Jiatong

Machine Learning for Classification

Classification Problem: recap

- Learn: a mapping from x to a discrete value y
- F(x) = y
- Examples:
  - Spam classification
  - Document topic classification
  - Identifying faces in images

Binary Classification

- We mainly focus on binary classification (y is from {0, 1})
- Usually easy to generalize to multi-class classification

How to evaluate

- Accuracy
  - (number of correct predictions) / (total number of predictions)
- Other Measurements appropriate for some tasks:
  - Ex. We care more about certain types of mistakes

Extend Regression to Classification

- Learn regression model $f(x) = y$
- If $y > 0$, predict True
- Else, predict False

Mismatch?

- Mismatch between regression loss
  - Classification use Accuracy
  - We don’t care about large vs. small values of output

- Outliers problematic
  - Prediction of 42 for example is fine for classification, bad for regression

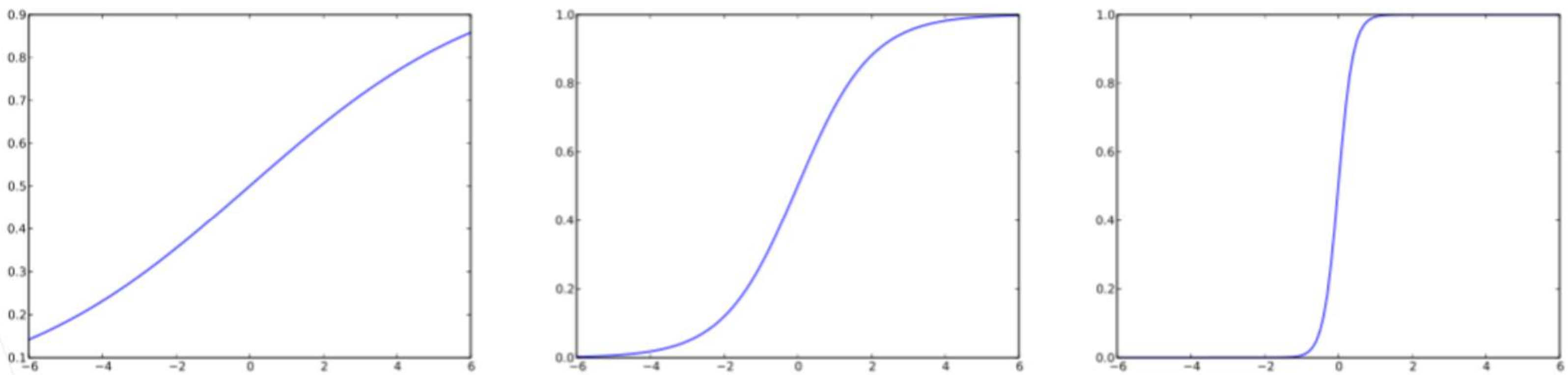
- We need output to be either 1 or 0

Logistic Function

- Quick fix: apply a function to the output of regression that gives desired value
- Logistic function
  - Outputs between 0 and 1
  - Scaling parameter alpha
  - Most outputs are close to 1 or 0
  - $g(x) = \frac{1}{1+e^{-\alpha x}}$

Logistic Function

$$g_\alpha(x) = \frac{1}{1 + e^{-\alpha x}}$$



$\alpha = 0.3$ $\alpha = 1$ $\alpha = 5$

Logistic Regression

- We can combine the logistic function and our regression model

- $g(w^T \cdot x_i) = \frac{1}{1+e^{-w^T \cdot x_i}}$

- Notice that: as $w^T \cdot x_i$ becomes
  - Large, output closer to 1
  - Small, output closer to 0

Probabilistic View

- We want to model the probability of a label given the example
- Conditional Likelihood p(y|x)
- Consider
  - We could maximize the joint p(x,y)
  - Which can be factored as p(x|y)p(y)
  - However, the best label y is the same under both
    - $\arg\max_{y=0,1} p(x|y)p(y) = argmax_{y=0,1} p(y|x)$
  - Because x is fixed/given

Probabilistic View

- We can now write the distribution as $p_w(y = 1|x) = \frac{1}{1+e^{-w^T x}}$
- Which implies that $p_w(y = 0|x)$ = $\frac{e^{-w^T x}}{1+e^{-w^T x}}$
- The odds of the event is then
  - $\frac{p_w(y = 1|x)}{p_w(y = 0|x)} = e^{w^T x}$
- The log odds is
  - $log \frac{p_w(y = 1|x)}{p_w(y = 0|x)} = w^T x$

Logistic Regression Decisions

- Given parameters w, how do we make decisions
- $p_w(y = 1|x) = \frac{1}{1+e^{-w^T x}}$
- If output > 0.5 predict 1 else 0
- In addition to prediction, we have confidence in prediction
  - Confidence is the probability of the prediction

Objective Function: Likelihood

- Conditional Data likelihood
- $p(Y|X,w) = \prod_{i=1}^{n} p(y_i|x_i,w)$
- $l(Y,X,w) = \log p(Y|X,w) = \sum_{i=1}^{n} \log p(y_i|x_i,w)$
- $p_w(y=1|x) = \frac{1}{1+e^{-w^T x}}$
- $p_w(y=0|x) = \frac{e^{-w^T x}}{1+e^{-w^T x}}$

