CS 1675: Intro to Machine Learning
Intro to Classification
(Nearest Neighbors, Logistic Regression, Perceptron)

Prof. Adriana Kovashka
University of Pittsburgh
September 27, 2018
Classification

• Given features $\mathbf{x}$, predict categorical output $y$

• For example:
  – Given attributes of a house (e.g. square footage and age built), predict whether it will be bought for the asking price or for less
  – Given temperature, predict whether it will rain, snow, or be sunny

• The rest of the course will cover different supervised approaches to classification
Plan for this lecture

• The simplest classifier: K-Nearest Neighbors
  – Algorithm and example use
  – Generalizing: Distance metrics, weighing neighbors
  – Problems: curse of dimensionality, picking K

• Logistic regression
  – Probability: review
  – Linear regression for classification?
  – Maximum likelihood solution for logistic regression
  – Related algorithm: perceptron
Nearest Neighbors: Key Idea

• A type of supervised learning: We want to learn to predict, for a new data point \( x \), its label \( y \) (e.g. spam / not spam)
• Don’t learn an explicit function \( F: X \rightarrow Y \)
• Keep all training data \( \{X, Y\} \)
• For a test example \( x \), find the training example \( x_i \) closest to it (e.g. using Euclidean distance)
• Then copy the target label \( y_i \) as the label for \( x \)
Related Methods / Synonyms

• Instance-based methods
• Exemplar methods
• Memory-based methods
• Non-parametric methods
Instance/Memory-based Learning

Four things make a memory based learner:

• A distance metric

• How many nearby neighbors to look at?

• A weighting function (optional)

• How to fit with the local points?

Slide credit: Carlos Guestrin
1-Nearest Neighbor Classifier

Four things make a memory based learner:

- A *distance metric*
  - Euclidean (and others)

- *How many nearby neighbors to look at?*
  - 1

- A *weighting function (optional)*
  - Not used

- *How to fit with the local points?*
  - Predict the same output as the nearest neighbor
1-Nearest Neighbor Classifier

\[ f(x) = \text{label of the training example nearest to } x \]

Adapted from Lana Lazebnik
K-Nearest Neighbor Classifier

Four things make a memory based learner:

• A distance metric
  – Euclidean (and others)

• How many nearby neighbors to look at?
  – K

• A weighting function (optional)
  – Not used

• How to fit with the local points?
  – Predict the average output among the nearest neighbors
K-Nearest Neighbor Classifier

- For a new point, find the $k$ closest points from training data (e.g. $k=5$)
- Labels of the $k$ points “vote” to classify

If query lands here, the 5 NN consist of 3 negatives and 2 positives, so we classify it as negative.

Black = negative
Red = positive

Slide credit: David Lowe
1-nearest neighbor
3-nearest neighbor
What are the tradeoffs of having a too large $k$? Too small $k$?
Formal Definition

• Let \( x \) be our test data point, and \( N_K(x) \) be the indices of the \( k \) nearest neighbors of \( x \)

• Classification:

\[
y = \arg\max_c \#(y_i = c)
\]

\[
y = \arg\max_c \sum_{i \in N_K(x)} I(y_i = c)
\]

• Regression:

\[
y = \frac{1}{K} \sum_{i \in N_K(x)} y_i
\]
Example: Predict where this picture was taken

Example: Predict where this picture was taken
Example: Predict where this picture was taken
6+ million geotagged photos by 109,788 photographers

Scene Matches

Scene Matches

Scene Matches

The Importance of Data

k-Nearest Neighbor

Four things make a memory based learner:

1. A distance metric
   - Euclidean (and others)

2. How many nearby neighbors to look at?
   - k

3. A weighting function (optional)
   - Not used

4. How to fit with the local points?
   - Just predict the average output among the nearest neighbors

Slide credit: Carlos Guestrin
Distances

• Suppose I want to charge my overall distance more for differences in $x_2$ direction as opposed to $x_1$ direction
• Setup A: equal weighing on all directions
• Setup B: more weight on $x_2$ direction
• Will my neighborhoods be longer in the $x_1$ or $x_2$ direction?
Voronoi partitioning

- Nearest neighbor regions
- All points in a region are closer to the seed in that region than to any other seed (black dots = seeds)
Multivariate distance metrics

Suppose the input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are two dimensional:

$\mathbf{x}_1 = (x_1^1, x_1^2), \mathbf{x}_2 = (x_2^1, x_2^2), \ldots, \mathbf{x}_N = (x_N^1, x_N^2)$.

