Support Vector Machines

Mark Dredze

Machine Learning
CS 601.475

Algorithm: Logistic Regression

$$p(y=1|\mathbf{x},\mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}^T\cdot\mathbf{x}}}$$

- Train: given data X and Y
  - Initialize w to starting value
  - Repeat until convergence
    - Compute the value of the derivative for X,Y and w
    - Update w by taking a gradient step
- Predict: using the learned w, compute p(y|x,w)
- Loss function: logistic
  - Modeled data as probability

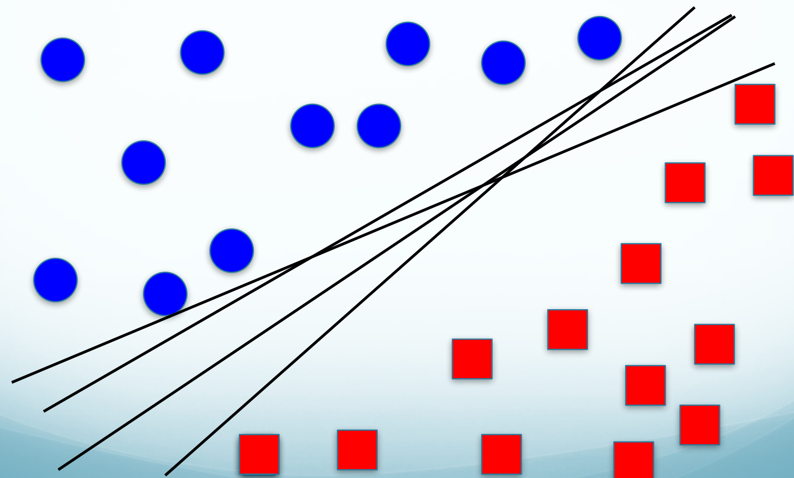
Algorithm: Perceptron

- Initialize w and η
- On each round
  - Receive example x
  - Predict $\hat{y} = \mathrm{sign}(w \cdot x)$
  - Receive correct label $y \in \{+1,-1\}$
  - Suffer loss $\ell_{0/1}(y,\hat{y})$
  - Update w: $w^{i+1} = w^i + \eta\, y_i\, x_i$
- Loss function: 0/1 discriminant classifier

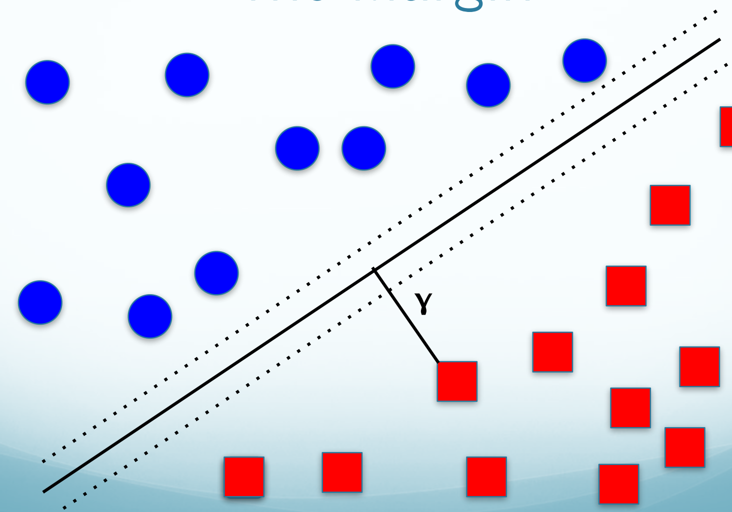
Lingering Questions

- Perceptron picks one separating hyperplane (of many)
  - What would we do if we saw all of the data (batch)?
  - We'd pick the best separating hyperplane!
- Which separating hyperplane is the best?
  - Let's look at the geometric model
- Better solutions for non-linear data?

