Introduction to Machine Learning

Varun Chandola <chandola@buffalo.edu>

June 30, 2014

Outline

Outline

Supervised Learning

- **Classification Task**: Use a model (or hypothesis) to assign a label (or value) to a previously unseen instance, $\mathbf{x}$
- **Learning Task**: *Learn* the hypothesis or model parameters using training data $\langle \mathbf{x}_i, y_i \rangle_{i=1}^{N}$

Ideal Learning Algorithm

1. Fast
2. **Generalizable**

Learning Parameters Via Loss Minimization

- **Notion of Loss** - how badly does the model ($\mathcal{M}$) perform on a $\mathbf{x}$ if y is the correct output?
- What can the loss function be ($loss(\mathbf{x}, y; \mathcal{M})$)?
- How do you measure this loss?
- If we had all possible realizations of $\mathbf{x}$ and corresponding y
- If we have a probability distribution defined over $\mathbf{x}, y$

$$\mathbb{E}_{p(\mathbf{x},y)}[loss(\mathbf{x}, y; \mathcal{M})]$$

  - **Risk minimization**
- Usually we do not have any of these

Learning Parameters Via Loss Minimization

- **Notion of Loss** - how badly does the model ($\mathcal{M}$) perform on a $\mathbf{x}$ if y is the correct output?
- What can the loss function be ($loss(\mathbf{x}, y; \mathcal{M})$)?
- How do you measure this loss?
- If we had all possible realizations of $\mathbf{x}$ and corresponding y
- If we have a probability distribution defined over $\mathbf{x}, y$

$$\mathbb{E}_{p(\mathbf{x},y)}[loss(\mathbf{x}, y; \mathcal{M})]$$

  - **Risk minimization**
- Usually we do not have any of these

Empirical Risk Minimization

- What we do have is an *empirical distribution* (from the training data)!
- **Empirical Risk**

$$\mathbb{E}_{\tilde{p}(\mathbf{x},y)}[loss(\mathbf{x},y;\mathcal{M})] = \frac{1}{N}\sum_{i=1}^{N}[loss(\mathbf{x}_i,y_i;\mathcal{M})]$$

- Find model parameters (or hypothesis) that minimizes the empirical risk

$$\arg\min_{\mathcal{M}} \mathbb{E}_{\tilde{p}(\mathbf{x},y)}[loss(\mathbf{x},y;\mathcal{M})] = \arg\min_{\mathcal{M}} \frac{1}{N}\sum_{i=1}^{N}[loss(\mathbf{x}_i,y_i;\mathcal{M})]$$

Structural Risk Minimization

- What if ERM gives multiple solution with "equal" risk?
  - Which model to choose?
  - Choose the one with least complexity (Akaike Information Criterion, Minimum Description Length principle, **Structural Risk Minimization** (SRM))
  - Ensures better generalizability
- SRM provides a *trade-off* between complexity of an algorithm (**VC Dimension**) and quality of *fit* on the training data (**empirical error**)
  - Low VC dimension $\Rightarrow$ Low complexity $\Rightarrow$ Better generalizability

Structural Risk Minimization

- What if ERM gives multiple solution with "equal" risk?
  - Which model to choose?
  - Choose the one with least complexity (Akaike Information Criterion, Minimum Description Length principle, **Structural Risk Minimization** (SRM))
  - Ensures better generalizability
- SRM provides a *trade-off* between complexity of an algorithm (**VC Dimension**) and quality of *fit* on the training data (**empirical error**)
  - Low VC dimension $\Rightarrow$ Low complexity $\Rightarrow$ Better generalizability

Structural Risk Minimization

- What if ERM gives multiple solution with "equal" risk?
  - Which model to choose?
  - Choose the one with least complexity (Akaike Information Criterion, Minimum Description Length principle, **Structural Risk Minimization** (SRM))
  - Ensures better generalizability
- SRM provides a *trade-off* between complexity of an algorithm (**VC Dimension**) and quality of *fit* on the training data (**empirical error**)
  - Low VC dimension $\Rightarrow$ Low complexity $\Rightarrow$ Better generalizability

Regularized Risk Minimization

- *Regularize* the complexity of the model

$$\arg\min_{\mathcal{M}} \frac{1}{N}\sum_{i=1}^{N}[loss(\mathbf{x}_i, y_i; \mathcal{M})] + R(\mathcal{M})$$

Outline

How do classification algorithms differ from each other?

