Linear SVM continued...

from last time: $\vec{w}^T \vec{x}_\Delta + b \geq +1$ for all training data ($y = +1$ (Δ))

bias

$\vec{w}^T \vec{x}_O + b \leq -1$ for all training data ($y = -1$ (O))

b is to find $\vec{w}$

Trick: combine both constraints into one constraint

$y_j (\vec{w}^T \vec{x}_j + b) \geq +1$ $\forall$ training data $j = 1, \ldots, N$ (#)

"for all"

Recall, for LINEAR classification, the boundary is that Hyperplane, "H", where $g(\vec{x}) = \vec{w}^T \vec{x} + b = 0$ $\forall \vec{x} \in H$

discriminant function

this linear equation define H

say b is integrated into $\vec{w}$

$\Rightarrow g_{\text{bias inside}}(\vec{x}) = \vec{w}_{\text{bias}} \vec{x} = 0$

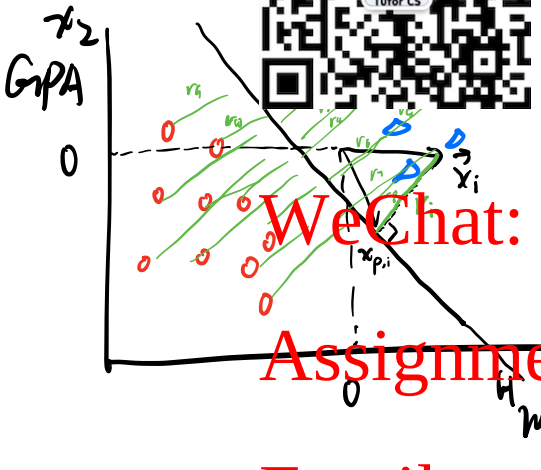
$\forall \vec{x} \in H$

$\Rightarrow 2 \cdot \vec{w}_{\text{bias}}^T \vec{x} = 2 \cdot 0 = 0$ still same boundary

$\Rightarrow 1000 \cdot \vec{w}_{\text{bias}} \vec{x} = 1000 \cdot 0 = 0$

⇒ We can scale $\vec{w}$ by any scalar value and it still would define the same boundary, H.

We need the margin between $\vec{x}$ and boundary, H



We know $\vec{r}_i \perp H$ (perpendicular)

Vector addition:

$\vec{x}_{p,i} + \vec{r}_i = \vec{x}_i$

$\vec{x}_{p,i} \in H$

$\vec{x}_{p,i} = \vec{x}_i - \vec{r}_i$

$\in H$

$\Rightarrow \vec{w}^T \vec{x}_{p,i} + b = 0$

$\Rightarrow \vec{w}^T(\vec{x}_{p,i} - \vec{r}_i) + b = 0$

we know from before is our measure of confidence

$\vec{r}_i = \frac{r_i}{\|\vec{w}\|}\vec{w}$

$\Rightarrow w^T\left(\vec{x} - \frac{r_i}{\|\vec{w}\|}\vec{w}\right) + b = 0$

We can find scalar distance r_i by solving this equation

$\vec{w}^T\vec{x} - \frac{r_i}{\|\vec{w}\|}\vec{w}^T\vec{w} + b = 0$ ($\vec{w}^T\vec{w} = \|\vec{w}\|^2$)

$\Rightarrow \vec{w}^T\vec{x} - \frac{r_i}{\|\vec{w}\|}\|\vec{w}\|^2 + b = 0$

$\vec{w}^T\vec{x} + b = r_i\|\vec{w}\| \Rightarrow r_i = \frac{w^Tx}{\|\vec{w}\|} + \frac{b}{\|\vec{w}\|}$

Def "margin for our SVM problem" $\gamma(\vec{w}, b) = \min_{\vec{x}_i \in D} |\dots|$

gama = smallest of all r_i among our training data

among $\vec{x}_i \in D$ ← training set

$= \min_{\vec{x}_i \in D} \frac{|\vec{w}^T \vec{x}_i + b|}{\|\vec{w}\|}$

Our job is to find H, ($\vec{w}$ and b), that has the largest margin $\gamma(\vec{w}, b)$ subject to our constrain (?)

goal: $\max_{\vec{w}, b} \{\gamma(\vec{w}, b)\}$

$= \max_{\vec{w}, b} \left\{ \min_{\vec{x}_i \in D} \frac{|\vec{w}^T \vec{x}_i + b|}{\|\vec{w}\|} \right\}$

Subject to constraint

$y_i(\vec{w}^T \vec{x}_i + b) \geq +1 \quad \forall \vec{x}_i \in D$

the same as the constrained optimization:

$$\max_{\vec{w}, b} \left\{ \frac{1}{\|\vec{w}\|} \min_{\vec{x}_i \in D} |\vec{w}^T \vec{x}_i + b| \right\}$$

Subject to constrain

$y_i(\vec{w}^T x_i + b) \geq +1 \quad \forall x_i \in D$

We know that we could scale $\vec{w}$ by any constant and it would yeid the same "h":

⇒ Let's choose it so

$$\frac{1}{\|\vec{w}\|}$$

This reduces our SVM optimization to

SVM formulation ☆

$$\max_{\vec{w},b}\left\{\frac{1}{\|\vec{w}\|}\right\} \Leftrightarrow \min_{\vec{w},b}\left\{\|\vec{w}\|^2\right\} \Leftrightarrow \max_{\vec{w},b}\left\{\frac{1}{2}\|\vec{w}\|^2\right\}$$

Subject to $y_i(\vec{w}\cdot\vec{x}_i+b) \geq +1 \quad \forall x_i \in D$

Linear SVM with slack

What if our training data is NOT linearly seperable?

