CS 6840: Natural Language Processing

ML Algorithms for Classification

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

Binary Classification: Sentiment Analysis

Movie reviews:

Positive: This was a great movie, which I thoroughly enjoyed.

Negative: I was very disappointed in this movie, it was a waste of time.

- Lexical features, e.g. presence of words such as *great* or *disappointed,* can be used to determine the sentiment orientation.
  - Can you think of examples where the same word may be used for both types of sentiment? How would you fix that?
- Represent each review as a *bag-of-words* feature vector:
  - High dimensional, sparse feature vector => use sparse representations that map features to indeces.
  - Feature value is 1 if word is present, 0 otherwise:
    - Can use more sophisticated word weighting schemes from IR, such as tf.idf.
    - Can use stems instead of tokens.

Sentiment Analysis

Movie reviews:

Positive: This was a great movie, which I thoroughly enjoyed.

Positive: Despite the bad reviews I read online, I liked this move.

Negative: The movie was not as good as I expected.

- It appears that the bag-of-words approach is not sufficient.
- Can try to address negation:
  - Use bigram NOT_X for all words X following the negation [Pang et al. EMNLP'02].
- Model sentiment compositionality:
  - Train recursive deep models over sentiment treebanks [Socher et al., EMNLP'13]
- Apply more sophisticated classifiers:
  - Convolutional Neural Networks (CNNs) [Kim, 2014]

Sentiment Analysis

More examples showing the limitations of *bag-of-words* models [Eisenstein, 2019]:

a. That's not bad for the first day.
b. This is not the worst thing that can happen.
c. It would be nice if you acted like you understood.
d. There is no reason at all to believe that the polluters are suddenly going to become reasonable. (Wilson et al., 2005)
e. This film should be brilliant. The actors are first grade. Stallone plays a happy, wonderful man. His sweet wife is beautiful and adores him. He has a fascinating gift for living life fully. It sounds like a great plot, **however**, the film is a failure. (Pang et al., 2002)

Classification Algorithms

- Train a classification algorithm on the labeled feature vectors, i.e. training examples.

- Use trained model to determine the sentiment orientation of new, unseen reviews.

- (Generalized) Linear models:
  - **Perceptron**
  - **Support Vector Machines**
  - **Logistic Regression**

Linear Discriminant Functions

- Use a linear function of the input vector:

$$\underbrace{h(\mathbf{x}) = \mathbf{w}^T}_{\textit{weight vector}} \varphi(\mathbf{x}) + \underbrace{w_0}_{\textit{bias} = -\textit{threshold}}$$

