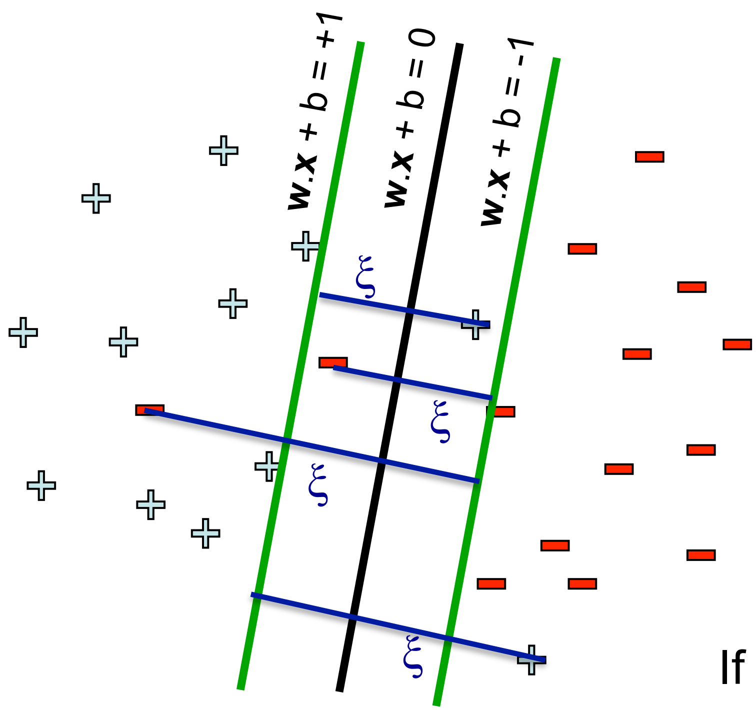
Support vector machines
Lecture 4

David Sontag
New York University

Slides adapted from Luke Zettlemoyer, Vibhav Gogate, and Carlos Guestrin

Allowing for slack: "Soft margin" SVM



$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\,\xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

"slack variables"

What is the (optimal) value of ξ_j as a function of **w** and b?

If $(w \cdot x_j + b)\, y_j \geq 1$, then $\xi_j = 0$

If $(w \cdot x_j + b)\, y_j < 1$, then $\xi_j = 1 - (w \cdot x_j + b)\, y_j$

$$\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$$

Sometimes written as

$$\left(1 - (w \cdot x_j + b)\, y_j\right)_+$$

Equivalent hinge loss formulation

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\,\xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad ,\forall j \quad \xi_j \geq 0$$

Substituting $\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$ into the objective, we get:

$$\boxed{\min ||w||^2 + C \sum_j \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)}$$

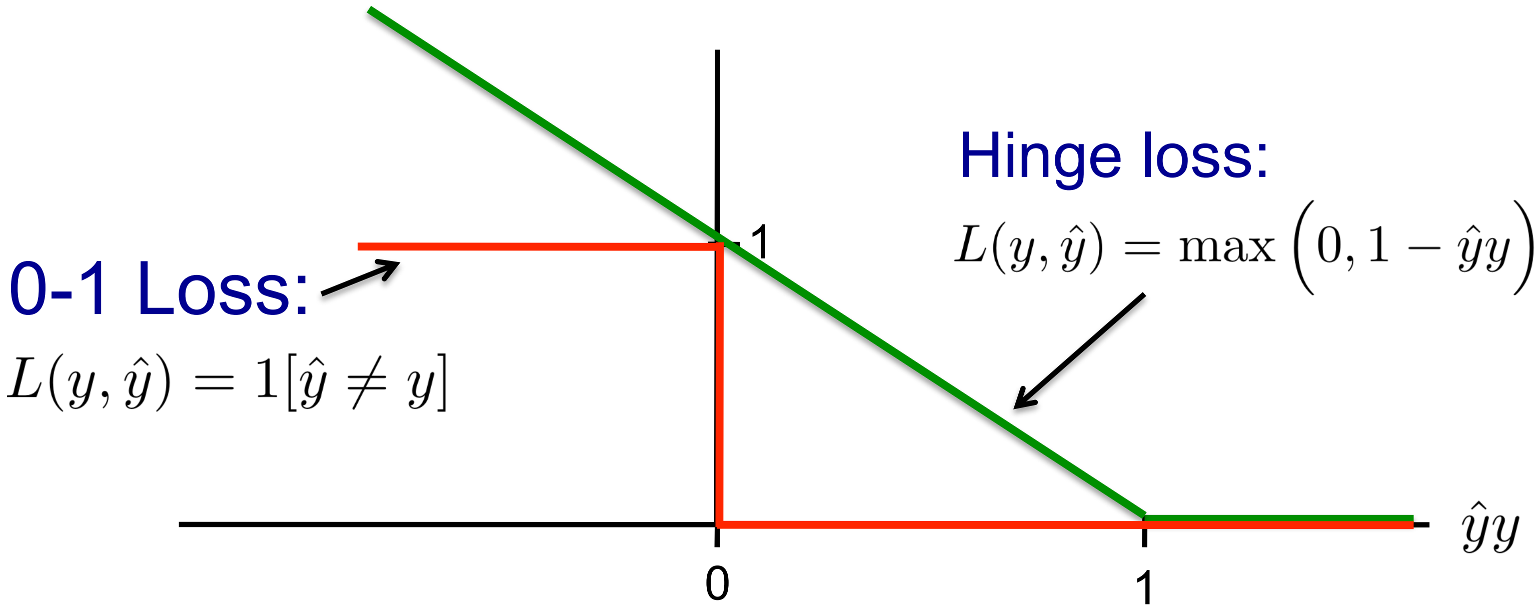
The **hinge loss** is defined as $L(y, \hat{y}) = \max\Big(0, 1 - \hat{y}y\Big)$