- What is the output?
  - Probability ($P(y|\mathbf{x})$) - Bayesian Classifier
  - Score ($s(\mathbf{x})$) with a threshold? - Support Vector Machines, Decision Trees
- How is $P(y|\mathbf{x})$ computed?
  - From $P(\mathbf{x}|y)$ (Generative) - Naive Bayes Classifier
  - Directly (Discriminative) - Logistic Regression
- What is the loss function?
  - 0-1 loss (Not easy to regularize ☹)
  - Log loss (Logistic Regression, CRF, Max ent models)
  - Hinge Loss (SVM)
- What is the regularizer?

Jumping to Linear Classifiers

- Calculate $\mathbf{w}^\top \mathbf{x} + b$ - linear combination of features
- Learning can be posed as a general optimization problem

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^\top \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

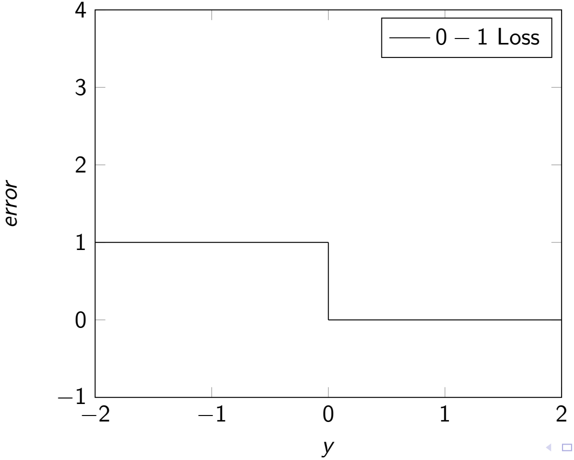
- $\mathbb{I}$ is an **indicator function** (1 if (.) is 0, 0 otherwise)
- Objective function = **Loss function** + λ**Regularizer**
- Objective function wants to **fit training data well** and **have simpler solution**

0-1 Loss is Hard to Optimize

- Combinatorial optimization problem
- NP-hard
- No polynomial time algorithm
- Loss function is non-smooth, non-convex
- Small changes in $\mathbf{w}, b$ can change the loss by lot

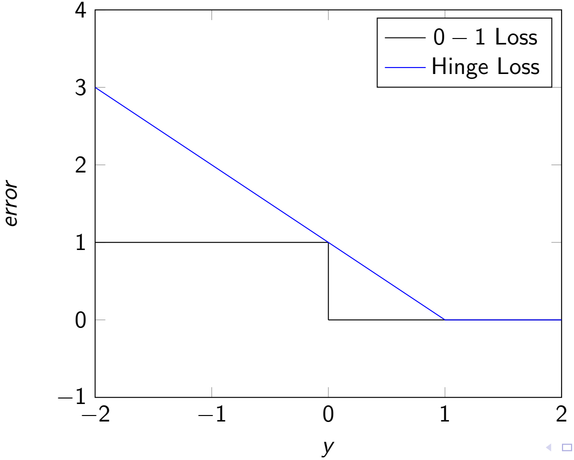
Approximations to 0-1 Loss

- Different linear classifiers use different approximations to 0-1 loss
  - Also known as *surrogate loss functions*



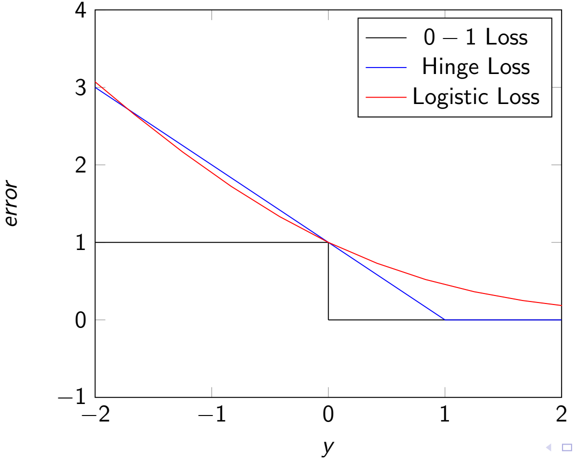
Approximations to 0-1 Loss

- Different linear classifiers use different approximations to 0-1 loss
  - Also known as *surrogate loss functions*



