Review: Supervised Learning

CS 6355: Structured Prediction

THE UNIVERSITY OF UTAH

Previous lecture

- A broad overview of structured prediction
- The different aspects of the area
  - Basically the syllabus of the class
- *Questions?*

Important dates

Date	Deadline
17-Jan	Start signing up for class presentations
24-Jan	Project team information due
5-Feb	Review 1 due
14-Feb	Project proposals due
9-Mar	Review 2 due
2-Apr	Project intermediate status report due
9-Apr	Review 3 due
23-Apr	Project final report due

Announcements

- Office hours today at 2 PM
- Reading for next lecture (Wednesday):
  - Erin L. Allwein, Robert E. Schapire, Yoram Singer, Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, ICML 2000.

Supervised learning, Binary classification

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

Supervised learning: General setting

- Given: Training examples of the form <**x**, *f*(**x**)>
  - The function f is an unknown function
- The input **x** is represented in a *feature space*
  - Typically $\mathbf{x} \in \{0,1\}^n$ or $\mathbf{x} \in \Re^n$
- For a training example **x**, *f*(**x**) is called the *label*
- Goal: Find a good approximation for *f*
- Different kinds of problems
  - Binary classification: $f(\mathbf{x}) \in \{-1,1\}$
  - Multiclass classification: $f(\mathbf{x}) \in \{1, 2, 3, \cdots, K\}$
  - Regression: $f(\mathbf{x}) \in \Re$

Nature of applications

- There is no human expert
  - Eg: Identify DNA binding sites
- Humans can perform a task, but can’t describe how they do it
  - Eg: Object detection in images
- The desired function is hard to obtain in closed form
  - Eg: Stock market

Binary classification

- Spam filtering
  - Is an email spam or not?
- Recommendation systems
  - Given user’s movie preferences, will she like a new movie?
- Malware detection
  - Is an Android app malicious?
- Time series prediction
  - Will the future value of a stock increase or decrease with respect to its current value?
- Spambot detection
  - Is a Twitter account a spam bot?

Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

Linear Classifiers

- Input is a n dimensional vector **x**
- Output is a label y $\in$ {-1, 1}   For now

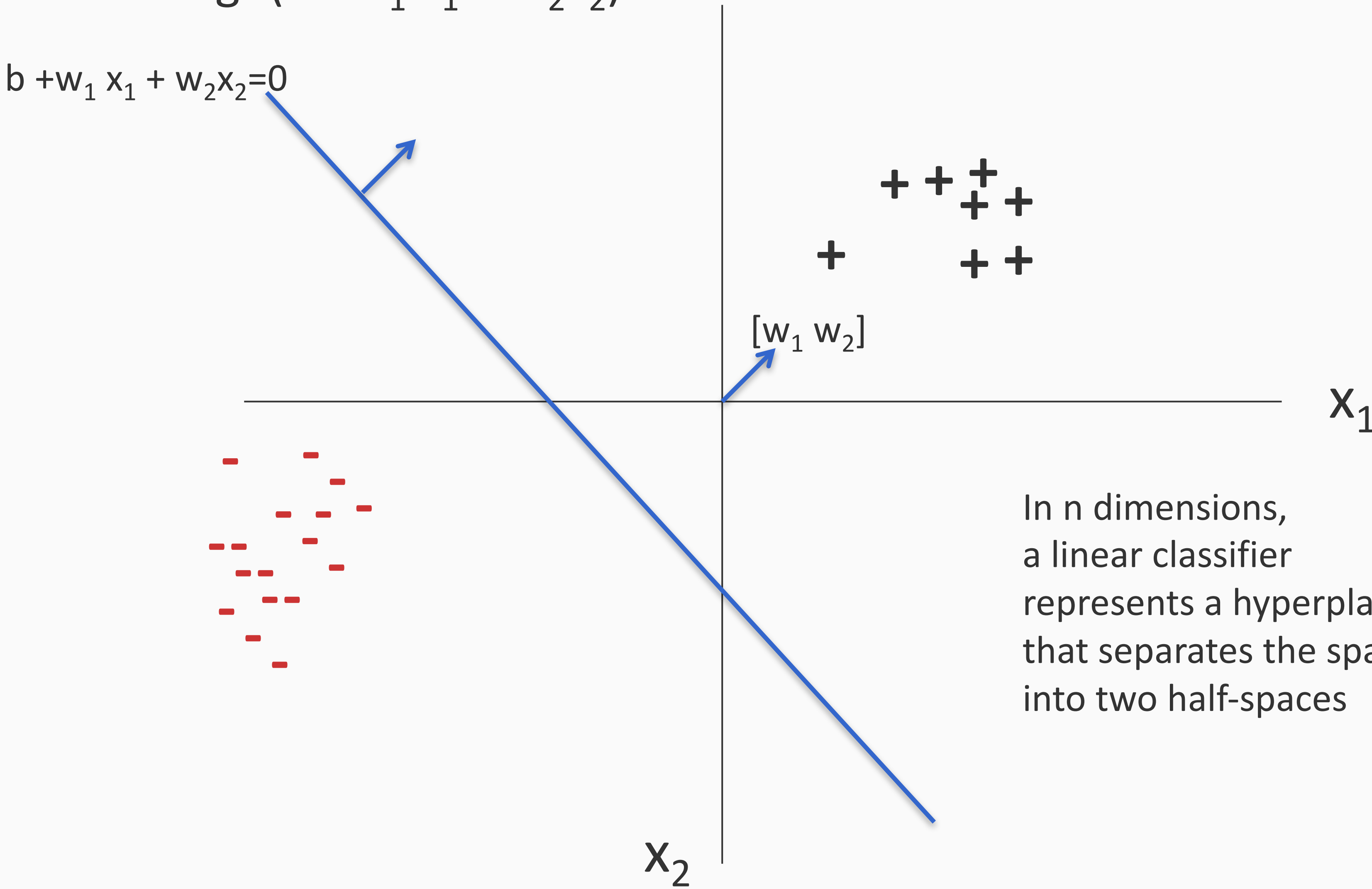
- Linear threshold units classify an example $\boldsymbol{x}$ using the classification rule

$$\mathrm{sgn}(b + \boldsymbol{w}^T \boldsymbol{x}) = \mathrm{sgn}(b + \sum_i w_i x_i)$$

  - $b + \boldsymbol{w}^T \boldsymbol{x} \geq 0 \Rightarrow$ Predict y = 1
  - $b + \boldsymbol{w}^T \boldsymbol{x} < 0 \Rightarrow$ Predict y = -1