$$\min_{w,b} ||w||_2^2 + C \underline{\sum_j L(y_j, \mathbf{w} \cdot x_j + b)}$$

This is called **regularization**; used to prevent overfitting!

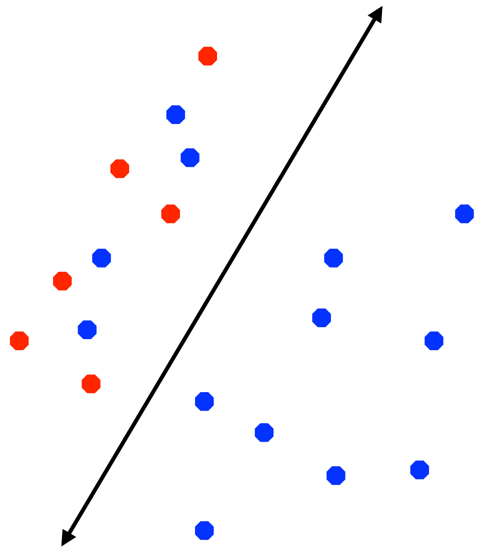
This part is empirical risk minimization, using the hinge loss

Hinge loss vs. 0/1 loss



Hinge loss upper bounds 0/1 loss!

How to deal with imbalanced data?

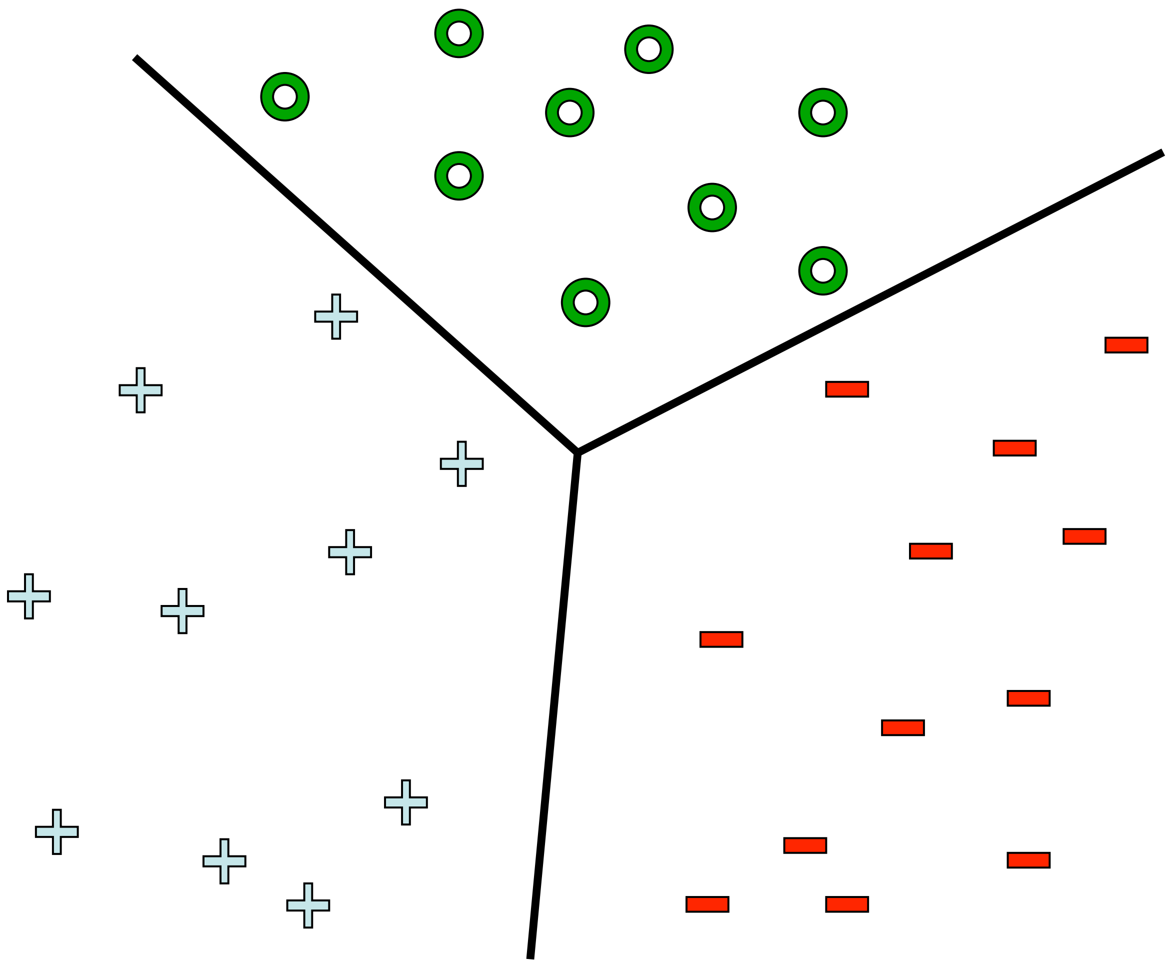


- In many practical applications we may have **imbalanced** data sets