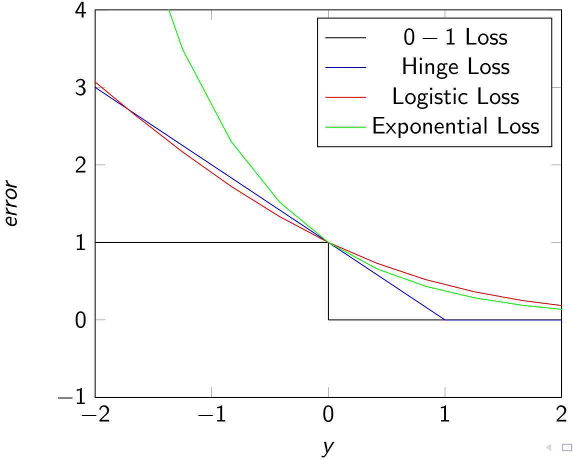
Approximations to 0-1 Loss

- Different linear classifiers use different approximations to 0-1 loss
  - Also known as *surrogate loss functions*



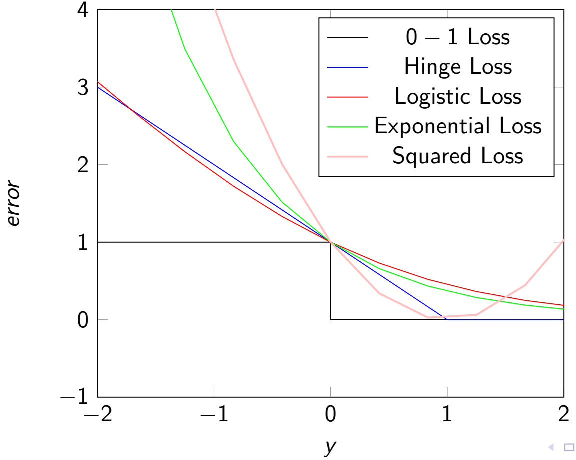
Approximations to 0-1 Loss

- Different linear classifiers use different approximations to 0-1 loss
  - Also known as *surrogate loss functions*



Approximations to 0-1 Loss

- Different linear classifiers use different approximations to 0-1 loss
  - Also known as *surrogate loss functions*



Role of Regularizers

- Recall the optimization problem for linear classification

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^\top \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

- What is the role of the regularizer term?
  - Ensure simplicity
- Ideally we want most entries of $\mathbf{w}$ to be zero
- Why?
- Desired minimization

$$R(\mathbf{w}, b) = \sum_{d=1}^{D} \mathbb{I}(w_d \neq 0)$$

- NP Hard

Role of Regularizers

- Recall the optimization problem for linear classification

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^\top \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

- What is the role of the regularizer term?
  - Ensure simplicity
- Ideally we want most entries of $\mathbf{w}$ to be zero
- Why?
- Desired minimization

$$R(\mathbf{w}, b) = \sum_{d=1}^{D} \mathbb{I}(w_d \neq 0)$$

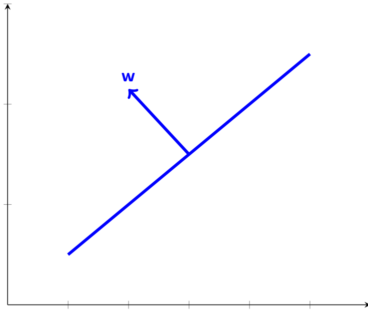
- NP Hard

Approximate Regularization

- **Norm based regularization**
  - l_2 squared norm
$$\|\mathbf{w}\|_2^2 = \sum_{d=1}^{D} w_d^2$$
  - l_1 norm
$$\|\mathbf{w}\|_1 = \sum_{d=1}^{D} |w_d|$$
  - l_p norm
$$\|\mathbf{w}\|_p = (\sum_{d=1}^{D} w_d^p)^{1/p}$$
  - Norm becomes non-convex for $p < 1$
  - l_1 norm gives best results
  - l_2 norm is easiest to deal with

Linear Hyperplane

- Separates a D-dimensional space into two half-spaces
- Defined by $\mathbf{w} \in \Re^D$
  - *Orthogonal* to the hyperplane
  - This **w** goes through the origin
  - How do you check if a point lies "above" or "below" **w**?
  - What happens for points **on w**?