The relative scalings in the distance metric affect region shapes.

Adapted from Carlos Guestrin
Distance metrics

- Euclidean: \[ d(x, z) = \left[ \sum_{i=1}^{D} (x_i - z_i)^2 \right]^{\frac{1}{2}} \]

- Minkowski: \[ d(x, z) = \left[ \sum_{i=1}^{D} |x_i - z_i|^P \right]^{\frac{1}{P}} \]

- Mahalanobis: \[ d(x, z) = \sqrt{\sum_{i=1}^{D} \frac{(x_i - z_i)^2}{\sigma_i^2}} \]
  \[ d(x, z) = (x - z)^T A (x - z) \]

(Where \( A \) is a positive semidefinite matrix, i.e. symmetric matrix with all non-negative eigenvalues)
Distance metrics

Voronoi diagrams of 20 points under two different metrics

Euclidean distance

Manhattan distance

Figures from Wikipedia
Another generalization: Weighted K-NNs

• Neighbors weighted differently:
  – Use all samples, i.e. K = N
  – Weight on i-th sample: \( w_i = e^{\frac{-||x-x_i||^2}{\sigma^2}} \)
  – \( \sigma \) = the bandwidth parameter, expresses how quickly our weight function “drops off” as points get further and further from the query \( x \)

• Classification:
  \[ y = \arg\max_c \sum_{i=1}^{N} w_i I(y_i = c) \]

• Regression:
  \[ y = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i} \]
Another generalization: Weighted K-NNs

• Extremes
  – Bandwidth = infinity: prediction is dataset average
  – Bandwidth = zero: prediction becomes 1-NN
Kernel Regression/Classification

Four things make a memory based learner:

• **A distance metric**
  – Euclidean (and others)

• **How many nearby neighbors to look at?**
  – All of them

• **A weighting function (optional)**
  – $w_i = \exp(-d(x_i, \text{query})^2 / \sigma^2)$
  – Nearby points to the query are weighted strongly, far points weakly. The $\sigma$ parameter is the kernel width / bandwidth.

• **How to fit with the local points?**
  – Predict the weighted average of the outputs

Adapted from Carlos Guestrin
Problems with Instance-Based Learning

• Too many features?
  – Doesn’t work well if large number of irrelevant features, distances overwhelmed by noisy features
  – Distances become meaningless in high dimensions (the curse of dimensionality)

• What is the impact of the value of K?

• Expensive
  – No learning: most real work done during testing
  – For every test sample, must search through all dataset – very slow!
  – Must use tricks like approximate nearest neighbor search
  – Need to store all training data

Adapted from Dhruv Batra
Curse of Dimensionality

How many neighborhoods are there?

Fruit data

#bins = 10x10
\[ d = 2 \]

#bins = \(10^d\)
\[ d = 1000 \]

Atoms in the universe
\[ \sim 10^{80} \]
Curse of Dimensionality

Regions become more sparsely populated given the same amount of data

Need more data to densely populate them

kNN Decision Boundary

- Increasing $k$ **complicates** decision boundary
kNN Decision Boundary

- Increasing k “simplifies” decision boundary
  - Majority voting means less emphasis on individual points

K = 1

K = 3
kNN Decision Boundary

- Increasing $k$ “simplifies” decision boundary
  - Majority voting means less emphasis on individual points

$K = 5$  
$K = 7$
kNN Decision Boundary

- Increasing k “simplifies” decision boundary
  - Majority voting means less emphasis on individual points

K = 25
Error rates and K

Use a validation set to pick K

K=1? Zero error! Training data have been memorized...

