Announcements

- Project proposal due tonight!

Stochastic Gradient Descent

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

$$\frac{1}{n}\sum_{i=1}^n \ell_i(w)$$

Each $\ell_i(w)$ is convex.

Stochastic Gradient Descent

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\frac{1}{n}\sum_{i=1}^n \ell_i(w)$$

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n}\sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

Stochastic Gradient Descent

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

$$\frac{1}{n}\sum_{i=1}^n \ell_i(w)$$

Each $\ell_i(w)$ is convex.

Gradient Descent:

$$w_{t+1} = w_t - \eta \boxed{\nabla_w \left(\frac{1}{n}\sum_{i=1}^n \ell_i(w) \right)}\Bigg|_{w=w_t}$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w)\Big|_{w=w_t}$$

I_t drawn uniform at random from $\{1, \ldots, n\}$

$$\mathbb{E}[\nabla \ell_{I_t}(w)] =$$

$$\mathbb{E}[f(\bar{w}_T)] - f(w_*) \leq \frac{C}{\sqrt{T}}$$

Stochastic Gradient Descent: A Learning perspective

Machine Learning – CSE546

Kevin Jamieson

University of Washington

October 24, 2017

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution p(**x**) on features:
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:

$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

$$\ell_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} \ell(\mathbf{w}, \mathbf{x}^j)$$

- However, we should really minimize expected loss on all data:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w}, \mathbf{x})d\mathbf{x}$$

- So, we are approximating the integral by the average on the training data

Gradient descent in Terms of Expectations

- "True" objective function:

$$E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right]$$

- Taking the gradient:

$$\nabla \mathbb{E}_x[\ell(w,x)] = \mathbb{E}_x[\nabla \ell(w,x)]$$

- "True" gradient descent rule:

$$w_{t+1} = w_t - \eta \mathbb{E}_x[\nabla \ell(w,x)]$$

- How do we estimate expected gradient?

$$w_{t+1} = w_t - \eta \nabla \ell_i(w, x_i)$$

SGD: Stochastic Gradient Descent

- "True" gradient: $$\nabla \ell(\mathbf{w}) = E_{\mathbf{x}}\left[\nabla \ell(\mathbf{w}, \mathbf{x})\right]$$

- One iid sample estimate:

- How many iid samples do we have?

n iid samples (not infinite)

So we cannot get infinite stream of iid samples

See [Hardt, Recht, Singer 2016] for resolution based on stability

$$\ell_i(w, x_i) = (y_i - w^T x_i)^2$$

Perceptron

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 24, 2017

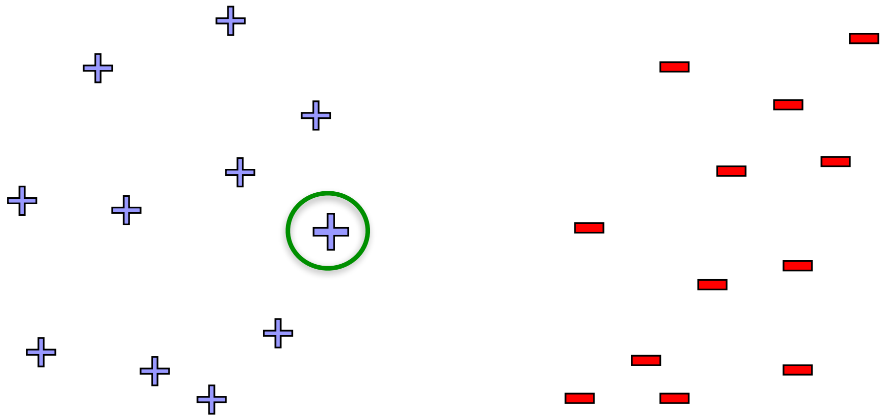
Online learning

- Click prediction for ads is a streaming data task:
  - User enters query, and ad must be selected
    - Observe $\mathbf{x}^j$, and must predict y^j

  - User either clicks or doesn't click on ad
    - Label y^j is revealed afterwards
      - Google gets a reward if user clicks on ad

  - Update model for next time

Online classification



New point arrives at time k

The Perceptron Algorithm [Rosenblatt '58, '62]

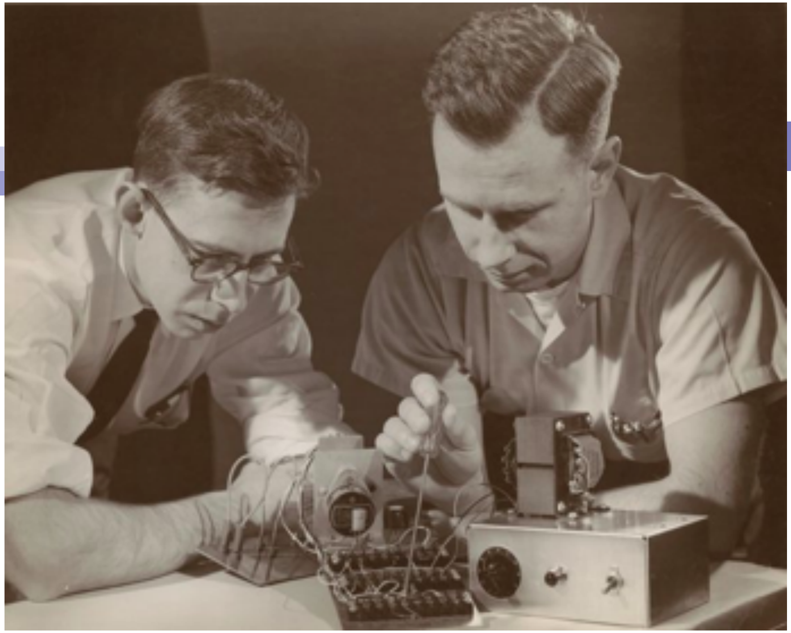
- Classification setting: y in {-1,+1}
- Linear model
  - Prediction: $y_k \approx \text{sign}(x_n^T w + b)$
- Training:
  - Initialize weight vector: $w_0 = 0, \; b_0 = 0$
  - At each time step: k
    - Observe features: x_n
    - Make prediction: $\hat{y}_n = \text{sign}(x_n^T w_n + b_n)$
    - Observe true class: y_n
    - Update model: If $y_n = \hat{y}_n$ then $w_{n+1} = w_n$, $b_{n+1} = b_n$
      - If prediction is not equal to truth

