YILDIZ TEKNİK ÜNİVERSİTESİ
Fen Edebiyat Fakültesi

YAPAY ZEKAYA GİRİŞ

5. Hafta

Optimizasyon Algoritmaları

- Optimizasyon algoritmaları, karmaşık problemleri çözmek için doğal ve matematiksel yöntemlerden yararlanarak en iyi çözümü bulmayı hedefleyen yöntemlerdir. Bu algoritmalar, doğada gözlemlenen süreçlerden esinlenerek geliştirilmiş ve birçok alanda uygulanabilir hale getirilmiştir.
- Örneğin, genetik algoritmalar, doğal seçilim ve evrim süreçlerini taklit ederken, karınca kolonisi optimizasyonu ve parçacık sürü optimizasyonu gibi yöntemler, sürü davranışlarını temel alarak çözüm arama sürecini yönetir. Bu tür algoritmalar, optimizasyon problemlerinin çözümünde sağladıkları esneklik ve verimlilik nedeniyle özellikle yapay zeka, mühendislik ve veri bilimi gibi alanlarda önemli bir yere sahiptir.

Optimizasyon Algoritmaları

- Bu algoritmaların temel amacı, belirli bir hedefe ulaşırken en iyi sonuçları elde etmektir.
- Optimizasyon, sadece bir çözüm bulmakla kalmaz, aynı zamanda çözümün kalitesini artırmaya yönelik süreçleri de içerir.
- Yapay zeka alanında kullanılan optimizasyon algoritmaları, makine öğrenimi modellerinin eğitimi, parametre ayarlamaları ve veri analizi gibi çeşitli uygulamalarda kritik bir rol oynar.
- Gelişen teknoloji ve veri analitiği ile birlikte, bu algoritmaların uygulanabilirliği ve etkinliği her geçen gün artmaktadır, bu da onların önemini daha da pekiştirmektedir.

Optimizasyon Algoritmaları

1. Yapay Sinir Ağları (Artificial Neural Networks - ANN)

Önem: Derin öğrenme uygulamaları ile birlikte, birçok alanda önemli bir yere sahiptir. Görüntü işleme, doğal dil işleme ve oyun oynama gibi uygulamalarda yaygın olarak kullanılır. (7. Hafta konusu)

2. Genetik Algoritmalar (Genetic Algorithms - GA)

Önem: Karmaşık optimizasyon problemlerinin çözümünde etkili bir yöntemdir. Çok sayıda değişkenin olduğu problemlerde uygulanabilir. (6. Hafta konusu)

3. Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)

Önem: Kombinatoryal optimizasyon problemlerinde (örneğin, seyahat eden satıcı problemi) etkilidir ve veri analizinde kullanılabilir.

4. Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

Önem: Sürekli ve kesikli optimizasyon problemlerinde, makine öğrenimi uygulamalarında sıkça kullanılır.

Optimizasyon Algoritmaları

5. Simüle Tavlama (Simulated Annealing)

Önem: Yerel minimumları geçmek için etkili bir yöntemdir ve çeşitli alanlarda optimizasyon için kullanılabilir.
(4. Hafta işlendi)

6. Hedefli Arama (Greedy Search)

Önem: Çoğu zaman hızlı sonuçlar sunar, ancak optimal çözüm garantisi yoktur. Belirli durumlarda kullanılabilir.
(4. Hafta işlendi)

7. Vezir Problemi (N-Queens Problem)

Önem: Eğitim ve öğretim için sıkça ele alınır; arama ve optimizasyon tekniklerinin uygulanmasına olanak tanır.

8. Gradyan İnişi (Gradient Descent)

Önem: Yapay zeka ve makine öğreniminde temel bir rol oynar; modelin başarısını doğrudan etkiler.

Kullanım Alanları: Makine öğrenimi modellerinin (örneğin, regresyon, yapay sinir ağları) eğitimi için temel bir yöntemdir. Model parametrelerini optimize etmek için yaygın olarak kullanılır.

Optimizasyon Algoritmaları

9. Dinamik Programlama (Dynamic Programming)

Önem: Problem çözme sürecinde verimlilik sağlar; karmaşık optimizasyon problemlerini yönetmek için etkilidir.

Kullanım Alanları: Karmaşık problemleri daha basit alt problemlere bölerek çözme yeteneği ile, özellikle reinforcement learning (pekiştirmeli öğrenme) ve bazı algoritmalarda önemli bir rol oynar.

10. Doğrusal Programlama (Linear Programming)

Önem: Optimum çözümler bulmak için güçlü bir araçtır; birçok endüstride uygulanabilir.

Kullanım Alanları: Lojistik, kaynak dağılımı ve karar verme süreçlerinde sıkça kullanılır. Yine, bazı yapay zeka uygulamalarında karar optimizasyonu için faydalıdır.

11. Newton Yöntemi

Önem: Özellikle daha karmaşık fonksiyonların optimizasyonu için kullanılır; bazı makine öğrenimi uygulamalarında yer alabilir.

Kullanım Alanları: Hızlı çözüm arayışında, gradyan bilgisi kullanarak optimizasyon problemlerinde tercih edilir.

Optimizasyon Algoritmaları

12. Konjuge Gradyan (Conjugate Gradient)

Önem: Bellek verimliliği açısından avantaj sağlar, ancak daha az yaygın olarak kullanılır.

Kullanım Alanları: Büyük ölçekli doğrusal sistemlerde ve bazı makine öğrenimi algoritmalarında kullanılır.

13. Lagrange Çarpanları

Önem: Bazı durumlarda faydalı olsa da, genel optimizasyon problemlerinde daha az sık kullanılır.

Kullanım Alanları: Kısıtlı optimizasyon problemlerini çözmek için kullanılır, ancak yapay zeka uygulamalarında daha sınırlı bir kullanım alanına sahiptir.

14. Boyer-Moore Algoritması

Önem: Yapay zeka alanında öncelikli bir rol oynamaz; daha çok veri analizi ve metin madenciliği gibi uygulamalarda yer alır.

Kullanım Alanları: Metin arama ve eşleştirme problemlerinde kullanılır; yapay zeka ile doğrudan bağlantısı zayıftır.