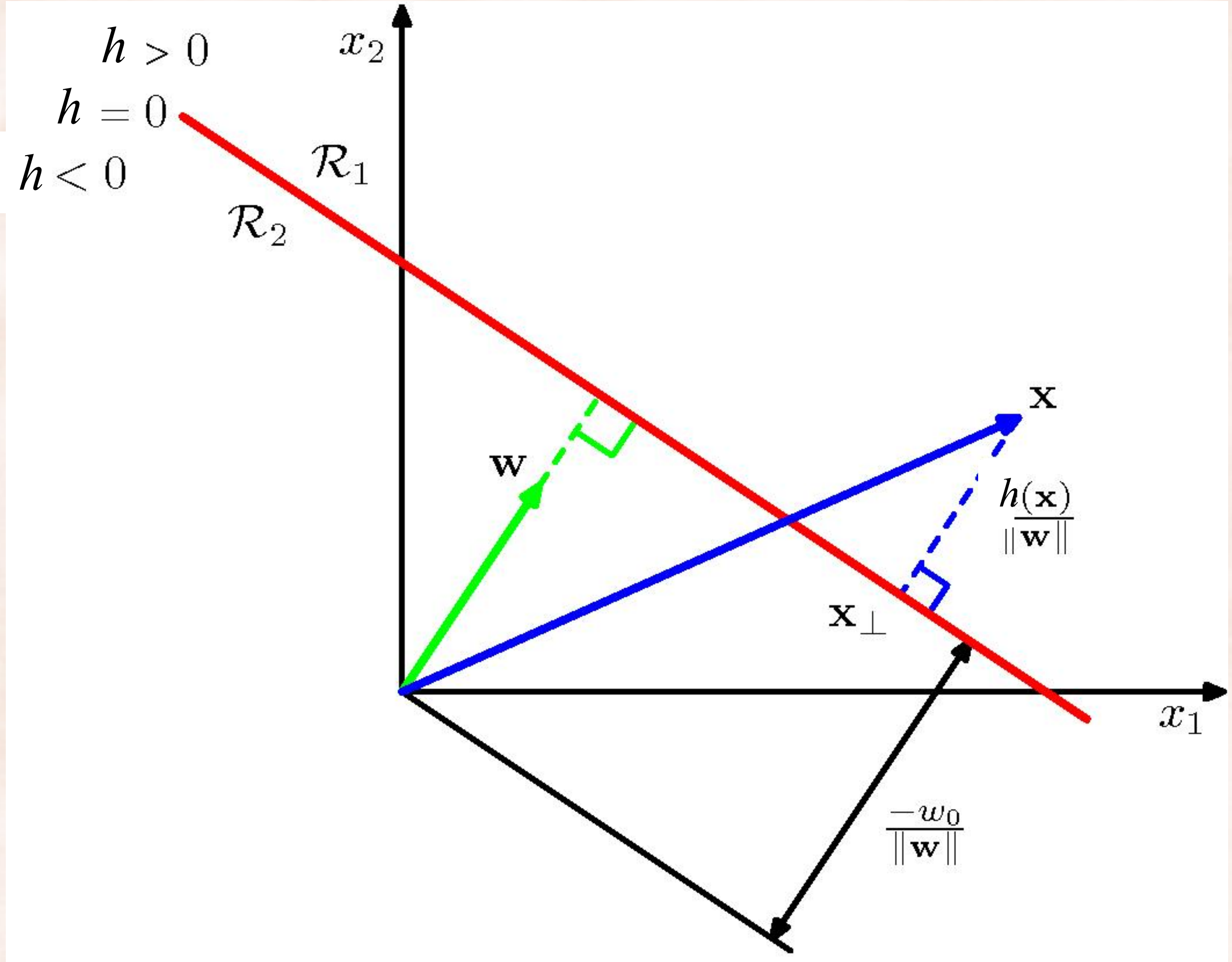
- Decision:

  $\mathbf{x} \in C_1$ if $h(\mathbf{x}) \geq 0$, otherwise $\mathbf{x} \in C_2$.

  $\Rightarrow$ decision boundary is hyperplane $h(\mathbf{x}) = 0$.

- Properties:
  - $\mathbf{w}$ is orthogonal to vectors lying within the decision surface.
  - w_0 controls the location of the decision hyperplane.

Geometric Interpretation



The Perceptron Algorithm: Two Classes $t_n \in \{+1, -1\}$

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $n = 1 \ldots N$
3. $\quad y_n = sgn(\mathbf{w}^T \mathbf{x}_n)$
4. $\quad$ **if** $y_n \neq t_n$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + t_n \mathbf{x}_n$

Repeat:
a) until convergence.
b) for a number of epochs E.

Theorem [Rosenblatt, 1962]:

If the training dataset is linearly separable, the perceptron learning algorithm is guaranteed to find a solution in a finite number of steps.

- see Theorem 1 (Block, Novikoff) in [Freund & Schapire, 1999].

Perceptron as Stochastic Gradient Descent

- **Perceptron Criterion**:
  - Set labels to be +1 or − 1. Want $\mathbf{w}^{\mathrm{T}}\mathbf{x}_n > 0$ for $t_n = 1$, and $\mathbf{w}^{\mathrm{T}}\mathbf{x}_n < 0$ for $t_n = -1$.

    $\Rightarrow$ would like to have $\mathbf{w}^{\mathrm{T}}\mathbf{x}_n t_n > 0$ for all patterns.