$$\begin{pmatrix} w_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} w_n \\ b_n \end{pmatrix} + y_n \begin{pmatrix} x_n \\ 1 \end{pmatrix}$$

The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: y in {-1,+1}
- Linear model
  - Prediction: $\text{sign}(w^T x_i + b)$
- Training:
  - Initialize weight vector: $w_0 = 0, b_0 = 0$
  - At each time step:
    - Observe features: x_k
    - Make prediction: $\text{sign}(x_k^T w_k + b_k)$
    - Observe true class: y_k
    - Update model:
      - If prediction is not equal to truth

$$\begin{bmatrix} w_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} w_k \\ b_k \end{bmatrix} + y_k \begin{bmatrix} x_k \\ 1 \end{bmatrix}$$

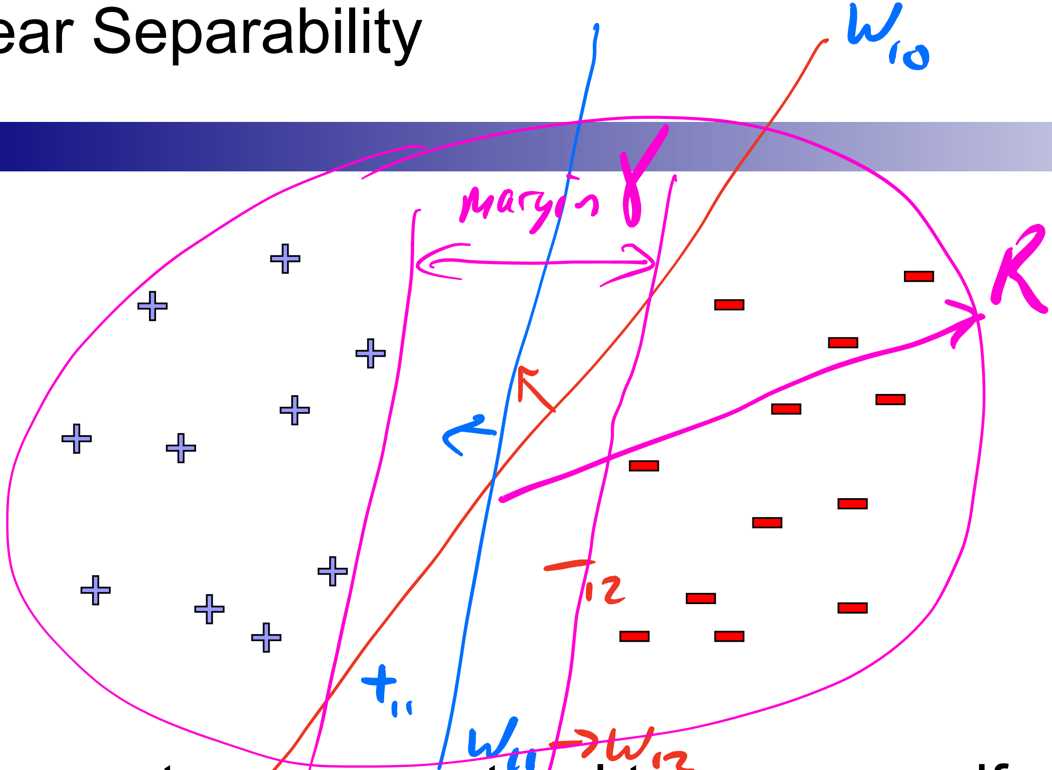


Rosenblatt 1957

"the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

The New York Times, 1958

Linear Separability



- Perceptron guaranteed to converge if
  - Data linearly separable:

Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: $(x_i, y_i) \quad i=1,2,\ldots$
  - Each feature vector has bounded norm: $\|x_i\|_2 \le R$
  - If dataset is linearly separable: $\exists w_*, b_* \quad \text{sign}(w_*^T x_i + b_*) = y_i \ \forall i$

- Then the number of mistakes made by the online perceptron on any such sequence is bounded by