Optimizasyon Algoritmaları

- *Yukarıda kısaca bahsedilen algoritmalardan bazıları için detaylı açıklamalar ilerleyen slaytlarda yer almaktadır.*

Optimizasyon Algoritmaları

- **Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**

❑ **Genel Açıklamalar:** Karınca Kolonisi Optimizasyonu, doğada karınca kolonilerinin yiyecek bulma davranışlarından esinlenerek geliştirilen bir meta-heuristik (meta-sezgisel) optimizasyon algoritmasıdır. Bu algoritma, karıncaların bir besin kaynağına giden en kısa yolu bulmak için feromon adı verilen kimyasal maddeleri nasıl bıraktığını ve bu feromonların zamanla nasıl buharlaştığını taklit eder. Karıncalar, her geçtikleri yol boyunca feromon bırakarak diğer karıncalara yönlendirme yapar. Zamanla, daha fazla feromon bırakan yollar, diğer karıncalar tarafından daha sık tercih edilir. Bu süreç, en iyi yolların giderek daha fazla feromon biriktirmesine ve sonunda en kısa yolun belirlenmesine yol açar.

❑ **Yapay Zeka ile İlişkisi:** Karınca Kolonisi Optimizasyonu, yapay zeka alanında özellikle karmaşık problemlerin çözümünde yaygın olarak kullanılmaktadır. ACO, keşif ve sömürü stratejileri arasında denge kurarak, karmaşık çözüm alanlarında etkin bir şekilde arama yapar. Bu, özellikle optimizasyon problemleri için önemlidir; zira ACO, yerel optimumdan kaçınmayı ve global optimumu bulmayı hedefler. Yapay zeka sistemlerinde ACO, çözüm uzayını daha verimli bir şekilde araştırmak için farklı kombinasyonlar denemeye olanak tanır.

Optimizasyon Algoritmaları

- **Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**

❑ Yapay Zekadaki Kullanım Alanları ve Örnekleri: Karınca Kolonisi Optimizasyonu, yapay zeka uygulamalarında birçok alanda kullanılmaktadır. Özellikle aşağıdaki alanlarda etkili sonuçlar elde edilmiştir:

1. Rota Optimizasyonu: ACO, lojistik ve ulaşım problemlerinde en kısa yol bulma, dağıtım rotalarının belirlenmesi gibi uygulamalarda kullanılmaktadır. Örneğin, bir dağıtım merkezinin en verimli yolunu bulmak için ACO algoritması kullanılabilir.
2. Tedarik Zinciri Yönetimi: Tedarik zinciri problemlerinde, malzeme akışlarının en iyi şekilde düzenlenmesi için ACO'dan yararlanılabilir. Bu, maliyetlerin azaltılmasına ve verimliliğin artırılmasına yardımcı olur.
3. Makine Öğrenimi: ACO, makine öğrenimi modellerinin parametre optimizasyonunda da kullanılabilir. Örneğin, destek vektör makineleri (SVM) gibi algoritmaların optimizasyonunda ACO uygulanabilir.
4. Görüntü İşleme: Görüntü işleme alanında ACO, kenar tespiti ve segmentasyon gibi işlemlerde kullanılmaktadır. Bu sayede görüntülerdeki belirli nesnelerin daha iyi tanımlanmasına olanak sağlar.

Optimizasyon Algoritmaları

- **Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**
- ACO'nun temel hesaplama adımları ve gerekli tanımlamalar:

❑ **Temel Tanımlar:**

1. **Feromon (τ)**: Karıncaların bıraktığı kimyasal bir madde. Feromon düzeyi, bir yolun ne kadar cazip olduğunu gösterir. Daha fazla feromon, yolun daha iyi bir çözüm olduğunu ifade eder.
2. **Heuristik Bilgi (η)**: Problemin çözümünde yardımcı olan ek bilgiler. Örneğin, mesafe veya maliyet gibi bilgileri içerebilir.
3. **Karıncalar (k)**: ACO'da, problem alanında dolaşan ve çözüm yollarını keşfeden sanal birimlerdir.
4. **Görüş Alanı (R)**: Her bir karıncanın hangi düğümlere gidebileceğini belirleyen kısıtlama. Karıncalar, sadece komşu düğümlerle etkileşimde bulunur.

Optimizasyon Algoritmaları

- **Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**

❑ **Hesaplama Adımları:**

1. **Başlatma**: Algoritmanın başlangıcında karınca sayısı, feromon düzeyi ve heuristik bilgi belirlenir. Başlangıçta feromon düzeyleri genellikle eşit olarak dağıtılır.

2. **Yol Seçimi**: Her karınca, belirli bir düğümden hareket ederken, feromon ve heuristik bilgilere dayalı olarak bir yol seçer.

3. **Yolun Tamamlanması**: Her karınca, belirli bir hedefe ulaşana kadar yollarını izler ve bu süreçte oluşturdukları çözümleri kaydeder.

4. **Feromon Güncelleme**: Karıncalar, buldukları çözüme göre feromon miktarlarını günceller. Daha iyi çözümler, daha fazla feromon bırakır.

5. **Döngüsel İşlem**: Adımlar, belirli bir durdurma kriteri (örneğin, belirli bir iterasyon sayısı veya yeterli çözüm kalitesi) karşılanana kadar tekrarlanır.