- We may want errors to be equally distributed between the positive and negative classes
- A slight modification to the SVM objective does the trick!

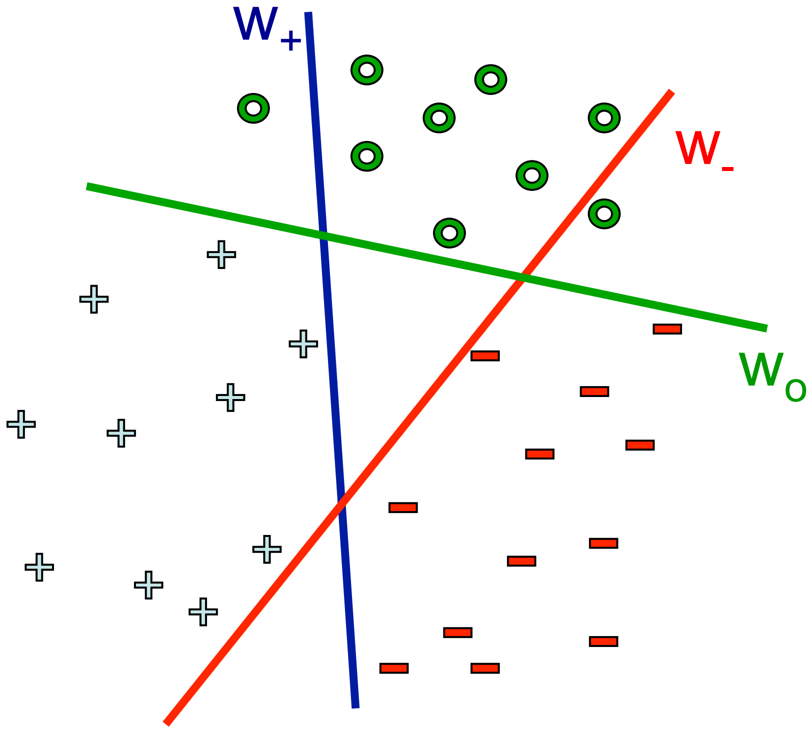
$$\min_{w,b} \frac{1}{2}||w||_2^2 + \frac{C}{N_+} \sum_{j\,:\,y_j=+1} \xi_j + \frac{C}{N_-} \sum_{j\,:\,y_j=-1} \xi_j$$

Class-specific weighting of the slack variables

How do we do multi-class classification?