Here, we define a cost function that allows — but punishes — when feature vectors fall too close to h

[between the boundary roads "gutters"]

and violate

SVM formulation with slack (non-differentiable)

$$(\vec{w}, b)_{SVM} = \arg\min_{\vec{w},b} \left\{ \underbrace{\vec{w}^T \vec{w}}_{\text{regularizer}} + C \cdot \sum_{j=1}^{N} \underbrace{\max\{0, 1 - y_j(\vec{w}^T \vec{x}_j + b)\}}_{\text{hinge loss}} \right\}$$

or equivalently (scale by $\frac{1}{C} = \lambda$)

we get the "squared loss SVM" formulation (yes differentiable)

$$(\vec{w}, b)_{SVM} = \arg\min_{\vec{w},b} \left\{ \lambda \vec{w}^T \vec{w} + \sum_{j=1}^{N} \left(\max\{0, 1 - y_j(\vec{w}^T \vec{x}_j + b)\} \right)^2 \right\}$$

Once we are equipped with those $\vec{w}$ th b, we can find our decision boundary:

$g(\vec{x}) = \vec{w}^T \vec{x} + b = 0 \quad \forall \vec{x} \in$ "H"

For new (non-training) instances, we decide in which side they are by $g(\vec{x}) \geq 0$ or not

" Hyperparame λ) allows us to tune how important it is for us to allow violation of strict constraints

- C BIG → you punish the objective a lot so minimizing ξ_i is very important
- C small → you don't care very much ... where you know there are many misclassified instances in training data

In machine learning:

Def "Hyperparameter" = parameter that is not learned by the system, but is tuned to control the learning

Ex: C (for SVM with slack), λ (for regularization), η learning rate for gradient descent

Like all hyp... you test with different orders of magnitude

(0.001, 0.01, 0.1, 1, 10, 100, etc)

and then you test around the best of those

(0.1, 0.2, 0.3) etc.

Nonlinear Binary Classification SVM: kernels

Here: training data is not linearly seperable:
and we are willing to relax linearly constraine

We want a (non-linear) transformation
to allow us to seperate training instances
that are not linearly seperable

Let's call the transformation that takes
each $\vec{x}$ into this new higher dimensional space $h(\vec{x})$

We repeat SVM process but with transformed data, instead of raw data.

Note: We don't really care about $h(\vec{x})$ ... as long as some function (kernel) which provides the dot product of these functions

DEF) kernel = measure of similarity used between instances $\vec{x}$ and some landmark $\vec{\ell}$ where measure complies with Mercer's theorem: $= \phi(\vec{x}, \vec{\ell})$

Mercer's theorem:

a function $\phi(\vec{x}, \vec{\ell})$ is continuous, symmetric ($\phi(\vec{x}, \vec{\ell}) = \phi(\vec{\ell}, \vec{x})$), and positive semi-definite

another space (possibly of higher dimension) such that $\phi(\vec{x}, \vec{\ell}) = h(\vec{x})^T h(\vec{\ell})$

Which kernels should we try first?

1) Linear kernel: no kernel: no transformation

2) Gaussian kernel = RBF (radial basis function)
$= \phi(\vec{x}, \vec{\ell}) = e^{\frac{-\|\vec{x}-\vec{\ell}\|^2}{2\sigma^2}}$

3) Polynomial kernel = $\phi(\vec{x}, \vec{l}) = (\vec{x}^T \vec{l} + \gamma)^d$

4) sigmoid kernel = $\phi(\vec{x}, \vec{l}) = \tanh(\gamma \vec{x}^T \vec{l} + r)$ where $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$

Note: If you use feature scaling, then similarity metrics will heavily bias in favor of features with large dynamic range

→ you need to apply feature scaling

landmarks you use are feature vectors from training data

our new decision will be:

$$\vec{w}^T \begin{bmatrix} 1 \\ \phi(\vec{x}, \vec{x}^{(1)}) \\ \phi(\vec{x}, \vec{x}^{(2)}) \\ \vdots \\ \phi(\vec{x}, \vec{x}^{(n)}) \end{bmatrix} \begin{matrix} \text{decide } w_1 \\ \geq \\ < \\ \text{decide } w_0 \end{matrix} 0$$

Clustering

Unsupervised Classification Algorithm

Unsupervised: no labels provided in training data

Classification: not prediction

$D = \{\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}\}$

Def clustering unlabeled data sample to same similarity metric

k-means Algorithm

The most commonly used clustering algorithm.