

Loss Minimization (and a bit about model selection)

Lecture 11

Machine Learning
Fall 2015



Overview

- Learning via Loss Minimization
- Model selection
- Bias and Variance

Overview

- Learning via Loss Minimization
- Model selection
- Bias and Variance

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

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

But distribution D is unknown

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

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

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

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

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?

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

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 linear classifiers: $\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i L(y_i, \mathbf{x}_i, \mathbf{w})$
- 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'

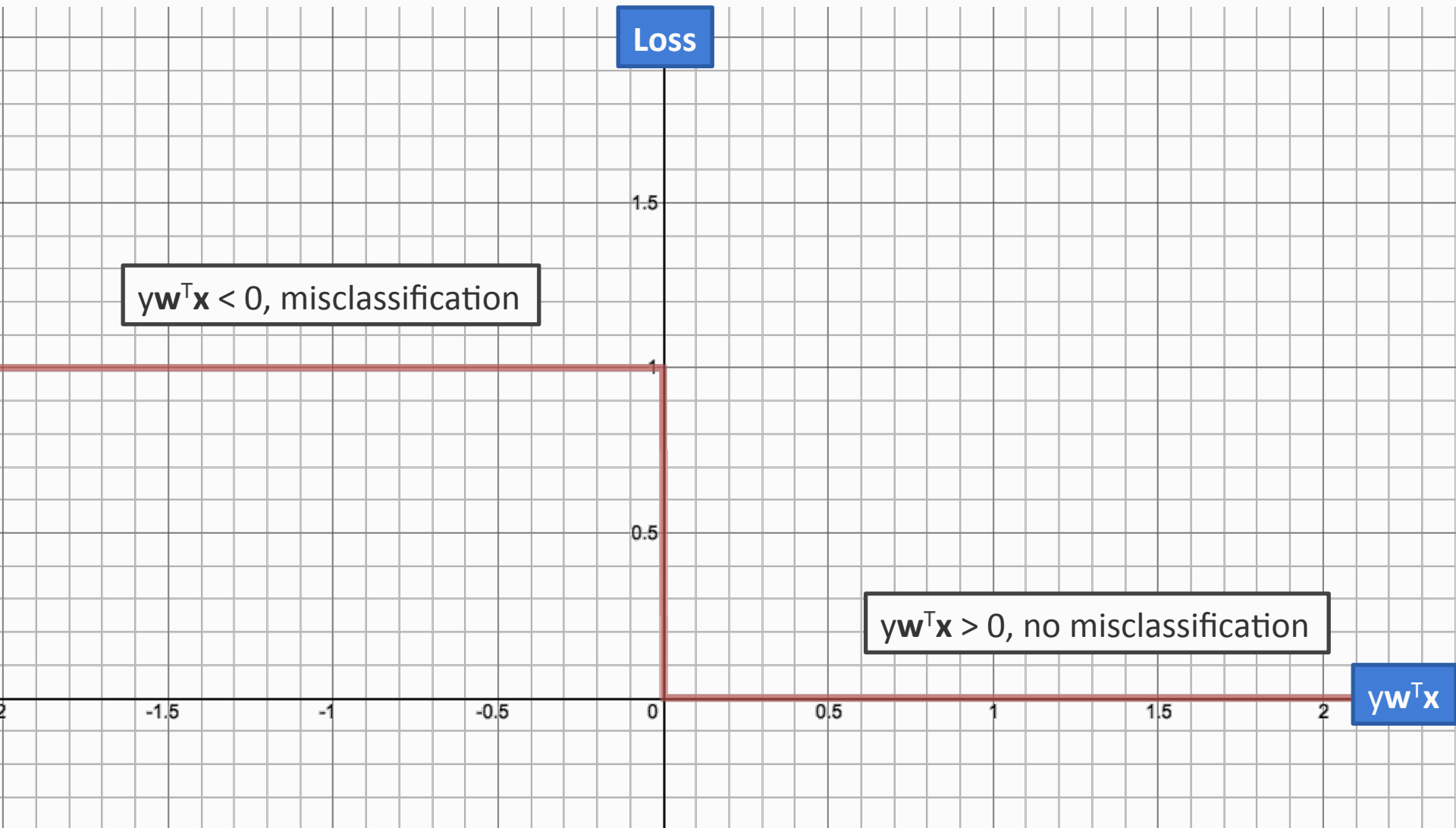
$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y \neq y', \\ 0 & \text{if } y = y'. \end{cases}$$