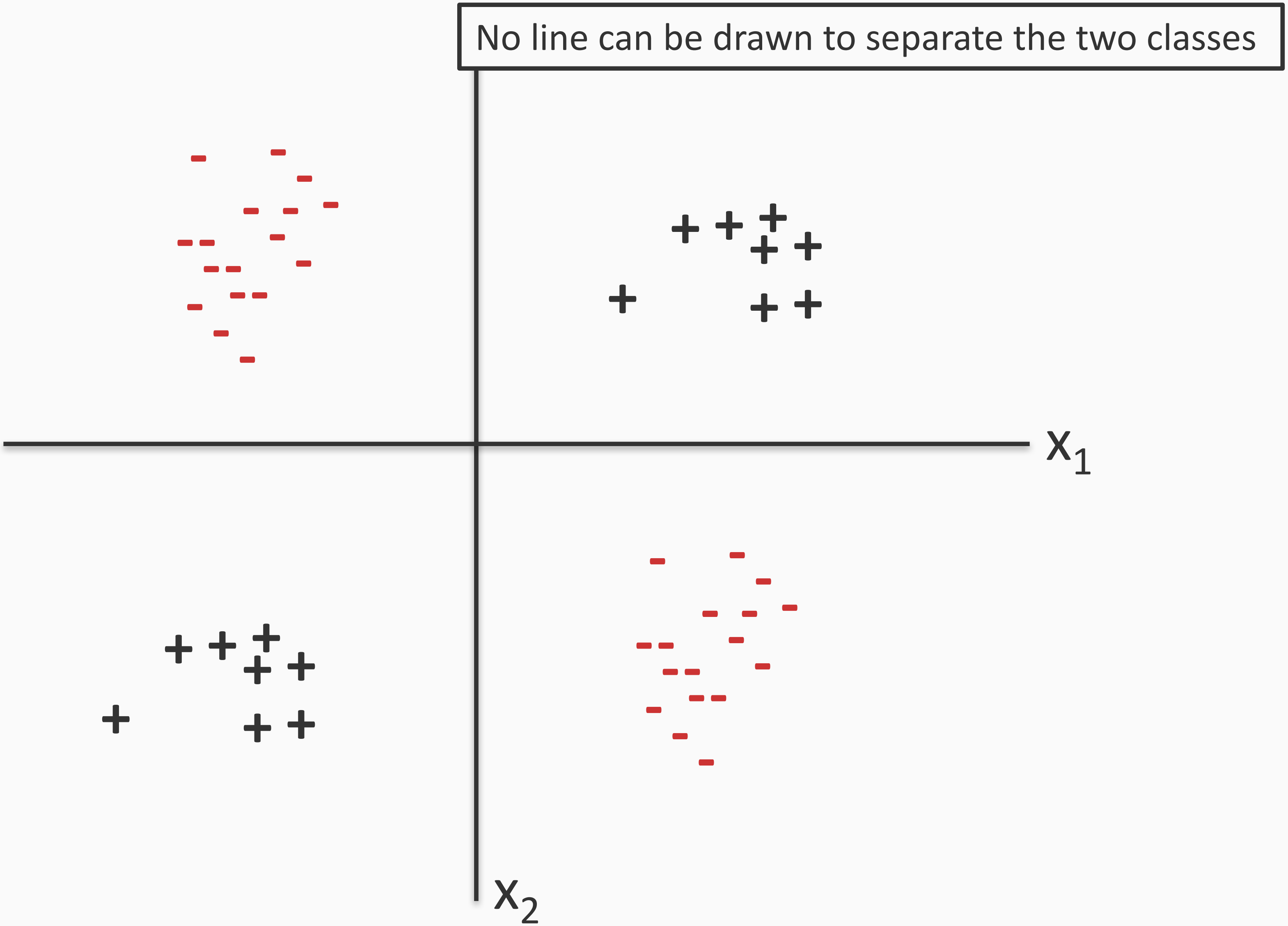
The geometry of a linear classifier

$\mathrm{sgn}(b + w_1 x_1 + w_2 x_2)$



In n dimensions,
a linear classifier
represents a hyperplane
that separates the space
into two half-spaces

XOR is not linearly separable



Even these functions can be *made* linear

These points are not separable in 1-dimension by a line

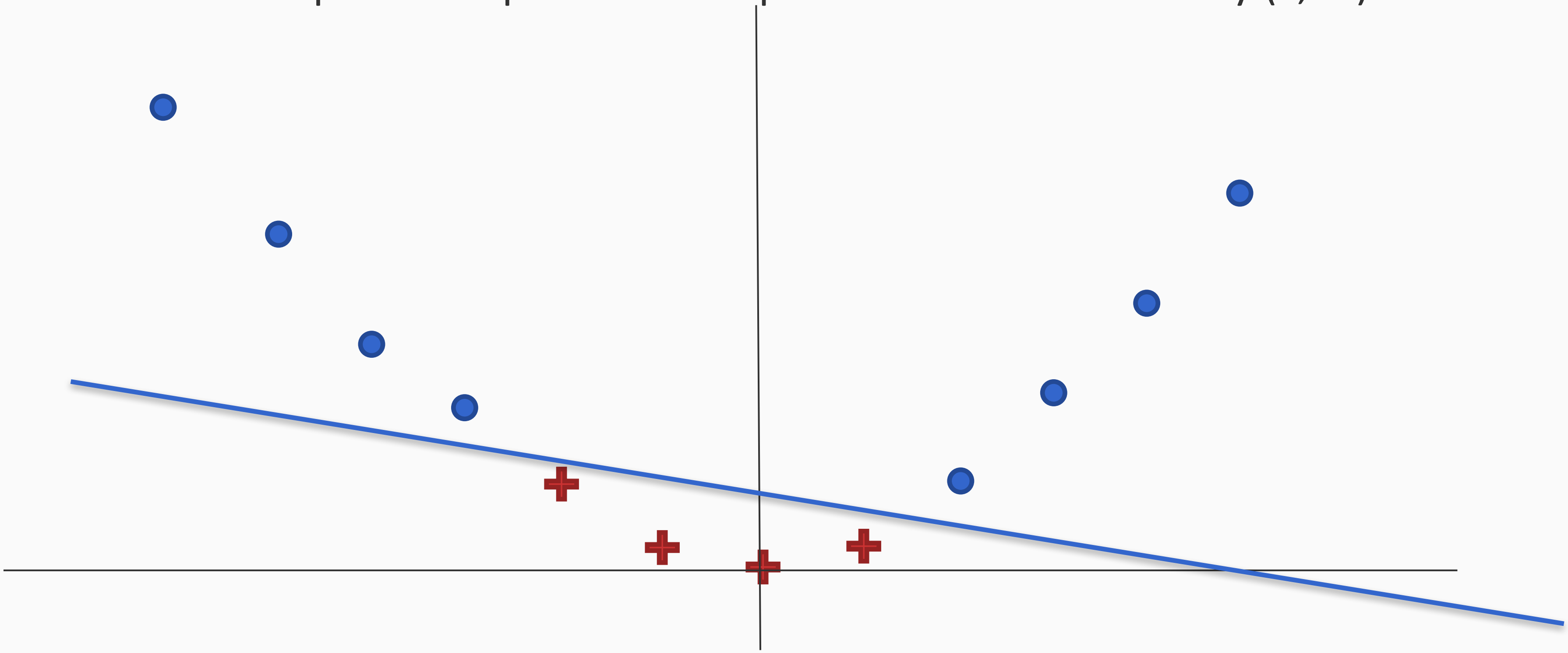
What is a one-dimensional line, by the way?

The trick: Change the representation

Even these functions can be *made* linear

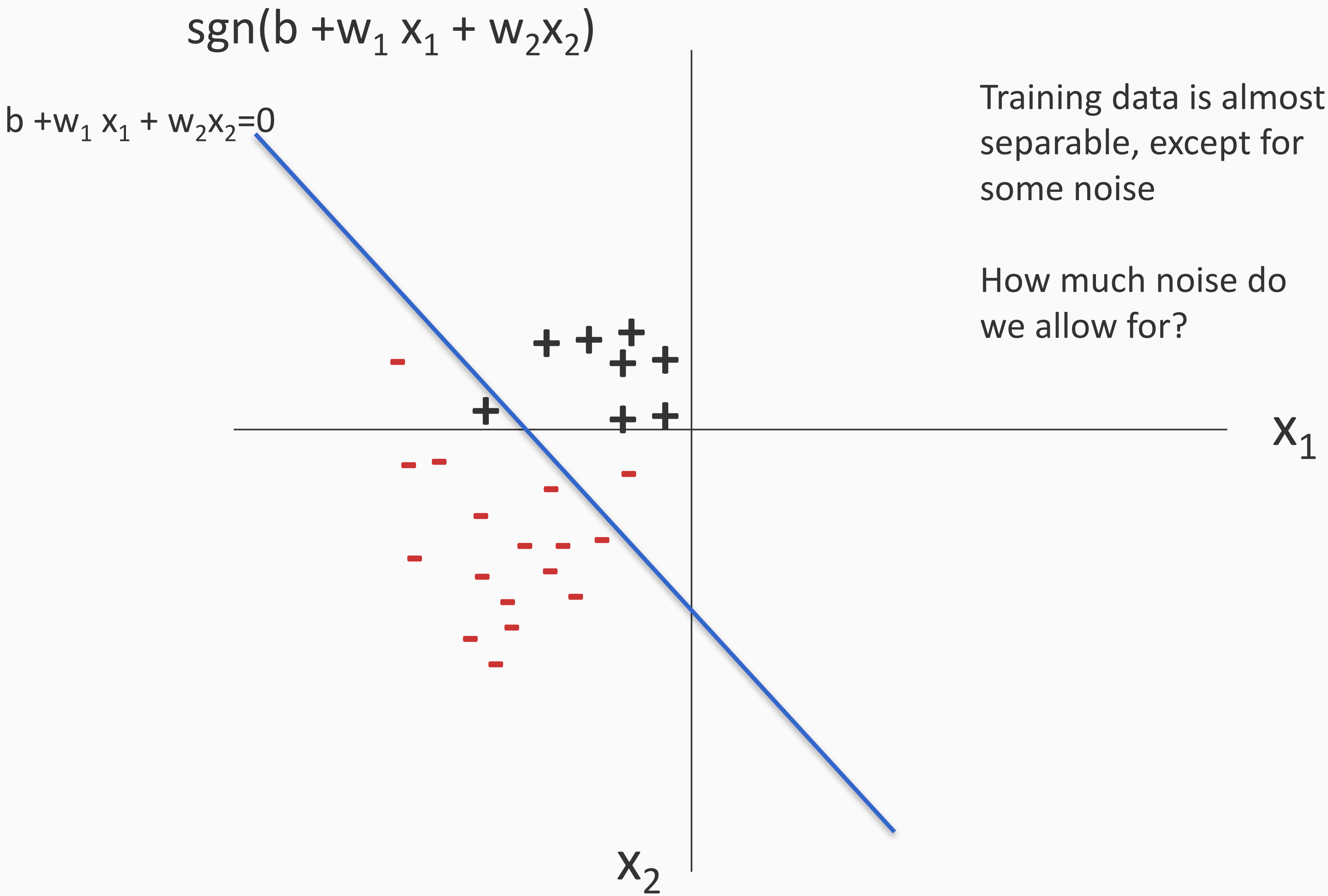
The trick: Use feature *conjunctions*

Transform points: Represent each point x in 2 dimensions by (x, x^2)



Now the data is linearly separable in this space!

Almost linearly separable data



Training data is almost separable, except for some noise

How much noise do we allow for?