Too “complex”

Best value of K

Predictive Error

Error on Test Data

Error on Training Data

K (# neighbors)
Summary

- K-Nearest Neighbor is the most basic and simplest to implement classifier
- Cheap at training time, expensive at test time
- Unlike other methods we’ll see later, naturally works for any number of classes
- Pick K through a validation set, use approximate methods for finding neighbors
- Success of classification depends on the amount of data and the meaningfulness of the distance function (also true for other algorithms)
Plan for this lecture

• The simplest classifier: K-Nearest Neighbors
  – Algorithm and example use
  – Generalizing: Distance metrics, weighing neighbors
  – Problems: curse of dimensionality, picking K

• Logistic regression
  – Probability: review
  – Linear regression for classification?
  – Maximum likelihood solution for logistic regression
  – Related algorithm: perceptron
Probability Review

A is non-deterministic event
Can think of A as a Boolean-valued variable

Examples
A = your next patient has cancer
A = Steelers win Super Bowl LIII
Interpreting Probabilities

What does P(A) mean?

Frequentist View

\[
\lim_{N \to \infty} \frac{\text{#}(A \text{ is true})}{N}
\]

frequency of a repeating non-deterministic event

Bayesian View

P(A) is your “belief” about A

Adapted from Dhruv Batra
Axioms of Probability

0 <= P(A) <= 1
P(false) = 0
P(true) = 1
P(A v B) = P(A) + P(B) - P(A ^ B)

Event space of all possible worlds
Its area is 1

P(A) = Area of reddish oval
Axioms of Probability

0 <= P(A) <= 1

P(false) = 0

P(true) = 1

P(A \lor B) = P(A) + P(B) - P(A \land B)

The area of A can’t get any smaller than 0

And a zero area would mean no world could ever have A true
Axioms of Probability

0\leq P(A) \leq 1
P(false) = 0
P(true) = 1
P(A \lor B) = P(A) + P(B) - P(A \land B)

The area of A can’t get any bigger than 1

And an area of 1 would mean all worlds will have A true
Axioms of Probability

0 <= P(A) <= 1
P(false) = 0
P(true) = 1
P(A v B) = P(A) + P(B) - P(A ^ B)

Simple addition and subtraction
Probabilities: Example Use

Apples and Oranges
**Marginal, Joint, Conditional**

Marginal Probability

\[ p(X = x_i) = \frac{c_i}{N}. \]

Joint Probability

\[ p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} \]

Conditional Probability

\[ p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i} \]
Joint Probability

- \( P(X_1, \ldots, X_n) \) gives the probability of every combination of values (an \( n \)-dimensional array with \( v^n \) values if all variables are discrete with \( v \) values, all \( v^n \) values must sum to 1):

<table>
<thead>
<tr>
<th></th>
<th>circle</th>
<th>square</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0.20</td>
<td>0.02</td>
</tr>
<tr>
<td>blue</td>
<td>0.02</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>circle</th>
<th>square</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0.05</td>
<td>0.30</td>
</tr>
<tr>
<td>blue</td>
<td>0.20</td>
<td>0.20</td>
</tr>
</tbody>
</table>

- The probability of all possible conjunctions (assignments of values to some subset of variables) can be calculated by summing the appropriate subset of values from the joint distribution.

\[
P(red \land circle) = \frac{P(red)}{P(red)}
\]

- Therefore, all conditional probabilities can also be calculated.

\[
P(positive \mid red \land circle)
\]

Adapted from Ray Mooney
Marginal Probability

\[ p(x, y) = \sum_{z \in \mathcal{Z}} p(x, y, z) \quad p(x) = \sum_{y \in \mathcal{Y}} p(x, y) \]
Conditional Probability

\[ P(Y=y \mid X=x): \text{What do you believe about } Y=y, \text{ if I tell you } X=x? \]

\[ P(\text{Andy Murray wins Australian Open 2019})? \]

What if I tell you:

- He has won it five times before
- He is currently ranked #307
Conditional Probability

\[ p(X, Y) \]

\[ p(Y) \]

\[ p(X) \]

\[ p(X|Y = 1) \]
Conditional Probability

\[ p(x, y \mid Z = z) = \frac{p(x, y, z)}{p(z)} \]
Sum and Product Rules

Sum Rule
\[ p(X = x_i) = \frac{c_i}{N} = \frac{1}{N} \sum_{j=1}^{L} n_{ij} \]
\[ = \sum_{j=1}^{L} p(X = x_i, Y = y_j) \]

Product Rule
\[ p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} \]
\[ = p(Y = y_j | X = x_i) p(X = x_i) \]
Chain Rule

Generalizes the product rule:

\[ P\left( \bigcap_{k=1}^{n} A_k \right) = \prod_{k=1}^{n} P \left( A_k \mid \bigcap_{j=1}^{k-1} A_j \right) \]

Example:

\[ P(A_4, A_3, A_2, A_1) = P(A_4 \mid A_3, A_2, A_1) \cdot P(A_3 \mid A_2, A_1) \cdot P(A_2 \mid A_1) \cdot P(A_1) \]
Independence

A and B are independent iff:

\[ P(A | B) = P(A) \]
\[ P(B | A) = P(B) \]

These two constraints are logically equivalent.