Make hyperplane not go through origin

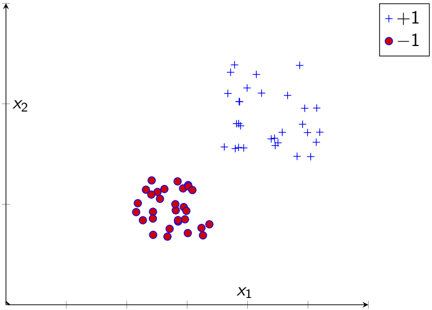
- Add a bias b
  - $b > 0$ - move along $\mathbf{w}$
  - $b < 0$ - move opposite to $\mathbf{w}$
- How to check if point lies above or below $\mathbf{w}$?
  - If $\mathbf{w}^\top \mathbf{x} + b > 0$ then $\mathbf{x}$ is *above*
  - Else, *below*

Outline

Maximum Margin Classifiers

$$y = \mathbf{w}^\top \mathbf{x} + b$$

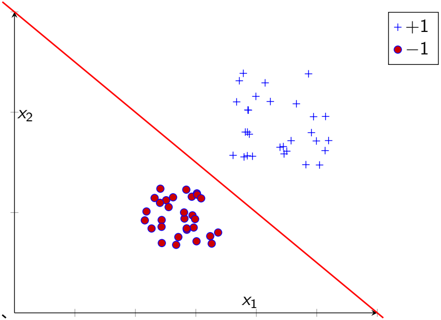
- Remember the Perceptron!
- If data is linearly separable
  - Perceptron training guarantees learning the decision boundary
- There can be other boundaries
  - Depends on initial value for $\mathbf{w}$
- **But what is the best boundary?**



Maximum Margin Classifiers

$$y = \mathbf{w}^\top \mathbf{x} + b$$

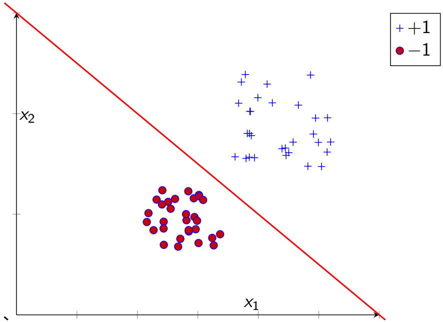
- Remember the Perceptron!
- If data is linearly separable
  - Perceptron training guarantees learning the decision boundary
- There can be other boundaries
  - Depends on initial value for $\mathbf{w}$
- **But what is the best boundary?**



Maximum Margin Classifiers

$$y = \mathbf{w}^\top \mathbf{x} + b$$

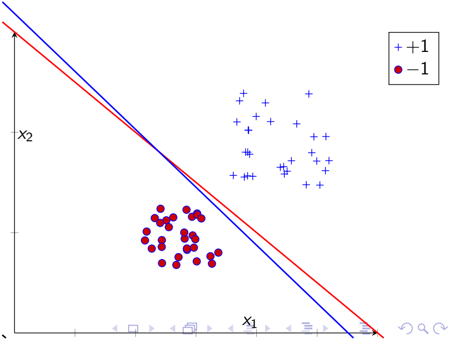
- Remember the Perceptron!
- If data is linearly separable
  - Perceptron training guarantees learning the decision boundary
- There can be other boundaries
  - Depends on initial value for $\mathbf{w}$
- **But what is the best boundary?**