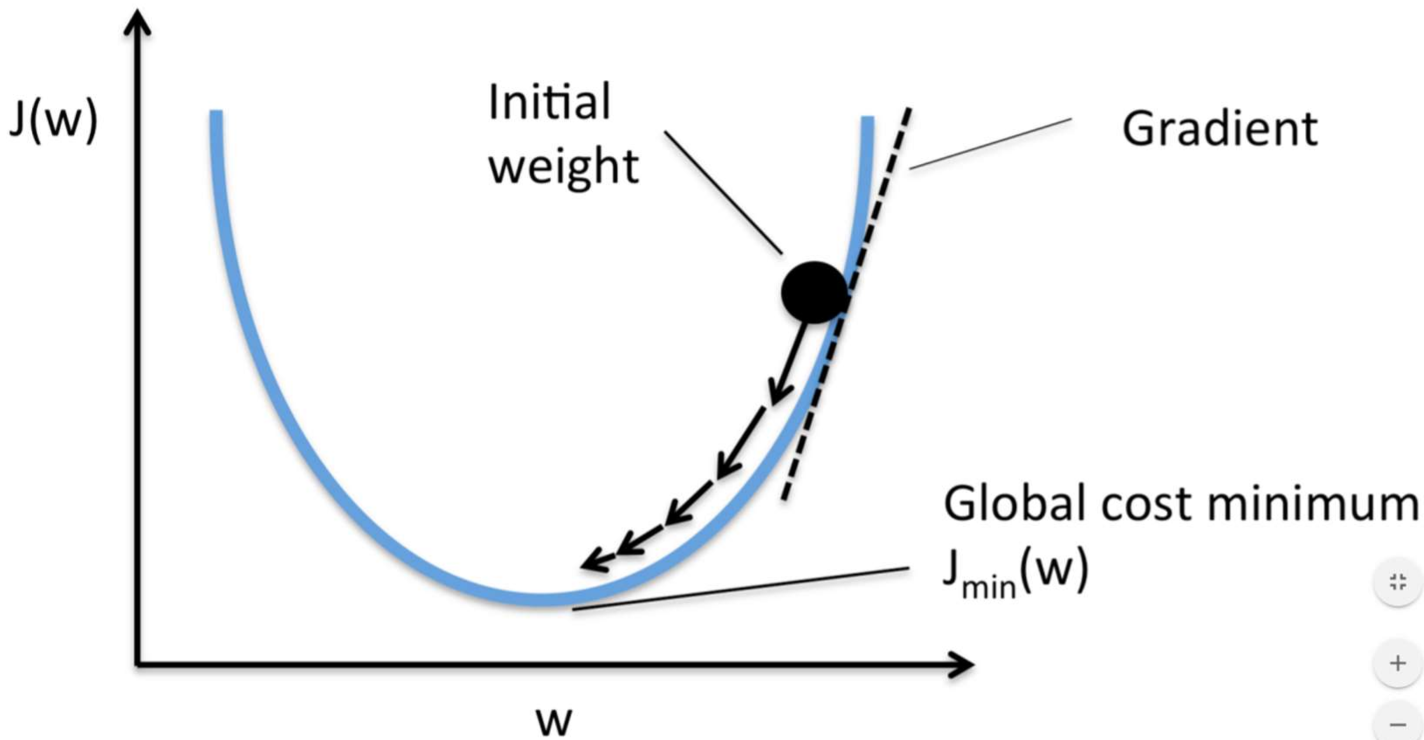
Function Optimization

- We have a function and want to maximize/minimize it
- How do we find the point at which the function reaches its max/min

- One possible solution: take the derivative, set it equal to 0, solved!

- Will this work?

Gradient Descent



Derivatives

Objective:
conditional log likelihood $\ell(Y, X, w) = \log p(Y \mid X, w) = \sum_{i=1}^{n} \log p(y_i \mid x_i, w)$

Given the sigmoid as $h_w(x) = \dfrac{1}{1 + e^{-w^T \cdot x}}$

We can rewrite compactly $p(y \mid x) = (h_w(x))^y (1 - h_w(x))^{1-y}$

New objective $\ell(Y, X, w) = \sum_{i=1}^{N} \log\{(h_w(x))^y (1 - h_w(x))^{1-y}\}$

Derivative

$$\frac{\partial \ell(Y, X, w)}{\partial w} = \sum_{i=1}^{N} (y_i - h_w(x_i)) x_i$$