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

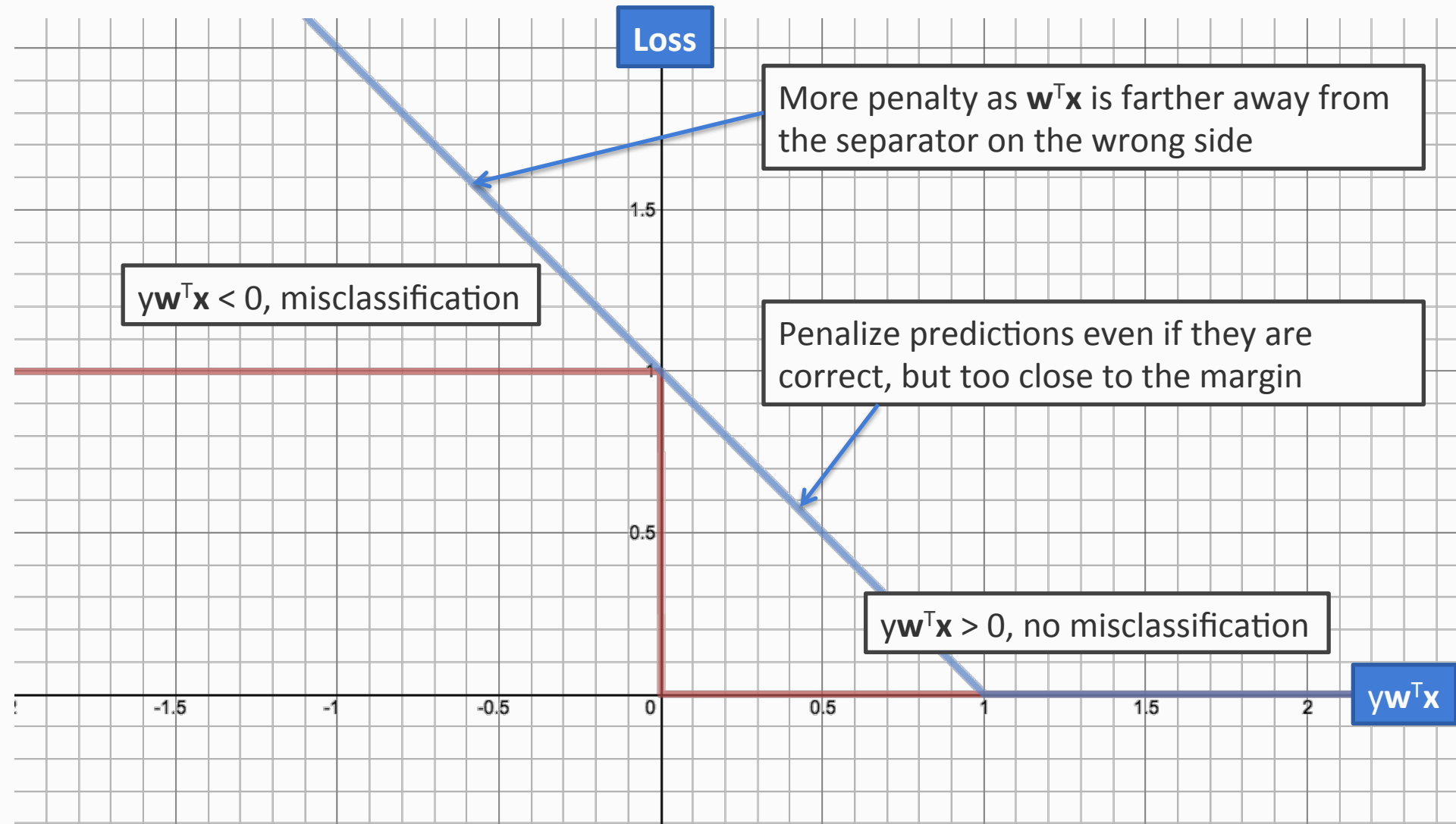
$$L_{0-1}(y, \mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } y \mathbf{w}^T \mathbf{x} \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Minimizing 0-1 loss is intractable. Need surrogates

