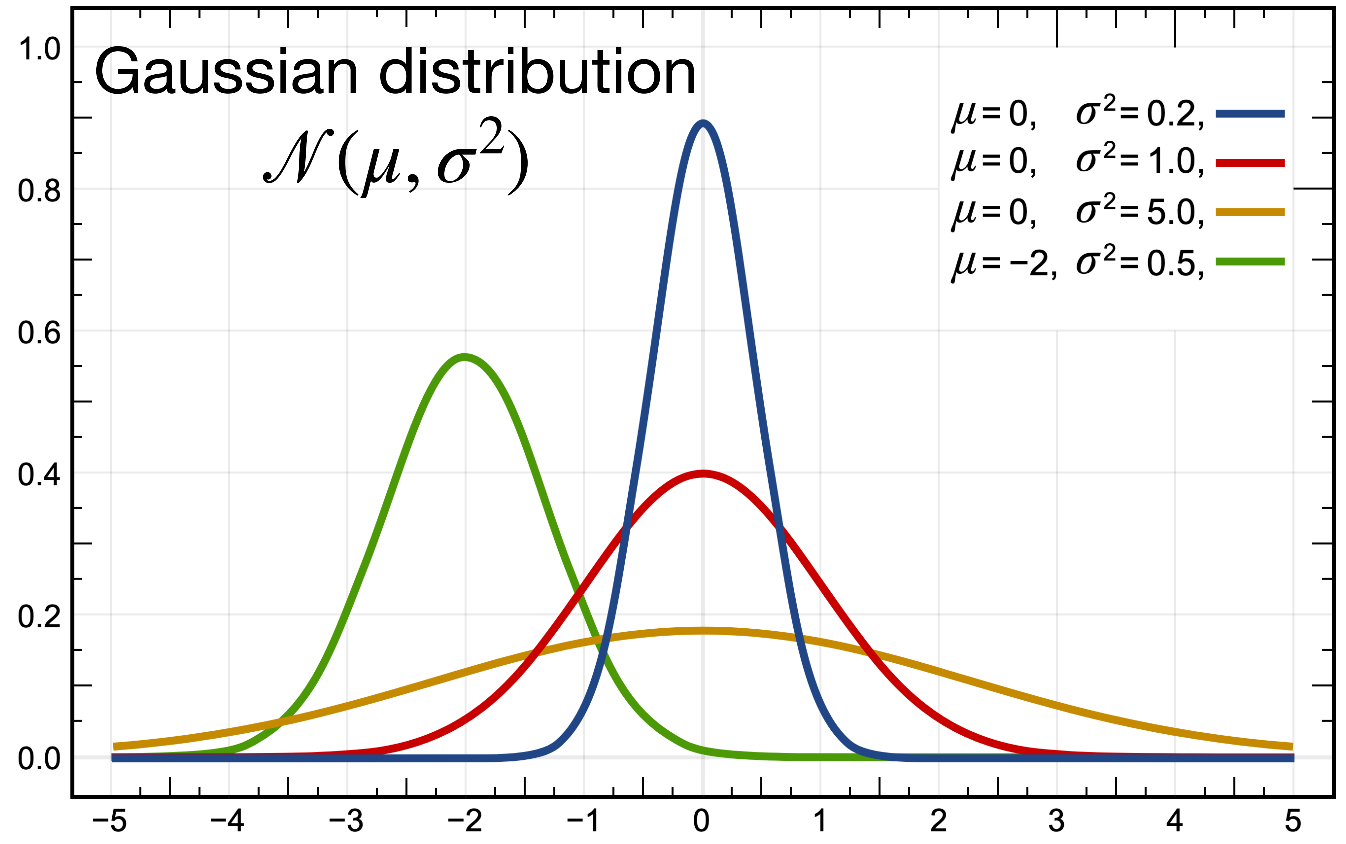
- Maximum likelihood estimation (MLE): find w that maximizes the (log) likelihood:

$$\log p(S \mid w) = \sum_{i=1}^{N} \log p(y_i \mid x_i; w) = -l(w)$$

Equivalently, minimize the loss function!