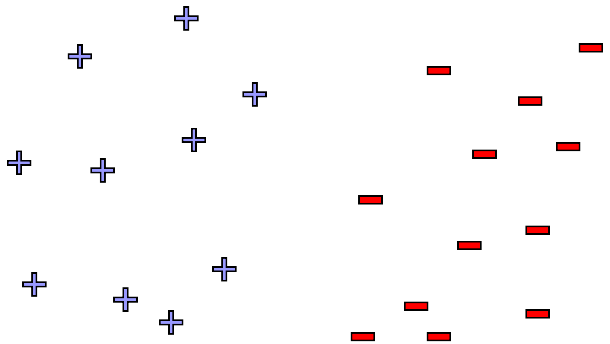
$$\frac{R^2}{\gamma^2}$$

$\gamma =$ "margin"

Gap between classes

Beyond Linearly Separable Case

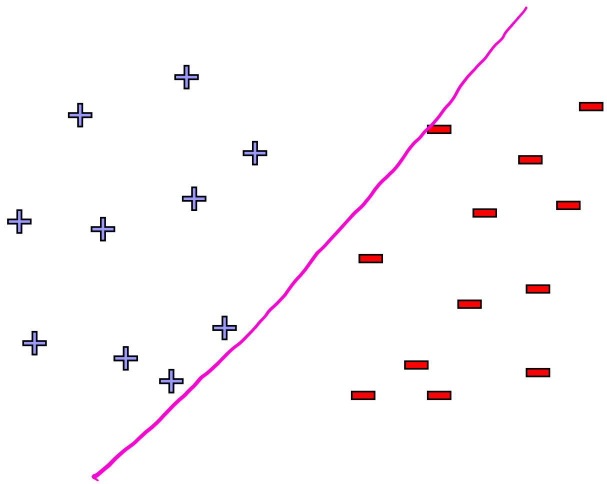
- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data



Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data

- Perceptron is useless in practice!
  - Real world not linearly separable
  - If data not separable, cycles forever and hard to detect
  - Even if separable may not give good generalization accuracy (small margin)



What is the Perceptron Doing???

- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood

- When we discussed the Perceptron:
  - Started from description of an algorithm

- What is the Perceptron optimizing????

(x_i, y_i) arrives w/ loss $\max\{0, -(x_i^T w + b) y_i\}$

$w_{k+1} = w_k - \nabla \ell_k(w_k, b)$ where $\ell_k = \max\{0, -y_i(x_i^T w + b)\}$