The 0-1 loss



Compare to the hinge loss



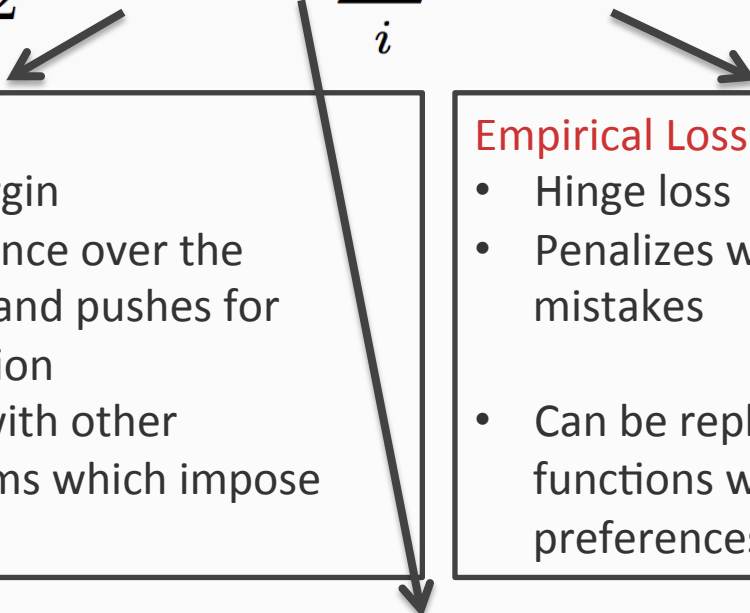
Support Vector Machines

- SVM = linear classifier combined with regularization
- Ideally, we would like to minimize 0-1 loss,
 - But we can't for computational reasons
- SVM minimizes hinge loss