Linear regression revisited





- Assume labels y are distributed as $\mathcal{N}(\bar{w}^\intercal x, \sigma^2)$

True value of w

Fixed (unknown) variance

- Likelihood of w (assuming training samples are i.i.d.):

$$p(S \mid w) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - w^\intercal x_i)^2)}{2\sigma^2}\right)$$

- Maximizing the likelihood is equivalent to minimizing

$$-\log P(S \mid w) \simeq \sum_{i=1}^{N} (y_i - w^\intercal x_i)^2$$

Binary classification revisited

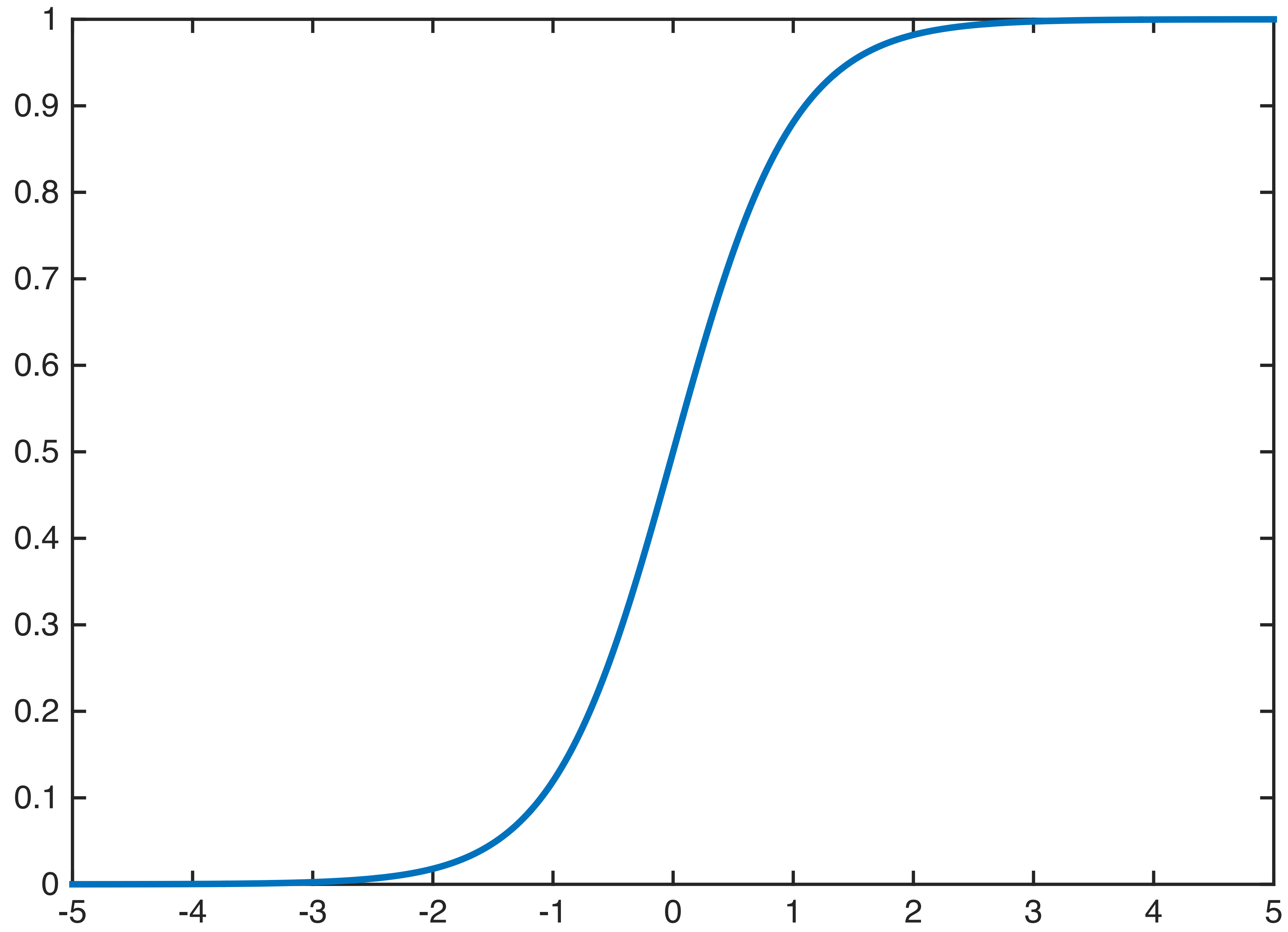
- Linear model for binary classification: $f(x \mid w) = \text{sign}(\underbrace{w^\intercal x}_{\text{Raw score}}) \in \{\overset{+}{+1}, \overset{\triangle}{-1}\}$