    $\Rightarrow$ want to minimize $-\mathbf{w}^{\mathrm{T}}\mathbf{x}_n t_n$ for all missclassified patterns M.

$$\boxed{\text{minimize } E_p(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T \mathbf{x}_n t_n}$$

- Update parameters **w** sequentially **after each mistake**:

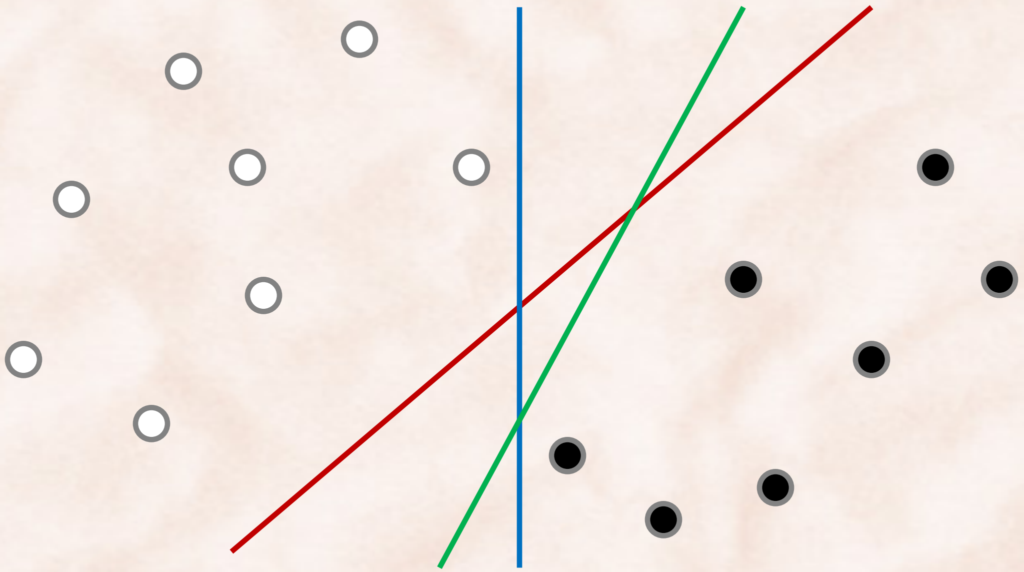
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}^{(\tau)}, \mathbf{x}_n)$$

$$= \mathbf{w}^{(\tau)} + \eta \mathbf{x}_n t_n$$

- The magnitude of **w** is inconsequential => set $\eta = 1$.

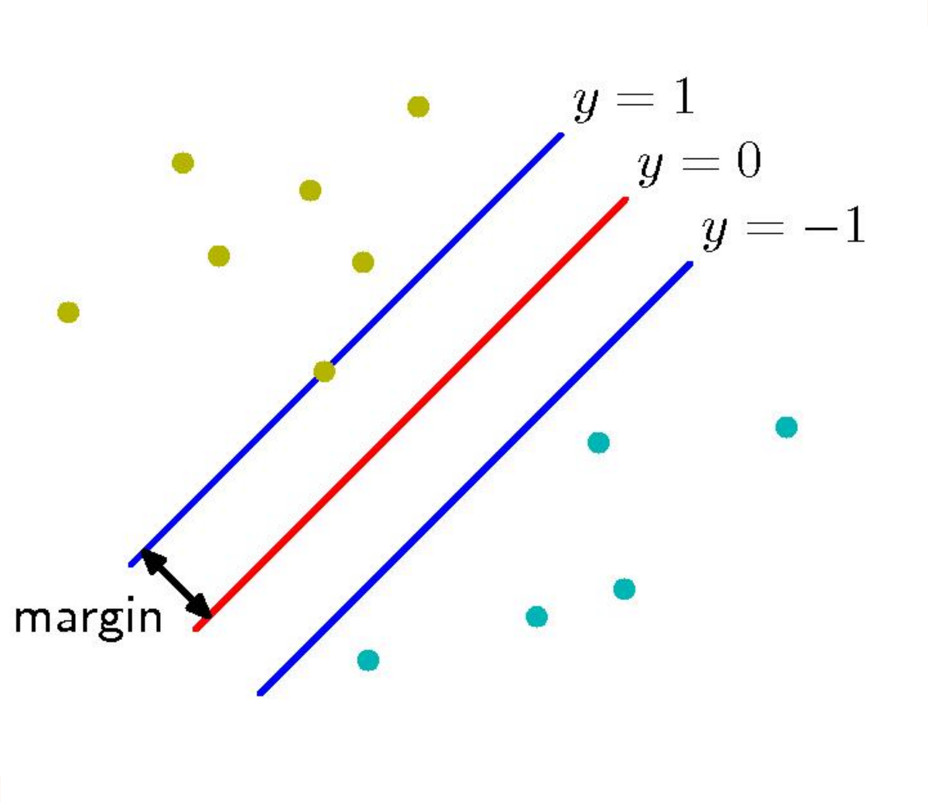
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \mathbf{x}_n t_n$$