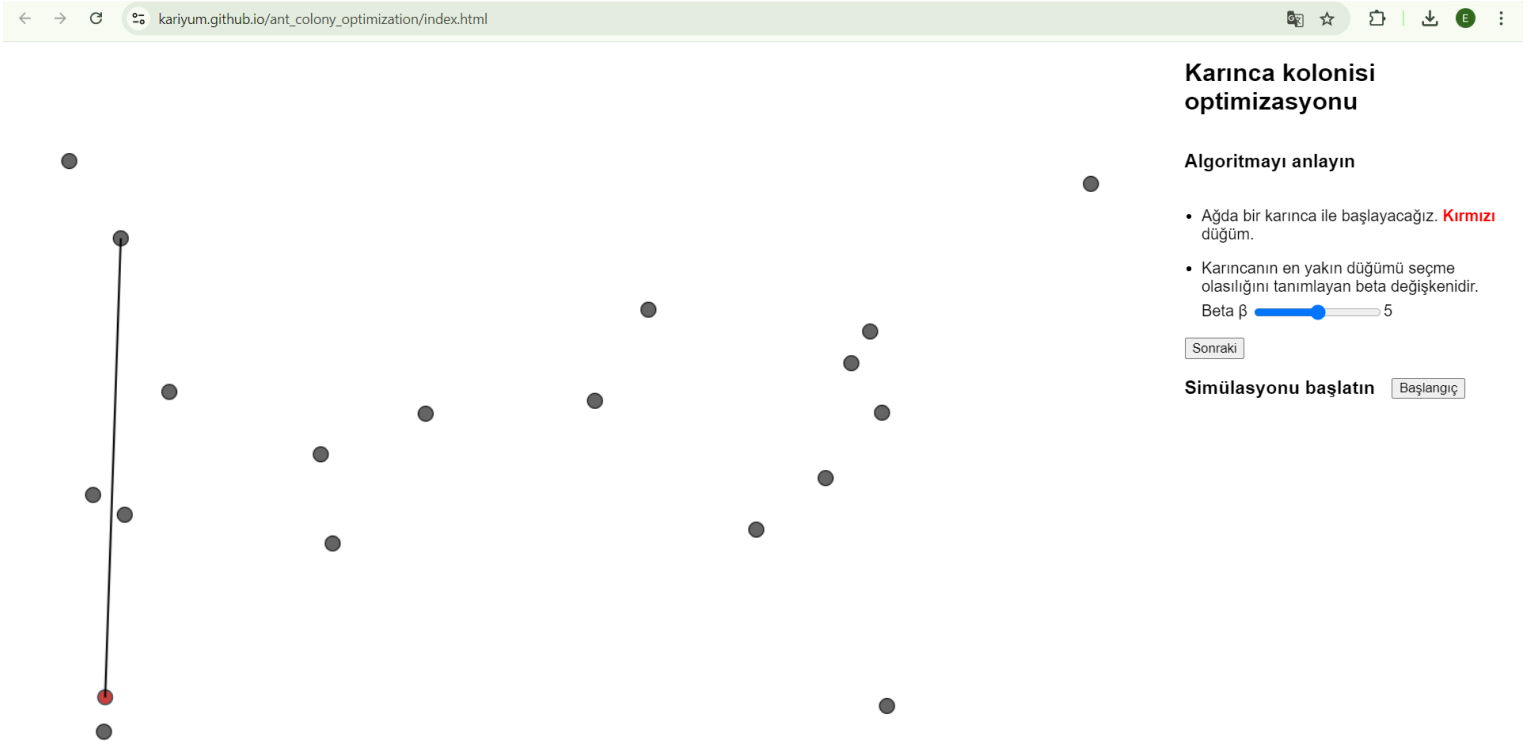
❑ **Sonuç:** Bu adımlar ve tanımlar, Karınca Kolonisi Optimizasyonu algoritmasının temelini oluşturur ve belirli problemleri çözmek için kullanılabilir.

Optimizasyon Algoritmaları

- **Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**

❑ **Uygulama: https://kariyum.github.io/ant_colony_optimization/index.html**

Bu interaktif simülatör, kullanıcıların düğüm ve karınca sayısını ayarlayarak ACO algoritmasını denemelerine olanak tanır. Simülasyon, feromon izlerini ve mevcut maliyetleri gösterir.



Optimizasyon Algoritmaları

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

- **Genel Açıklamalar:** Parçacık Sürü Optimizasyonu, doğada bulunan kuş sürülerinin ve balık gruplarının hareket davranışlarından esinlenerek geliştirilmiş bir meta-heuristik optimizasyon algoritmasıdır. Bu algoritma, her bir "parçacığın" (bir çözüm) problem uzayındaki konumunu güncelleyerek en iyi çözümü bulmaya çalışır. Her parçacık, kendi en iyi konumunu (kişisel en iyi) ve sürüdeki en iyi konumu (global en iyi) takip eder. Bu sayede parçacıklar, hem kendi deneyimlerinden hem de sürülerinin deneyimlerinden yararlanarak problem uzayında etkili bir şekilde dolaşırlar. PSO, basitliği ve düşük hesaplama maliyeti nedeniyle birçok optimizasyon probleminde tercih edilen bir yöntemdir.
- **Yapay Zeka ile İlişkisi:** Parçacık Sürü Optimizasyonu, yapay zeka alanında karmaşık optimizasyon problemlerinin çözümünde yaygın olarak kullanılmaktadır. PSO, keşif ve sömürü arasındaki dengeyi sağlamak için etkili bir yöntem sunar. Bu algoritmanın en önemli avantajlarından biri, yerel optimumdan kaçınabilmesi ve global optimumu bulma potansiyelidir. Yapay zeka sistemlerinde, PSO, parametre optimizasyonu ve model seçimi gibi uygulamalarda kullanılarak öğrenme süreçlerini iyileştirir.

Optimizasyon Algoritmaları

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

- ❑ Yapay Zekadaki Kullanım Alanları ve Örnekleri: Parçacık Sürü Optimizasyonu, yapay zeka uygulamalarında çeşitli alanlarda etkili sonuçlar sağlamaktadır. Aşağıda bazı kullanım alanları ve örnekleri verilmiştir:

1. Parametre Optimizasyonu: PSO, makine öğrenimi modellerinin hiperparametrelerini optimize etmek için kullanılmaktadır. Örneğin, bir yapay sinir ağı modelinin katman sayısını ve nöron sayısını optimize etmek için PSO algoritması uygulanabilir.
2. Fonksiyon Optimizasyonu: PSO, karmaşık matematiksel fonksiyonların minimum veya maksimum değerlerini bulmak için kullanılabilir. Bu, mühendislik tasarımında ve araştırma geliştirmede önemli bir yer tutar.
3. Görüntü İşleme: Görüntü segmentasyonu ve nesne tanıma gibi alanlarda PSO, parametre ayarlama ve model seçimi için etkili bir yöntemdir. Bu, görüntülerdeki belirli nesnelerin doğru bir şekilde tanımlanmasına yardımcı olur.
4. Kontrol Sistemleri: PSO, kontrol sistemlerinin optimizasyonunda da kullanılabilir. Örneğin, bir robotun hareketinin optimize edilmesi ya da dinamik sistemlerin kontrol parametrelerinin ayarlanması için PSO kullanılabilir.
5. İş Süreçleri ve Tedarik Zinciri Yönetimi: PSO, iş süreçlerinin optimize edilmesi ve tedarik zinciri yönetiminde daha iyi karar verme için de etkili bir yöntemdir.

