Machine Learning EN.601.475/675

DR. PHILIP GRAFF

Support Vector Machines

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: given an example **x**

- Using the learned **w**, compute *p(y|**x**,**w**)*

Note: many other optimization routines available

Perceptron Algorithm

Initialize **w** and η

On each round

1. Receive example **x**
2. Predict $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x})$
3. Receive correct label $y \in \{-1, +1\}$
4. Suffer loss $l_{0/1}(y, \hat{y})$
5. Update **w** $\mathbf{w}^{i+1} = \mathbf{w}^{i} + \eta(y_i - \hat{y}_i)\mathbf{x}_i$

Fitting a function to data

Fitting: Stochastic gradient descent

Function: 0/1 loss with linear function

Data: Update using a single example at a time

Questions

Perceptron picks one separating hyperplane (of many)

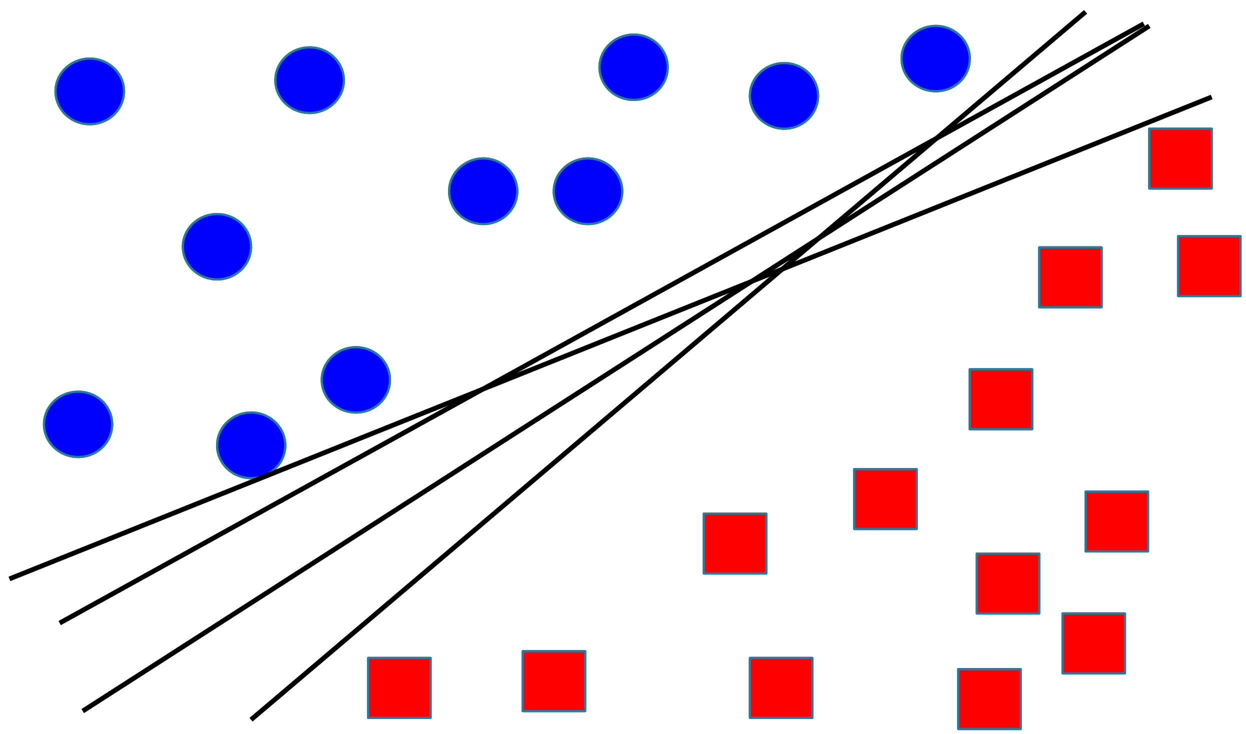
- What would we do if we saw all of the data (batch)?
- We'd pick the best separating hyperplane!

Which separating hyperplane is the best?