Linear classifiers are an expressive hypothesis class

- Many functions are linear
  - Conjunctions, disjunctions
  - At least m-of-n functions
- Often a good guess for a hypothesis space
- Some functions are not linear
  - The XOR function
  - Non-trivial Boolean functions

We will see later in the class that many structured predictors are linear functions too

Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

The Perceptron algorithm

- Rosenblatt 1958

- The goal is to find a separating hyperplane
  - For separable data, guaranteed to find one

- An online algorithm
  - Processes one example at a time

- Several variants exist

The algorithm

Given a training set D = {($\mathbf{x}$,y)}, $x \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
    1. For each training example ($\mathbf{x}$, y) $\in$ D:
        1. Predict $y' = \mathrm{sgn}(\mathbf{w}^T\mathbf{x})$
        2. If $y \neq y'$, update $\mathbf{w} \leftarrow \mathbf{w} + y\,\mathbf{x}$
3. Return $\mathbf{w}$

Prediction: $\mathrm{sgn}(\mathbf{w}^T\mathbf{x})$

The algorithm

Given a training set D = $\{(\mathbf{x},y)\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. For each training example $(\mathbf{x}, y) \in D$:
      1. Predict $y' = \text{sgn}(\mathbf{w}^T\mathbf{x})$
      2. If $y \neq y'$, update $\mathbf{w} \leftarrow \mathbf{w} + y\,\mathbf{x}$
3. Return $\mathbf{w}$

T is a hyperparameter to the algorithm

In practice, good to shuffle D before the inner loop

Update only on an error. Perceptron is an mistake-driven algorithm.

Prediction: $\text{sgn}(\mathbf{w}^T\mathbf{x})$

Convergence

- Convergence theorem
  - If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge. [Novikoff 1962]
- Cycling theorem
  - If the training data is *not* linearly separable, then the learning algorithm will eventually repeat the same set of weights and enter an infinite loop

Beyond the separable case

- The good news
  - Perceptron makes no assumption about data distribution
  - Even adversarial
  - After a fixed number of mistakes, you are done. Don't even need to see any more data

- The bad news: Real world is not linearly separable
  - Can't expect to *never* make mistakes again
  - What can we do: more features, try to be linearly separable if you can

Variants of the algorithm

- The original version: Return the final weight vector

- Averaged perceptron
  - Returns the average weight vector from the entire training time (i.e longer surviving weight vectors get more say)

  - Widely used

  - A practical approximation of the Voted Perceptron

Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

Learning as loss minimization

- Collect some annotated data. More is generally better
- Pick a hypothesis class (also called model)
  - Eg: linear classifiers, deep neural networks
  - Also, decide on how to impose a preference over hypotheses
- Choose a loss function
  - Eg: negative log-likelihood, hinge loss
  - Decide on how to penalize incorrect decisions
- Minimize the expected loss
  - Eg: Set derivative to zero, more complex algorithm

Learning as loss minimization

- The setup
  - Examples $\mathbf{x}$ drawn from a fixed, unknown distribution D
  - Hidden oracle classifier f labels examples
  - We wish to find a hypothesis h that mimics f

Learning as loss minimization

- The setup
  - Examples $\mathbf{x}$ drawn from a fixed, unknown distribution D
  - Hidden oracle classifier f labels examples
  - We wish to find a hypothesis h that mimics f
- The ideal situation
  - Define a function L that penalizes bad hypotheses
  - **Learning:** Pick a function h $\in$ H to minimize expected loss

$$\min_{h \in H} E_{\mathbf{x} \sim D} \left[L\left(h(\mathbf{x}), f(\mathbf{x}) \right) \right]$$

But distribution D is unknown

Learning as loss minimization

- The setup
  - Examples $\mathbf{x}$ drawn from a fixed, unknown distribution D
  - Hidden oracle classifier f labels examples
  - We wish to find a hypothesis h that mimics f
- The ideal situation
  - Define a function L that penalizes bad hypotheses
  - **Learning:** Pick a function h $\in$ H to minimize expected loss

$$\min_{h \in H} E_{\mathbf{x} \sim D} \left[L \left(h(\mathbf{x}), f(\mathbf{x}) \right) \right]$$

But distribution D is unknown

- Instead, minimize *empirical loss* on the training set

$$\min_{h \in H} \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

Empirical loss minimization

Learning = minimize *empirical loss* on the training set

$$\min_{h \in H} \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

Is there a problem here?

Overfitting!

We need something that biases the learner towards simpler hypotheses

- Achieved using a regularizer, which penalizes complex hypotheses
- Capacity control for better generalization

Regularized loss minimization

- Learning: $\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$

Regularized loss minimization

- Learning: $\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$
- With L2 regularization: $\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i L(y_i, \mathbf{x}_i, \mathbf{w})$

Regularized loss minimization

- Learning: $\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$
- With L2 regularization: $\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i L(y_i, \mathbf{x}_i, \mathbf{w})$
- What is a loss function?
  - Loss functions should penalize mistakes
  - We are minimizing average loss over the training data
- What is the ideal loss function for classification?

The 0-1 loss

Penalize classification mistakes between true label y and prediction y'

$$L_{0\text{-}1}(y, y') = \begin{cases} 1 & \text{if } y \neq y', \\ 0 & \text{if } y = y'. \end{cases}$$

- For linear classifiers, the prediction y' = sgn($\mathbf{w}^T\mathbf{x}$)
  - Mistake if y $\mathbf{w}^T\mathbf{x} \leq 0$

$$L_{0\text{-}1}(y, \mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } y\ \mathbf{w}^T\mathbf{x} \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Minimizing 0-1 loss is intractable. Need surrogates

$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

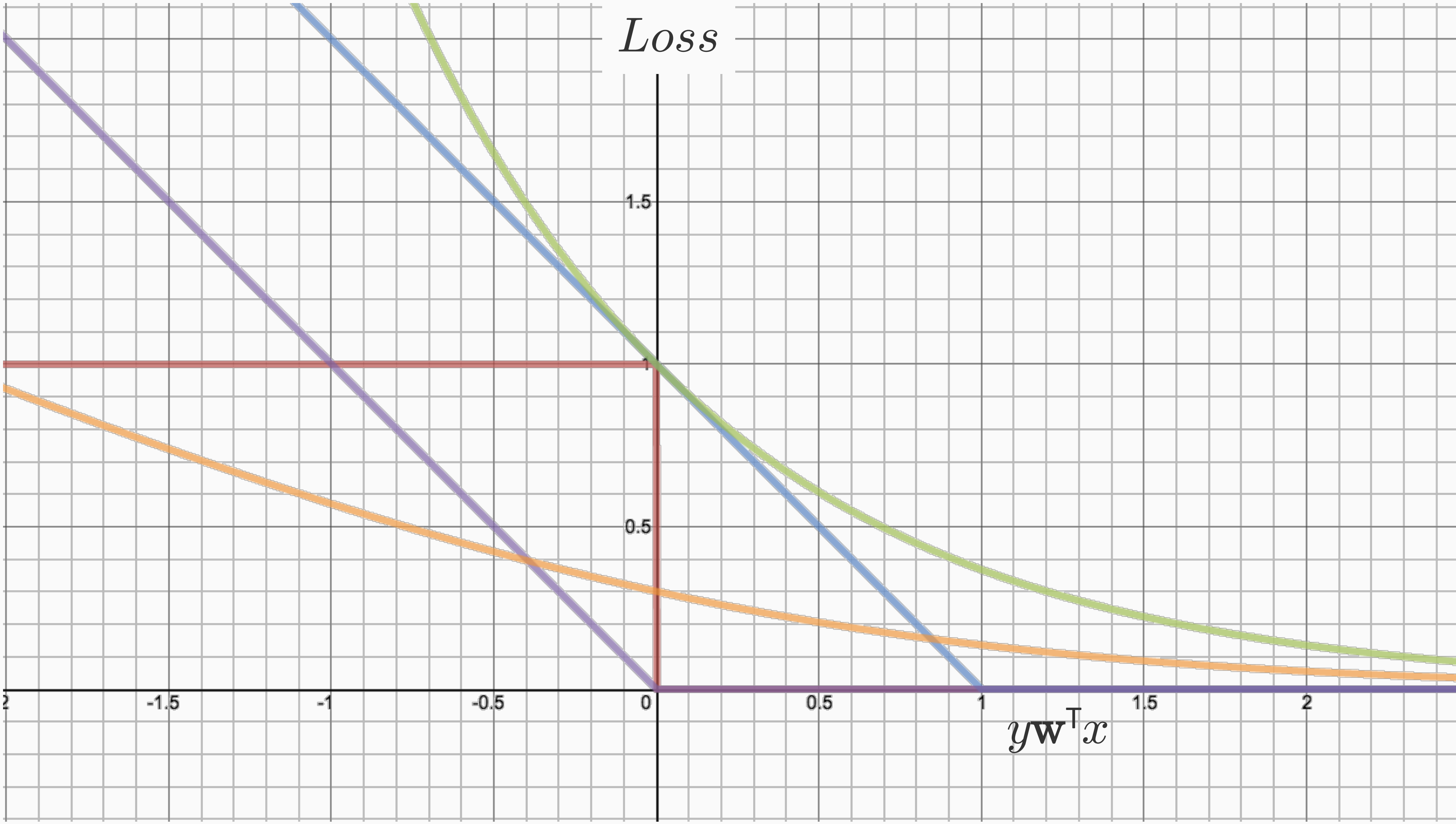
The loss function zoo

Many loss functions exist

- Perceptron loss $$L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$$
- Hinge loss (SVM) $$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$
- Exponential loss (AdaBoost) $$L_{Exponential}(y, \mathbf{x}, \mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}}$$
- Logistic loss (logistic regression) $$L_{Logistic}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