Optimizasyon Algoritmaları

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

❑ Temel Tanımlar

1. **Parçacık (Particle)**: PSO algoritmasındaki her birey, bir çözüm adayıdır. Parçacıklar çözüm uzayında dolaşır.
2. **Konum (Position)**: Her parçacığın çözüm uzayındaki mevcut pozisyonu, çözüm için bir noktayı temsil eder.
3. **Hız (Velocity)**: Parçacığın konum değiştirme hızıdır. Her iterasyonda parçacığın pozisyonunu günceller.
4. **Kişisel En İyi (Personal Best - pbest)**: Her parçacığın şimdiye kadar keşfettiği en iyi çözüm.
5. **Küresel En İyi (Global Best - gbest)**: Tüm parçacıkların şimdiye kadar keşfettiği en iyi çözüm.

Optimizasyon Algoritmaları

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

❑ **Hesaplama Adımları**

1. **Başlatma**: Parçacıkların konumları ve hızları rastgele seçilir. Her parçacık çözüm uzayında rastgele bir konumda başlar.

2. **Uygunluk Değerinin (Fitness) Hesaplanması**: Her parçacığın mevcut konumuna karşılık gelen uygunluk (fitness) değeri hesaplanır. Bu değer, çözülen probleme göre optimize edilen ölçüttür (örneğin, minimum maliyet, maksimum kazanç).

3. **Kişisel ve Küresel En İyi Değerlerin Güncellenmesi**: Her parçacık için o ana kadar keşfettiği en iyi çözüm pbest ve tüm parçacıkların bulduğu en iyi çözüm gbest belirlenir.

4. **Hız ve Konum Güncellenmesi**: Her parçacığın hızı ve pozisyonu güncellenir:

5. **Döngüsel İşlem**: Bu adımlar, belirlenen iterasyon sayısına ulaşılana veya çözüm uygunluk değeri yeterince optimize edilene kadar tekrarlanır.

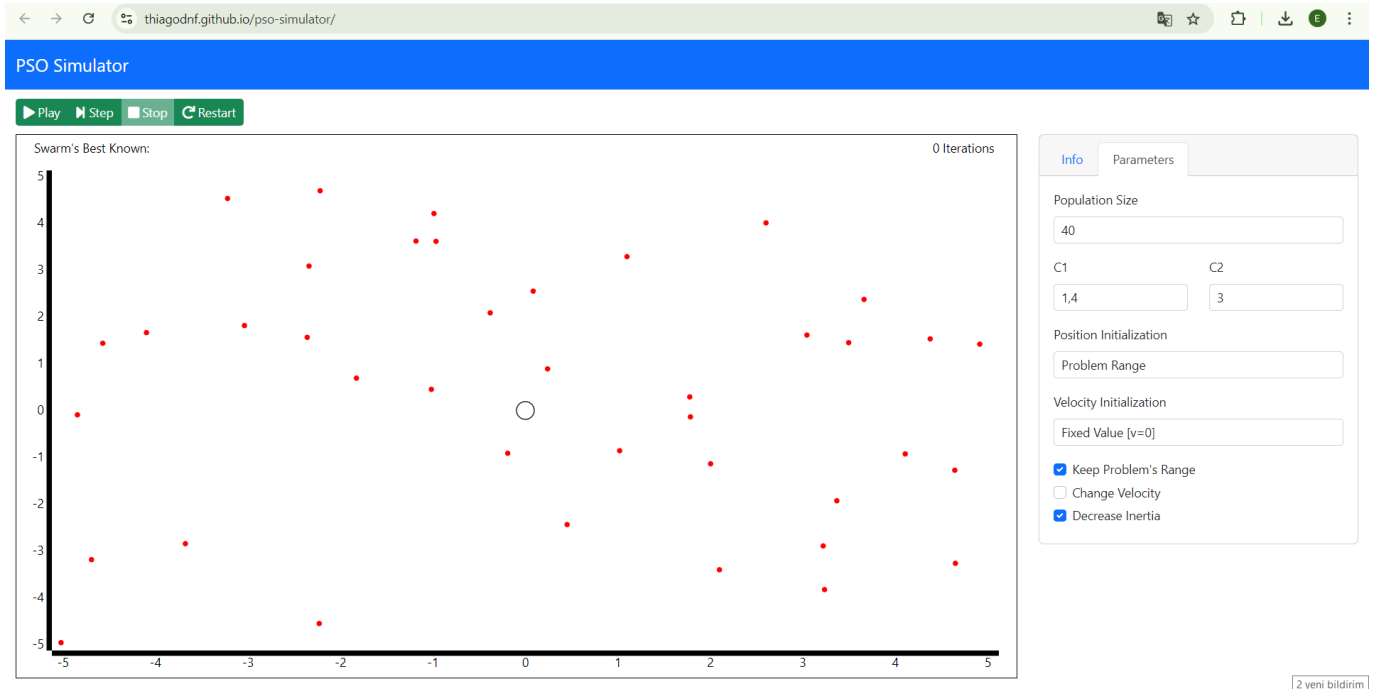
❑ **Sonuç:** Parçacık Sürü Optimizasyonu, parçacıkların grup davranışını taklit ederek en iyi çözümü bulmaya çalışır. Sürekli olarak kişisel ve küresel en iyi çözümler üzerinden güncellemeler yaparak, global optimuma yakın sonuçlar elde edebilir.

Optimizasyon Algoritmaları

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

❑ Uygulama: https://thiagodnf.github.io/pso-simulator/

Web tarayıcısı üzerinden çalışan basit bir PSO simülatörü. Kullanıcılar, popülasyon boyutunu, hız ayarlarını ve diğer parametreleri düzenleyerek simülasyonu interaktif olarak deneyimleyebilirler. Bu simülatör, PSO'nun optimizasyon sürecini ve sonuçlarını görsel olarak takip etmek için uygun bir araçtır.



C1: Parçacığın kendi en iyi çözümüne doğru yönelme eğilimini ifade eder.

C2: Parçacığın sürüdeki en iyi çözüme doğru yönelme eğilimini ifade eder.