Geometric Representation



The Margin



Functional Margin

- Prediction and y should agree to get large margin

$$\hat{\gamma}^i = y_i(w^T x + b)$$

- What if we double w?

$$\hat{\gamma}^i = y_i(2w^T x + 2b)$$

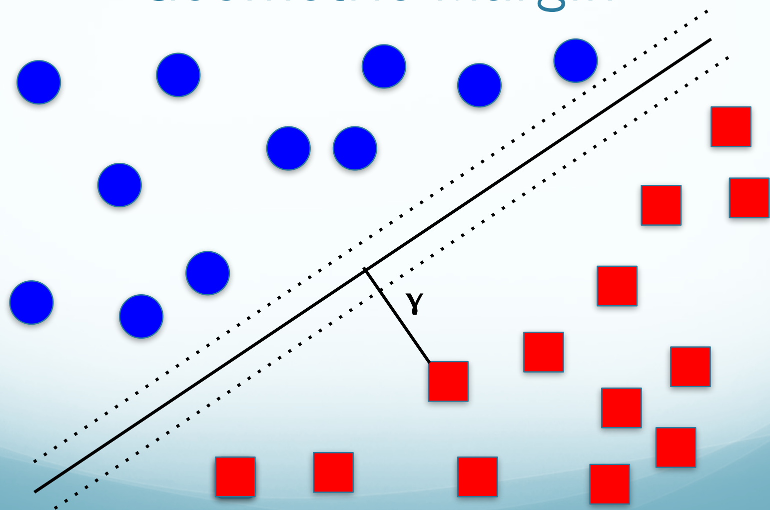
- Doubles margin, but no practical change
  - We will address this in a moment

Functional Margin of Data

- Given a training set of size N:
  - Smallest margin

$$\hat{\gamma} = \min_{i=1,\ldots,N} \hat{\gamma}^i$$

Geometric Margin



Geometric Margin

- Size of γ?
- $\frac{w}{||w||}$ is a unit length vector pointing in the direction of w
- γ intersects with the decision boundary at
$$x_i - \gamma^i \cdot \frac{w}{||w||}$$
and points on the boundary must give a prediction of 0
$$\gamma^i = y_i \left(\left(\frac{w}{||w||} \right)^T x_i + \frac{b}{||w||} \right)$$
if ||w|| = 1 then functional = geometric margin

Max-Margin Principle

- Assuming the observed data is linearly separable
- Select the hyperplane that separates the data with the maximal margin
- Why?
  - New examples are likely to be close to old examples
  - Gives the best generalization error on new data

Maximum Geometric Margin

$$\max_{\gamma, w, b} \quad \gamma$$
$$s.t. \quad y_i(w^T x_i + b) \geq \gamma, i = 1, \ldots, N$$
$$||w|| = 1$$

- Every training instance has margin at least γ
- ||w|| constraint means geometric = functional margin
- Problem: ||w|| constraint is non-convex!

Maximum Geometric Margin

- Functional and geometric related by

$$\gamma = \frac{\hat{\gamma}}{||w||}$$

- Equivalently consider

$$\max_{\hat{\gamma},w,b} \quad \frac{\hat{\gamma}}{||w||}$$

$$s.t. \quad y_i(w^T x_i + b) \geq \hat{\gamma}, i = 1, \ldots, N$$

Maximum Geometric Margin

- Recall: we can arbitrarily scale w!
  - Arbitrarily set $\gamma = 1$

$$\min_{w,b} \quad \frac{1}{2}||w||^2$$

$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1, i = 1, \ldots, N$$

  - min $||w||^2$ same as max $1/||w||$
- Quadratic program (QP): quadratic objective with linear constraints
- Result is optimal margin classifier

Support Vector Machines

Fitting a function to data

- Fitting: Batch optimization method: QP solver
- Function: hyperplane with functional margin >= 1
  - New loss function?
- Data: Train in batch mode