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

- Variants exist

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

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

The loss function zoo

Many loss functions exist

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

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

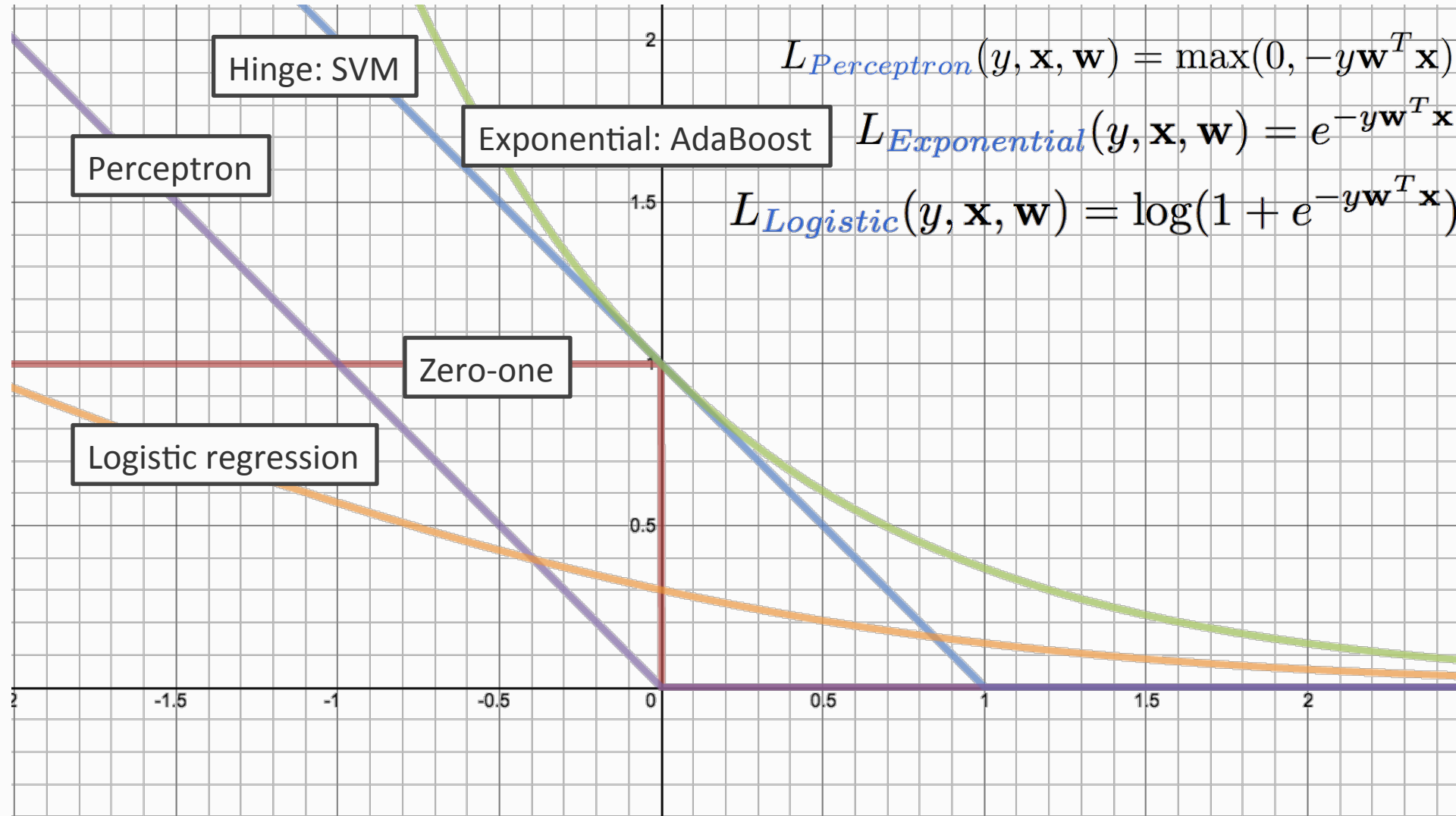
The loss function zoo

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

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

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

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



Overview

- Learning via Loss Minimization
- Model selection
- Bias and Variance

Model selection

Very broadly: Choosing the best model using given data

- What makes a model
 - Features
 - Hyper-parameters
 - Loss
 - Actual weights/parameters
- The learning algorithms we have seen so far only find the last one
 - What about the rest?

Model selection strategies

- Many, many different approaches out there
 - (Chapter 7 of Elements of Statistical Learning Theory)
 - Minimum description length
 - VC dimension and risk minimization
 - *Cross-validation*
 - Bayes factor, AIC, BIC,

Cross-validation

We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters.

How do we know what the best feature set and hyper-parameters are?

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

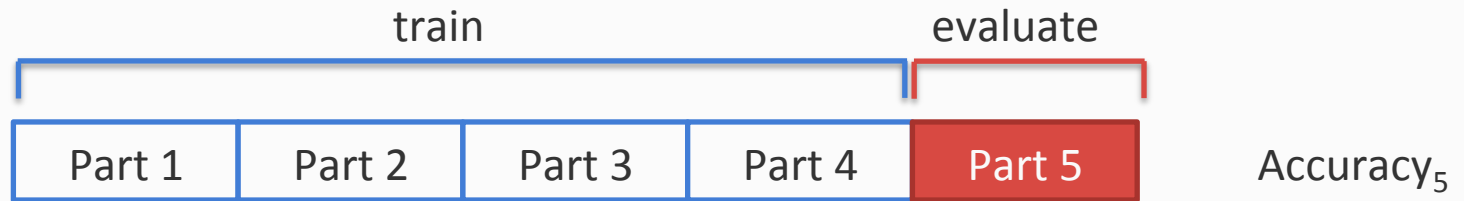
1. Split the data into K (say 5 or 10) equal sized parts

Part 1	Part 2	Part 3	Part 4	Part 5
--------	--------	--------	--------	--------

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts
2. Train a classifier on four parts and evaluate it on the fifth one



K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts
2. Train a classifier on four parts and evaluate it on the fifth one
3. Repeat this using each of the K parts as the *validation set*

Part 1	Part 2	Part 3	Part 4	Part 5	Accuracy ₅
Part 1	Part 2	Part 3	Part 4	Part 5	Accuracy ₄
Part 1	Part 2	Part 3	Part 4	Part 5	Accuracy ₃
Part 1	Part 2	Part 3	Part 4	Part 5	Accuracy ₂
Part 1	Part 2	Part 3	Part 4	Part 5	Accuracy ₁

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts
2. Train a classifier on four parts and evaluate it on the fifth one
3. Repeat this using each of the K parts as the *validation set*
4. The quality of this feature set/hyper-parameter is the average of these K estimates

$$\text{Performance} = (\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5)/5$$

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts
2. Train a classifier on four parts and evaluate it on the fifth one
3. Repeat this using each of the K parts as the *validation set*
4. The quality of this feature set/hyper-parameter is the average of these K estimates
$$\text{Performance} = (\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5)/5$$
5. Repeat for every feature set/hyper parameter choice

Cross-validation

We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters

How do we know what the best feature set and hyper-parameters are?

1. Evaluate every feature set and hyper-parameter using cross-validation (could be computationally expensive)
2. Pick the best according to cross-validation performance
3. Train on full data using this setting

Overview

- Learning via Loss Minimization
- Model selection
- Bias and Variance
 - An informal introduction

Bias and variance

Every learning algorithm requires assumptions about the hypothesis space.