Therefore, if A and B are independent:

\[ P(A | B) = \frac{P(A \land B)}{P(B)} = P(A) \]
\[ P(A \land B) = P(A)P(B) \]
Independence

Marginal: $P$ satisfies $(X \perp Y)$ if and only if
\[ P(X=x, Y=y) = P(X=x) \cdot P(Y=y), \]
\[ \forall x \in \text{Val}(X), y \in \text{Val}(Y) \]

Conditional: $P$ satisfies $(X \perp Y \mid Z)$ if and only if
\[ P(X, Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z), \]
\[ \forall x \in \text{Val}(X), y \in \text{Val}(Y), z \in \text{Val}(Z) \]
Independence

\[ P(x, y) \]

\[ X \perp Y \]

\[ p(x, y) = p(x)p(y) \]

for all \( x \in \mathcal{X}, y \in \mathcal{Y} \)
Bayes’ Theorem

\[ p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \]

\[ p(X) = \sum_Y p(X|Y)p(Y) \]

posterior \( \propto \) likelihood \( \times \) prior
Expectations

\[ \mathbb{E}[f] = \sum_{x} p(x) f(x) \]

\[ \mathbb{E}[f] = \int p(x) f(x) \, dx \]

\[ \mathbb{E}_x[f|y] = \sum_{x} p(x|y) f(x) \]

Conditional Expectation (discrete)

\[ \mathbb{E}[f] \approx \frac{1}{N} \sum_{n=1}^{N} f(x_n) \]

Approximate Expectation (discrete and continuous)
Entropy

\[ H[x] = - \sum_x p(x) \log_2 p(x) \]

Important quantity in
- coding theory
- statistical physics
- machine learning
Entropy

\[ H = 1.77 \]

\[ H = 3.09 \]
The Kullback-Leibler Divergence

\[
\text{KL}(p\|q) = - \int p(x) \ln q(x) \, dx - \left( - \int p(x) \ln p(x) \, dx \right)
\]

\[
= - \int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} \, dx
\]

\[
\text{KL}(p\|q) \simeq \frac{1}{N} \sum_{n=1}^{N} \left\{ - \ln q(x_n|\theta) + \ln p(x_n) \right\}
\]

\[
\text{KL}(p\|q) \geq 0 \quad \text{KL}(p\|q) \neq \text{KL}(q\|p)
\]
Mutual Information

\[ I[x, y] \equiv \text{KL}(p(x, y) \| p(x)p(y)) \]

\[ = - \iint p(x, y) \ln \left( \frac{p(x)p(y)}{p(x, y)} \right) \, dx \, dy \]

Likelihood / Prior / Posterior

• A hypothesis (model, function, parameter set, weights) is denoted as $h$; it is one member of the hypothesis space $H$
• A set of training examples is denoted as $D$, a collection of $(x, y)$ pairs for training
• $\Pr(h)$ – the prior probability of the hypothesis – without observing any training data, what is the probability that $h$ is the target function we want?

Adapted from Rebecca Hwa
Likelihood / Prior / Posterior

- Pr(D) – the prior probability of the observed data – chance of getting the particular set of training examples D
- Pr(h|D) – the posterior probability of h – what is the probability that h is the target given that we have observed D?
- Pr(D|h) – the probability of getting D if h were true (a.k.a. likelihood of the data)
- Pr(h|D) = Pr(D|h)Pr(h)/Pr(D)
MLE and MAP Estimation

Maximum likelihood estimation (MLE):
\[ h_{ML} = \arg\max Pr(D|h) \]

Maximum-a-posteriori (MAP) estimation:
\[ h_{MAP} = \arg\max_h Pr(h|D) \]
\[ = \arg\max_h Pr(D|h)Pr(h)/Pr(D) \]
\[ = \arg\max_h Pr(D|h)Pr(h) \]
Classification via regression