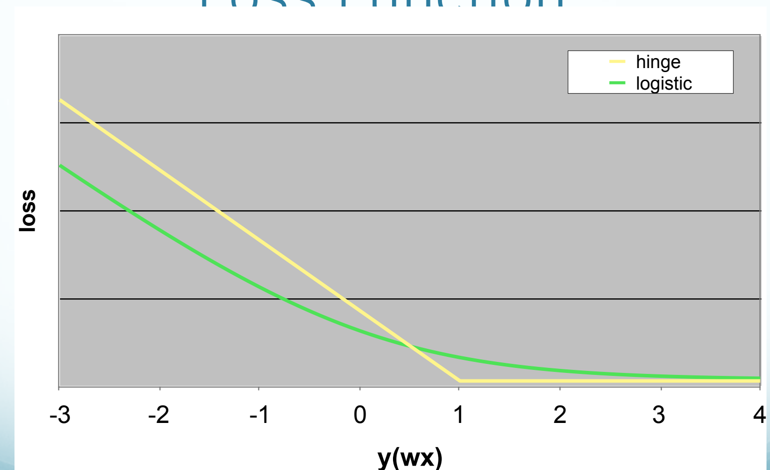
SVM vs. Logistic Regression

- Both minimize the empirical loss with some regularization
  - SVM: $$\frac{1}{n}\sum_{i=1}^{n}(1 - y_i[w \cdot x_i])^+ + \lambda\frac{1}{2}\|w\|^2$$
  - Logistic: $$\frac{1}{n}\sum_{i=1}^{n}\underbrace{-\log g(y_i[w \cdot x_i])}_{-P(y_i|x_i,w)} + \lambda\frac{1}{2}\|w\|^2$$
    - $(z)^+$ indicates only positive values
    - $g(z) = (1+\exp(-z))^{-1}$ is the logistic function

Loss Function

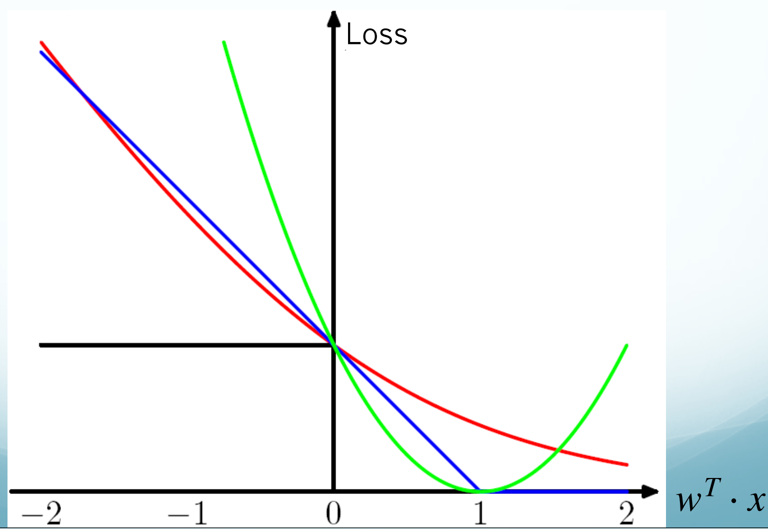
- Both minimize $\frac{1}{n}\sum_{i=1}^{n}\ell(y_i[w\cdot x_i]) + \lambda\frac{1}{2}\|w\|^2$
- Different loss functions
- SVM: Hinge Loss
  $\ell(w,x,y) = \max(0, 1 - y[w\cdot x])$
- Logistic regression: Logistic loss
  $\ell(w,x,y) = \log(1 + \exp\{-y[w\cdot x]\})$

Loss Function



Thanks to Jason Eisner for figure

Rethinking Loss Functions



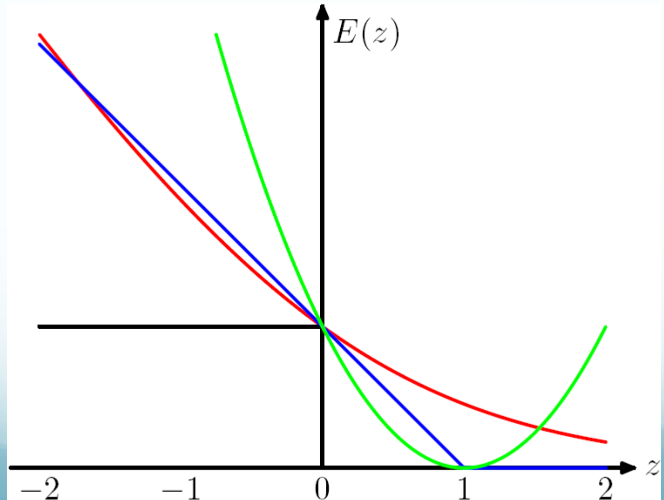
Perceptron: How to Update?