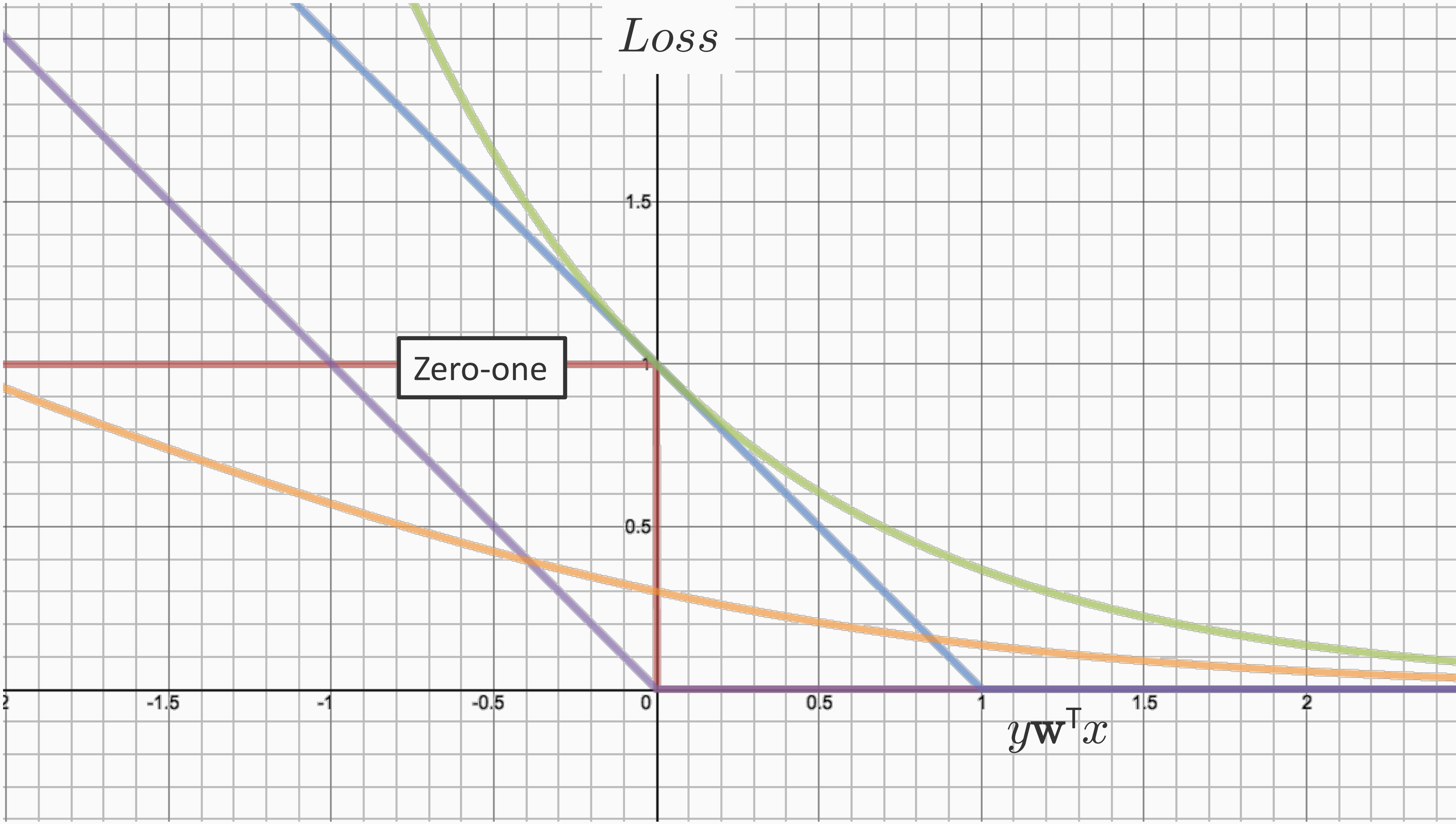
$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo



$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

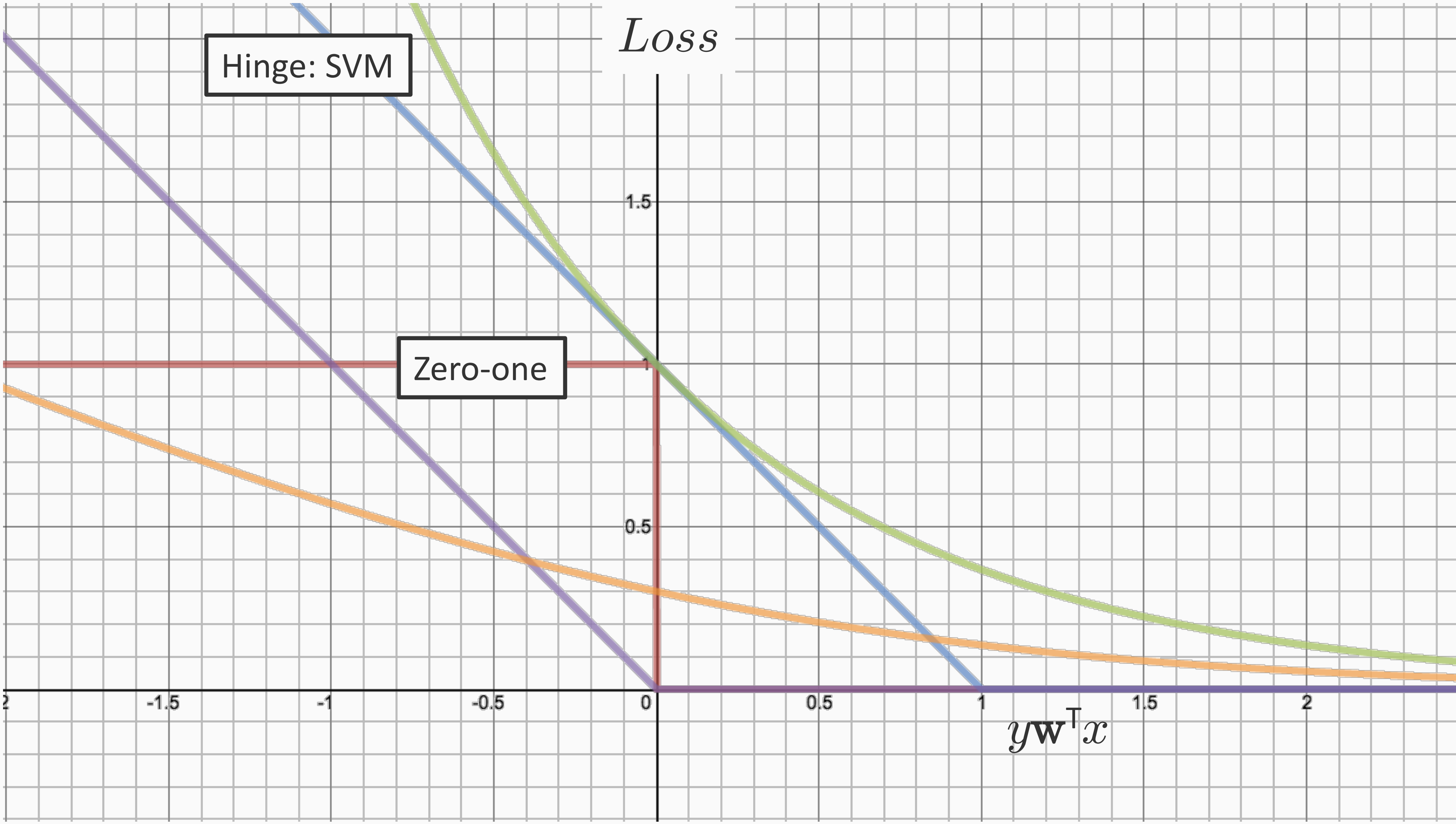
The loss function zoo



$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

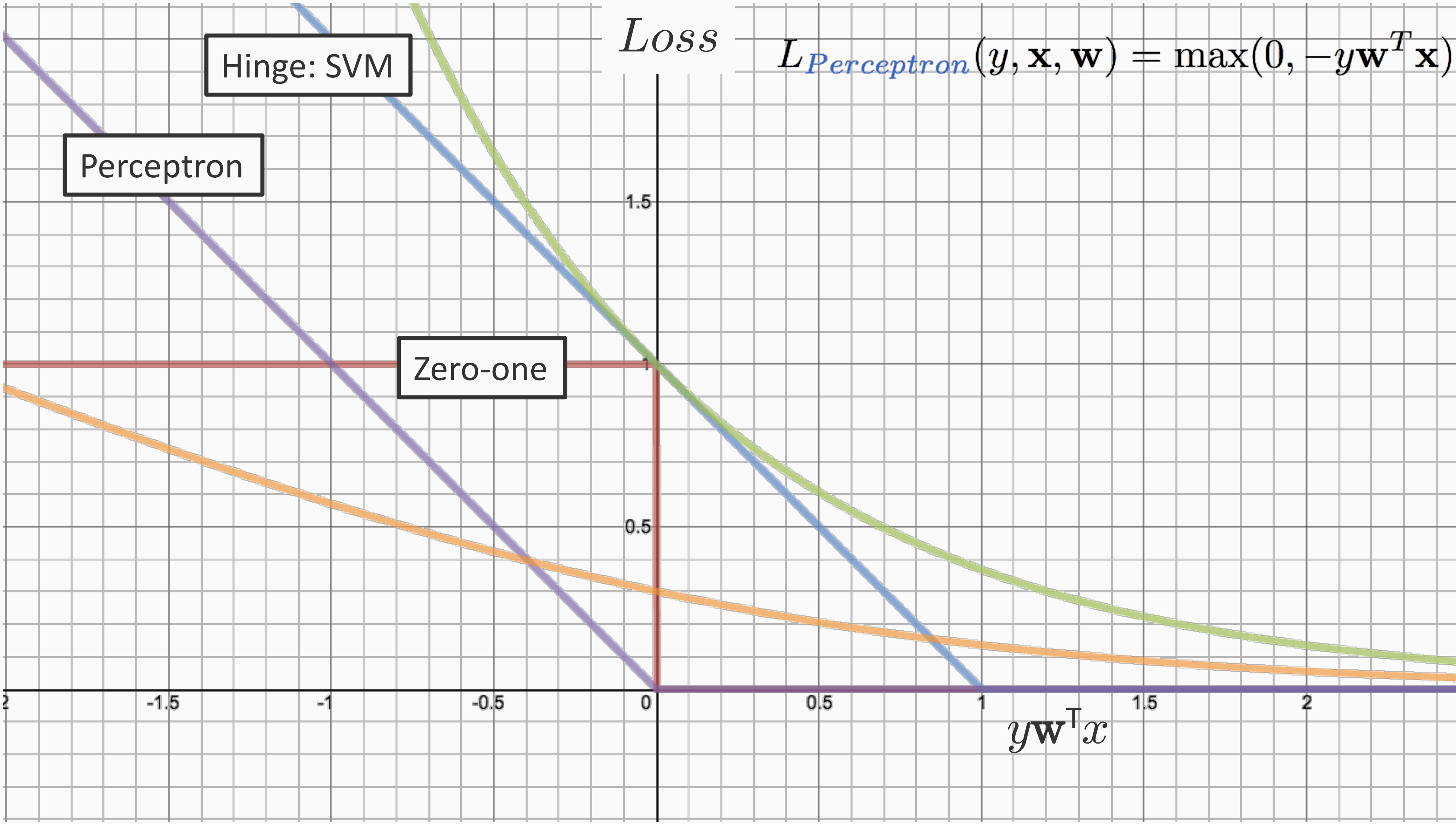


$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

$$L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$$



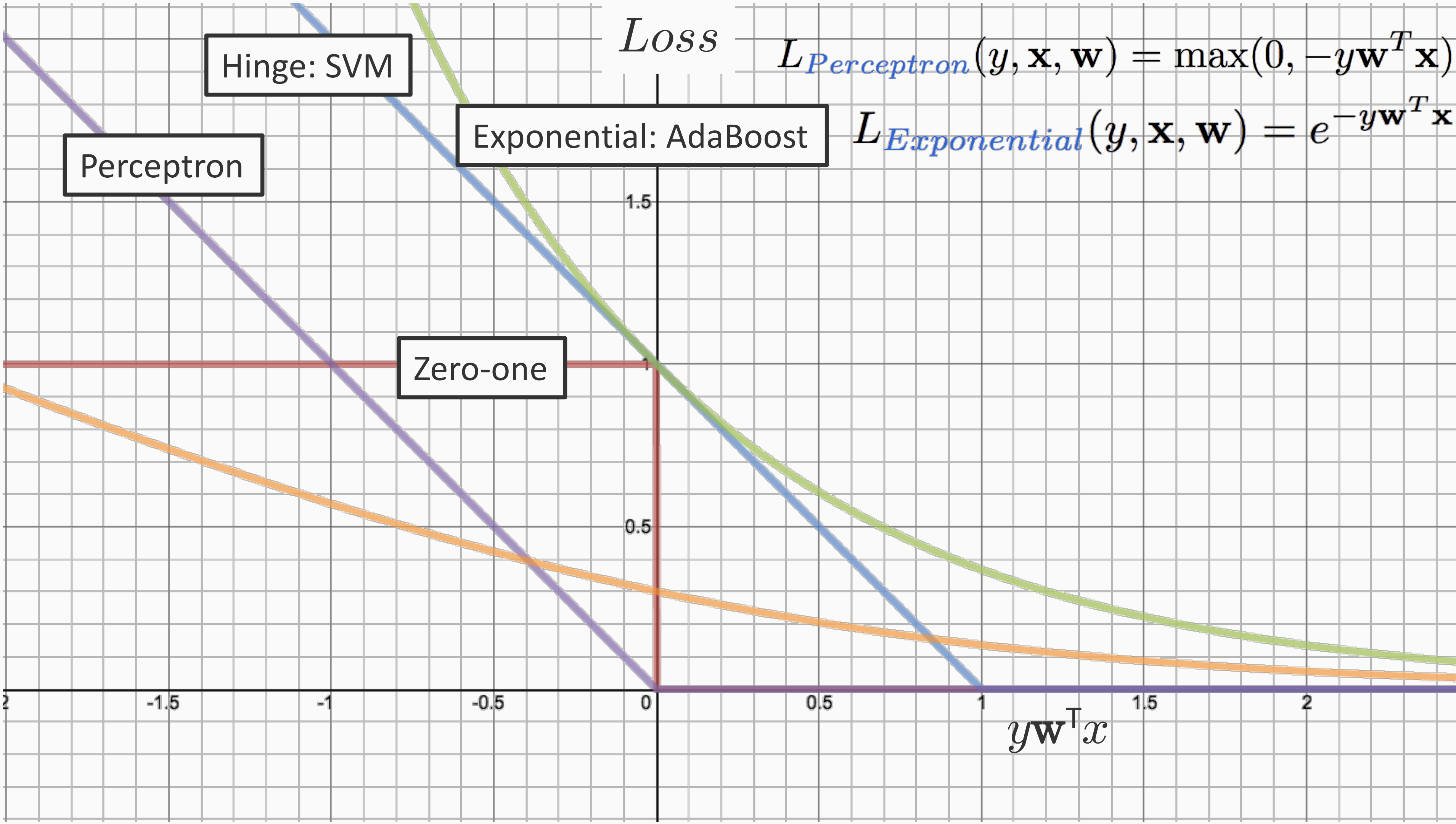
$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

$$L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$$

$$L_{Exponential}(y, \mathbf{x}, \mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}}$$