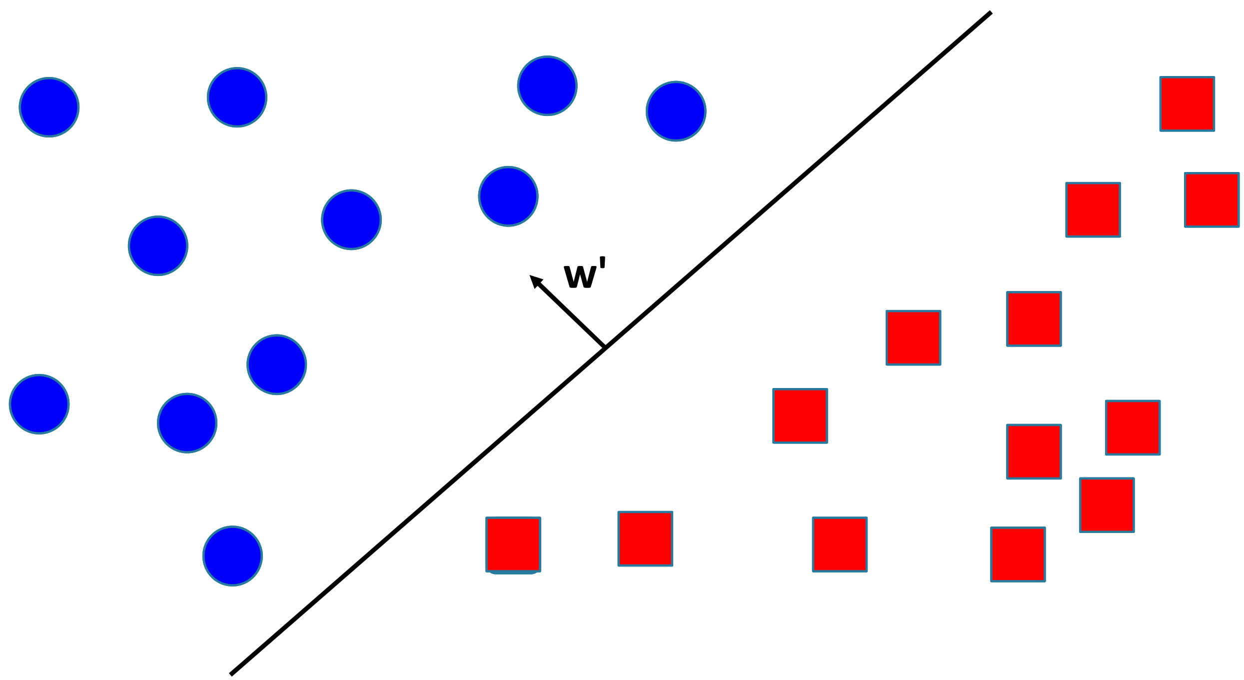
- Let's look at the geometric model

Better solutions for non-linear data?