Maximum Margin Classifiers

$$y = \mathbf{w}^\top \mathbf{x} + b$$

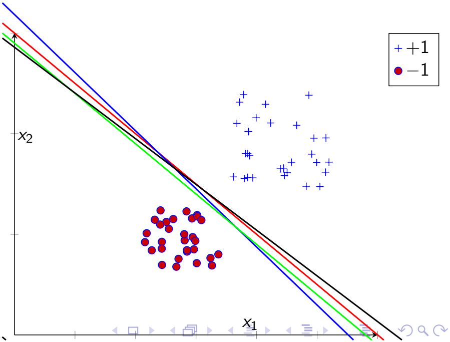
- Remember the Perceptron!
- If data is linearly separable
  - Perceptron training guarantees learning the decision boundary
- There can be other boundaries
  - Depends on initial value for $\mathbf{w}$
- **But what is the best boundary?**



Maximum Margin Classifiers

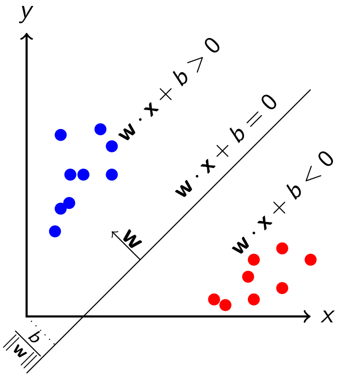
$$y = \mathbf{w}^\top \mathbf{x} + b$$

- Remember the Perceptron!
- If data is linearly separable
  - Perceptron training guarantees learning the decision boundary
- There can be other boundaries
  - Depends on initial value for $\mathbf{w}$
- **But what is the best boundary?**



The Notion of Maximum Margin

- If multiple solutions classify the training data perfectly
- Find one which will give the smallest *generalization error*
- Equivalent to choosing the decision surface with **Maximum Margin**



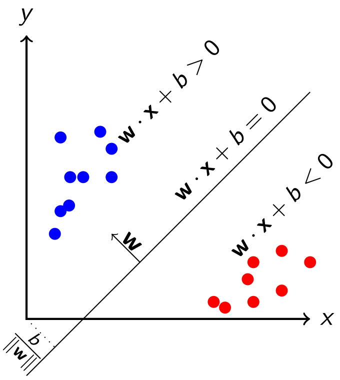
Line as a Decision Surface

- Decision boundary represented by the hyperplane **w**
- For binary classification, **w** points **towards** the positive class

Decision Rule

$$y = sign(\mathbf{w}^\top \mathbf{x} + b)$$

- $\mathbf{w}^\top \mathbf{x} + b > 0 \Rightarrow y = +1$
- $\mathbf{w}^\top \mathbf{x} + b < 0 \Rightarrow y = -1$



What is a Margin?

- **Margin** is the distance between an example and the decision line
- Denoted by γ
- For a positive point:

$$\gamma = \frac{\mathbf{w}^\top \mathbf{x} + b}{\|\mathbf{w}\|}$$

- For a negative point:

$$\gamma = -\frac{\mathbf{w}^\top \mathbf{x} + b}{\|\mathbf{w}\|}$$

Functional Interpretation

- Margin **positive** if prediction is **correct**; **negative** if prediction is **incorrect**

Outline

Support Vector Machines

- A hyperplane based classifier defined by **w** and b
- Find hyperplane with *maximum separation margin* on the training data
- Assume that data is linearly separable (will relax this later)
  - Zero training error (loss)

SVM Prediction Rule

$$y = sign(\mathbf{w}^\top \mathbf{x} + b)$$

SVM Learning

- **Input**: Training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$
- **Objective**: Learn **w** and b that maximizes the margin

SVM Learning

- SVM learning task as an optimization problem
- Find $\mathbf{w}$ and b that gives zero training error
- Maximizes the margin $(= \frac{2}{\|w\|})$
- Same as minimizing $\|\mathbf{w}\|$

Optimization Formulation

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{\|\mathbf{w}\|^2}{2} \\ \text{subject to} \quad & y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \ n = 1, \ldots, N. \end{aligned}$$

- **Optimization** with N linear inequality constraint

A Different Interpretation of Margin

- What impact does the margin have on $\mathbf{w}$?
- Large margin $\Rightarrow$ Small $\|\mathbf{w}\|$
- Small $\|\mathbf{w}\|$ $\Rightarrow$ regularized/simple solutions
- Simple solutions $\Rightarrow$ Better generalizability (*Occam's Razor*)