Support Vector Machines

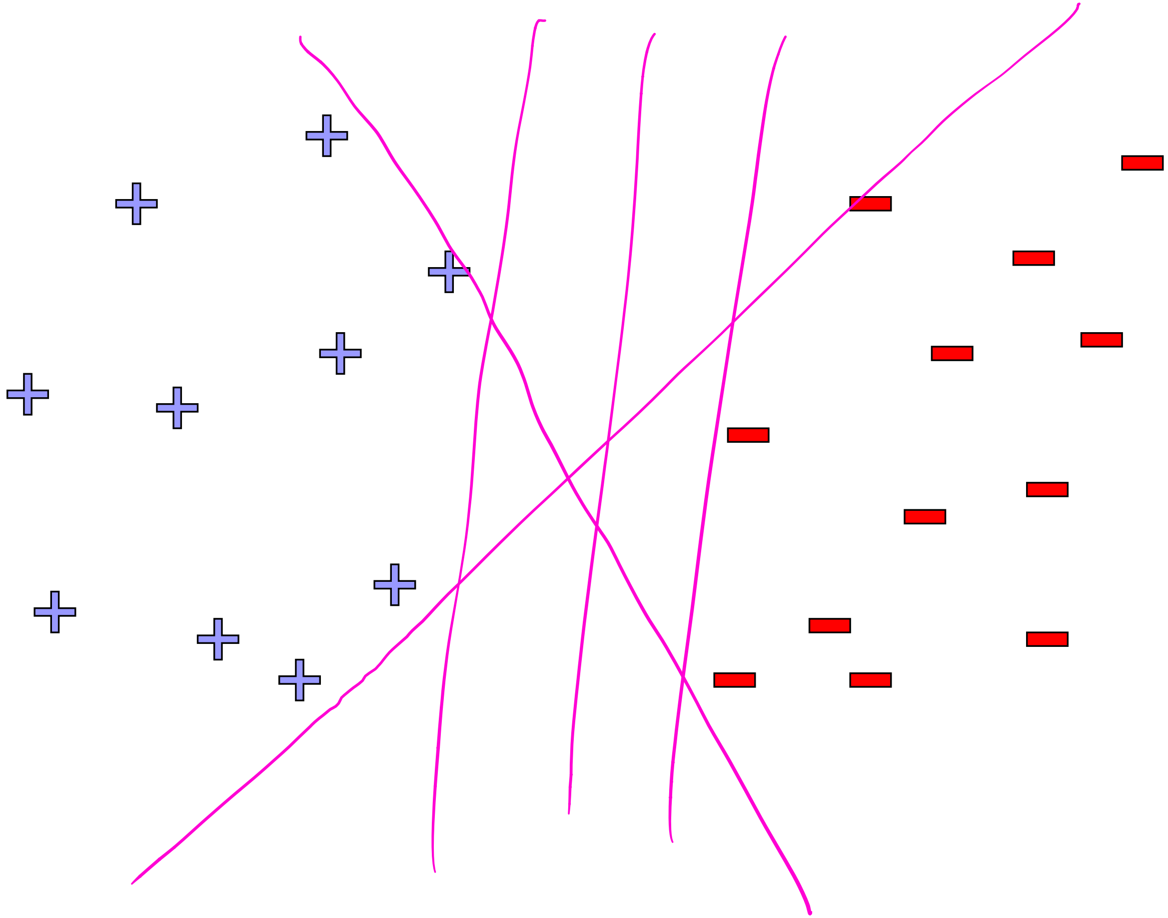
Machine Learning – CSE446

Kevin Jamieson

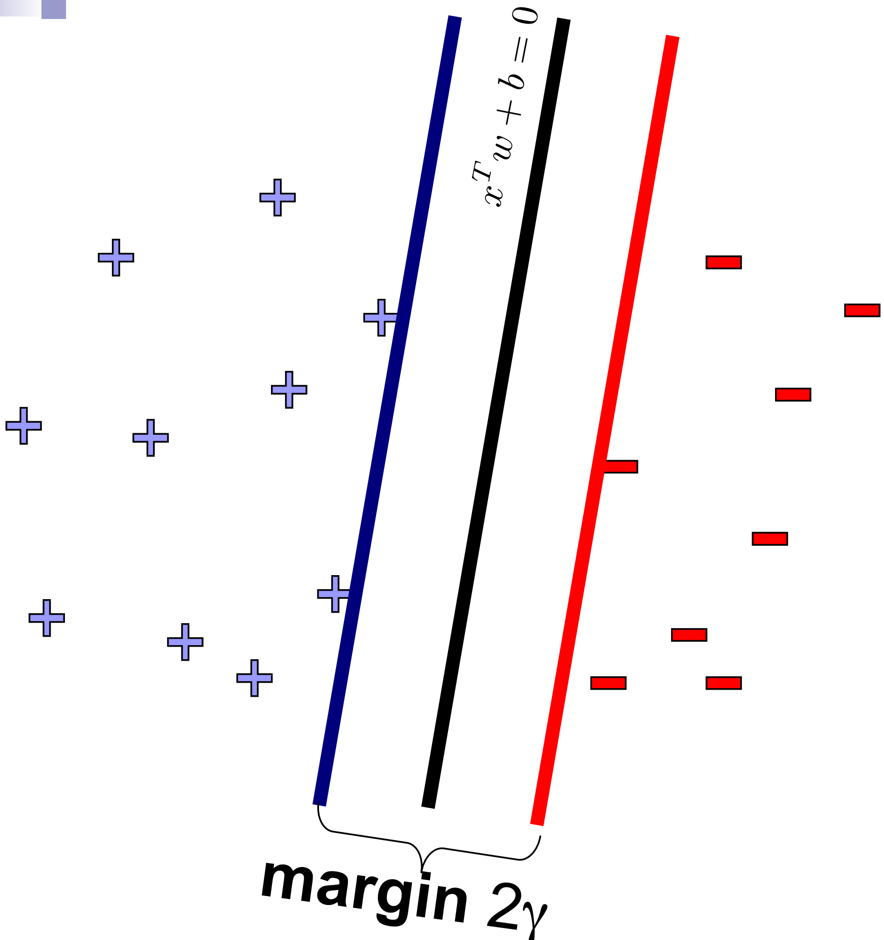
University of Washington

October 24, 2017

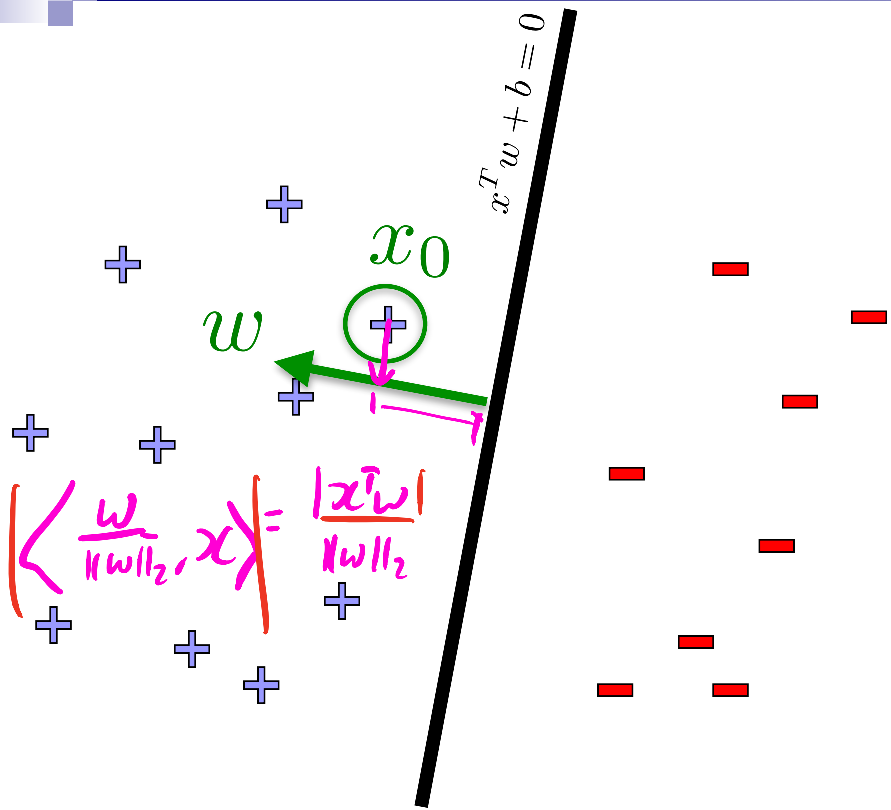
Linear classifiers – Which line is better?