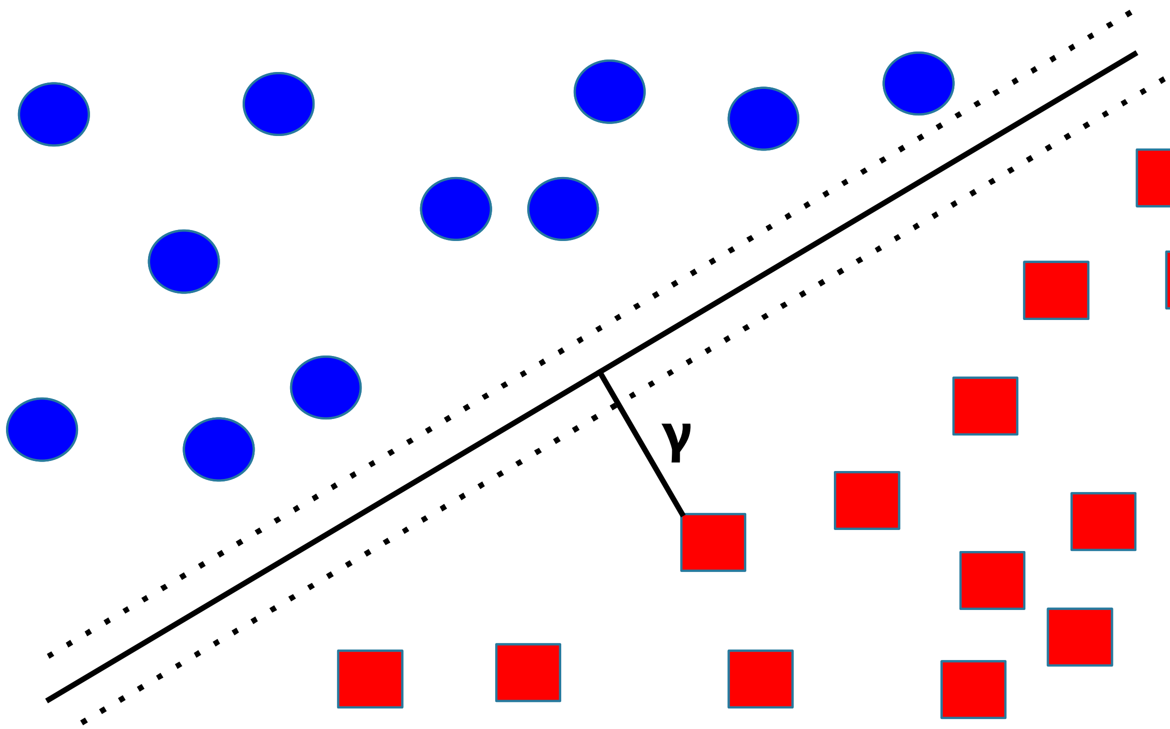
Geometric Representation



Geometric Representation



The Margin



Functional Margin

Prediction and y should agree to get large margin

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

What if we double **w**?