Optimizasyon Algoritmaları

Vezir Problemi (N-Queens Problem)

- **Genel Açıklamalar:** Vezir Problemi, klasik bir kombinatoryal optimizasyon problemidir. Amaç, N×N'lik bir satranç tahtasına N adet vezirin yerleştirilmesidir; bu yerleştirme sırasında, hiçbir vezirin birbirini tehdit etmemesi gerekmektedir. Yani, her vezir aynı satırda, sütunda veya çaprazda bulunmamalıdır. Bu problem, başlangıçta matematiksel bir bulmaca olarak ortaya çıkmış olsa da, çözüm yöntemleriyle birlikte bilgisayar bilimleri alanında da önemli bir yere sahip olmuştur. Vezir Problemi, özellikle arama ve optimizasyon algoritmalarının etkinliğini test etmek için kullanılan klasik örneklerden biridir.
- **Yapay Zeka ile İlişkisi:** Vezir Problemi, yapay zeka alanında genellikle arama algoritmaları ve heuristik yöntemlerin test edilmesi için kullanılır. Bu problem, bir çözüm bulmanın yanı sıra, çözüm bulma süreçlerinde karşılaşılan zorlukları ve optimizasyon yöntemlerini anlamak için de önemli bir örnek teşkil eder. Yapay zeka sistemleri, Vezir Problemi'ni çözmek için çeşitli arama stratejileri (örneğin, geri izleme, genetik algoritmalar) ve heuristikler (örneğin, en iyi ilk) kullanabilirler.

Optimizasyon Algoritmaları

Vezir Problemi (N-Queens Problem)

- **Yapay Zekadaki Kullanım Alanları ve Örnekleri:** Vezir Problemi, yapay zeka ve bilgisayar bilimleri alanında pek çok uygulama alanına sahiptir. Aşağıda bazı kullanım alanları ve örnekleri verilmiştir:

1. **Arama ve Optimizasyon Algoritmaları:** Vezir Problemi, geri izleme (backtracking) ve genetik algoritmalar gibi arama ve optimizasyon yöntemlerinin etkinliğini test etmek için yaygın olarak kullanılır. Bu algoritmalar, daha karmaşık problemlerin çözümünde de uygulanabilir.
2. **Kısıtlı Programlama:** Vezir Problemi, kısıtlı programlama alanında da kullanılmaktadır. Bu bağlamda, problem, kısıtları yönetme ve optimizasyon tekniklerinin geliştirilmesi için bir test alanı olarak hizmet eder.
3. **Eğitim ve Öğretim Araçları:** Vezir Problemi, algoritma derslerinde ve yapay zeka eğitiminde öğretici bir örnek olarak kullanılmaktadır. Öğrenciler, problem çözme stratejilerini ve algoritma geliştirme becerilerini uygulamak için bu problemi çözebilirler.
4. **Yapay Zeka Oyunları:** Vezir Problemi, yapay zeka tabanlı oyunlarda düşmanların veya rakiplerin hareketlerini planlamak için de kullanılabilir. Örneğin, satranç oyunlarında rakibin olası hamlelerini analiz etme yeteneği geliştirilir.
5. **Evrensel Çözüm Yöntemleri:** Vezir Problemi, bilgisayar bilimleri alanında genel bir çözüm yöntemi olarak değerlendirilir ve birçok karmaşık problemi anlamak için bir başlangıç noktası sağlar. Çözüm yöntemleri, çeşitli uygulamalara ve problemlere aktarılabilir.

Optimizasyon Algoritmaları

Gradyan İnişi (Gradient Descent)

- **Genel Açıklamalar:** Gradyan İnişi, bir fonksiyonun minimum noktasını bulmak için kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, türev bilgisi kullanarak bir fonksiyonun eğimini (gradyanını) hesaplar ve ardından bu eğim doğrultusunda belirli bir adım büyüklüğü ile geriye doğru hareket eder. Temel olarak, hedeflenen minimum değere ulaşmak için en hızlı yol olarak kabul edilir. Gradyan İnişi genellikle makine öğrenimi ve istatistikte model eğitimi sırasında, parametrelerin optimizasyonu için kullanılır. Algoritmanın çeşitli versiyonları (örneğin, Stokastik Gradyan İnişi ve Mini-Batch Gradyan İnişi) vardır ve her biri belirli durumlar için daha etkili olabilir.
- **Yapay Zeka ile İlişkisi:** Gradyan İnişi, yapay zeka ve özellikle derin öğrenme alanında temel bir rol oynamaktadır. Yapay sinir ağlarının eğitiminde, modelin ağırlıklarının ve parametrelerinin ayarlanması için en yaygın olarak kullanılan yöntemlerden biridir. Modelin kayıp fonksiyonunu minimize etmek için kullanılır; bu, modelin tahminlerinin doğruluğunu artırmak için gereklidir. Gradyan İnişi, yalnızca temel bir optimizasyon aracı değil, aynı zamanda daha karmaşık yapay zeka sistemlerinin geliştirilmesinde kritik bir bileşendir.

Optimizasyon Algoritmaları

Gradyan İnişi (Gradient Descent)

- Yapay Zekadaki Kullanım Alanları ve Örnekleri: Gradyan İnişi, yapay zeka ve makine öğrenimi alanında birçok uygulama alanına sahiptir:

1. Derin Öğrenme: Gradyan İnişi, yapay sinir ağlarının eğitilmesinde yaygın olarak kullanılır. Sinir ağlarının katmanlarındaki ağırlıklar, kayıp fonksiyonunu minimize etmek amacıyla gradyan inişi ile güncellenir.
2. Regresyon Analizi: Lineer regresyon ve lojistik regresyon gibi yöntemlerde, model parametreleri Gradyan İnişi kullanılarak optimize edilir. Bu, tahminlerin doğruluğunu artırmak için gereklidir.
3. Hiperparametre Optimizasyonu: Makine öğrenimi modellerinin hiperparametreleri, gradyan inişi ile optimize edilerek en iyi model performansına ulaşılabilir. Bu, modelin genel başarısını artırır.
4. Görüntü İşleme: Görüntü sınıflandırma ve nesne tanıma gibi görevlerde, gradyan inişi kullanılarak görüntü işleme algoritmalarının parametreleri optimize edilir.
5. Oyun Teorisi: Gradyan İnişi, oyun stratejilerinin belirlenmesi için de kullanılabilir. Özellikle, belirli bir stratejinin ödül fonksiyonunu en üst düzeye çıkarmak amacıyla uygulanır.
6. Doğal Dil İşleme: Dil modellerinin eğitilmesi sırasında, gradyan inişi kullanılarak modelin parametreleri optimize edilir ve dil anlama yeteneği artırılır.