$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

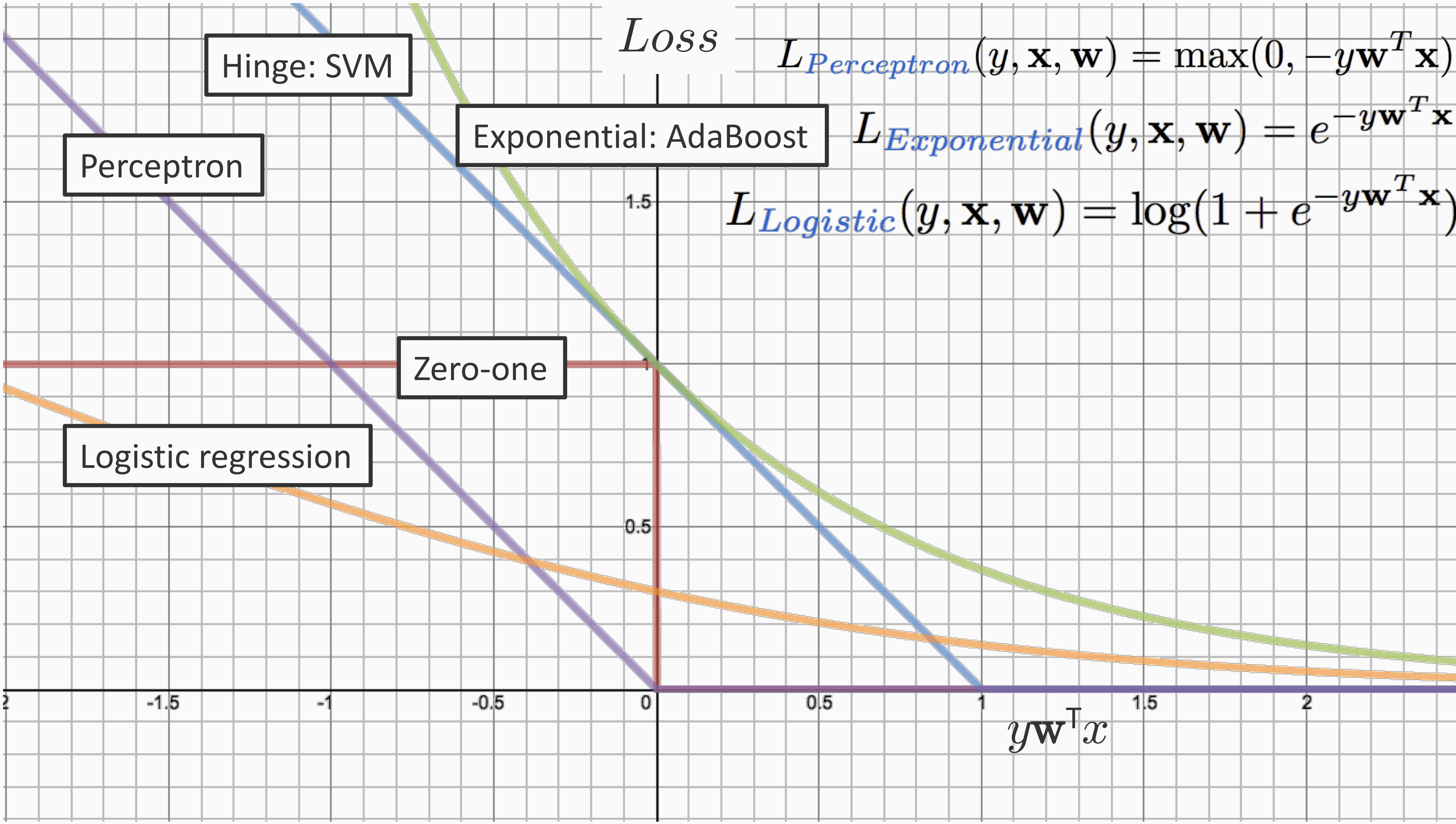
The loss function zoo

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

$$L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$$

$$L_{Exponential}(y, \mathbf{x}, \mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}}$$

$$L_{Logistic}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

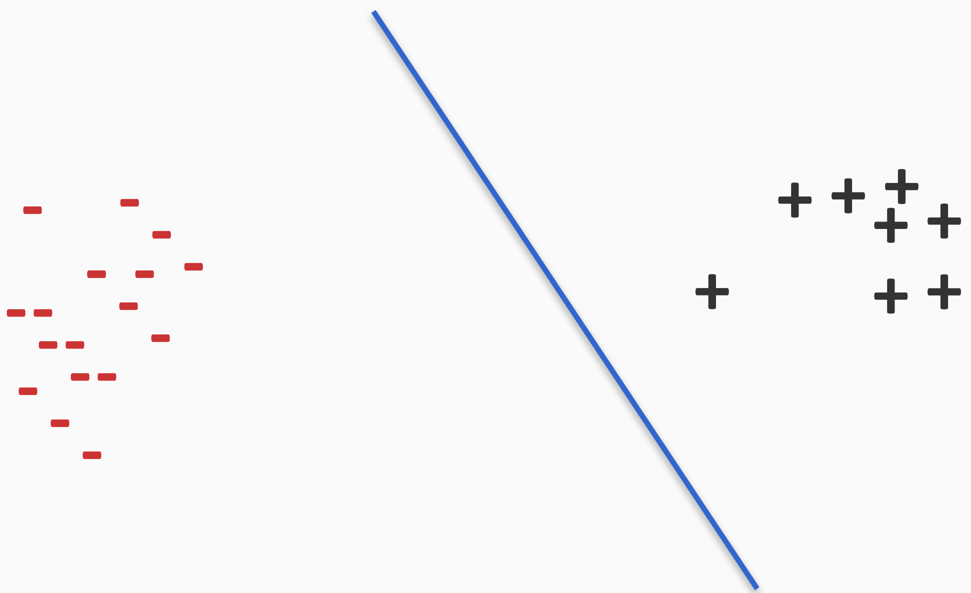


Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

Margin

The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Learning strategy

Find the linear separator that maximizes the margin

Maximizing margin and minimizing loss

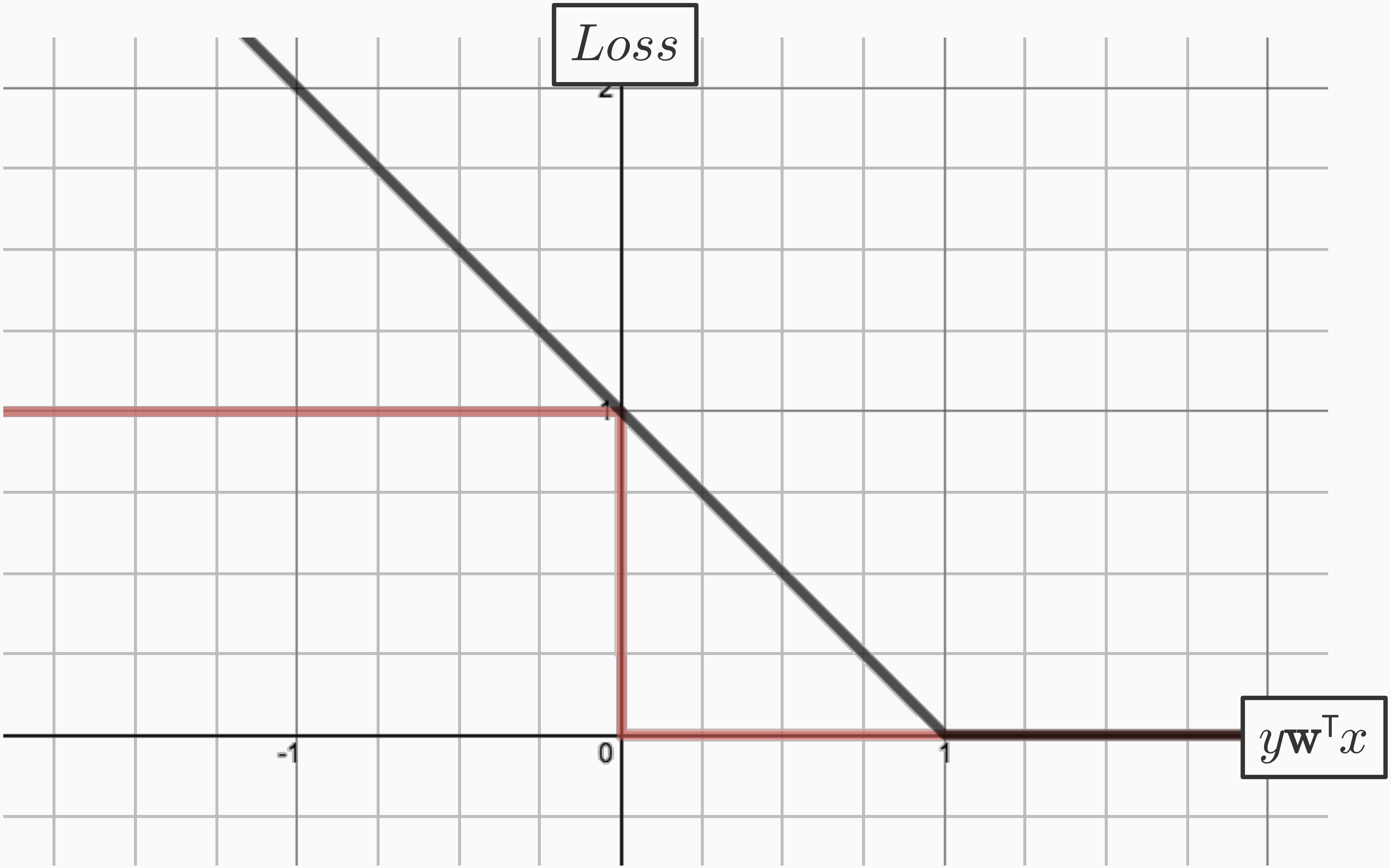
Find the linear separator that maximizes the margin

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \boxed{\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)}$$