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

Doubles the margin, but no practical change

- We will address this in a moment

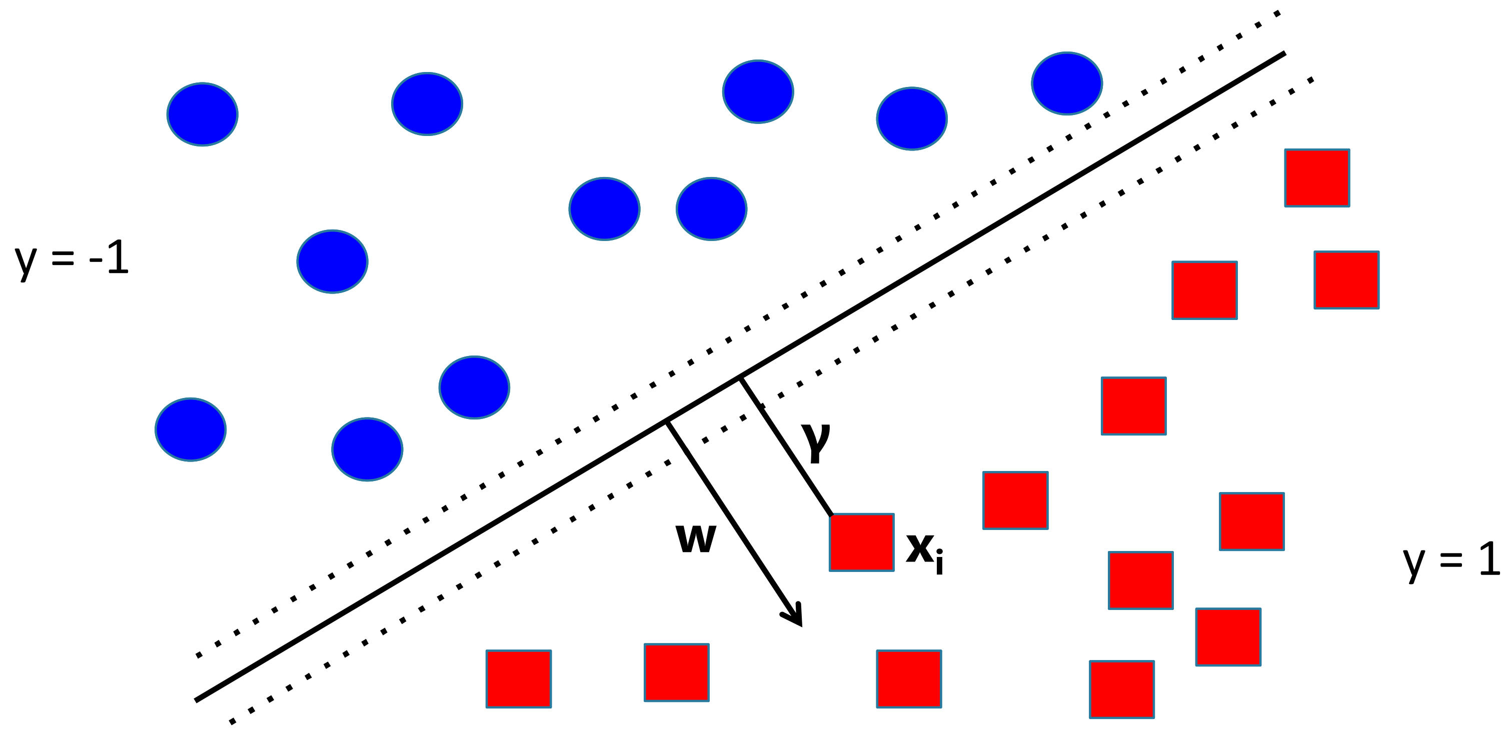
Functional Margin of Data

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?
- $\mathbf{w}/||\mathbf{w}||$ is a unit length vector pointing in the direction of **w**
- γ intersects with the decision boundary (for y=+1 side) at

$$\mathbf{x}_i - \gamma^i \times \mathbf{w}/||\mathbf{w}||$$

- Points on the boundary must satisfy $\mathbf{w}^T\mathbf{x}+b=0$

$$\mathbf{w}^{\mathrm{T}}\left(\mathbf{x}_i - \gamma^i \frac{\mathbf{w}}{||\mathbf{w}||}\right) + b = 0 \quad \Rightarrow \quad \gamma^i = y_i\left(\left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^T \mathbf{x}_i + \frac{b}{||\mathbf{w}||}\right)$$

- If $||\mathbf{w}|| = 1$ 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

Minimum Margin of the Data

- Closer points to the decision boundary will have a smaller γ
- The minimum margin is the smallest such value

$$\gamma = \min_i y_i \left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right)$$

- The best (**w**, b) solution will maximize this value

$$\text{argmax}_{\mathbf{w},b}\gamma = \text{argmax}_{\mathbf{w},b} \min_i y_i \left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right)$$

  - Subject to the constraint: $||\mathbf{w}|| = 1$

Maximum Geometric Margin

- We can also write this as:

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

- 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 margins are related by

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

- Equivalently consider

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

- No more constraint that we didn't like: $||\mathbf{w}|| = 1$

Maximum Geometric Margin

- Recall: we can arbitrarily scale **w**!
- Arbitrarily set $\hat{\gamma} = 1$
- Then we have $\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1, ..., N$
    - Because $\min||\mathbf{w}||^2$ is the same as $\max 1/||\mathbf{w}||$
- Quadratic program (QP): quadratic objective with linear constraints
- Result is optimal margin classifier

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: $\frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i) + \lambda\frac{1}{2}||\mathbf{w}||^2$

- Logistic: $\frac{1}{N}\sum_{i=1}^{N}\log(1 + \exp\{-y_i\mathbf{w}^T\mathbf{x}_i\}) + \lambda\frac{1}{2}||\mathbf{w}||^2$

$y \in \{-1, +1\}$, making this a valid loss function for logistic regression, although depicted differently than before

Loss Functions

- Both minimize

$$\frac{1}{N}\sum_{i=1}^{N} l(y_i(\mathbf{w}\cdot\mathbf{x}_i)) + \frac{\lambda}{2}||\mathbf{w}||^2$$