- Idea: raw score to model the probability of each class

$\sigma(w^\intercal x) \approx$ probability that $y = +1$

Logistic/sigmoid function $\sigma : \mathbb{R} \to (0,1)$

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



$a = w^\intercal x$

What is the right loss function?

- Assume that the true probability that $y = +1$ given x is $\sigma(\bar{w}^\intercal x)$ and that

  $p(y \mid \sigma(\bar{w}^\intercal x))$ is a Bernoulli distribution

  True value of w

- Likelihood of w:

  Only one of these two terms appears depending on y_i

$$p(S \mid w) = \prod_{i=1}^{N} \sigma(w^\intercal x_i)^{\delta_{\{y_i=+1\}}} (1 - \sigma(w^\intercal x_i))^{\delta_{\{y_i=-1\}}}$$

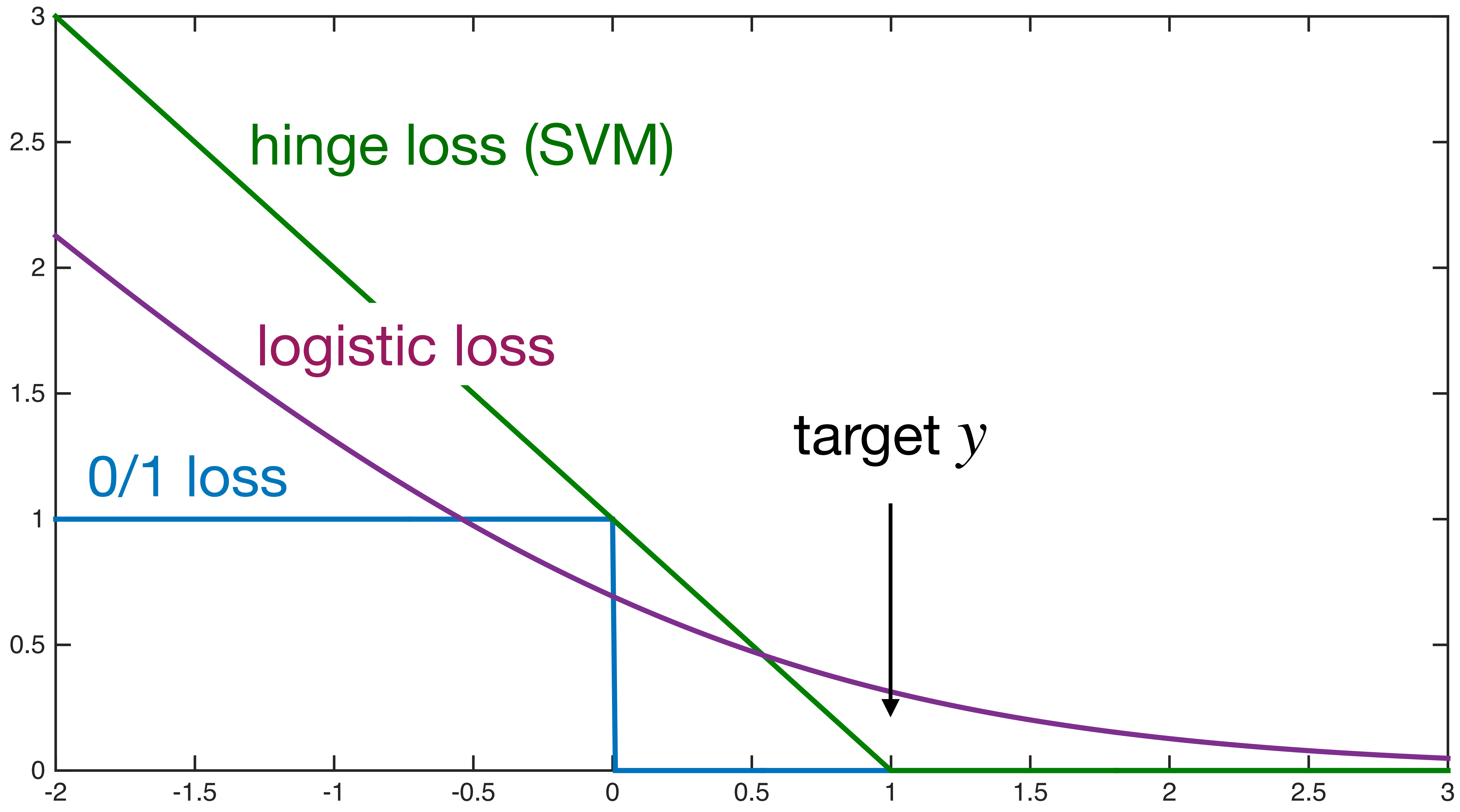
- Negative log likelihood of w a.k.a. logistic / log / binary cross-entropy loss:

$$-\log p(S \mid w) = -\sum_{i=1}^{N} \delta_{\{y_i=+1\}} \log \sigma(w^\intercal x_i) + \delta_{\{y_i=-1\}} \log(1 - \sigma(w^\intercal x_i))$$

Logistic loss

- Logistic / log / binary cross-entropy loss:

$$L(y, y') = -\,\delta_{\{y=+1\}} \log y' - \delta_{\{y=-1\}} \log(1 - y')$$



Logistic regression update

- Logistic loss: $-\left(\delta_{\{y_i=+1\}} \log \sigma(w^\intercal x_i) + \delta_{\{y_i=-1\}} \log(1-\sigma(w^\intercal x_i))\right)$

- Gradient: $-\left(\underbrace{\delta_{\{y_i=+1\}}}_{\substack{1 \text{ if } y_i=+1 \\ 0 \text{ otherwise}}} - \underbrace{\sigma(w^\intercal x_i)}_{\text{Model's prediction}}\right) x_i$

  Using: $\sigma'(a) = \sigma(a)(1-\sigma(a))$

- SGD update: (logistic regression) $$w(t+1) = w(t) + \eta\left(\delta_{\{y=+1\}} - \sigma(w^\intercal x)\right) x \text{ for } (x,y) \in S$$