Optimizasyon Algoritmaları

Gradyan İnişi (Gradient Descent)

- Gradyan iniş ve onun farklı varyasyonları, optimizasyon süreçlerinde yaygın olarak kullanılan yöntemlerdir. Her birinin kendine özgü avantajları ve kullanımları vardır:

1. Gradyan İnişi (Gradient Descent)

Tanım: Bir fonksiyonun minimumunu bulmak için, her adımda fonksiyonun türevine (gradyanına) bakarak en hızlı inişi takip eder.

Kullanımı: Basit ve etkili bir yöntemdir, ancak büyük veri setlerinde ve karmaşık fonksiyonlarda yavaş kalabilir. Öğrenme oranı uygun seçilmezse, optimizasyon süreci ya çok yavaş olabilir ya da atlama yapabilir.

2. Momentumlu Gradyan İnişi (Gradient Descent with Momentum)

Tanım: Gradyan inişin ivmeli bir versiyonudur. Önceki adımlardaki gradyanları hesaba katar ve bu sayede daha hızlı yakınsama sağlar.

Özellik: Yerel minimumlarda takılmayı önlemek için hareket hızını artırır. Momentum terimi sayesinde daha kararlı bir öğrenme sağlar.

3. AdaGrad (Adaptive Gradient Algorithm)

Tanım: Her parametre için farklı bir öğrenme oranı belirleyerek, nadir görülen özelliklere daha fazla dikkat eder.

Özellik: Özellikle seyrek veri setlerinde etkilidir. Ancak, zamanla öğrenme oranı çok küçülebilir ve algoritmanın durmasına yol açabilir.

Optimizasyon Algoritmaları

Gradyan İnişi (Gradient Descent)

4. RMSProp (Root Mean Square Propagation)

Tanım: AdaGrad'ın zayıf yönlerini gidermek için ortaya çıkmıştır. Öğrenme oranlarını zamanla dengeler.

Özellik: Öğrenme oranlarını normalize ederek, çok büyük ya da çok küçük gradyanların etkilerini dengeler. Daha uzun vadede stabil sonuçlar verir ve AdaGrad'ın yavaşlamasını önler.

5. Adam (Adaptive Moment Estimation)

Tanım: Gradyan iniş, momentum ve RMSProp yöntemlerini birleştirir. Hem öğrenme oranlarını adapte eder hem de momentum kullanır.

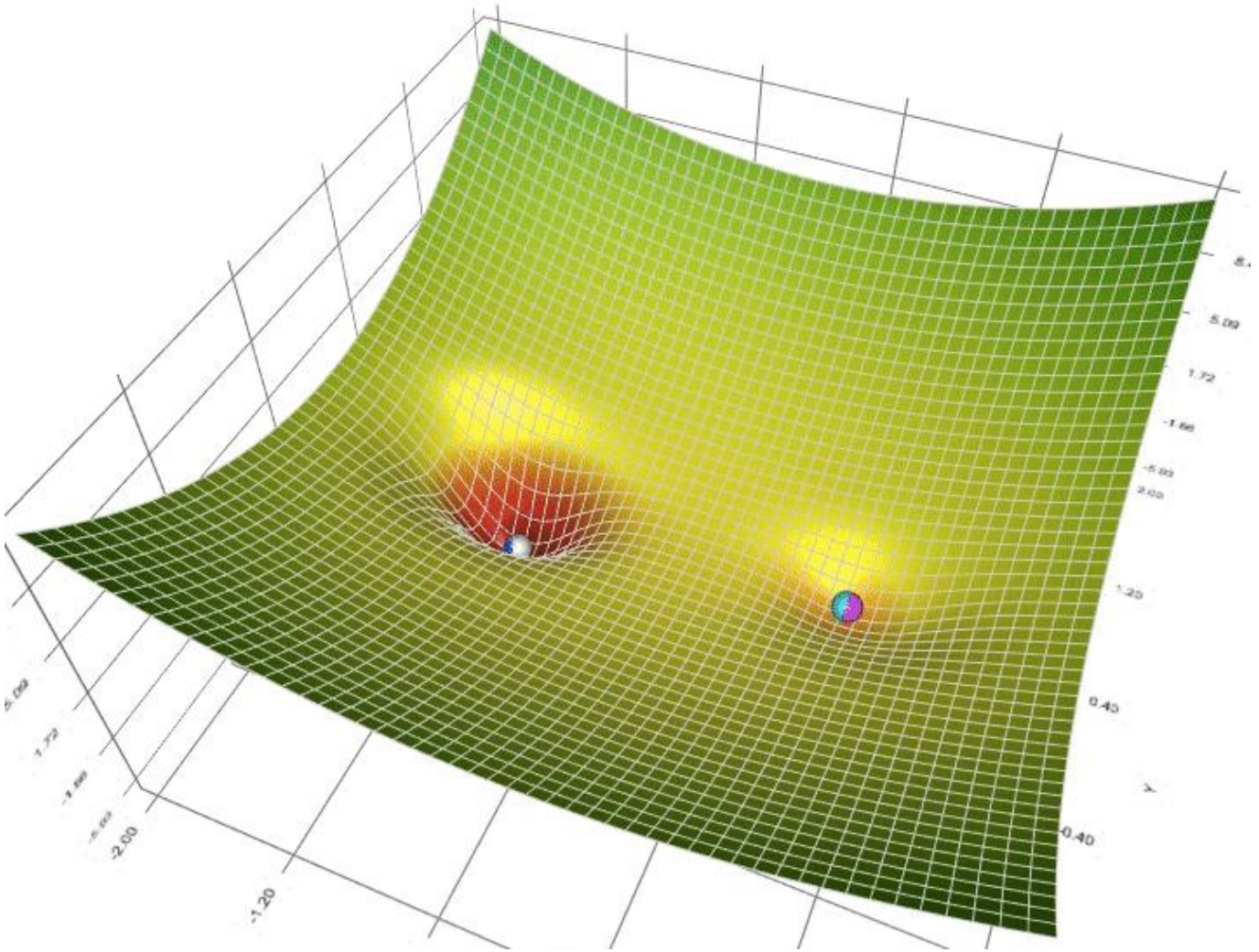
Özellik: Hızlı ve kararlı bir yakınsama sağlar. Hem seyrek veri setlerinde hem de büyük veri setlerinde oldukça etkili bir yöntemdir.

Bu yöntemler, yapay zeka ve makine öğrenimi modellerinin eğitim süreçlerinde optimizasyonu hızlandırmak ve daha iyi sonuçlar elde etmek için kullanılır. Her yöntemin farklı veri setlerine ve problemlere göre avantajları ve dezavantajları vardır.