Solving the Quadratic Optimization Problem

Optimization Formulation

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \quad y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \;\; n = 1, \ldots, N.$$

- There is one quantity to minimize and N constraints
- **Primal formulation** - Lagrange Multipliers
- Bring constraints into the objective function

Primal Lagrangian Formulation

$$\underset{\mathbf{w},b,\alpha}{\text{minimize}} \quad L_P(\mathbf{w}, b, \alpha) = \frac{\|\mathbf{w}\|^2}{2} + \sum_{n=1}^{N} \alpha_n(1 - y_n(\mathbf{w}^\top \mathbf{x}_n + b))$$

$$\text{subject to} \quad \alpha_n \geq 0; n = 1, \ldots, N.$$

Solving the Quadratic Optimization Problem

Optimization Formulation

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \quad y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \; n = 1, \ldots, N.$$

- There is one quantity to minimize and N constraints
- **Primal formulation** - Lagrange Multipliers
- Bring constraints into the objective function

Primal Lagrangian Formulation

$$\underset{\mathbf{w},b,\boldsymbol{\alpha}}{\text{minimize}} \quad L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{\|\mathbf{w}\|^2}{2} + \sum_{n=1}^{N} \alpha_n(1 - y_n(\mathbf{w}^\top \mathbf{x}_n + b))$$

$$\text{subject to} \quad \alpha_n \geq 0; n = 1, \ldots, N.$$

More About Quadratic Optimization

- The **Lagrangian** is a lower bound on the original problem
- Find *optimal* values of $\mathbf{w}$ and b, w.r.t. $\boldsymbol{\alpha}$ by setting the derivative to 0:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0$$

- Use the above results in the Primal Lagrangian L_P to get the Dual Lagrangian:

$$\underset{\mathbf{w},b,\boldsymbol{\alpha}}{\text{minimize}} \quad L_D(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^\top \mathbf{x}_n)$$

$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \alpha_n \geq 0; n = 1, \ldots, N.$$

Solving for α_n

- A **Quadratic Programming** problem in α
- Use "off-the-shelf" quadratic solvers for L_D
  - quadprog (MATLAB), CVXOPT
- Solution should satisfy certain conditions
- Also known as the **Karush-Kuhn-Tucker** (KKT) Conditions

The Karush-Kuhn-Tucker Conditions

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} L_P(\mathbf{w}, b, \alpha) &= \mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = 0 && (1)\\
\frac{\partial}{\partial b} L_P(\mathbf{w}, b, \alpha) &= -\sum_{n=1}^{N} \alpha_n y_n = 0 && (2)\\
y_n\{\mathbf{w}^\top \mathbf{x}_n + b\} - 1 &\geq 0 && (3)\\
\alpha_n &\geq 0 && (4)\\
\alpha_n(y_n\{\mathbf{w}^\top \mathbf{x}_n + b\} - 1) &= 0 && (5)
\end{aligned}$$

From Primal to Dual

- Using KKT conditions (1) and (2), we get:

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$
$$0 = \sum_{n=1}^{N} \alpha_n y_n$$

- Now we only need to optimize using α_n's
- This is done using the **dual formulation**

$$\underset{\mathbf{w},b,\alpha}{\text{maximize}} \quad L_D(\mathbf{w}, b, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^\top \mathbf{x}_n)$$
$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \alpha_n \geq 0 \; n = 1, \ldots, N.$$

Two Key Observations from Dual Formulation

Observation 1: Dot Product Formulation

- All training examples ($\mathbf{x}_n$'s) occur in *dot/inner products*
- Also recall the prediction using SVMs

$$
\begin{aligned}
y^* &= sign(\mathbf{w}^\top \mathbf{x}^* + b) \\
&= sign((\sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n)^\top \mathbf{x}^*) \\
&= sign(\sum_{n=1}^{N} \alpha_n y_n \boxed{(\mathbf{x}_n^\top \mathbf{x}^*)})
\end{aligned}
$$