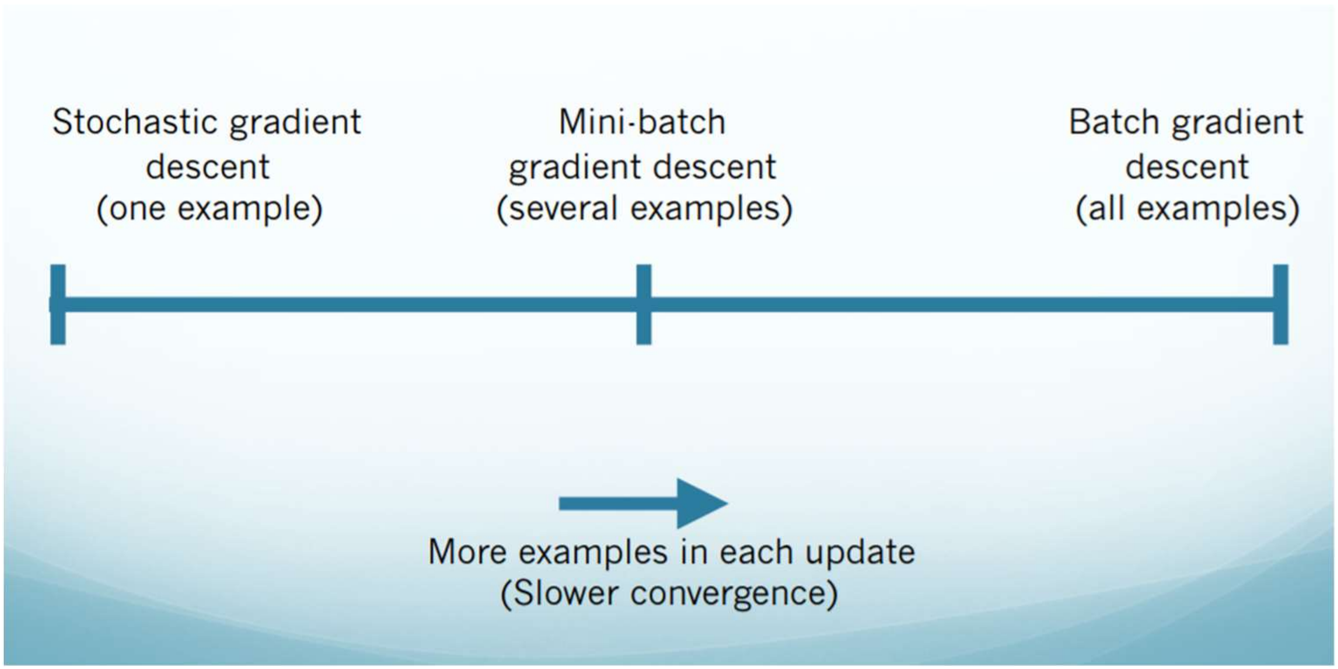
- The derivative is 0 when $y_i = p(y_i \mid x_i, w)$
- Maximizing likelihood = minimize logistic error

Gradient Descent Solution

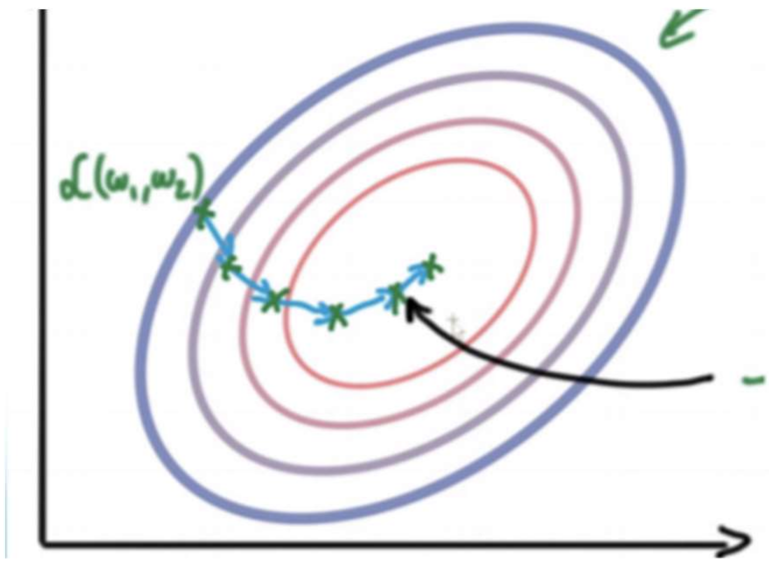
$$w^{(t+1)} = w^t + \gamma \frac{\partial \ell(Y, X, w)}{\partial w}$$

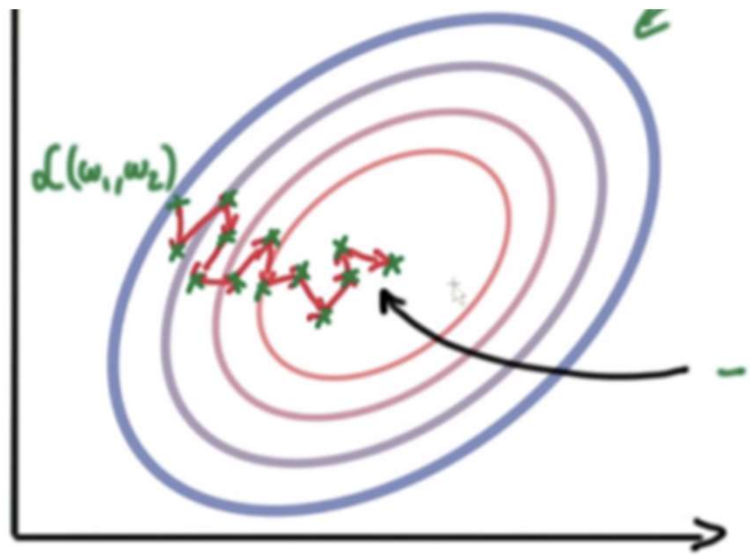
$$w^{t+1} = w^t + \gamma \sum_{i=1}^{N} (y_i - h_w(x_i))x_i$$