Optimizasyon Algoritmaları

Gradyan İnişi (Gradient Descent)

- Bir yüzeyde 5 gradyan iniş yönteminin animasyonu: gradyan iniş (turkuaz), momentum (mor), AdaGrad (beyaz), RMSProp (yeşil), Adam (mavi). Soldaki kuyu **küresel minimum**dur; sağdaki kuyu **yerel bir minimum**dur.



WEKA Programı

WEKA

- Waikato üniversitesinde açık kaynak kodlu olarak JAVA dili üzerinde geliştirilmiştir. İsmi de buradan gelir ve Waikato Environment for Knowledge Analysis kelimelerinin baş harflerinden oluşur.
- WEKA verileri basit bir dosyadan okur ve değişkenlerin sayısal veya nominal değerler olduğunu kabul eder. Aynı zamanda veri tabanı (database) üzerinden de veri çekebilir ancak bu durumda verilerin bir dosya verisi şeklinde olması beklenir.
- WEKA üzerinde makine öğrenmesi ve istatistik ile ilgili pek çok kütüphane hazır olarak gelmektedir.
- WEKA İndirme Linki: https://waikato.github.io/weka-wiki/downloading_weka/

WEKA Programı

WEKA

- Örneğin veri ön işlemesi (data preprocessing), ilkelleme (regression), sınıflandırma (classification), gruplandırma (clustering), özellik seçimi veya özellik çıkarımı (feature extraction) bunlardan bazılarıdır. Ayrıca bu işlemler sonucunda çıkan neticelerinde görsel olarak gösterilmesini sağlayan görüntüleme (visualization) araçları bulunmaktadır.

WEKA Programı

WEKA

- Dosya Yapısı
  - İngilizce, Attribute Relationship File Format kelimelerinin baş harflerinden oluşmuştur. ARFF dosya yapısı, Weka'ya özel olarak geliştirilmiştir ve dosya, metin yapısında tutulmaktadır.
  - Dosyanın ilk satırında, dosyadaki ilişki tipi (relation) tutulmakta olup ikinci satırdan itibaren veri kümesindeki özellikler (attributes) yazılmaktadır.
  - Özelliklerin hemen ardından veri kümesi yer alır ve veri kümesindeki her satır bir örneğe (instance) işaret etmektedir. Ayrıca veri kümesindeki her örneğin her özelliği arasında da virgül ayıracı kullanılmaktadır.

WEKA Programı

WEKA

- Dosya Yapısı
  - Şekil'de verilen örnek dosyada, hava tahmini için kullanılan nem, sıcaklık ve basınç değerleri bir dosya içerisinde 4 örnek içerecek şekilde gösterilmiştir. Bu değerler tip olarak sayısal değerler olduğundan "numeric" olarak ifade edilmiştir.
  - NUMERIC: Sayısal değerlerde kullanılır.

```
@relation havatahmini

@attribute nem numeric
@attribute sıcaklık numeric
@attribute basınç numeric
@attribute tahmin numeric

@data
53,25,1013,1
41,22,1011,-1
54,18,1012,-1
67,23,1000,1
```

WEKA Programı

WEKA

- Dosya Yapısı-Değişken Türleri
  - NOMINAL: [Küme Değerleri] Bir tanım kümesi içerisinden kategorik değerler alır. Örneğin {güneşli,yağmurlu,sisli} şeklinde tanımlanan bir kümede, bu özellik kümedeki tanımlı değerlerden birisini alabilir.
  - REAL: [Reel Sayılar] kümesinden bir değer verileceğinde kullanılır.
  - STRING: Veri kümesinin bu özelliğinin serbest yazı şeklinde olabileceğini ifade eder. Özellikle metin madenciliği çalışmaları için sıkça kullanılan bir tiptir.
  - DATE: Veri kümesinin bu özelliğinin tarih olduğunu ifade eder. Örneğin veri kümesindeki kişilerin doğum tarihi veya örneklerin toplanma tarihi gibi özelliklerin tutulmasında kullanılabilir.