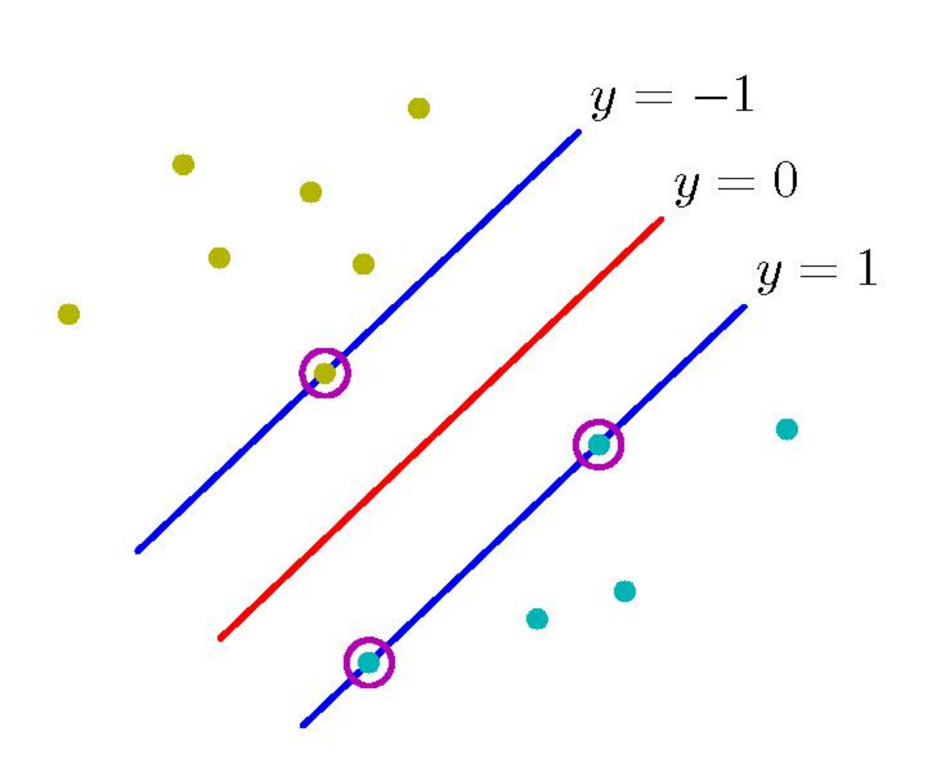
Linear Classifiers & Margin



- Perceptron solution depends on initial values of w and b and order of processing of data points.
- Which classifier has the smallest generalization error?
  - The one that maximizes the margin [Computational Learning Theory]
    - **margin** = the distance between the decision boundary and the closest sample.

Maximum Margin Classifiers





- The distance between $\mathbf{x}_n$ and hyperplane $y(\mathbf{x}) = 0$ is $\dfrac{|y(\mathbf{x}_n)|}{\|\mathbf{w}\|} = \dfrac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \dfrac{t_n(\mathbf{w}^T\varphi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$

Maximum Margin Classifiers

- Margin = the distance between hyperplane $y(\mathbf{x})$=0 and closest sample:

$$\min_n \left[\frac{t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \right]$$

- Find parameters **w** and b that maximize the margin:

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \right] \right\}$$

- Rescaling **w** and b does not change distances to the hyperplane:

$\Rightarrow$ for the closest point(s), set $t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) = 1$

$\Rightarrow$ $t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1, \quad \forall n \in \{1, \ldots, N\}$

Max-Margin: Quadratic Optimization

- Constrained optimization problem:

minimize:

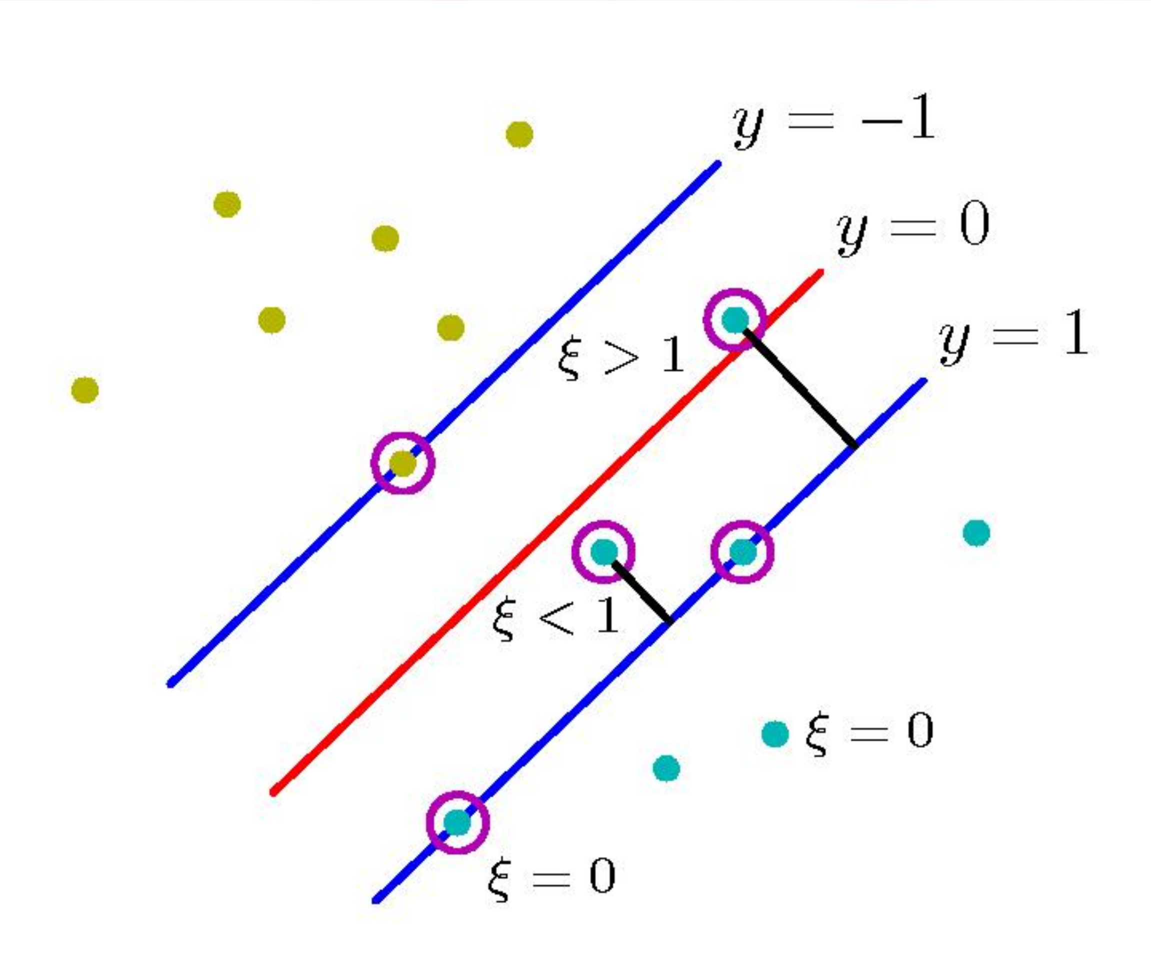
$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1, \quad \forall n \in \{1, \ldots, N\}$$

- But most real data is not linearly separable => allow for some *slack* in the constraints.

Max-Margin: Non-Separable Case



Max-Margin: Non-Separable Case

- Optimization problem:

minimize:

$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N} \xi_n$$

subject to:

$$t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1 - \xi_n, \quad \forall n \in \{1, \ldots, N\}$$
$$\xi_n \geq 0$$

- Tipically solved using the technique of Lagrange Multipliers, which enables the use of non-linear *kernels*.
- Here we will solve the linear SVM using gradient descent.