Eg: “My hypothesis space is

- ...linear”
- ...decision trees with 5 nodes”
- ...quadratic kernel”

Bias is the true error (loss) of the *best* predictor in the hypothesis set

- What will the bias be if the hypothesis set can not represent the target function? (high or low?)
 - Bias will be non zero, possibly high
- **Underfitting**: When bias is high

Bias and variance

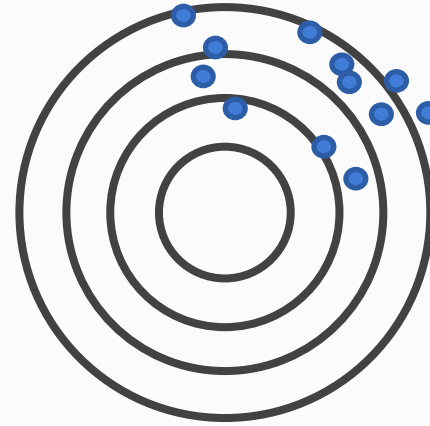
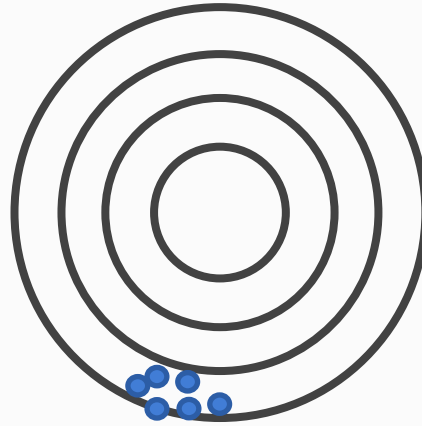
- The performance of a classifier is dependent on the specific training set we have
 - Perhaps the model will change if we slightly change the training set
- **Variance**: Describes how much the best classifier depends on the training set
- **Overfitting**: High variance
- Variance
 - Increases when the classifiers become more complex
 - Decreases with larger training sets

Let's play darts

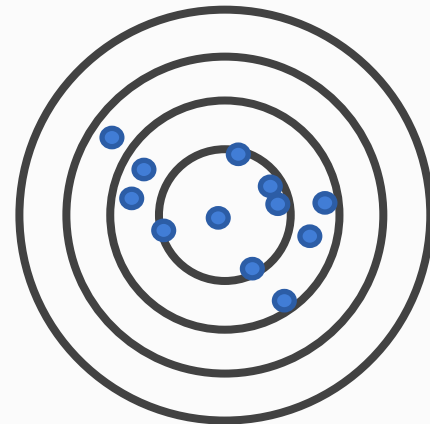
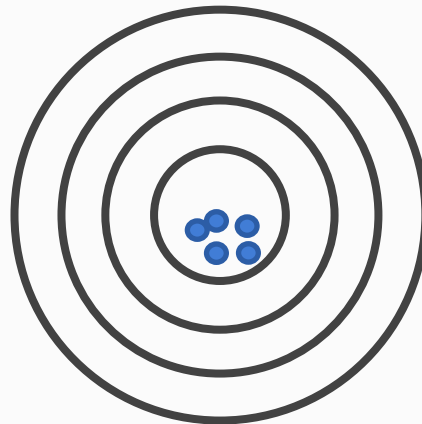
Suppose the true concept is the center

Each dot is a model that is learned from a different dataset

High bias



Low bias



Low variance

High variance

Bias variance tradeoffs

- Error = bias + variance (+ noise)
- High bias \Rightarrow both training and test error can be high
 - Arises when the classifier can not represent the data
- High variance \Rightarrow training error can be low, but test error will be high
 - Arises when the learner overfits the training set

Bias variance tradeoff has been studied extensively in the context of regression
Generalized to classification (Pedro Domingos, 2000)

Managing of bias and variance

- **Ensemble methods** reduce variance
 - Multiple classifiers are combined
 - Eg: Bagging, boosting
- **K nearest neighbors**
 - Increasing k generally increases bias, reduces variance
- **Decision trees of a given depth**
 - Increasing depth decreases bias, increases variance
- **SVMs**
 - Higher degree polynomial kernels decreases bias, increases variance
 - Stronger regularization increases bias, decreases variance

Summary of this section

- Learning via Loss Minimization
 - Write down a loss function
 - Minimize empirical loss
 - Regularize to avoid overfitting
 - Widely applicable, different loss functions and regularizers
- Model selection
 - How to select the model, features, hyper-parameters
 - Related to learning
- Bias and Variance
 - Rich exploration in statistics
 - Provides a different view of learning criteria