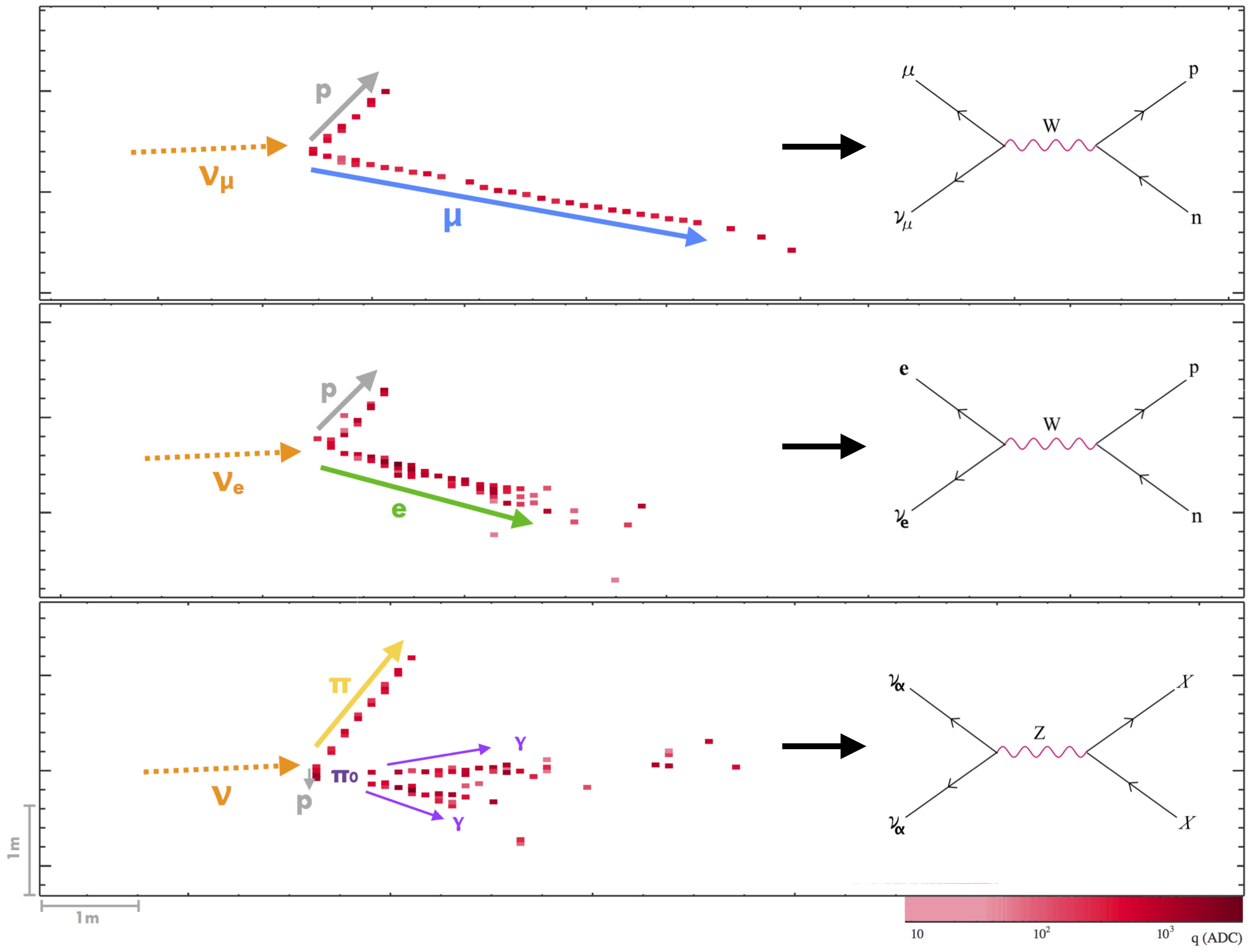
- SGD update: (linear regression) $$w(t+1) = w(t) + 2\eta(y - w^\intercal x)x \text{ for } (x,y) \in S$$