WEKA Programı

WEKA

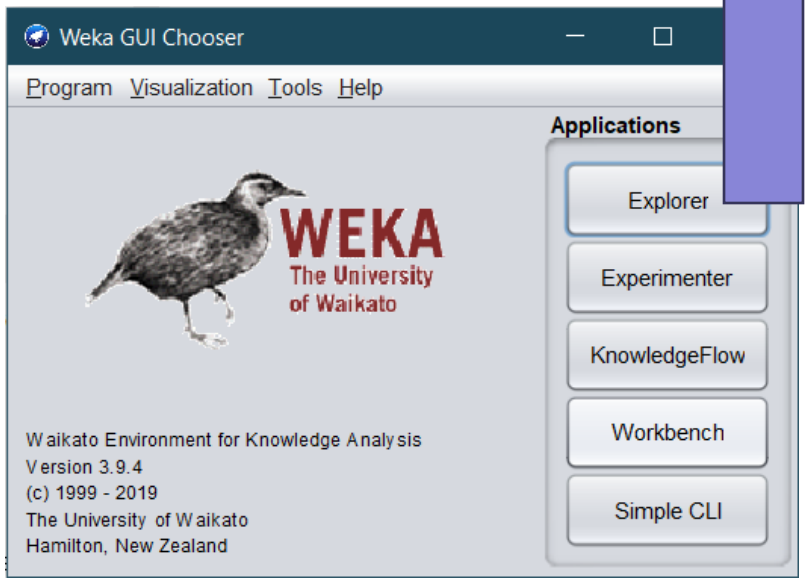
- Dosya Yapısı-Değişken Türleri

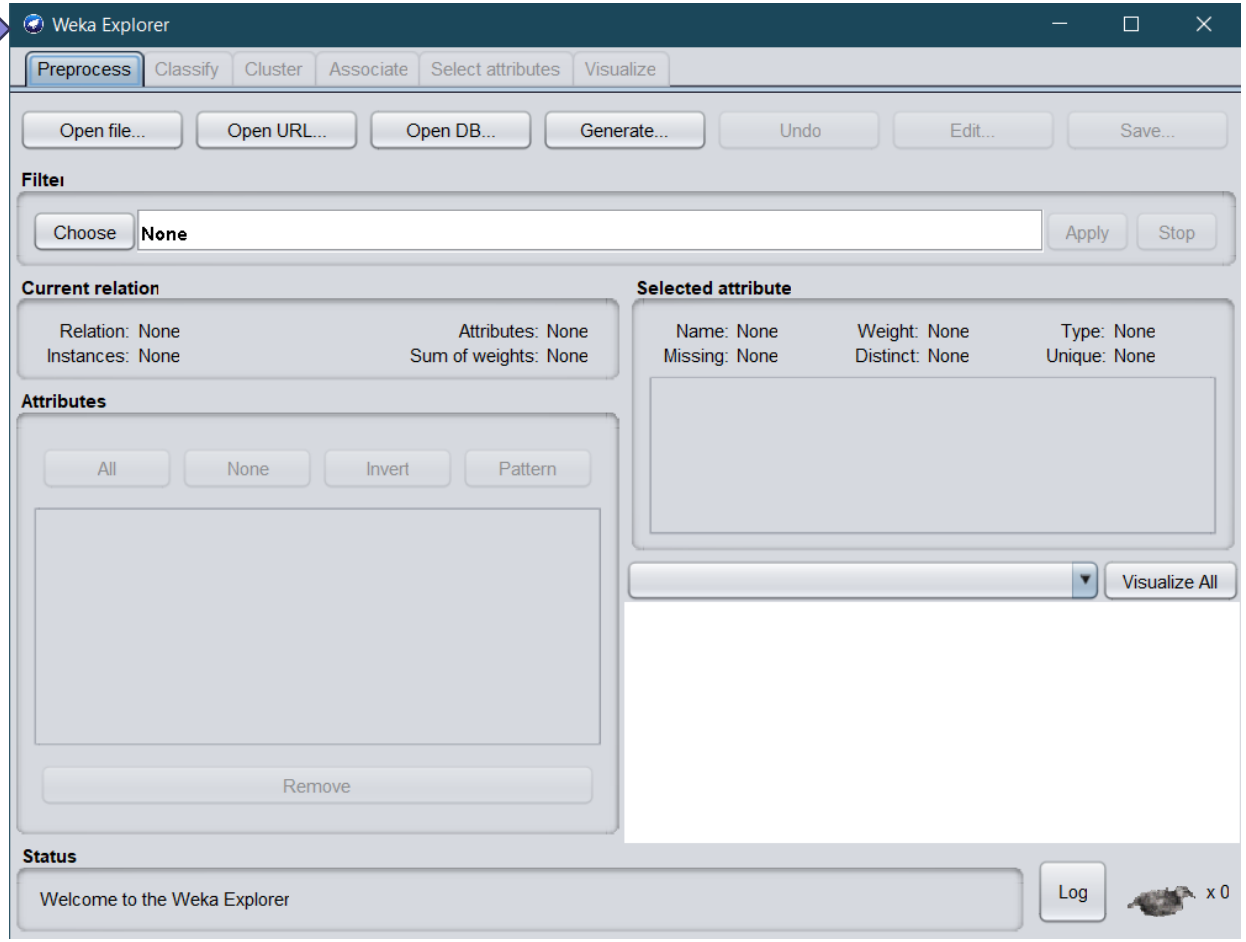
```
% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no
```

WEKA Programı

WEKA

- Uygulama Başlatma: Explorer

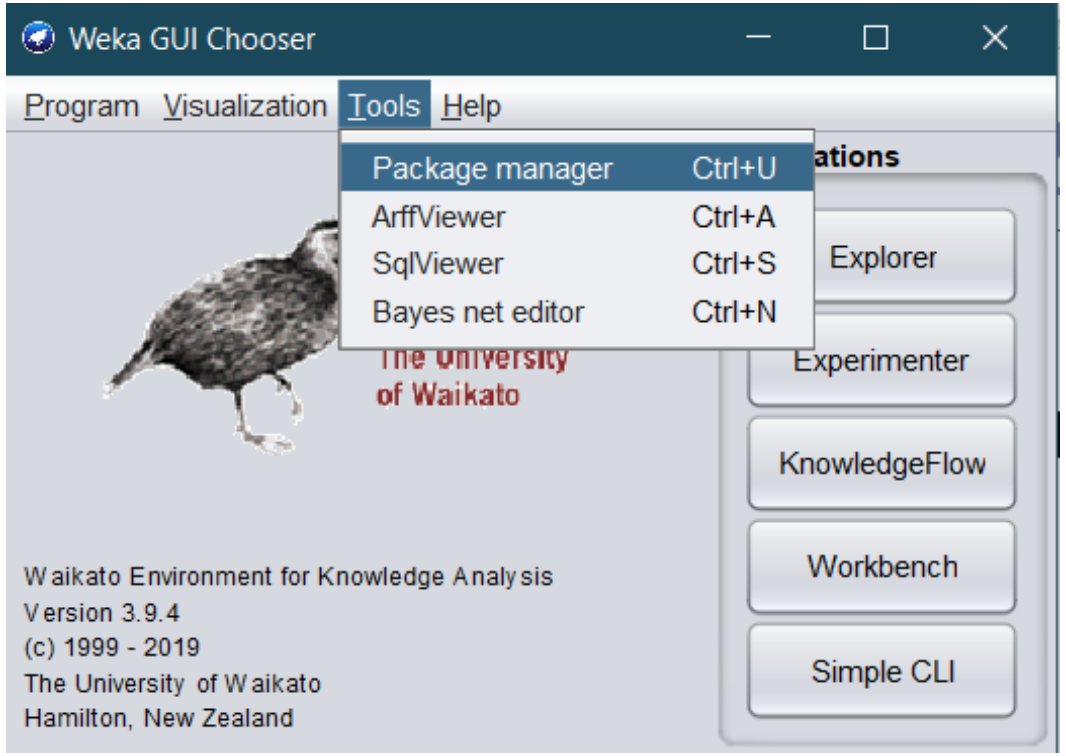




WEKA Programı

WEKA

- **Kütüphane Yükleme:** Tools – Package Manager
- Weka'da **kütüphaneleri yüklemek için** genellikle **Package Manager** kullanılır. Package Manager, Weka'ya yeni algoritmalar, veri ön işleme araçları, modelleme teknikleri gibi ek fonksiyonellikler eklemek için paketleri yönetme imkanı tanır. Bu sayede, mevcut Weka kurulumunun sunduğu araçların ötesine geçerek özelleştirilmiş çözümler geliştirilebilir.



WEKA Programı

WEKA

- **Görselleştirme:** Visualization