Maximize margin

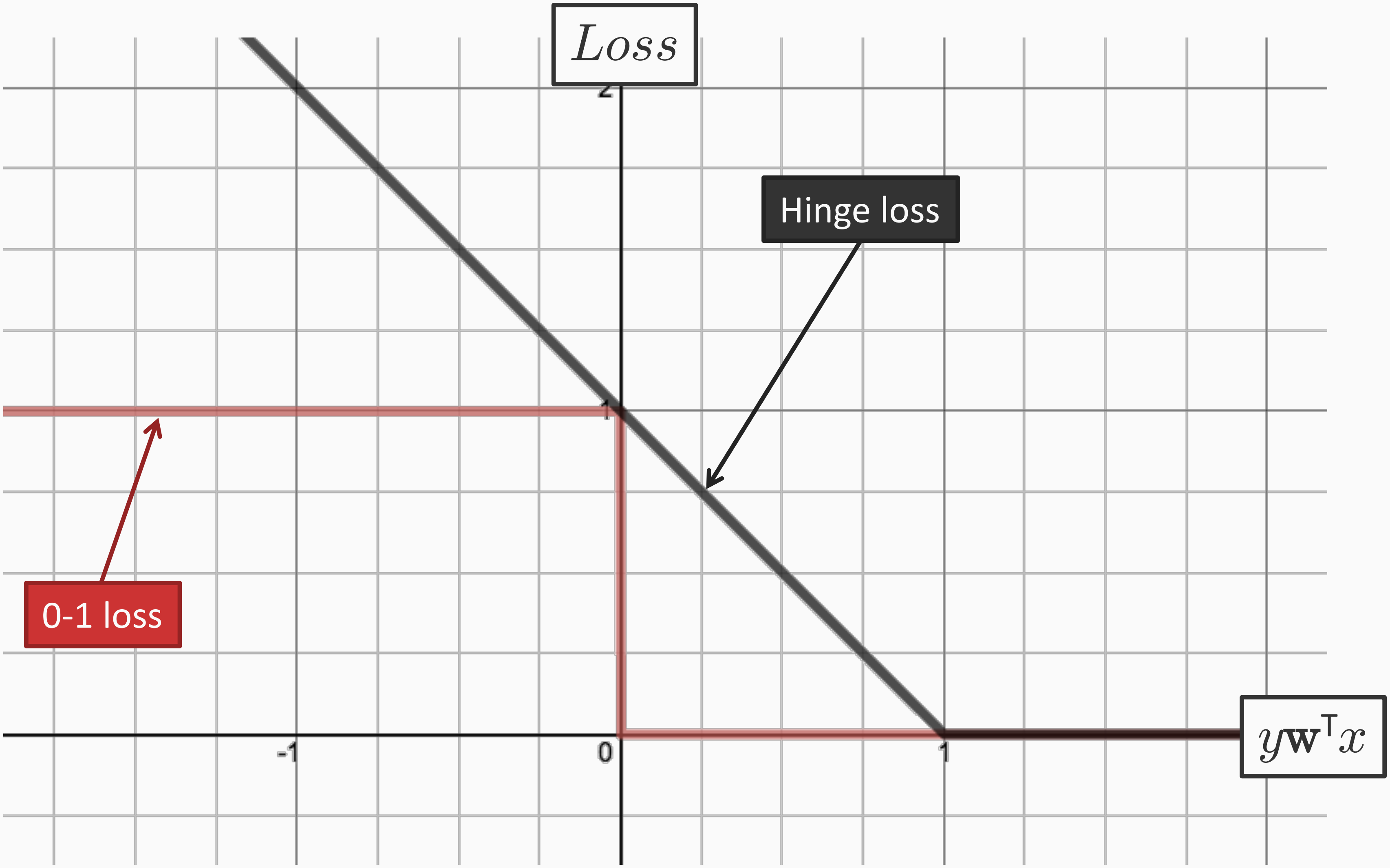
Penalty for the prediction:
The Hinge loss

The Hinge Loss



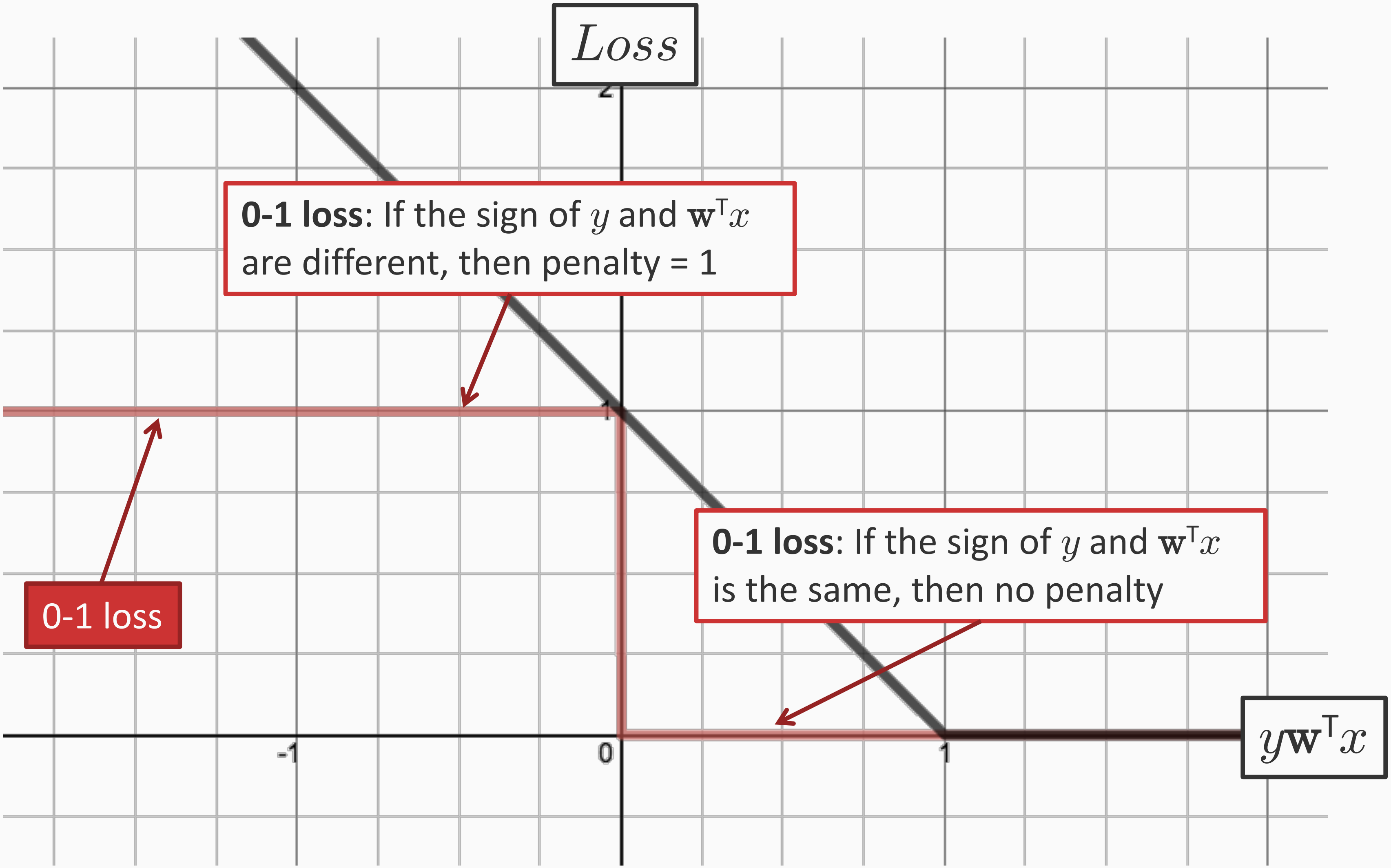
$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

The Hinge Loss



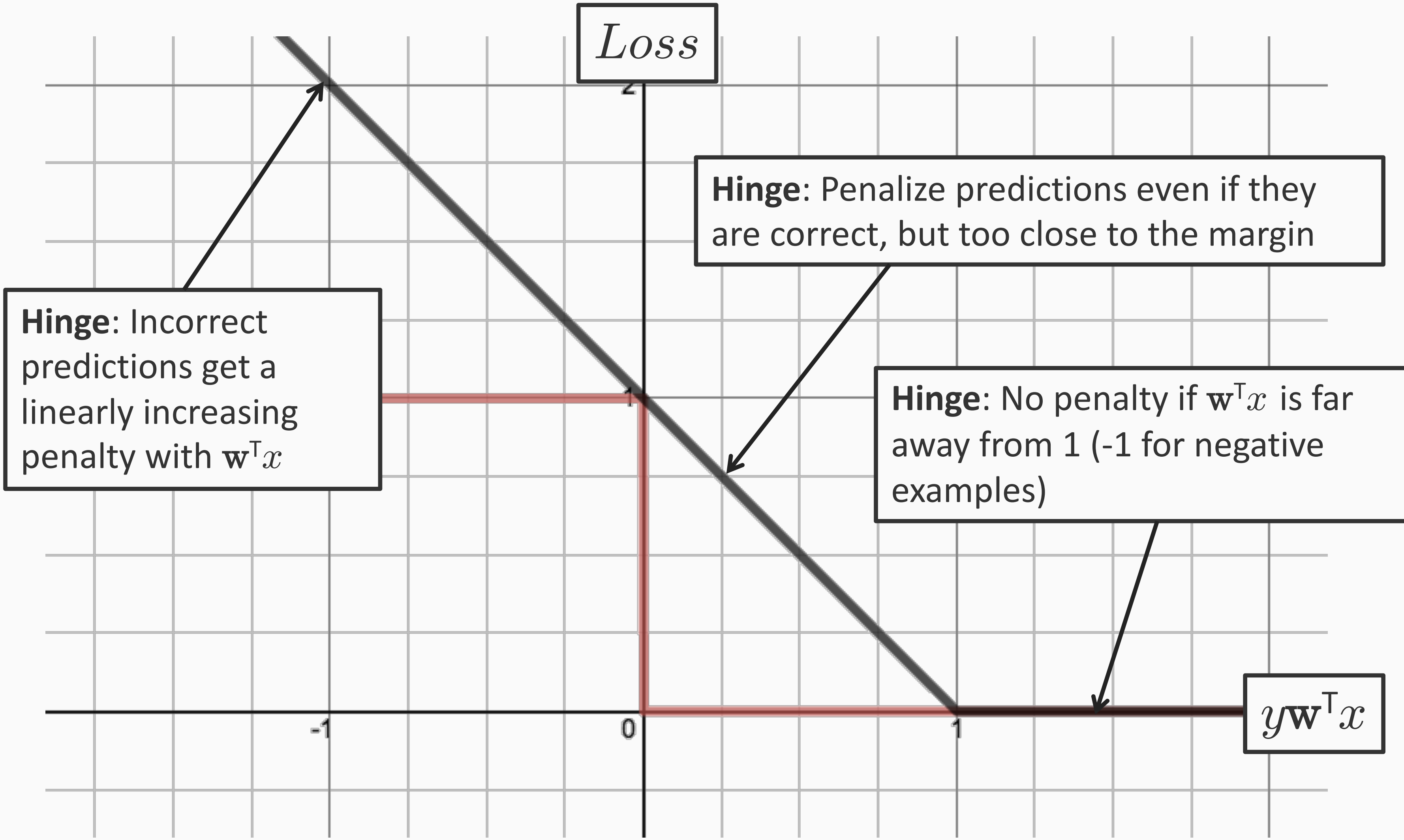
$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