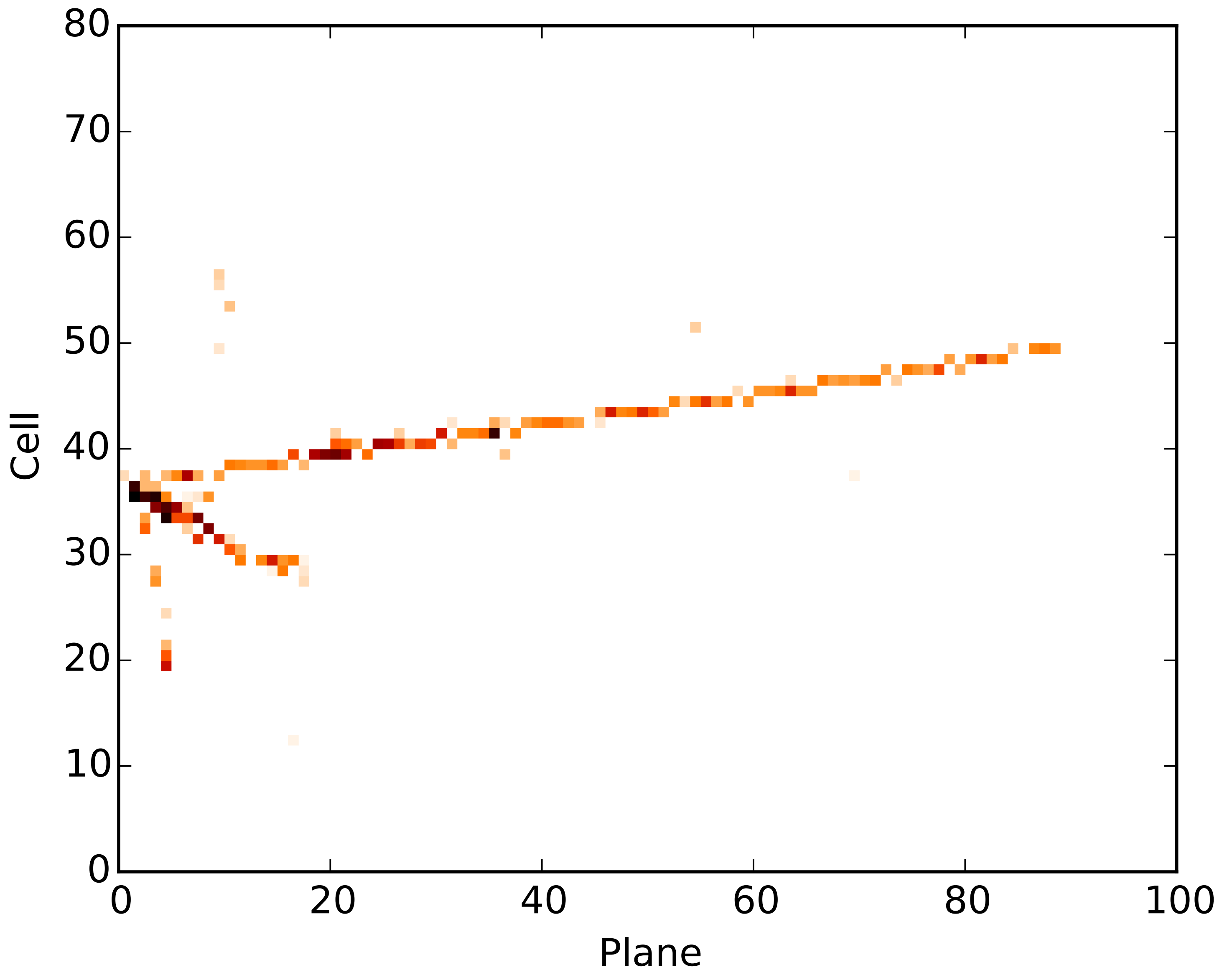
Multiclass logistic regression

arXiv:1604.01444

- Predict a raw score for each of K classes

- Example: $K = 3, Y = \{\nu_\mu \text{ CC}, \nu_e \text{ CC}, \text{NC}\}$





$x \in \mathbb{R}^D$

$$w^\intercal x = \begin{bmatrix} w_1^\intercal x \\ w_2^\intercal x \\ w_3^\intercal x \end{bmatrix} \begin{matrix} \longleftarrow \nu_\mu \text{ CC score} \\ \longleftarrow \nu_e \text{ CC score} \\ \longleftarrow \text{NC score} \end{matrix}$$