Stochastic Updates vs. Batch Updates



Stochastic Updates vs. Batch Updates





Perceptron

- Some Basic Settings
  - Stochastic Gradient Descent
    - One example at a time
  - Linear Classifier
    - w * x

  - Output: binary classification
  - Let’s use the simplest loss function we can think of

0/1 Loss Function

If we are wrong, loss of 1

If we are correct loss of 0

Implication:

Only make a change when we make a mistake

Hark to minimize

- Minimizing the 0/1 loss is difficult
- Replace with a similar form

- $L_w(y_i) = \sum_i^N \max(0, -y_i w \cdot x_i)$

Gradient

- $\partial L_w(y_i) = \begin{cases} 0 & y_i w x_i > 0 \\ -y_i x_i & y_i w x_i < 0 \end{cases}$

- What about when wx = 0
  - We will ignore this case, unlikely to occur

Update Rule

- $w^{i+1} = w^i + \eta \partial L_w(y_i)$
- $w^{i+1} = w^i + \eta(y_i - \hat{y}_i)x_i$

- $\hat{y}_i = sign(w \cdot x_i)$

Why is this a good update

- $w^{i+1} = w^i + \eta y_i x_i$
- $(w^{i+1} \cdot x_i) \cdot y_i = w^i \cdot x_i \cdot y_i + \eta(x_i y_i)(x_i y_i)$
- $= w^i \cdot x_i \cdot y_i + \eta y_i y_i (x_i x_i)$
- $= w^i \cdot x_i \cdot y_i + \eta \left|\left|x_i\right|\right|^2$
- $> (w^i \cdot x_i) y_i$

- Our prediction has improved
  - This says nothing about seeing the example in the future
  - The prediction may still be incorrect
  - We are just moving in the right direction

Algorithm: Perceptron

- Initialize w and eta

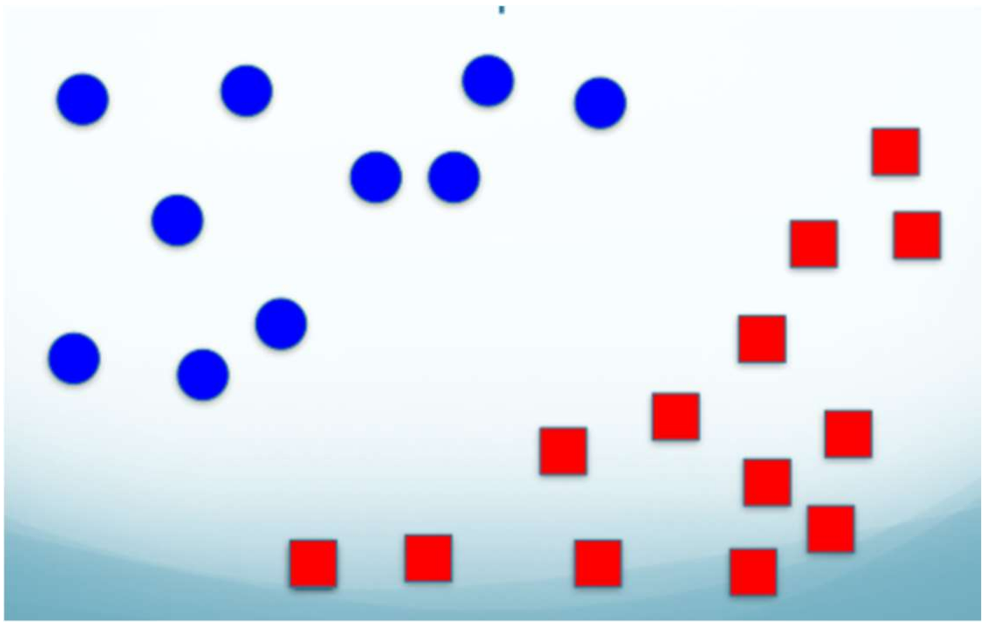
- On each round
- Receive example x
- Predict y=sign(wx)
- Receive correct label y=1 or -1
- Suffer loss
- Update w