One versus all classification



Learn 3 classifiers:

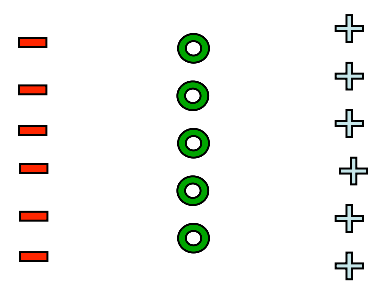
- \- vs {o,+}, weights w_-
- \+ vs {o,-}, weights w_+
- o vs {+,-}, weights w_o

Predict label using:

$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

Any problems?

Could we learn this dataset? →

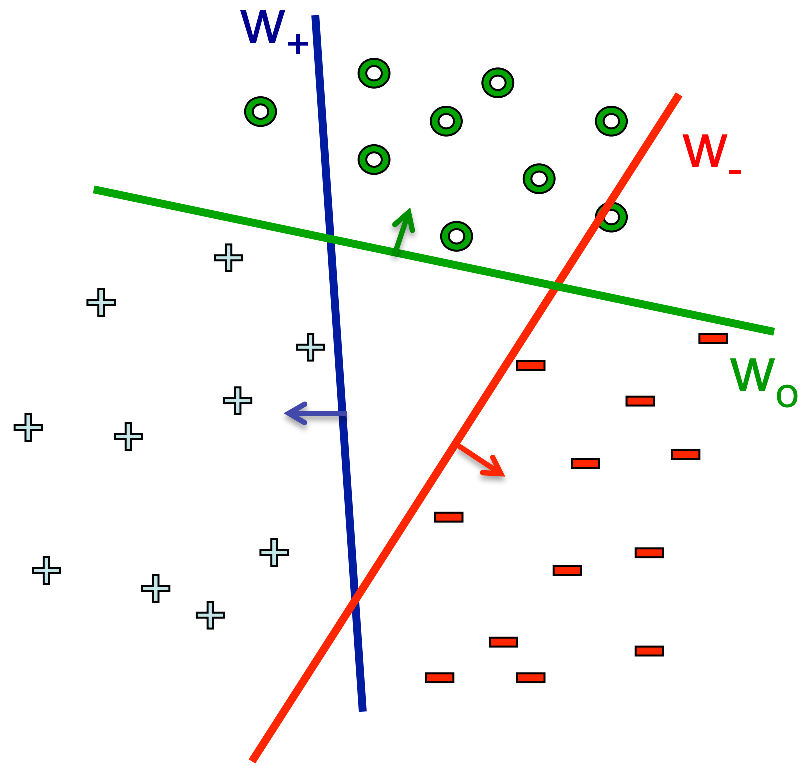


Multi-class SVM

Simultaneously learn 3 sets of weights:

•How do we guarantee the correct labels?

•Need new constraints!



The "score" of the correct class must be better than the "score" of wrong classes:

$$w^{(y_j)} \cdot x_j + b^{(y_j)} > w^{(y)} \cdot x_j + b^{(y)} \qquad \forall j,\ y \neq y_j$$

Multi-class SVM

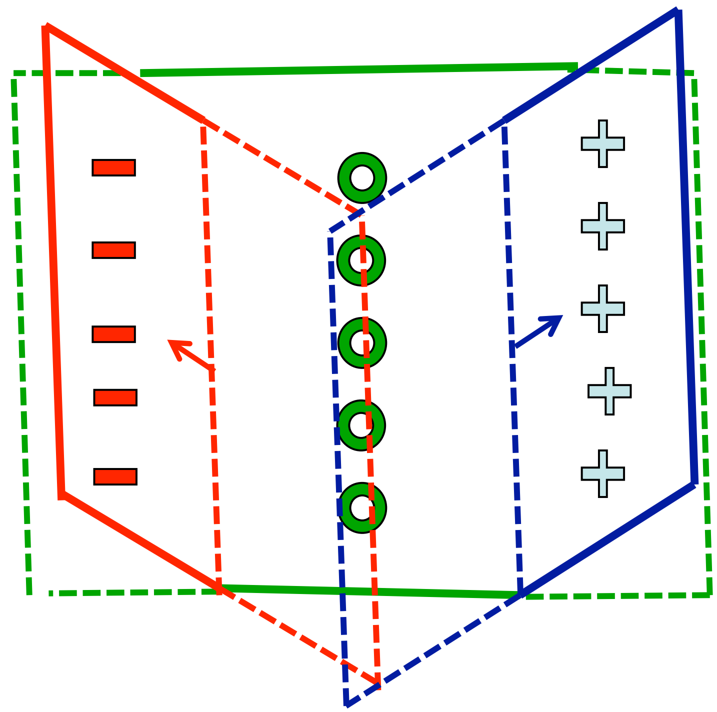
As for the SVM, we introduce slack variables and maximize margin:

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \xi_j$$
$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1 - \xi_j, \;\; \forall y' \neq y_j, \;\; \forall j$$
$$\xi_j \geq 0, \;\; \forall j$$

To predict, we use:

$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

Now can we learn it? →

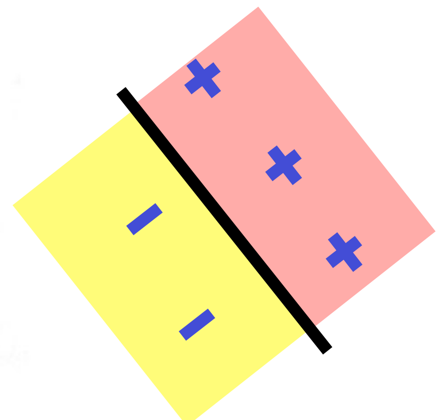


Software

- SVM*light*: one of the most widely used SVM packages. Fast optimization, can handle very large datasets, C++ code.
- LIBSVM (used within Python's scikit-learn)
- Both of these handle multi-class, weighted SVM for imbalanced data, etc.
- There are several new approaches to solving the SVM objective that can be much faster:
  - Stochastic subgradient method (up next!)
  - Distributed computation
- See http://mloss.org, "machine learning open source software"

PEGASOS [ICML 2007]

Primal Efficient sub-GrAdient SOlver for SVM



Shai Shalev-Shwartz

The Hebrew University
Jerusalem, Israel

Yoram Singer

Google

Nati Srebro

Toyota Technological Institute Chicago

Support Vector Machines

QP form:

$$\operatorname*{argmin}_{\mathbf{w},\xi_i \geq 0} \; \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t.} \quad \forall i, \; y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 - \xi_i$$

More “natural” form:

$\operatorname*{argmin}_{\mathbf{w}} f(\mathbf{w})$ where:

$$f(\mathbf{w}) \stackrel{\text{def}}{=} \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{Regularization term}} + \underbrace{\frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}}_{\text{Empirical loss}}$$

PEGASOS

A_t = S

Subgradient method

|A_t| = 1

Stochastic gradient

t S =

ation

Number of iterations T

INITIALIZE: Choose $\mathbf{w}_1$ s.t. $\|\mathbf{w}_1\| \leq 1/\sqrt{\lambda}$

FOR $t = 1, 2, \ldots, T$

Subgradient

- Choose $A_t \subseteq S$
- $A_t^+ = \{(\mathbf{x}, y) \in A_t : y\langle \mathbf{w}_t, \mathbf{x} \rangle < 1\}$
- $\nabla_t = \lambda \mathbf{w}_t - \frac{1}{|A_t|} \sum_{(\mathbf{x},y) \in A_t^+} y\,\mathbf{x}$
- $\eta_t = \frac{1}{t\,\lambda}$
- $\mathbf{w}'_t = \mathbf{w}_t - \eta_t \nabla_t$

Projection $\Leftarrow$ $\mathbf{w}_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}'_t\|}\right\} \mathbf{w}'_t$

OUTPUT: $\mathbf{w}_{T+1}$

Run-Time of Pegasos

- Choosing $|A_t|=1$

  ➔ Run-time required for Pegasos to find ε accurate solution w.p. $\geq$ 1-δ

$$\tilde{O}\left(\frac{n}{\delta\,\lambda\,\epsilon}\right)$$

- Run-time does not depend on #examples
- Depends on “difficulty” of problem (λ and ε)

Experiments

- **3 datasets** (provided by Joachims)
  - Reuters CCAT (800K examples, 47k features)
  - Physics ArXiv (62k examples, 100k features)
  - Covertype (581k examples, 54 features)

Training Time (in seconds):

	Pegasos	SVM-Perf	SVM-Light
Reuters	**2**	77	20,075
Covertype	**6**	85	25,514
Astro-Physics	**2**	5	80

What’s Next!

- Learn one of the most interesting and exciting recent advancements in machine learning
  - The “kernel trick”
  - High dimensional feature spaces at no extra cost!
- But first, a detour
  - Constrained optimization!