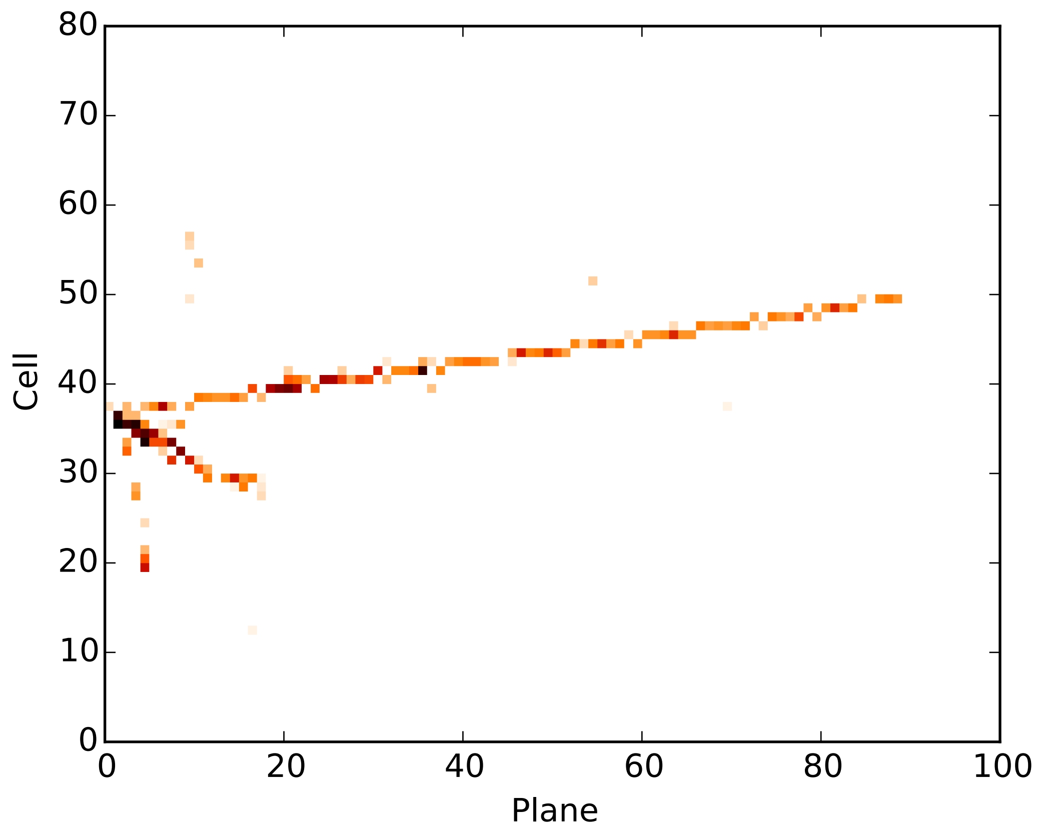
Model parameters: $w \in \mathbb{R}^{K \times D}$

Multiclass logistic regression

- Sigmoid is replaced by softmax:

$$\text{softmax}\left(\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_K \end{bmatrix}\right) = \frac{1}{\sum_{k=1}^{K} \exp(a_k)} \begin{bmatrix} \exp(a_1) \\ \exp(a_2) \\ \vdots \\ \exp(a_K) \end{bmatrix}$$