- Replace the dot products with kernel functions
  - Kernel or non-linear SVM

Two Key Observations from Dual Formulation

Observation 2: Most α_n's are 0

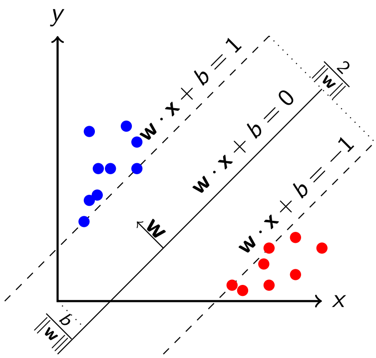
- KKT condition #5:

$$\alpha_n(y_n\{\mathbf{w}^\top \mathbf{x}_n + b\} - 1) = 0$$

- If $\mathbf{x}_n$ **not** on margin

$$y_n\{\mathbf{w}^\top \mathbf{x}_n + b\} > 1$$
$$\Rightarrow \qquad \alpha_n = 0$$

- $\alpha_n \neq 0$ only for $\mathbf{x}_n$ on margin
- These are the **support vectors**
- Only need these for prediction



What if data is not linearly separable?

- Cannot go for zero training error
- Still learn a maximum margin hyperplane
    1. Allow some examples to be misclassified
    2. Allow some examples to fall **inside** the margin
- How do you set up the optimization for SVM training

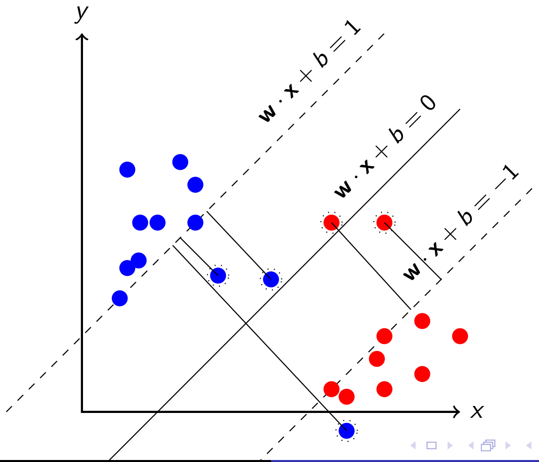
What if data is not linearly separable?

- Cannot go for zero training error
- Still learn a maximum margin hyperplane
  1. Allow some examples to be misclassified
  2. Allow some examples to fall **inside** the margin
- How do you set up the optimization for SVM training

What if data is not linearly separable?

- Cannot go for zero training error
- Still learn a maximum margin hyperplane
  1. Allow some examples to be misclassified
  2. Allow some examples to fall **inside** the margin
- How do you set up the optimization for SVM training

Cutting Some Slack



Introducing Slack Variables

- **Separable Case**: To ensure zero training loss, constraint was

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n = 1 \ldots N$$

- **Non-separable Case**: Relax the constraint

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n \quad \forall n = 1 \ldots N$$

- ξ_n is called **slack variable** ($\xi_n \geq 0$)
- For misclassification, $\xi_n > 1$

Introducing Slack Variables

- **Separable Case**: To ensure zero training loss, constraint was

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad \forall n = 1 \ldots N$$

- **Non-separable Case**: Relax the constraint

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n \quad \forall n = 1 \ldots N$$

- ξ_n is called **slack variable** ($\xi_n \geq 0$)
- For misclassification, $\xi_n > 1$

Relaxing the Constraint

- It is OK to have some misclassified training examples
  - Some ξ_n's will be non-zero
- Minimize the number of such examples
  - Minimize $\sum_{n=1}^{N} \xi_n$
- Optimization Problem for Non-Separable Case

$$\underset{\mathbf{w},b}{\text{maximize}} \quad f(\mathbf{w},b) = \|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

$$\text{subject to} \quad y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \xi_n \geq 0 \; n = 1, \ldots, N.$$

- C controls the impact of margin and the margin error.