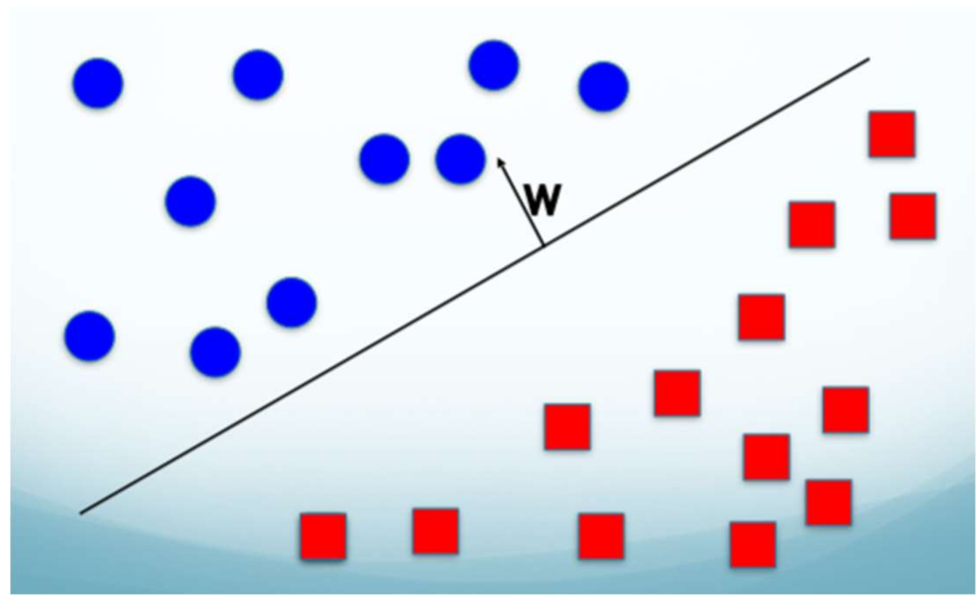
Geometric Representations

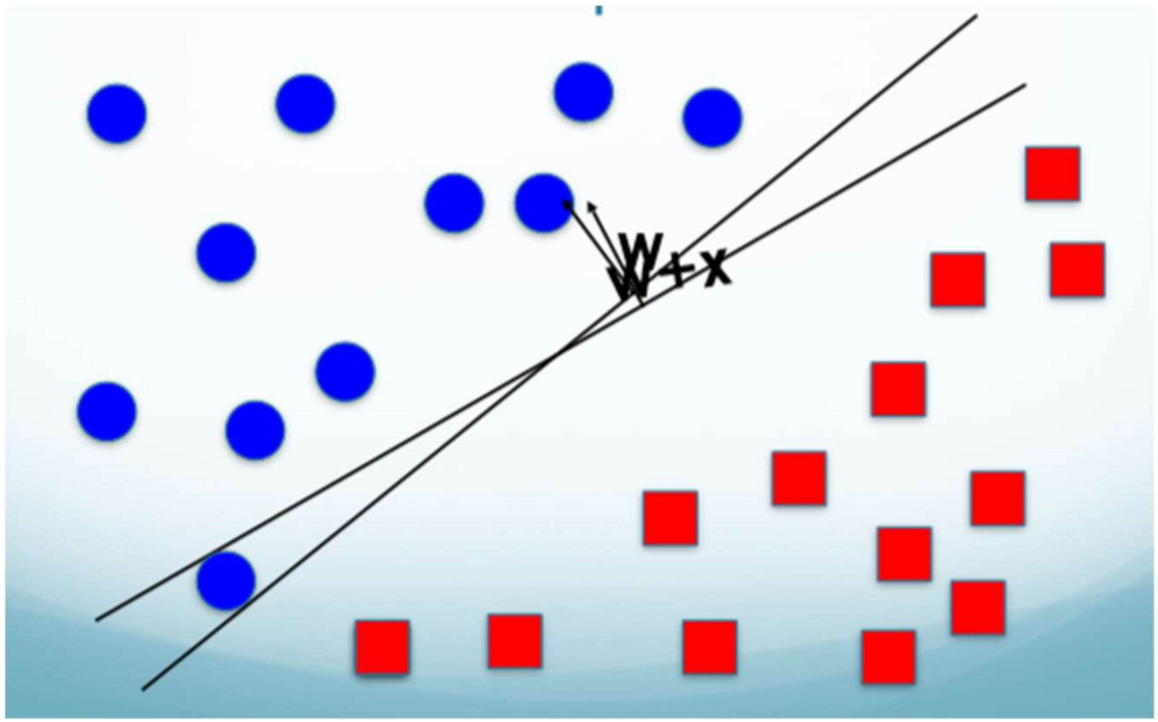
- Each example represents a point in an M dimensional space
- Classification divides space into 2 parts
- Example labeled according to where they are
- Linear Classification
  - A linear decision boundary
  - A hyper-plan (M-1 dimensions)

Discriminant Linear Classifiers

- Previously, we forced prediction by thresholding output
- Now: output either -1 or 1 directly
- Classification boundary represented by w
  - W is a vector that is orthogonal to the decision boundary
- Prediction
  - The sign of the prediction indicates which side of the boundary
  - Assume decision boundary passes through origin




w


W
W+x

Batch Algorithm

- Take a batch of examples at once
- Assumptions:
  - The data is labeled with a consistent hypothesis
  - Examples are drawn from a common IID

Remove Assumptions

- No train/test data
  - Instead access to a data stream
- Concept drift
  - No consistency in hypothesis used to label each data
- Example not IID (data distribution may change, examples may be dependent)
- Adversary Model (an adversary controls the stream)

Online Learning Algorithms

- Online Learning handles all of these cases
- Make no assumptions about the data