Pick the one with the largest margin!



Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

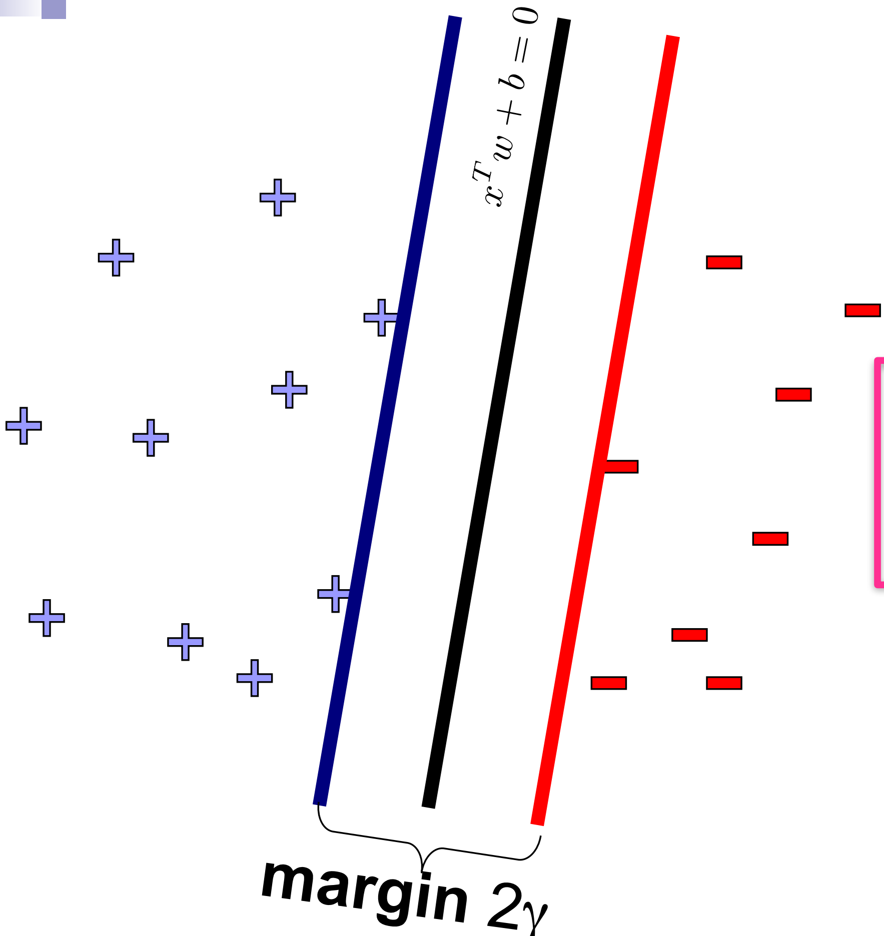
$$\left|\frac{1}{||w||_2}(x_0^T w + b)\right|$$