Linear SVM: The (Sub)Gradient

minimize:

$$J(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N}\xi_n$$

subject to:

$$t_n(\mathbf{w}^T\varphi(\mathbf{x}_n)+b) \geq 1-\xi_n, \quad \forall n \in \{1,\ldots,N\}$$
$$\xi_n \geq 0$$

The two constraints can be written as:

$$\xi_n = \max(0, 1 - t_n h(\mathbf{x}_n))$$

This leads to the equivalent unconstrained optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N}\max(0, 1 - t_n h(x_n))$$

- To compute the gradient ∇J we need to compute the gradient of each slack term ξ_n :

$$\frac{\partial \xi_n}{\partial \mathbf{w}} = 0 \text{ if } \xi_n = 0 \qquad\qquad \frac{\partial \xi_n}{\partial \mathbf{w}} = -t_n\mathbf{x}_n \text{ if } \xi_n > 0$$

Linear SVM: The (Sub)Gradient Descent Algorithm

$$\min_{\mathbf{w},b} J(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N} \max(0, 1 - t_n h(\mathbf{x}_n))$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \max(0, 1 - t_n h(\mathbf{x}_n))\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi_n\right) \quad \text{where } \lambda = 1/C$$

- Gradients are: $\frac{\partial \xi_n}{\partial \mathbf{w}} = 0$ if $\xi_n = 0$ $\quad \frac{\partial \xi_n}{\partial \mathbf{w}} = -t_n \mathbf{x}_n$ if $\xi_n > 0$

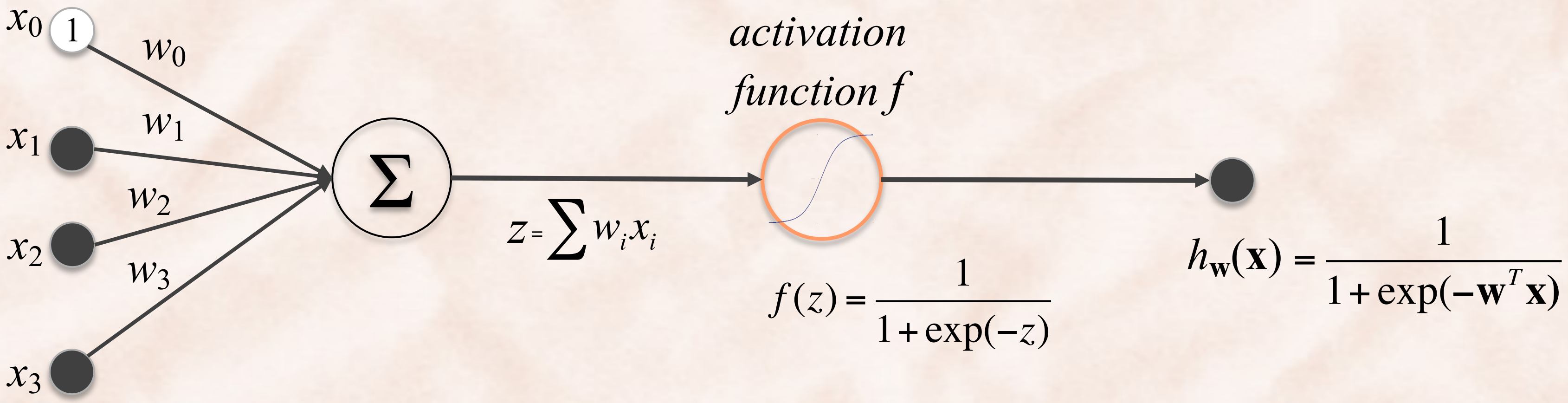
- Stochastic gradient Descent update is:

  $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta(\lambda \mathbf{w}^t - t_n \mathbf{x}_n)$ if $t_n h(\mathbf{x}_n) < 1$

  $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \lambda \mathbf{w}^t$ otherwise

  – In the Pegasos algorithm the learning rate is set at $\eta = \frac{1}{\lambda t}$

Logistic Regression for Binary Classification



- Used for binary classification of examples $\mathbf{x} = [1, x_1, x_2, ..., x_k]^T$
  - Labels $T = \{C_1, C_2\} = \{1, 0\}$
  - Output C_1 if and only if $h(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) > 0.5$
- Training set is $(\mathbf{x}_1,t_1), (\mathbf{x}_2,t_2), \ldots (\mathbf{x}_n,t_n)$.