The Hinge Loss



$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

The Hinge Loss



SVM objective function

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

Regularization term:
- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:
- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

SVM objective function

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:

- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

A hyper-parameter that controls the tradeoff between a large margin and a small hinge-loss

Solving the SVM optimization problem

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

This function is convex in $\mathbf{w}$

Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, \mathrm{y}_i)\}$, $\mathbf{x} \in \Re^n$, $\mathrm{y} \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$

3. Return $\mathbf{w}$

Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:

3. Return $\mathbf{w}$

Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
    1. For each training example $(\mathbf{x}_i, y_i) \in$ S:

       Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t \nabla J^t$

3. Return $\mathbf{w}$

Stochastic sub-gradient descent for SVM

$$\nabla J^t = \begin{cases} \mathbf{w} & \text{if } \max\left(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\right) = 0 \\ \mathbf{w} - C y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i\, \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1- \gamma_t)\, \mathbf{w} + \gamma_t\, C\, y_i\, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1- \gamma_t)\, \mathbf{w}$
3. Return $\mathbf{w}$

Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x} \in \Re^n$, $y \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. For each training example $(\mathbf{x}_i, y_i) \in$ S:

      If $y_i \mathbf{w}^T\mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1- \gamma_t)\, \mathbf{w} + \gamma_t\, C\, y_i\, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1- \gamma_t)\, \mathbf{w}$
3. Return $\mathbf{w}$

γ_t: learning rate, many tweaks possible

Important to shuffle examples at the start of each epoch

Stochastic sub-gradient descent for SVM

Given a training set S = $\{(\mathbf{x}_i, \mathrm{y}_i)\}$, $\mathbf{x} \in \Re^n$, $\mathrm{y} \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
   1. Shuffle the training set
   2. For each training example $(\mathbf{x}_i, \mathrm{y}_i) \in S$:

      If $\mathrm{y}_i\, \mathbf{w}^T \mathbf{x}_i \leq 1$,

      $\mathbf{w} \leftarrow (1- \gamma_t)\, \mathbf{w} + \gamma_t\, C\, \mathrm{y}_i\, \mathbf{x}_i$

      else

      $\mathbf{w} \leftarrow (1- \gamma_t)\, \mathbf{w}$
3. Return $\mathbf{w}$

γ_t: learning rate, many tweaks possible

Where are we?

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Learning as optimization
5. Support vector machines
6. Logistic Regression

Regularized loss minimization: Logistic regression

- Learning: $\min_{f \in H} \text{regularizer}(f) + C \sum_i L(y_i, f(\mathbf{x}_i))$
- With linear classifiers: $\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_i L(y_i, \mathbf{x}_i, \mathbf{w})$
- SVM uses the hinge loss
- Another loss function: The logistic loss

$$L_{logistic}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

The probabilistic interpretation

Suppose we believe that the labels are distributed as follows given the input:

$$\left.\begin{aligned} P(y=1|\mathbf{x},\mathbf{w}) &= \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} \\ P(y=-1|\mathbf{x},\mathbf{w}) &= \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}} \end{aligned}\right\} \quad P(y|\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp\left(-y\mathbf{w}^T\mathbf{x}\right)}$$

Predict label = 1 if P(1 | **x**, **w**) > P(-1 | **x**, **w**)

- Equivalent to predicting 1 if $\mathbf{w}^T\mathbf{x} \geq 0$

The probabilistic interpretation

Suppose we believe that the labels are distributed as follows given the input:

$$\left.\begin{aligned} P(y = 1|\mathbf{x}, \mathbf{w}) &= \frac{e^{\mathbf{w}^T\mathbf{x}}}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} \\ P(y = -1|\mathbf{x}, \mathbf{w}) &= \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}} \end{aligned}\right\} \quad P(y|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp\left(-y\mathbf{w}^T\mathbf{x}\right)}$$

The log-likelihood of seeing a dataset D = {($\mathbf{x}_i$, y_i)} if the true weight vector was **w**:

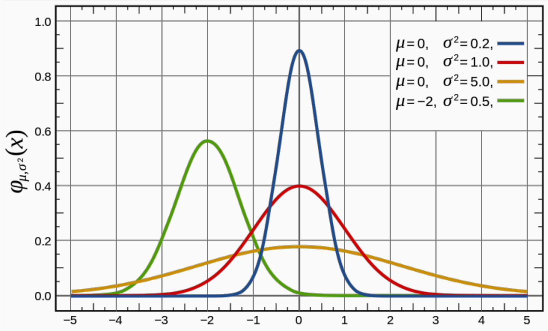
$$\log P(D|\mathbf{w}) = -\sum_i \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x}\right)$$

Prior distribution over the weight vectors

A prior balances the tradeoff between the likelihood of the data and existing belief about the parameters

- Suppose each weight w_i is drawn independently from the normal distribution centered at zero with variance σ^2
  - Bias towards smaller weights

$$P(w_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)$$



Source: Wikipedia

- Probability of the entire weight vector:

$$\log P(\mathbf{w}) = -\frac{1}{2\sigma^2}\mathbf{w}^T\mathbf{w} + \text{constant terms}$$

Regularized logistic regression

What is the probability of a weight vector **w** being the true one for a dataset D = {<$\mathbf{x}_i$, y_i>}?

$$P(\boldsymbol{w} \mid D) \propto P(\boldsymbol{w}, D) = P(\ D \mid \boldsymbol{w}\ )P(\ \boldsymbol{w})$$

Regularized logistic regression

What is the probability of a weight vector **w** being the true one for a dataset D = {<$\mathbf{x}_i$, y_i>}?

$$P(\boldsymbol{w} \mid D) \propto P(\boldsymbol{w}, D) = P(\ D \mid \boldsymbol{w}\ )P(\ \boldsymbol{w})$$

Learning: Find weight vector by maximizing the posterior distribution P(**w** | D)

$$\log P(D, \mathbf{w}) = -\frac{1}{2\sigma^2}\mathbf{w}^T\mathbf{w} - \sum_i \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x}\right)$$

Once again, regularized loss minimization! This is the Bayesian interpretation of regularization

Regularized logistic regression

What is the probability of a weight vector **w** being the true one for a dataset D = {<$\mathbf{x}_i$, y_i>}?

$$P(\boldsymbol{w} \mid D) \propto P(\boldsymbol{w}, D) = P(\ D \mid \boldsymbol{w}\ )P(\ \boldsymbol{w})$$

Learning: Find weight vector by <u>maximizing</u> the posterior distribution P(**w** | D)

$$\log P(D, \mathbf{w}) = -\frac{1}{2\sigma^2}\mathbf{w}^T\mathbf{w} - \sum_i \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x}\right)$$

Exercise: Derive the stochastic gradient descent algorithm for logistic regression.

Once again, regularized loss minimization! This is the Bayesian interpretation of regularization

Regularized loss minimization

Learning objective for both SVM & logistic regression:

"loss over training data + regularizer"

- Different loss functions
  - Hinge loss vs. logistic loss
- Same regularizer, but different interpretation
  - Margin vs prior
- Hyper-parameter controls tradeoff between the loss and regularizer
- Other regularizers/loss functions also possible

Questions?

Review of supervised binary classification

1. Supervised learning: The general setting
2. Linear classifiers
3. The Perceptron algorithm
4. Support vector machine
5. Learning as optimization
6. Logistic Regression

Questions?