$$= \frac{1}{\|w\|_2}(x_0^T w + b)\, y_0$$

If w classifies x_0 as y_0

$\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = x^T y$

Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

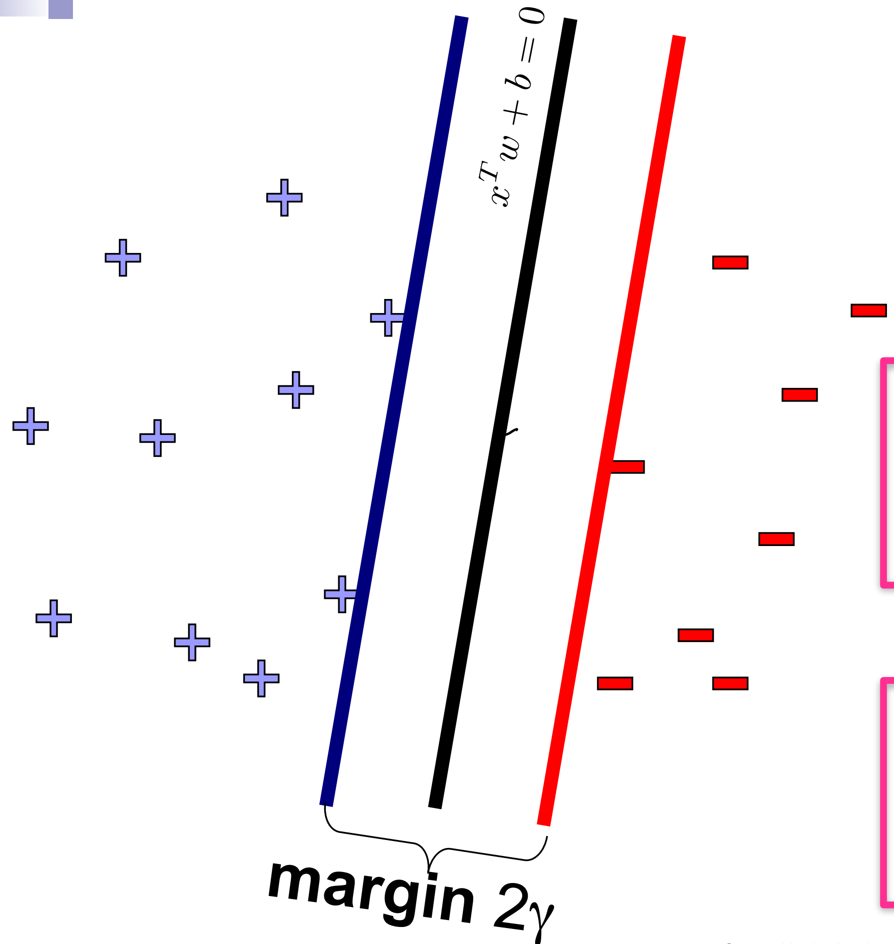
$$\frac{1}{||w||_2}(x_0^T w + b)$$

Optimal Hyperplane

$$\max_{w,b} \gamma$$

$$\text{subject to } \frac{1}{||w||_2^2} y_i (x_i^T w + b) \geq \gamma \quad \forall i$$

Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

$$\frac{1}{||w||_2}(x_0^T w + b)$$

Optimal Hyperplane

$$\max_{w,b} \gamma$$

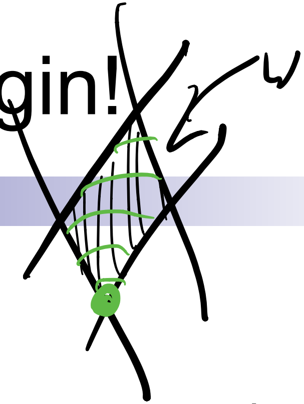
$$\text{subject to } \frac{1}{||w||_2} y_i(x_i^T w + b) \geq \gamma \quad \forall i$$

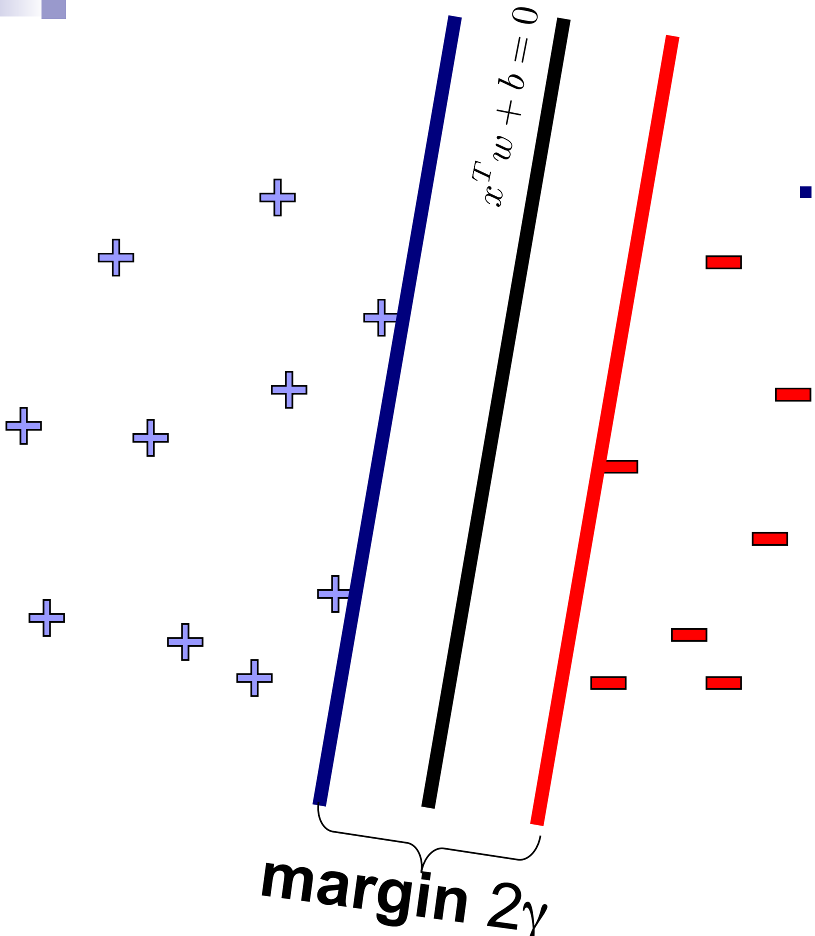
Optimal Hyperplane (reparameterized)

$$\min_{w,b} ||w||_2^2$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Pick the one with the largest margin!





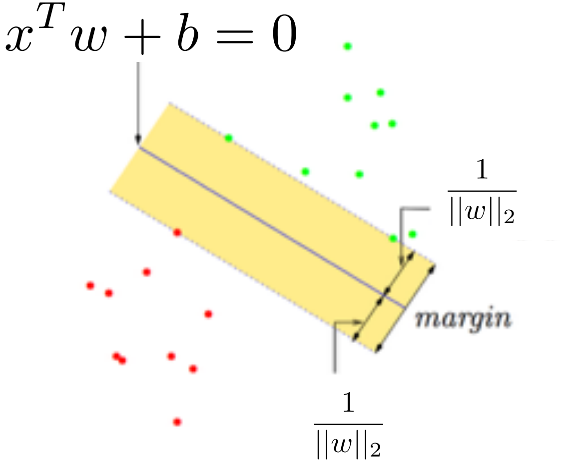
- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
    - Well-studied solution algorithms
  - Stochastic gradient descent
  - Coordinate descent (in the dual)

Optimal Hyperplane (reparameterized)

$$\min_{w,b} ||w||_w^2$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 \quad \forall i$$

What if the data is still not linearly separable?