- Model interacts with a data stream
- Predictions needed after each example

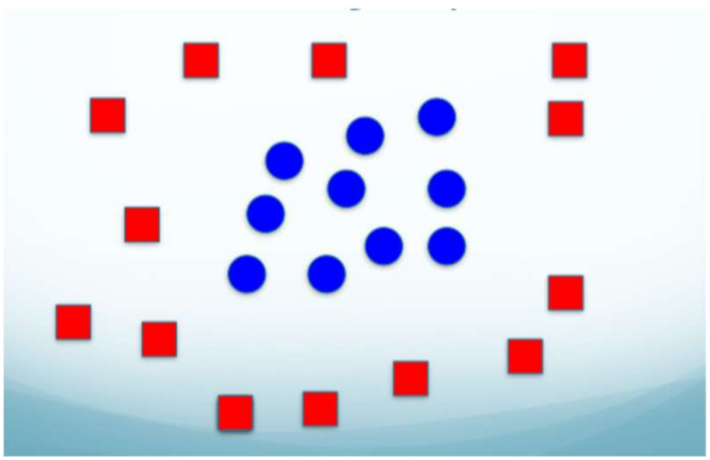
Why online Learning

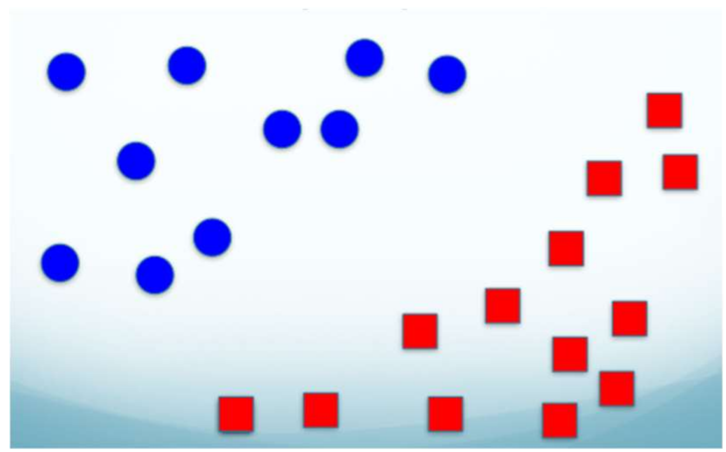
- Few assumptions means widely applicable
- Strong theoretical foundation
- Scales to large amounts of data
- Updates model without retraining
- Handles a changing world

Linear Separable

- Question: is there a linear boundary (hyper-plane) that correctly separates all of the examples
  - Yes: the examples are linearly separable
  - No: the examples are not linearly separable

- This is a separate issue from a consistent or optimal hypothesis
  - Could be not linearly separable but consistent





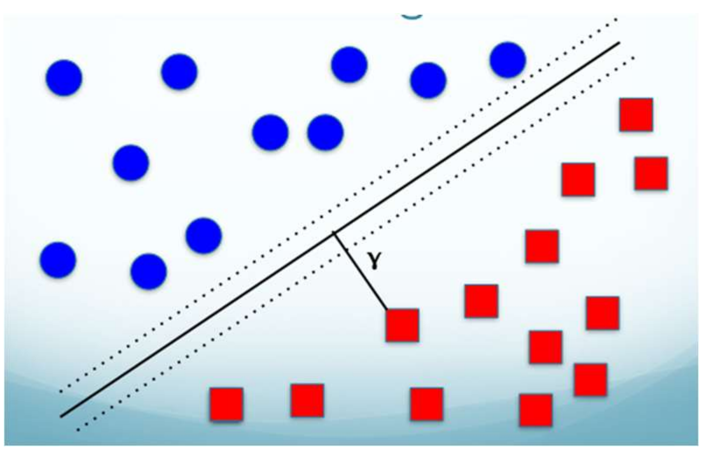
Convergence

- Assume the Data are Linear Separate
- Will the Perceptron algorithm converge
  - i.e. will it stop making updates

Convergence

- Theorem: if the provided data are linearly separable with margin γ, then Perceptron will terminate in iterations linear with respect to the number of examples

Margin



Functional Margin vs. Geometric Margin

- Functional Margin
- Prediction and y should agree to get large margin
- $\gamma^i = y_i(w^T x + b)$

- Geometric Margin
- $\gamma^i = y_i \left((\frac{w}{||w||})^T x + \frac{b}{||w||} \right)$ where $\frac{w}{||w||}$ is a unit length vector pointing in the direction of w

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
  - Givens the best generalization error on new data

Maximum Margin

- Maximum Geometric Margin

  - $\max\limits_{\gamma,w,b} \frac{\gamma}{||w||}$
  - $s.t.\, y_i(w^T x_i + b) > \gamma, i = 1, \ldots, N$

- Since γ can represent by w, for optimization problem,
- we can set it as 1

Optimization Problem

- $\min_{\gamma,w,b} 0.5||w||^2$
- $s.t. y_i(w^T x_i + b) > \gamma, i = 1, \dots, N$

- Set gamma =1, min ||w||2 same as 1/||w||

Non-separable data

- Not all data is linearly separable
  - A naïve solution, add a unique feature to every example to make it separable