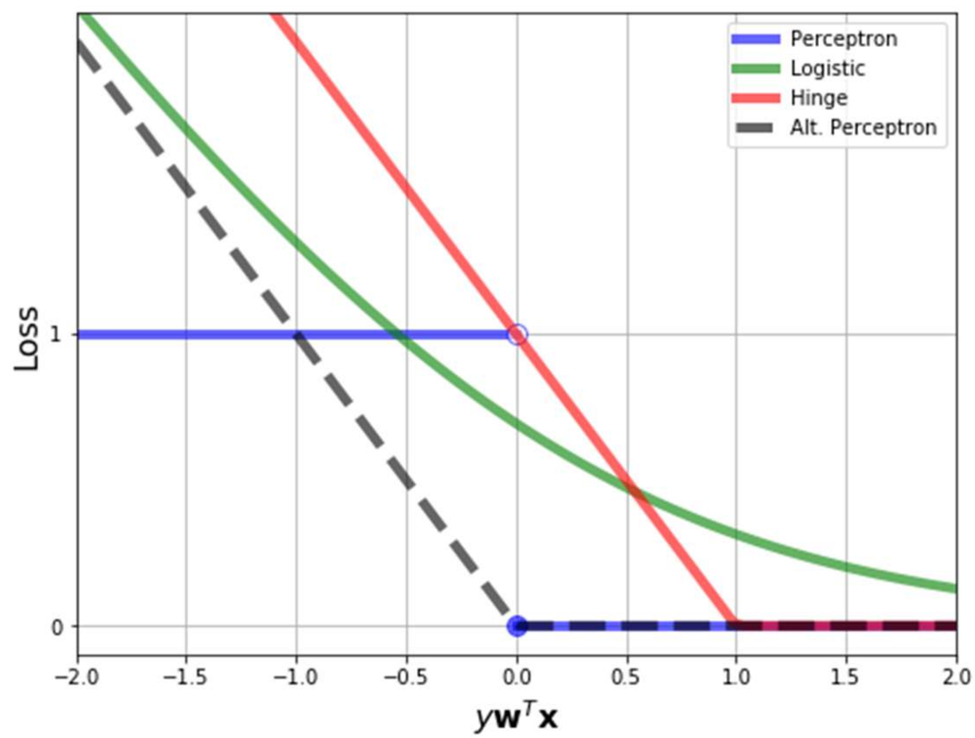
- Different loss functions
- SVM: Hinge Loss

$$l(\mathbf{w},\mathbf{x},y) = \max(0, 1 - y(\mathbf{w}\cdot\mathbf{x}))$$

- Logistic regression: Logistic loss

$$l(\mathbf{w},\mathbf{x},y) = \log(1 + \exp\{-y(\mathbf{w}\cdot\mathbf{x})\})$$

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
  - Hinge of loss function at 1, not 0
- Could we apply max-margin to online learning?
  - Yes! Perceptron with margin
  - Other methods as well

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 approach

Recall our objective function:

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

This is a constrained optimization problem…

→ **Lagrange multipliers**

Generalized Lagrangian Problem

We are trying to solve a constrained optimization problem

$$\min_{\mathbf{w}} f(\mathbf{w}) \quad \text{s.t.} \quad \begin{aligned} g_i(\mathbf{w}) &\leq 0 \quad \forall i \\ h_j(\mathbf{w}) &= 0 \quad \forall j \end{aligned}$$

The Lagrangian function for this is:

$$\mathcal{L}(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_i \alpha_i g_i(\mathbf{w}) + \sum_j \beta_j h_j(\mathbf{w})$$

The α and β values are the Lagrange multipliers

Primal Formulation

Consider $$\theta_P(\mathbf{w}) = \max_{\alpha,\beta;\alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \alpha, \beta)$$

If **w** violates the constraints, then $g_i(\mathbf{w})>0$ or $h_j(\mathbf{w})\neq 0$

$$\theta_P(\mathbf{w}) = \max_{\alpha,\beta;\alpha_i \geq 0} f(\mathbf{w}) + \sum_i \alpha_i g_i(\mathbf{w}) + \sum_j \beta_j h_j(\mathbf{w}) = \infty$$

If **w** satisfies constraints, then $g_i(\mathbf{w})\leq 0$ and $h_j(\mathbf{w})=0$

$$\theta_P(\mathbf{w}) = \max_{\alpha,\beta;\alpha_i \geq 0} f(\mathbf{w}) + \sum_i \alpha_i g_i(\mathbf{w}) + \sum_j \beta_j h_j(\mathbf{w}) = f(\mathbf{w})$$

In summary: $$\theta_P(\mathbf{w}) = \begin{cases} f(\mathbf{w}) & \text{if } \mathbf{w} \text{ satisfies constraints} \\ \infty & \text{otherwise} \end{cases}$$

Primal Formulation

$$\theta_P(\mathbf{w}) = \begin{cases} f(\mathbf{w}) & \text{if } \mathbf{w} \text{ satisfies constraints} \\ \infty & \text{otherwise} \end{cases}$$

Considering the minimization problem:

$$p^* = \min_{\mathbf{w}} \theta_P(\mathbf{w}) = \min_{\mathbf{w}} \max_{\alpha,\beta;\alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \alpha, \beta)$$

This is the same as our original problem!