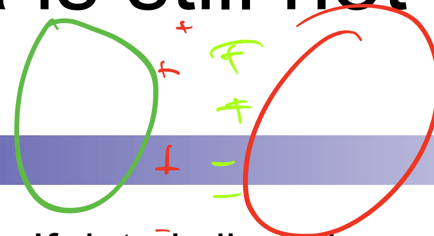


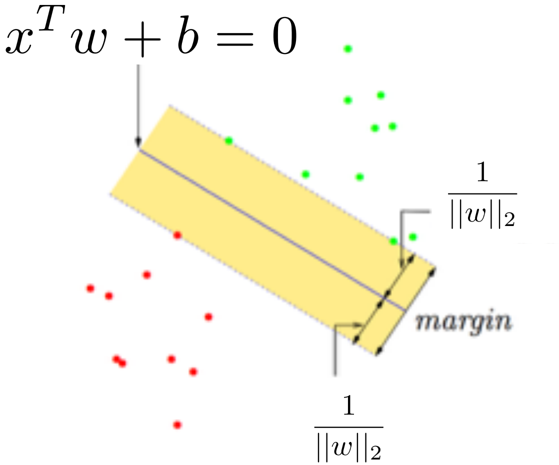
- If data is linearly separable

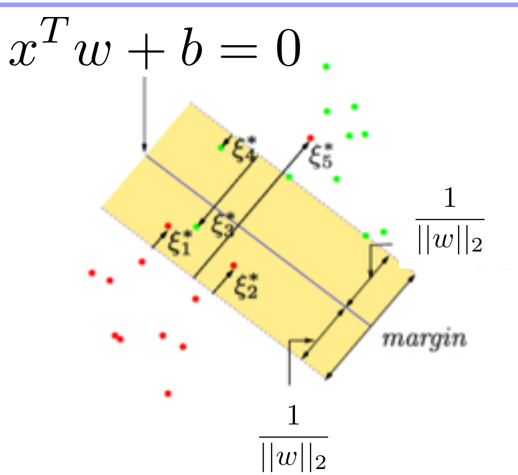
$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

What if the data is still not linearly separable?







- If data is linearly separable

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

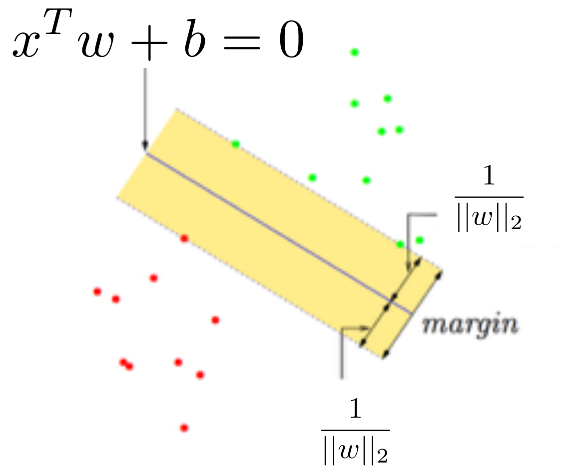
- If data is not linearly separable, some points don't satisfy margin constraint:

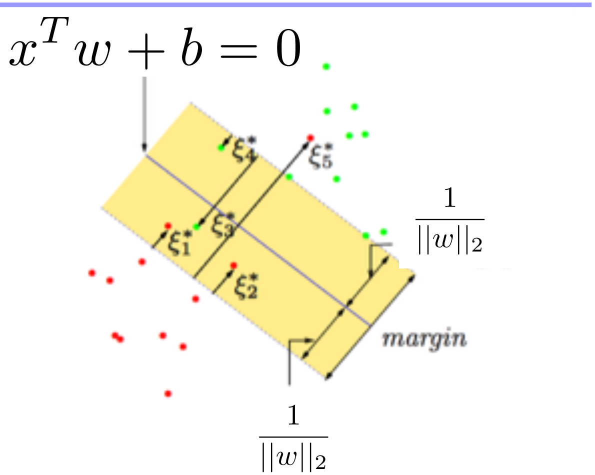
$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

What if the data is still not linearly separable?





- If data is linearly separable

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

- If data is not linearly separable, some points don't satisfy margin constraint:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

- What are "support vectors?"