- What will SVM 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

- $\min_{\gamma,w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$
- $s.t. \, y_i(w^T x_i + b) + \xi_i > 1, \text{and } \xi_i \geq 0, i = 1, \dots, N$

- We can always satisfy the margin using ξ_i
  - We want these to be small
  - Trade off parameter C (similar to lambda before)
  - They are called slack variables

Bias vs. Variance

- Smaller C means more slack
  - More training examples are wrong
  - More bias (less variance) in the output
- Larger C means less slack
  - Better fit to the data
  - Less bias (more variance) in the output

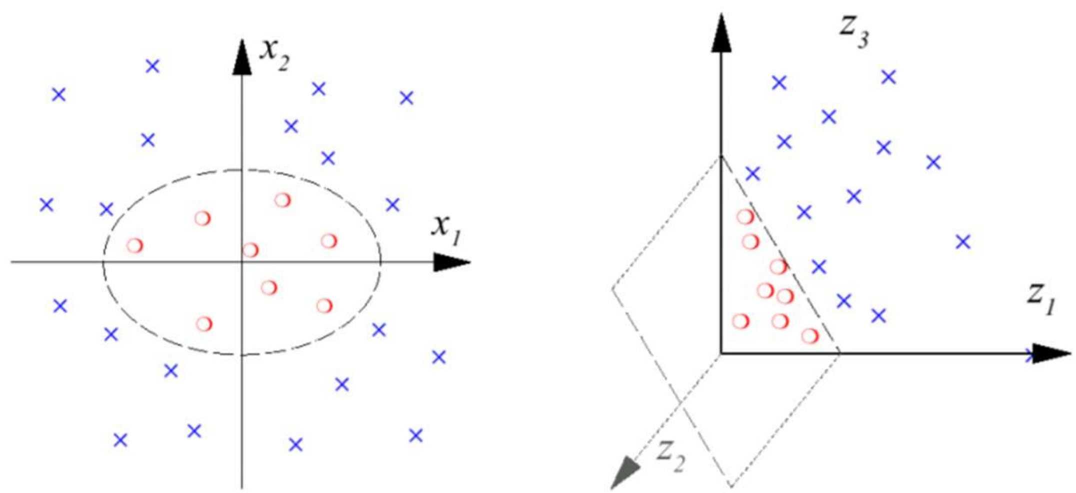
Handling Non-Linear Data

- Option 1: add features by hand that make the data separable
  - Requires feature engineering
- Option 2: Learn a small number of additional features that will suffice
  - We'll see this eventually
- Option 3: kernel trick

Feature Mapping Functions

- Assuming a two dimensional vector x=[x(1), x(2)]
  - X(i) is the ith position of x
- Let's apply a feature mapping function: 2nd order polynomial function

- $\phi([x_1, x_2]) = \left(x_1^2, \sqrt{2} \cdot x_1 x_2, x_2^2\right)$
- Why is it useful
  - If the boundary is $\phi_1 + 2\phi_3 < 3$
  - Not linear in x, but linear in $\phi([x_1, x_2])$



x_2
x_1
z_3
z_1
z_2

Why Feature Mapping Functions

- Any dataset is linearly separable if we use enough dimensions
  - In an n-dimensional space, almost any set of up to n+1 labeled points is linearly separable
- We can obtain linear separability by projecting data into higher dimensional spaces
  - Use smarter techniques to obtain generalizable separability

Feature Functions + SVM

- Replace x with a feature mapping function
  - $\min_{\gamma,w,b} \frac{1}{2}||w||^2$
  - $s.t.\, y_i(w^T \phi(x_i)) > \gamma, i = 1, \dots, N$
- The dot product is now taken over a higher dimensional feature space
- If ϕ is quadratic then the feature space is a quadratic space in terms of the inputs

Limitations

- We still have to learn w
  - W will grow in size of the feature space
  - E.g. quadratic kernel: |x|=100 → |phi(x)| = 10000
- Feature function just increase the feature space in a non-linear way
- Too limiting

With some magic from optimization theory

- We can reformat the SVM into dual format

- $\max_{\alpha} \sum \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i x_j^T)$
- $s.t.\, \alpha_i \geq 0 \; and \; \sum \alpha_i y_i = 0$

- (we skip a lot here, the derivation is complex, so I will leave it as after-class material)

Dual Formulation

- Primal Problem:
  - Objective function is a combination of the m variables (feature space)
  - Solution is a vector of m values that minimize function
  - Minimizing
- Dual problem:
  - Objective function is combination of the n variables (num of data samples)
  - Solution is a vector of n values called the dual variables
  - Maximizing

Use dual for kernel tricks

- $\max_{\alpha} \sum \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\phi \cdot \phi^T)$
- $s.t. \alpha_i \geq 0 \ and \ \sum \alpha_i y_i = 0$

- There is no modeling constraint that prevents us from making phi very large
- Alphas do not grow in the size of phi

Kernels

- $\max_{\alpha} \sum \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (K(x, x'))$
- $s.t. \alpha_i \geq 0 \ and \ \sum \alpha_i y_i = 0$

- Where $x^T w = x^T \sum_{i=1}^{n} \alpha_i y_i x_i = \sum_{i=1}^{n} \alpha_i y_i K(x, x_i)$

Intuition about over-fitting

- Assuming we project features then even using the simple projection shown so far, we'd have way to many features!
- Didn't we learn that too many features means overfitting?