- How do we update w to improve our loss?
- Define an error function based on 0/1 loss

$$L_w(y) = \sum_{i}^{N} max(0, -y_i w \cdot x_i)$$

- What is the difference?

Rethinking Loss Functions



The Perceptron Connection

- SVM minimizes the Perceptron but goes further
- Perceptron gives local updates, SVM gives global updates
- SVM is more aggressive: max-margin principle
- Could we apply max-margin to online learning?
  - Yes! Perceptron with margin
  - Other methods as well

Support Vector Machines

Fitting a function to data

- Fitting: Batch optimization method
- Function: select hyperplane that ensures a fixed margin, L2 regularization
  - Loss: hinge loss
- Data: Train in batch mode

Another Formulation

Lagrangians for Constrained Optimization

- Problem $\min_{\mathbf{w}} f(\mathbf{w})$

  s.t. $g_i(\mathbf{w}) \leq 0,\ i = 1,\ldots,n$

  $h_j(\mathbf{w}) = 0,\ j = 1,\ldots,l$
- Define

$$\mathscr{L}(\mathbf{w},\alpha,\beta) = f(\mathbf{w}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{w}) + \sum_{j=1}^{l} \beta_j h_j(\mathbf{w})$$

- Solve

$$\frac{\partial \mathscr{L}}{\partial w_i} = 0 \qquad \frac{\partial \mathscr{L}}{\partial \beta_i} = 0$$

Dual Formulation

- “Primal” and “dual are complimentary solutions of the Lagrangian problem

$$d^* = \max_{\alpha,\beta:\alpha_i \geq 0} \min_{\mathbf{w}} \mathscr{L}(\mathbf{w},\alpha,\beta) \leq \min_{\mathbf{w}} \max_{\alpha,\beta:\alpha_i \geq 0} \mathscr{L}(\mathbf{w},\alpha,\beta) = p^*$$

- This is true by the “Max-min” inequality

Conditions and Consequences of Equality

- Sufficient conditions: f and g_is convex, h_js are affine, constraints are strictly feasible. Then there exists a solution; moreover p* = d* and the following Karush-Kuhn-Tucker (KKT) conditions hold:

$$\frac{\partial}{\partial w_i}\mathscr{L}(\mathbf{w}^*,\alpha^*,\beta^*) = 0,\ i = 1,\ldots,m$$

$$\frac{\partial}{\partial \beta_i}\mathscr{L}(\mathbf{w}^*,\alpha^*,\beta^*) = 0,\ i = 1,\ldots,l$$

$$\alpha_i^* g_i(\mathbf{w}^*) = 0,\ i = 1,\ldots,n$$

$$g_i(\mathbf{w}^*) \leq 0,\ i = 1,\ldots,n$$

$$\alpha_i^* \geq 0,\ i = 1,\ldots,n$$

Dual Formulation

- The primal and dual formulations are complimentary
  - Solving one will give the solution for the other

- Primal problem: objective function is a combination of the m variables
  - Minimize the objective function
  - Solution is a vector of m values that minimize function
- Dual problem: objective function is a combination of n variables
  - Maximize the objective function
  - Solution is a vector of n values called the dual variables

Application to SVM

- Recall our problem

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$
$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1, ..., N$$

- The relevant Lagrangian is

$$\mathcal{L} = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

- Conditions for d* = p* hold here
- Solve using dual form

SVM Solution

- Select αs that maximize

$$\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i x_j^T)$$

such that $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

- Predictions for new examples

$$x^T \cdot w = x^T \cdot \sum_{i=1}^{n}[\alpha_i y_i x_i] = \sum_{i=1}^{n}\alpha_i y_i (x^T \cdot x_i)$$