- Suppose we ignore the fact that the target output $y$ is binary (e.g., 0/1) rather than a continuous variable.
- So we will estimate a linear regression function
  \[ f(x; w) = w_0 + w_1 x_1 + \ldots + w_d x_d \]
  \[ = w_0 + x^T w_1, \]
  based on the available data as before.
- Objective we want to minimize:

\[ J_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i; w))^2 \]
Classification via regression cont’d

- We can use the resulting regression function

\[ f(x; \hat{w}) = w_0 + x^T \hat{w}_1, \]

- to classify any new (test) example \( x \) according to \( \text{label} = 1 \) if \( f(x; w) > 0.5 \), and \( \text{label} = 0 \) otherwise

- \( f(x; \hat{w}) = 0.5 \) therefore defines a linear decision boundary that partitions the input space into two class specific regions (half spaces)
Classification via regression cont’d

- Given the dissociation between the objective (classification) and the estimation criterion (regression) it is not clear that this approach leads to sensible results.
Classification via regression cont’d

\[ f(x, \mathbf{w}) = 0.5 \]

Figures adapted from Andrew Ng
Classification via regression cont’d

\[ f(x, w) = 0.5 \]

Figures adapted from Andrew Ng
The effect of outliers: Another example

Magenta = least squares, green = logistic regression

Figures from Bishop
Logistic regression

• Also has “regression” in the name but it’s a method for classification
• Also uses a linear combination of the features to predict label, but in a slightly different way
• Fit a sigmoid function to model the probability of the data belonging to a certain class

\[ P(y=1|x) \]

\[ f(x) = \text{dot}(w, x) + b \]
Background: simple decision theory

- Suppose we know the class-conditional densities $p(x|y)$ for $y = 0, 1$ as well as the overall class frequencies $P(y)$.

How do we decide which class a new example $x'$ belongs to so as to minimize the overall probability of error?
Background: simple decision theory

Suppose we know the class-conditional densities \( p(x|y) \) for \( y = 0, 1 \) as well as the overall class frequencies \( P(y) \).

How do we decide which class a new example \( x' \) belongs to so as to minimize the overall probability of error?

The minimum probability of error decisions are given by

\[
y' = \operatorname*{arg\,max}_{y=0,1} \{ p(x'|y)P(y) \} = \operatorname*{arg\,max}_{y=0,1} \{ P(y|x') \}
\]
Logistic regression

- The optimal decisions are based on the posterior class probabilities $P(y|x)$. For binary classification problems, we can write these decisions as

$$y = 1 \text{ if } \log \frac{P(y = 1|x)}{P(y = 0|x)} > 0$$

and $y = 0$ otherwise.
Logistic regression

- The optimal decisions are based on the posterior class probabilities $P(y|x)$. For binary classification problems, we can write these decisions as

$$y = 1 \text{ if } \log \frac{P(y = 1|x)}{P(y = 0|x)} > 0$$

and $y = 0$ otherwise.

- We generally don't know $P(y|x)$ but we can parameterize the possible decisions according to

$$\log \frac{P(y = 1|x)}{P(y = 0|x)} = f(x; w) = w_0 + x^T w_1$$
Logistic regression cont’d

- Our log-odds model

\[
\log \frac{P(y = 1|\mathbf{x})}{P(y = 0|\mathbf{x})} = w_0 + \mathbf{x}^T \mathbf{w}_1
\]

gives rise to a specific form for the conditional probability over the labels (the logistic model):

\[
P(y = 1|\mathbf{x}, \mathbf{w}) = \sigma \left( w_0 + \mathbf{x}^T \mathbf{w}_1 \right)
\]

where

\[
\sigma(z) = \frac{1}{1 + \exp(-z)}
\]

is a logistic “squashing function” that turns linear predictions into probabilities.
Logistic regression: decisions

- Logistic regression models imply a linear decision boundary

\[
\log \frac{P(y = 1 | x)}{P(y = 0 | x)} = w_0 + x^T w_1 = 0
\]
Fitting logistic regression models

We can fit the logistic models using the maximum (conditional) log-likelihood criterion

\[ l(D; w) = \sum_{i=1}^{n} \log P(y_i|x_i, w) \]

where

\[ P(y = 1|x, w) = \sigma (w_0 + x^T w_1) \]

Solution: find roots of

\[ \left( y_i - P(y_i = 1|x_i, w) \right) \left[ \begin{array}{c} 1 \\ x_i \end{array} \right] = 0 \]
Stochastic gradient ascent

- We can try to maximize the log-likelihood in an *on-line* or incremental fashion.