Saved by the Dual

We aren't free to choose a parameter for each feature

- w is a linear combination of the inputs
- We can only choose the parameters for αs
- There are only n αs, no matter how large our feature space projection
- The inputs x put a constraint on our flexibility in high dimensional space

Kernel Trick

- Take a linear SVM
- Substitute a non-linear kernel
- Optimize objective in the dual
- We get non-linear Classification
- Without
  - Over-fitting
  - Learning too many parameters
  - Computing a large feature space

What is kernel

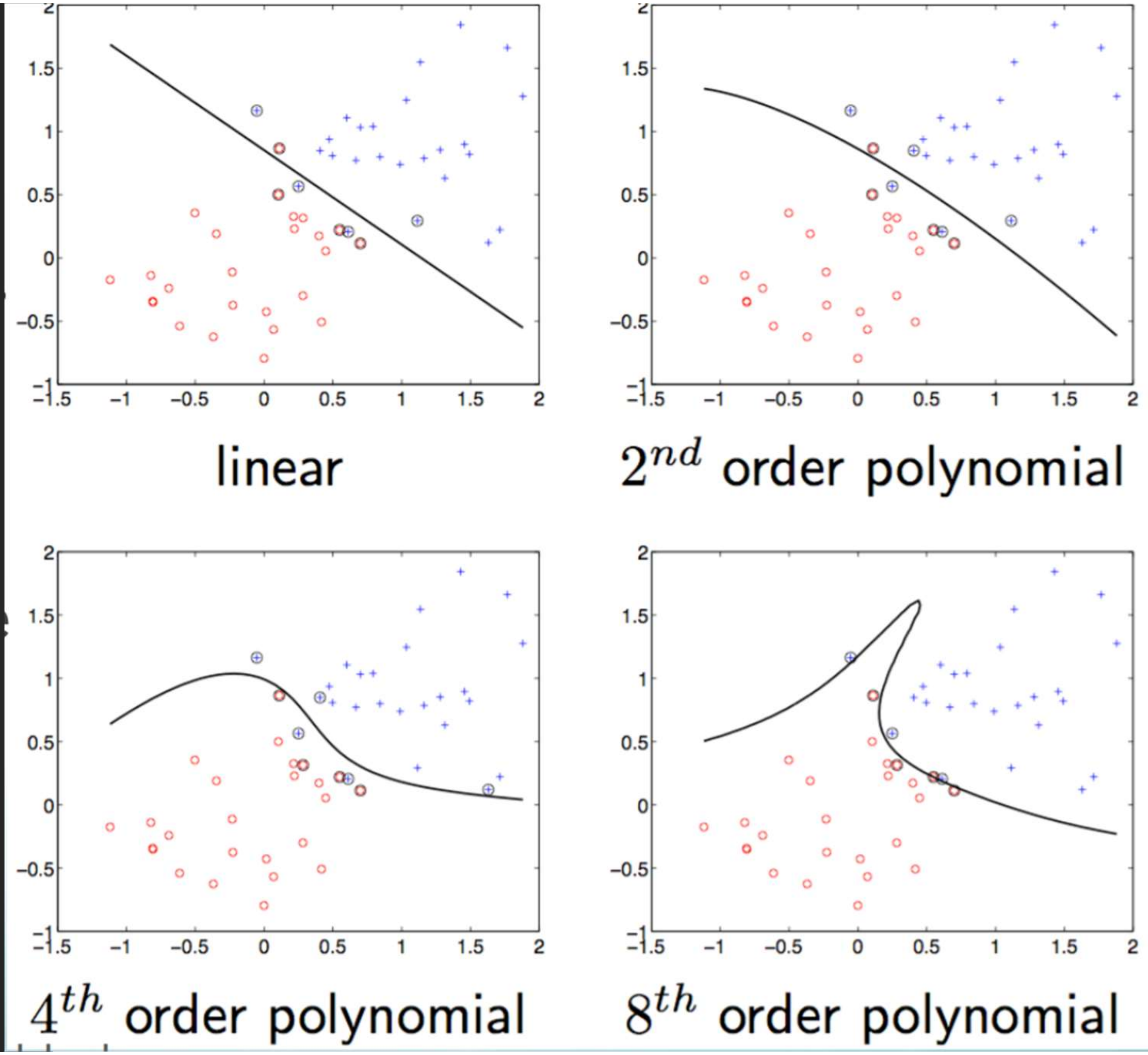
- A kernel is a scalar product between two high dimensional feature vectors
- We can define any mapping function and then compute the kernel

Quadratic Kernel

- $K(x, x') = (x \cdot x')^2$
- Why this a valid kernel
- $K(x, x') = (x \cdot x')^2 = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \cdot (x_1'^2, x_2'^2, \sqrt{2}x_1'x_2')$

Polynomial Kernel

$K(x, x') = \left(1 + (x^T x')\right)^p$



Q&A