Logistic Regression for Binary Classification

- Model output can be interpreted as **posterior class probabilities**:

$$p(C_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}))}$$

$$p(C_2 \mid \mathbf{x}) = 1 - \sigma(\mathbf{w}^T \mathbf{x}) = \frac{\exp(-\mathbf{w}^T \mathbf{x})}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

- Inference:
  - Output C_1 if $\mathrm{p}(C_1|\mathrm{x}) \geq 0.5$, else output C_2.
  - Show that it corresponds to a linear decision boundary.

- Training:
  - What **error/cost/loss function** to minimize?

Logistic Regression: Training

- Training set is $D = \{\langle \mathbf{x}_n, t_n \rangle \mid t_n \in \{0,1\}, n \in 1 \ldots N\}$
- Training = finding the "right" parameters $\mathbf{w}^T = [w_0, w_1, \ldots, w_k]$
  - Find $\mathbf{w}$ that minimizes an *error function* $E(\mathbf{w})$ which measures the misfit between $h(\mathbf{x}_n, \mathbf{w})$ and t_n.
  - Expect that $h(\mathbf{x},\mathbf{w})$ performing well on training examples $\mathbf{x}_n \Rightarrow h(\mathbf{x},\mathbf{w})$ will perform well on arbitrary test examples $\mathbf{x} \in X$.

Maximum Likelihood (ML) principle: find parameters that maximize the likelihood of the labels.

- The likelihood function is: $p(\mathbf{t}|\mathbf{w}, X) = \prod_{n=1}^{N} p(t_n|\mathbf{w}, x_n)$
- The negative log-likelihood (cross entropy): $-\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \ln p(t_n|x_n)$

Logistic Regression: Training

- The Maximum Likelihood solution is:

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} p(\mathbf{t} \mid \mathbf{w}) = \arg\min_{\mathbf{w}} \boxed{E(\mathbf{w})}$$

convex in **w**

- Maximum Likelihood solution is given by $\nabla E(\mathbf{w}) = 0$
  - Cannot solve analytically => solve numerically with gradient based methods: (stochastic) **gradient descent**, conjugate gradient, L-BFGS, etc.
  - Gradient is (prove it): $\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \boxed{(h_n - t_n)\mathbf{x}_n^T}$

What does this represent for binary features?

where $h_n = \sigma(\mathbf{w}^T \mathbf{x}_n)$

Regularized Logistic Regression

- Maximum a Posteriori solution:

$$\mathbf{w}_{MAP} = \arg\min_{\mathbf{w}} E_D(\mathbf{w}) + E_{\mathbf{w}}(\mathbf{w}) = \text{argmin} - \sum_{n=1}^{N} \ln p(t_n|x_n) + \frac{\alpha}{2} \|\mathbf{w}\|^2$$

- MAP solution is given by $\nabla E(\mathbf{w}) = \nabla E_D(\mathbf{w}) + \nabla E_{\mathbf{w}}(\mathbf{w}) = 0$.

  – Cannot solve analytically => solve numerically using (**stochastic**) **gradient descent**, conjugate gradient, L-BFGS, …

  – Gradient is (prove it): $\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (h_n - t_n)\mathbf{x}_n^T + \alpha \mathbf{w}^T$

where $h_n = \sigma(\mathbf{w}^T \mathbf{x}_n)$

Logistic Regression for Multiclass Classification

- Multiclass classification:

  $T = \{C_1, C_2, ..., C_K\} = \{1, 2, ..., K\}$.

- Training set is $(\mathbf{x}_1,t_1), (\mathbf{x}_2,t_2), \dots (\mathbf{x}_n,t_n)$.