Given each training input $x_i$ and the binary (0/1) label $y_i$, we can change the parameters $w$ slightly to increase the corresponding log-probability

$$w \leftarrow w + \eta \frac{\partial}{\partial w} \log P(y_i | x_i, w)$$

$$= w + \eta \left( y_i - P(y_i = 1 | x_i, w) \right) \begin{bmatrix} 1 \\ x_i \end{bmatrix}$$

where $\eta$ is the *learning rate*.
Whiteboard: solution
Logistic Regression / MLE Example

- Want to find the weight vector that gives us the highest $P(y_i | x_i, w)$,
- where $P(y_i=1 | x_i, w) = 1 / (1 + \exp(-w'x))$
- Consider two weight vectors and three samples, with corresponding likelihoods:

|     | $P(y_1 = 1 | x_1, w)$ | $P(y_2 = 1 | x_2, w)$ | $P(y_3 = 1 | x_3, w)$ |
|-----|------------------------|------------------------|------------------------|
| $w_1$ | 0.3                    | 0.1                    | 0.4                     |
| $w_2$ | 0.7                    | 0.8                    | 0.2                     |
| **True label:** | **1**                  | **0**                  | **1**                   |
Logistic Regression / MLE Example

• Then the value of the objective for $w_i$ is:
  \[
P(y_1 = 1 \mid x_1, w_i) \ast (1 - P(y_2 = 1 \mid x_2, w_i)) \ast P(y_3 = 1 \mid x_3, w_i)
  \]

• So the score for $w_1$ is: $0.3 \ast 0.9 \ast 0.4$

• And the score for $w_2$ is: $0.7 \ast 0.2 \ast 0.2$

• Thus, $w_1$ is a better weight vector = model
Plan for this lecture

• The simplest classifier: K-Nearest Neighbors
  – Algorithm and example use
  – Generalizing: Distance metrics, weighing neighbors
  – Problems: curse of dimensionality, picking K

• Logistic regression
  – Probability: review
  – Linear regression for classification?
  – Maximum likelihood solution for *logistic* regression
  – Related algorithm: perceptron
The perceptron algorithm

- Rosenblatt (1962)
- Prediction rule:
  \[ y(x) = f \left( w^T \phi(x) \right) \]
  where
  \[ f(a) = \begin{cases} 
  +1, & a \geq 0 \\
  -1, & a < 0 
  \end{cases} \]

- Want:
  \[ w^T \phi(x_n) t_n > 0 \quad (t_n = +1 \text{ or } -1) \]
- Loss:
  \[ E_P(w) = - \sum_{n \in \mathcal{M}} w^T \phi_n t_n \]
  (just using the Misclassified examples)
The perceptron algorithm

• Loss: \( E_P(w) = -\sum_{n \in M} w^T \phi_n t_n \)

• Learning algorithm update rule:

\[ w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_P(w) = w^{(\tau)} + \eta \phi_n t_n \]

• Interpretation:
  – If sample is misclassified and is positive, make the weight vector more like it
  – If sample is misclassified and negative... *unlike* it
The perceptron algorithm (red=pos)

\[ w = w + x \]

(x is pos)

Figures from Bishop
Summary: Tradeoffs of classification methods thus far

• Nearest neighbors
  – Non-parametric method; basic formulation cannot ignore/focus on different feature dimensions
  – Slow at test time (large search problem to find neighbors)
  – Need to store all data points (unlike SVM, coming next)
  – Decision boundary not necessarily linear
  – Naturally handles multiple classes

• Logistic regression (a classification method)
  – Models the probability of a label given the data
  – Decision boundary corresponds to $w^T x = 0$ (a line)

• Perceptron
  – Same decision boundary as logistic regression (a line)
  – Simple update rule
  – Won’t converge for non-linearly-separable data