New Approach

Fitting a function to data

- Fitting: Maximize objective in the dual using a QP solver
- Function: max margin linear classifier

$$\hat{y} = \text{sign}(x^T \cdot w) = \text{sign}(\sum_{i=1}^{n}\alpha_i y_i (x^T \cdot x_i))$$

- Data: Train in batch mode

Dual vs. Primal Formulation

- In the primal we have M variables to solve
  - Solve for the vector **w** (length of features)
- In the dual we have N variables to solve
  - Solve for the vector **α** (length of examples)
- When to use the primal?
  - Lots of examples without many features
- When to use the dual?
  - Lots of features without many examples
  - Some other reasons (we'll talk about later)

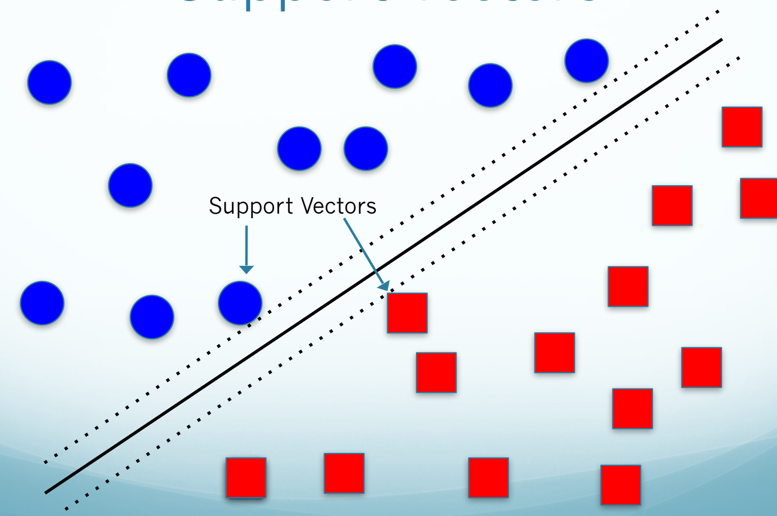
Support Vectors

- Why is it called support vector machine?
- Only some of the αs will be non-zero
  - All misclassified examples will be support vectors

$$\sum_{i=1}^{n} \alpha_i y_i (x^T \cdot x_i)$$

  - Only these vector support the hyperplane
  - These are the vectors closest to the hyperplane
- These are called "support vectors"

Support Vectors



By the Way

- We represented w in terms of the input X
- w is a *linear combination* of the inputs
  - Before: prediction was linear combination of w and x

$$w = \sum_{i=1}^{n} [\alpha_i y_i x_i]$$

- The same is true of Perceptron
  - If we store the support examples

Dual Perceptron

Non-Separable Data

- But not all data is linearly separable
  - Previous solution: add a unique feature to every example to make it separable
- What will SVMs do?
  - The regularization forces the weights to be small
  - But it must still find a max margin solution
  - Result: even with significant regularization, still leads to over-fitting

Slack Variables

$$\min_{\mathrm{w}} \; \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{such that} \quad (wx_i)y_i + \xi_i \geq 1, \quad \forall i$$

$$\xi_i \geq 0, \quad \forall i$$

- We can always satisfy the margin using ξ
  - We want these ξs to be small
  - Trade off parameter C (similar to λ before)
- ξs are called slack variables
  - The cut the margin some "slack"

Non-Separable Solution

- Similar form to the separable solution
- Extra term added to objective

Bias vs. Variance

- Smaller C means more slack (larger ξ)
  - More training examples are wrong
  - More bias (less variance) in the output
- Larger C means less slack (smaller ξ)
  - Better fit to the data
  - Less bias (more variance) in the output
- For non-separable data we can't learn a perfect separator so we don't want to try too hard
  - Finding the right balance is a tradeoff

Lingering Questions

- What would we do if we saw all of the data (batch)?
  - We'd pick the best separating hyperplane!
- Which separating hyperplane is the best?
  - The maximum margin separator
  - Use a quadratic regularizer on the weights
- What can we do for non-linear data?
  - It's not separable, use slack variables
  - Can we do better?

Next Time

Kernel Methods and
Non-Linear Support Vector Machines