Dual Formulation

Now we consider a similar problem

$$\theta_D(\mathbf{w}) = \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \alpha, \beta)$$

This is called the *dual*

The dual optimization problem is

$$d^* = \max_{\alpha,\beta;\alpha_i \geq 0} \theta_D(\mathbf{w}) = \max_{\alpha,\beta;\alpha_i \geq 0} \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \alpha, \beta)$$

The order of min and max has been changed

Dual Formualtion

It can be shown that (max min) ≤ (min max)

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

If *f* and *g* are convex and *h* is affine (i.e. linear)…

$$\exists\, \mathbf{w}^*, \alpha*, \beta* \quad \text{s.t.} \quad p^* = d^* = \mathcal{L}(\mathbf{w}^*, \alpha*, \beta*)$$

The Karush-Kuhn-Tucker (KKT) conditions are satisfied

$$\frac{\partial}{\partial w_i}\mathcal{L}(\mathbf{w}^*, \alpha*, \beta*) = 0 \quad \forall i \qquad\qquad \alpha_i^* g_i(\mathbf{w}^*) = 0 \quad \forall i$$

$$g_i(\mathbf{w}^*) \leq 0 \quad \forall i$$

$$\frac{\partial}{\partial \beta_i}\mathcal{L}(\mathbf{w}^*, \alpha*, \beta*) = 0 \quad \forall i \qquad\qquad \alpha_i^* \geq 0 \quad \forall i$$

Lagrange Multipliers for SVMs

Let's go back to our original problem:

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

Our constraints are: $g_i(\mathbf{w}) = 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0$

From KKT conditions, $\alpha_i > 0$ only where $g_i = 0$

- These are points **on the margin**

Solving the Dual

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

Taking partial derivatives w.r.t. **w** and *b*, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad 0 = \sum_{i=1}^{N} \alpha_i y_i$$

Solving the Dual

We can plug this solution for **w** back into the Lagrangian

$$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j) - b \sum_{i=1}^{N} \alpha_i y_i$$

The last term =0, so we are left with

$$\tilde{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Solve: $\max_{\alpha} \tilde{L}(\alpha) = \max_{\alpha} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j) \right]$ **s.t.** $\alpha_i \geq 0 \quad \forall i$, $\sum_{i=1}^{N} \alpha_i y_i = 0$

Primal vs. Dual

Primal and dual are complementary problems, solving one will solve the other

SVM formulation meets conditions that ensure equality in the solution

Primal problem: objective function is a combination of the *M* variables

- Minimize the objective function
- Solution, **w**, is a vector of *M* values that minimize function

Dual problem: objective function is a combination of *N* variables

- Maximize the objective function
- Solution, α, is a vector of *N* values called the dual variables
- This will be sparse (many α_i=0)

Predictions

Predictions for new inputs, **x**, are simple:

$$\begin{aligned}\hat{y} &= \text{sign}(\mathbf{w}^T\mathbf{x}) \\ &= \text{sign}\left(\left[\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i^T\right]\mathbf{x}\right) \\ &= \text{sign}\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i^T \mathbf{x}\right)\end{aligned}$$

Note: We only need those $\mathbf{x}_i$ with non-zero α_i

Fitting a function to data

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

$$\hat{y} = \mathrm{sign}(\mathbf{w}^T\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}\right)$$