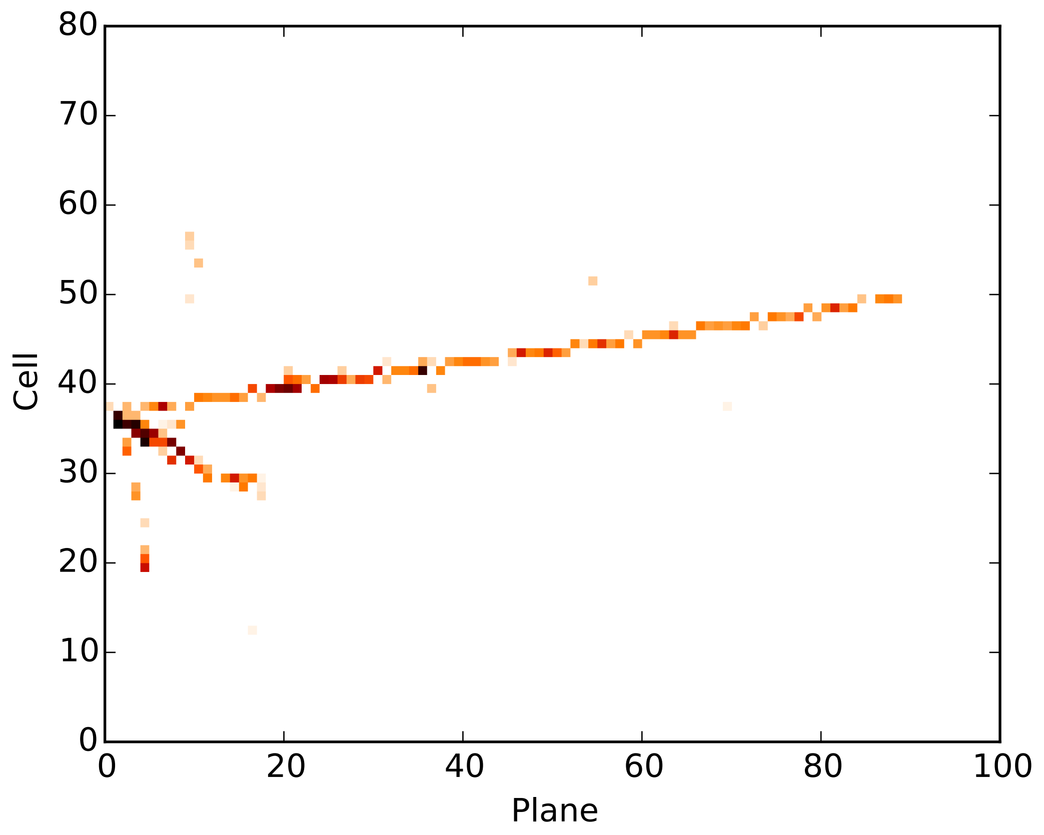
K numbers between 0 and 1 that sum to 1



$x \in \mathbb{R}^D$

$$\longrightarrow \quad w^\intercal x = \begin{bmatrix} w_1^\intercal x \\ w_2^\intercal x \\ w_3^\intercal x \end{bmatrix} \quad \longrightarrow \quad \text{softmax}(w^\intercal x) = \begin{bmatrix} f_1(x \mid w) \\ f_2(x \mid w) \\ f_3(x \mid w) \end{bmatrix}$$

Multiclass logistic regression example



$x \in \mathbb{R}^D$

$$w^\intercal x = \begin{bmatrix} 3.1 \\ 0.5 \\ -1.2 \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} 0.919 \\ 0.068 \\ 0.013 \end{bmatrix}$$

ν_μ CC: 93%

ν_e CC: 7%

NC: 1%

softmax outputs sum to 1

Categorical cross-entropy loss

- Negative log likelihood of w a.k.a. categorical cross-entropy loss:

$$-\log p(S \mid w) = -\sum_{i=1}^{N} \sum_{k=1}^{K} \delta_{\{y_i = k\}} \log f_k(x \mid w)$$

- Generalizes the binary cross-entropy loss (and equivalent when $k = 2$)

Recap: Activations and loss functions

- Linear regression:
  - Activation: linear; loss: mean-squared error
- Binary classification:
  - Activation: linear; loss: perceptron (PLA)
  - Activation: linear; loss: hinge (SVM)
  - Activation: sigmoid; loss: binary cross-entropy
- Multiclass classification:
  - Activation: softmax; loss: categorical cross-entropy

Next time

- (Boosted) decision trees
- Tabular data
    - Kaggle Higgs boson classification challenge