  $\mathbf{x} = [1, x_1, x_2, ..., x_M]$

  $t_1, t_2, \dots t_n \in \{1, 2, ..., K\}$

- K weight vectors, one per class: $p(C_k \mid \mathbf{x}) = \dfrac{\exp(\mathbf{w}_k^T \mathbf{x}))}{\sum_j \exp(\mathbf{w}_j^T \mathbf{x})}$

- One weight vector:

Logistic Regression (K ≥ 2)

- Inference:

$$C_* = \arg\max_{C_k} p(C_k \mid \mathbf{x})$$

$$= \arg\max_{C_k} \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\boxed{\sum_j \exp(\mathbf{w}_j^T \mathbf{x})}}$$

Z(**x**) *is the partition function*

$$= \arg\max_{C_k} \exp(\mathbf{w}_k^T \mathbf{x})$$

$$= \arg\max_{C_k} \mathbf{w}_k^T \mathbf{x}$$

- Training using:
  – Maximum Likelihood (ML)
  – Maximum A Posteriori (MAP) with a Gaussian prior on **w**.

Logistic Regression (K ≥ 2)

- The **negative log-likelihood** error function is:

convex in $\mathbf{w}$

$$E_D(\mathbf{w}) = -\frac{1}{N}\ln\prod_{n=1}^{N} p(t_n \mid \mathbf{x}_n) = \boxed{-\frac{1}{N}\sum_{n=1}^{N}\ln\frac{\exp(\mathbf{w}_{t_n}^T\mathbf{x}_n)}{Z(\mathbf{x}_n)}}$$

- The ML solution is:

$$\mathbf{w}_{ML} = \arg\min_{\mathbf{w}} E_D(\mathbf{w})$$

- The **gradient** is (prove it):

$$\nabla_{\mathbf{w}_k} E_D(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left(\delta_k(t_n) - p(C_k \mid \mathbf{x}_n)\right)\mathbf{x}_n$$

where $\delta_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$ is the *Kronecker delta* function.

L_2 Regularized Logistic Regression ($K \geq 2$)

- The new **cost** function is:

$$E(\mathbf{w}) = E_D(\mathbf{w}) + E_{\mathbf{w}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} \ln \frac{\exp\left(\mathbf{w}_{t_n}^T \mathbf{x}_n\right)}{Z(\mathbf{x}_n)} + \frac{\alpha}{2}\|\mathbf{w}\|^2$$

- The new **gradient** is (prove it):

$$\nabla_{\mathbf{w}_k} E(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left(\delta_k(t_n) - p(C_k \mid \mathbf{x}_n)\right)\mathbf{x}_n^T + \alpha \mathbf{w}_k^T$$

Logistic Regression ($K \geq 2$)

- ML solution is given by $\nabla E(\mathbf{w}) = 0$.
  - Cannot solve analytically.
  - Solve numerically, by pluging [*cost*, *gradient*] = [$E(\mathbf{w})$, $\nabla E(\mathbf{w})$] values into general convex solvers:
    - L-BFGS
    - Newton methods
    - conjugate gradient
    - (stochastic / minibatch) gradient-based methods.
      - gradient descent (with / without momentum).
      - AdaGrad, AdaDelta
      - RMSProp
      - ADAM, ...

Logistic Regression

- Stochastic gradient update for binary case:
  - No regularization:
    $\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta(h_n - t_n)\mathbf{x}_n$ where $h_n = \sigma(\mathbf{w}^{t}\mathbf{x}_n)$
  - With L2 regularization:
    $\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta(\alpha\mathbf{w}^{t} + (h_n - t_n)\mathbf{x}_n)$

- Stochastic gradient update for multiclass case:
  - No regularization:
    $\mathbf{w}_k^{t+1} = \mathbf{w}_k^{t} - \eta(P(C_k|\mathbf{x}_n) - \delta_k(t_n))\mathbf{x}_n$ where $P(C_k|\mathbf{x}_n) = softmax(\mathbf{w}_k^{t}\mathbf{x}_n)$
  - With L2 regularization:
    $\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta(\alpha\mathbf{w}_k^{t} + (P(C_k|\mathbf{x}_n) - \delta_k(t_n))\mathbf{x}_n)$

SVMs for multiclass classification