SVM as penalization method

- Original quadratic program with linear constraints:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

SVM as penalization method

- Original quadratic program with linear constraints:

for some $c > 0$

$$\min_{w,b} ||w||_2^2 + c\nu$$

$$y_i(x_i^T w + b) \geq \underline{1 - \xi_i} \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^n \xi_j \leq \nu$$

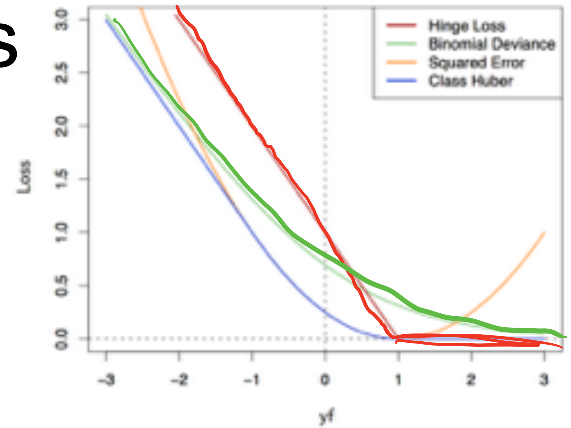
- Using same constrained convex optimization trick as for lasso:

(for any c)

For any $\nu \geq 0$ there exists a $\lambda \geq 0$ such that the solution the following solution is equivalent:

$$\boxed{\sum_{i=1}^n \max\{0, \underline{1 - y_i(b + x_i^T w)}\} + \lambda \underline{||w||_2^2}}$$

Machine Learning Problems



- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

  Each $\ell_i(w)$ is convex.

Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

How do we solve for *w*? The last two lectures!

SVMs vs logistic regression

- We often want probabilities/confidences, logistic wins here?

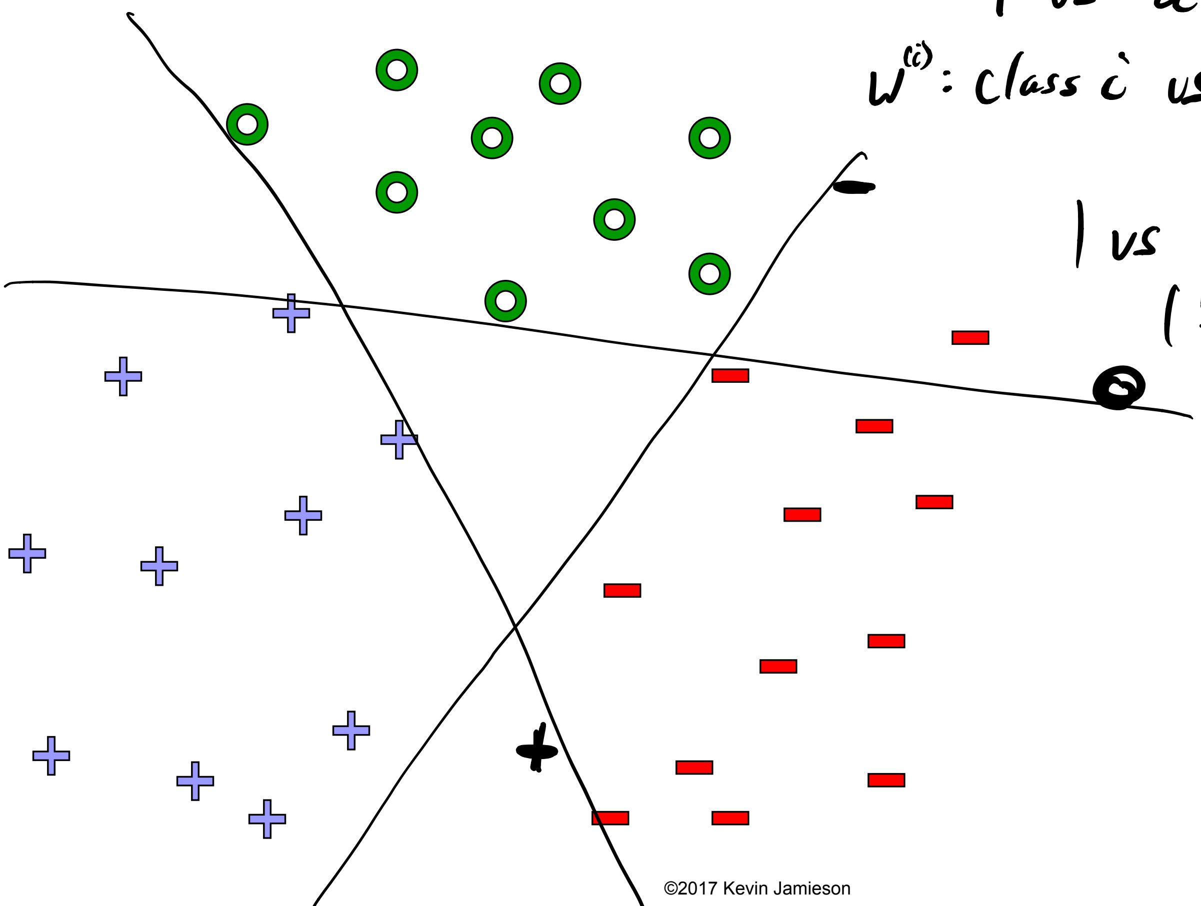
SVMs vs logistic regression

- We often want probabilities/confidences, logistic wins here?
- No! Perform isotonic regression or non-parametric bootstrap for probability calibration. Predictor gives some score, how do we transform that score to a probability?

SVMs vs logistic regression

- We often want probabilities/confidences, logistic wins here?
- No! Perform isotonic regression or non-parametric bootstrap for probability calibration. Predictor gives some score, how do we transform that score to a probability?
- For classification loss, logistic and svm are comparable
- Multiclass setting:
  - Softmax naturally generalizes logistic regression
  - SVMs have
- What about good old least squares?

What about multiple classes?



1 vs all

$w^{(i)}$: class i vs $j \neq i$

1 vs 1

$\binom{k}{2}$ classifiers