- Data: Train in batch mode

Primal vs. Dual Formulation

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

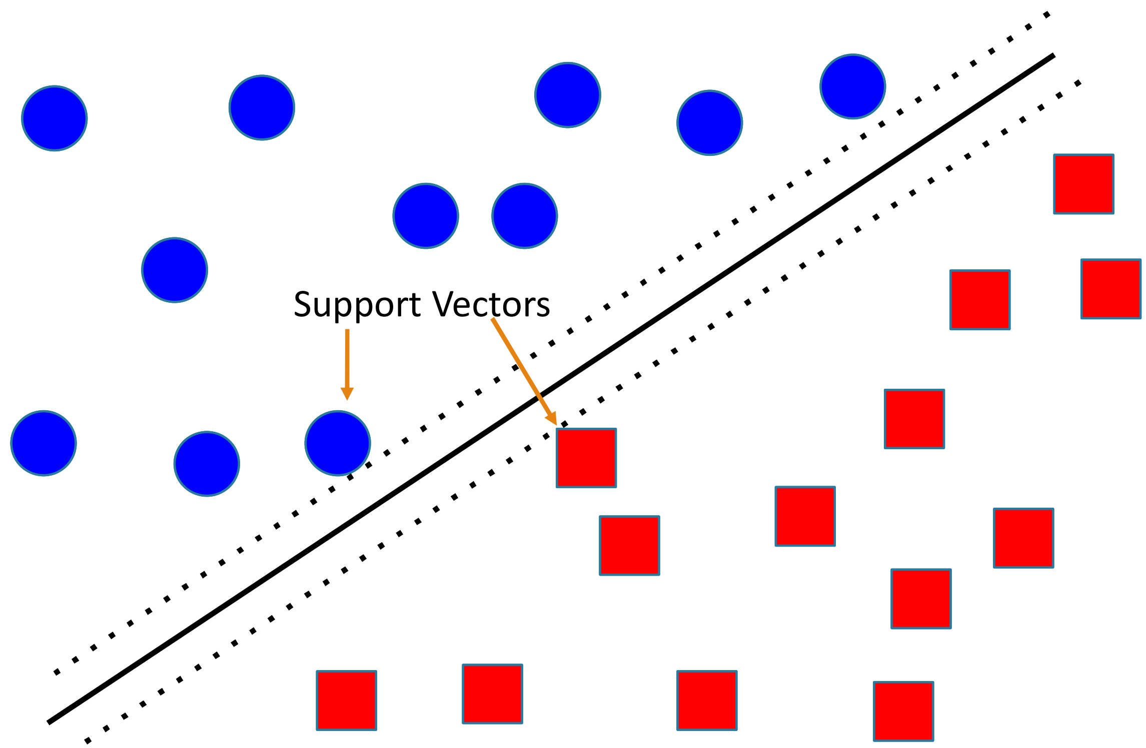
Why is it called support vector machine?

Only some of the αs will be non-zero

- All examples on the margin will be support vectors
- Only these vectors support 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

$$\mathbf{w} = \sum_{i=1}^{n} [\alpha_i y_i \mathbf{x}_i]$$

- Prediction is a linear combination of **w** and **x**

The same is true of Perceptron

- If we store the support examples

Dual Perceptron

Let's look back at the perceptron model

Base case: $\mathbf{w}^0 = \mathbf{0}$

Inductive case: $\mathbf{w}^{i+1} = \mathbf{w}^i + y_i \mathbf{x}_i$

Substitute in for $\mathbf{w}^i$ based on step i-1 until we get to i=0

$$\mathbf{w}^{i+1} = \mathbf{0} + \sum_{j=1}^{i} y_j \mathbf{x}_j \quad \Rightarrow \quad \mathbf{w}^{i+1} = \mathbf{0} + \sum_{j \in \mathcal{M}} y_j \mathbf{x}_j$$

$$\hat{y} = \operatorname{sign}(\mathbf{w}^T \mathbf{x}) = \operatorname{sign}\left(\left[\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \right]^T \mathbf{x} \right) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} \right) \qquad \alpha_i = \begin{cases} 1 & \text{prediction } i \text{ incorrect} \\ 0 & \text{prediction } i \text{ correct} \end{cases}$$

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

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \qquad \text{s.t.} \quad (\mathbf{w}^T\mathbf{x}_i)y_i + \xi_i \geq 1, \forall i$$
$$\xi_i \geq 0, \forall i$$

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

- They cut the margin some "slack"

Non-Separable Solution

Similar form to the separable solution

Extra term added to objective and extra constraint term

$$L(\mathbf{w}, b, \xi, \alpha, r) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i[y_i\mathbf{w}^T\mathbf{x} - 1 + \xi_i] - \sum_{i=1}^{N} r_i\xi_i$$

Perform similar solution according to the dual formulation

$$\tilde{L}(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T\mathbf{x}_j)$$

s.t.

$$0 \le \alpha_i \le C \quad \forall i$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0$$

Bias vs. Variance

Smaller C means more slack (larger ξ, smaller α)

- More training examples can be wrong
- More bias (less variance) in the output

Larger C means less slack (smaller ξ, larger α)

- Better fit to the training 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