Relaxing the Constraint

- It is OK to have some misclassified training examples
  - Some ξ_n's will be non-zero
- Minimize the number of such examples
  - Minimize $\sum_{n=1}^{N} \xi_n$
- Optimization Problem for Non-Separable Case

$$\begin{aligned} & \underset{\mathbf{w},b}{\text{maximize}} && f(\mathbf{w}, b) = \|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ & \text{subject to} && y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \xi_n \geq 0 \; n = 1, \ldots, N. \end{aligned}$$

- C controls the impact of margin and the margin error.

Estimating Weights

- What is the role of C?
- Similar optimization procedure as for the separable case (QP for the dual)
- Weights have the same expression

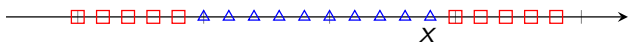
$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

- All training examples exist in dot products (*kernelizable*)
- Support vectors are slightly different
  1. Points on the margin ($\xi_n = 0$)
  2. Inside the margin but on the correct side ($0 < \xi_n < 1$)
  3. On the wrong side of the hyperplane ($\xi_n \geq 1$)

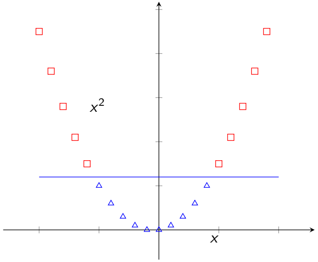
Outline

Why Use Kernels?

- $x \in \Re$
- No linear separator

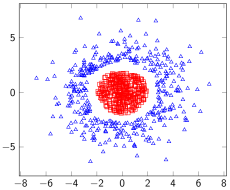


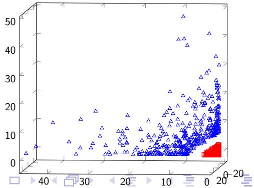
- Map $x \to \{x, x^2\}$
- Separable in 2D space



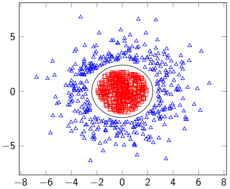
Another Example

- $\mathbf{x} \in \Re^2$
- No linear separator
- Map $\mathbf{x} \rightarrow \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$
- A circle as the decision boundary

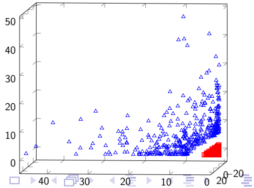




Another Example



- $\mathbf{x} \in \Re^2$
- No linear separator
- Map $\mathbf{x} \rightarrow \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$
- A circle as the decision boundary



The Gaussian Kernel

- The *squared dot product* kernel ($\mathbf{x}, \mathbf{x}' \in \Re^2$):

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}' \triangleq \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

$$\phi(\mathbf{x}) = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$$

- What about the Gaussian kernel (radial basis function)?

$$k(\mathbf{x}, \mathbf{x}') = exp\left(-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}'||^2\right)$$

Why is the Gaussian Kernel Mapping to Infinite Dimensions

- Assume $\sigma = 1$ and $\mathbf{x} \in \Re$ (denoted as x)

$$\begin{aligned} k(x, x') &= exp(-x^2)exp(-x'^2)exp(2xx') \\ &= exp(-x^2)exp(-x'^2)\sum_{k=0}^{\infty}\frac{2^k x^k x'^k}{k!} \\ &= \sum_{k=0}^{\infty}\left(\frac{2^{k/2}}{\sqrt{k!}}x^k exp(-x^2)\right)\left(\frac{2^{k/2}}{\sqrt{k!}}x'^k exp(-x'^2)\right) \end{aligned}$$

- *Using Maclaurin Series Expansion*

$$k(x, x') = \begin{pmatrix} 1 \\ 2^{1/2}x^1 exp(-x^2) \\ \frac{2^{2/2}}{2}x^2 exp(-x^2) \\ \vdots \end{pmatrix} \times \begin{pmatrix} 1 \\ 2^{1/2}x'^1 exp(-x'^2) \\ \frac{2^{2/2}}{2}x'^2 exp(-x'^2) \\ \vdots \end{pmatrix}^\top$$

SVM Extensions

1. Multiple classes
    - One vs. Rest
    - One vs. One
2. One class SVM
3. Transductive SVM (Semi-supervised Learning)
4. Support Vector Regression
5. Custom kernels (Use a *Gram* matrix)

References